TWO APPLICATIONS OF COMBINATORIAL BRANCH-AND-BOUND IN COMPLEX NETWORKS AND TRANSPORTATION

A Dissertation
submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Saeid Rasti

In Partial Fulfillment of the Requirements
for the Degree of
DOCTOR OF PHILOSOPHY

Major Program:
Industrial and Manufacturing Engineering

September 2020

Fargo, North Dakota

North Dakota State University
Graduate School

Title

TWO APPLICATIONS OF COMBINATORIAL BRANCH-AND-BOUND IN COMPLEX NETWORKS AND TRANSPORTATION

By

Saeid Rasti

The Supervisory Committee certifies that this **disquisition** complies with North Dakota State University's regulations and meets the accepted standards for the degree of

DOCTOR OF PHILOSOPHY

Approved:

11/08/2020	Dr. David Grewell
Date	Department Chair

ABSTRACT

In this dissertation, we show two significant applications of combinatorial branch-and-bound as an exact solution methodology in combinatorial optimization problems. In the first problem, we propose a set of new group centrality metrics and show their performance in estimating protein importance in protein-protein interaction networks. The centrality metrics introduced here are extensions of well-known nodal metrics (degree, betweenness, and closeness) for a set of nodes which is required to induce a specific pattern. The structures investigated range from the "stricter" induced stars and cliques, to a "looser" definition of a representative structure. We derive the computational complexity for each of the newly proposed metrics. Then, we provide mixed integer programming formulations to solve the problems exactly; due to the computational complexity of the problem and the sheer size of protein-protein interaction networks, using a commercial solver with the formulations is not always a viable option. Hence, we also propose a combinatorial branch-and-bound approach to solve the problems introduced. Finally, we conclude this work with a presentation of the performance of the proposed centrality metrics in identifying essential proteins in helicobacter pylori. In the second problem, we introduce the asymmetric probabilistic minimum-cost Hamiltonian cycle problem (APMCHCP) where arcs and vertices in the graph are possible to fail. APMCHCP has applications in many emerging areas, such as post-disaster recovery and electronic circuit design. For each vertex, we define a chance-constraint to guarantee that the probability of arriving at the vertex must be greater than or equal to a given threshold. Four mixed-integer programming (MIP) formulations are proposed for modeling the problem, including two direct formulations and two recursive formulations. A combinatorial branch-and-bound (CBB) algorithm is proposed for solving the APMCHCP, where data preprocessing steps, feasibility rules, and approaches of finding upper and lower bounds are developed. In the numerical experiments, the CBB algorithm is compared with formulations on a test-bed of two popular benchmark instance sets. The results show that the proposed CBB algorithm significantly outperforms formulations in terms of both the size of optimally solved instances and the computing time.

ACKNOWLEDGEMENTS

I wish to thank my committee members who were more than generous with their expertise and precious time during the past 5 years. A special thanks to *Dr. Chrysafis Vogiatzis* and *Dr. Yiwen Xu* my committee co-chairmans for their countless hours of reflecting, reading, encouraging, and most of all patience throughout the entire process. Thank you Dr. David Grewell, Dr. Om Yadav, and Dr. Mijia Yang for agreeing to serve on my committee.

Last but not the least, I would like to acknowledge and thank my school division for allowing me to conduct my research and providing any assistance requested.

Fargo, September 2020

Saeid Rasti

DEDICATION

“I dedicate this dissertation to my loving wife *Shiva* who has been by my side through all the ups and downs of my doctorate program and whose words of encouragement has motivated me over the years. A special feeling of gratitude to my parents who have thought me the value of hard work and have been a positive light in my life.”

Saeid Rasti

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

TSP	Traveling Salesman Problem
GPP	Graph Partitioning Problem
QAP	Quadratic Assignment Problem
B&B	Branch-and-Bound
MIP	Mixed Integer Program
ILP	Integer Linear Programming
CBFS	Cyclic Best-First Search
DFS	Depth-First Search
BrFS	Breadth-First Search
BFS	Best-First Search
SOS	Special Ordered Sets
SOS1	Special Ordered Sets type 1
SOS2	Special Ordered Sets type 2
CBB	Combinatorial Branch-and-Bound
LMP	Linear Master Problem
RLMP	Restricted Linear Relaxation of the Master Problem
PPIN	Protein-Protein Interaction Network
GRN	Gene Regulatory Network
MN	Metabolic Network
COBIM	COmplex BIological Modules
ECOBIM	Essential COmplex BIological Modules
Y2H	Yeast 2 Hybrid

AP/MS	Affinity Purification followed by Mass Spectrometry
IPR	Interaction Pathway Reliability index
MCODE	Molecular COmplex DEtection
COACH	COre/AttaCHment-based method
ClusterONE	Clustering with Overlapping Neighborhood Expansion
MCL	Markov Clustering
DECAFF	Dense Neighborhood Extraction using Connectivity And confidence Features
RNSC	Restricted Neighborhood Search Clustering
GO	Gene Ontology
SWEMODE	Semantic WEights for MODule Elucidation
CEZANNE	Co-Expression Zone Analysis using NEtworks
TC-PINs	Time Course Protein Interaction Network
CoEWCC	Co- Expression Weighted by Clustering Coefficient
HC	Harmonic Centrality
UC	United complex Centrality
NTPGE	Network Topology-based Prediction of Gene Essentiality
SVM	Support Vector Machine
RD	Representative Degree
SD	Star Degree
CD	Clique Degree
MCSP	Most Closeness-central Star Problem

MCCP	Most Closeness-central Clique Problem
MCRP	Most Closeness-central Representative Problem
MDSP	Most Degree-central Star Problem
MDCP	Most Degree-central Clique Problem
MDRP	Most Degree-central Representative Problem
MBSP	Most Betweenness-central Star Problem
MBCP	Most Betweenness-central Clique Problem
MBRP	Most Betweenness-central Representative Problem
MCHCP	Minimum-Cost Hamiltonian Cycle Problem
APMCHCP	Asymmetric Probabilistic Minimum-Cost Hamiltonian Cycle Problem
PDR	Post-Disaster Recovery
WSN	Wireless Sensor Networks
MTZ	Miller-Tucker-Zemlin
PHC	Probabilistic Hamiltonian Cycle
RAPMCHCP	Restricted APMCHCP
ESPPRC	Elementary Shortest Path Problem with Resource Constraints
ESPPCC	Elementary Shortest Path Problem with Chance Constraints
SPPCC	Shortest Path Problem with Chance Constraints
PSP	Probabilistic Shortest Path
MPNPR	Maximum Possible Number of Precedence Relationship

LIST OF SYMBOLS

$\mathscr{U}$	unexplored subproblems
T	search tree
b	branching factor
d	search tree depth
M	bound on the length of time needed to explore a subproblem
x^*	optimal solution
$\hat{x}$	feasible solution
μ	measure-of-best function
λ	Lagrange multipliers
μ	measure-of-best function
G	graph
V	set of vertices
E	set of edges
A	set of arcs
$N(i)$	open neighborhood of vertex i
$G[S]$	subgraph induced by S in G
γ_{ij}	number of shortest paths between vertices i and j
P_{ij}	all shortest (geodesic) paths connecting vertices i and j
$d(i,j)$	length of a shortest path between vertices i and j
$\gamma_{ij}(S)$	number of shortest paths between vertices i and j that intersect set S
$diam(G)$	diameter of the graph G

$C^d(S)$	degree centrality of set S
$C^c(S)$	average closeness centrality of set S
$C^b(S)$	betweenness centrality of set S
$\mathcal{C}^b(S)$	monotone betweenness centrality of set S
$\mathcal{C}^c(S)$	monotone average closeness centrality of set S
$\mathcal{C}^d(S)$	monotone degree centrality of set S
S_l	set of structure vertices at tree node l
C_l	set of candidate vertices at tree node l
c_{ij}	cost of arc (i,j)
p_{ij}	success probability of arc (i,j)
p_i	success probability of vertex i
β_i	success arriving probability threshold of vertex i
P^i	arrival probability at vertex i
$p_{\mathscr{R}}$	probability of route $\mathscr{R}$
$LAP(j,i)$	largest arrival probability at vertex i from vertex j
π_j	set of all vertices that must precede vertex j
$\mathscr{P}$	precedence digraph
$\mathscr{R}_t$	ordered set of visited vertices at tree node t
$\mathscr{C}_t$	set of candidate vertices at tree node t
z_t	cost of tree node t
$SDP(i,j)$	smallest departure probability from vertex i such that β_j is satisfied
$MPP(i,j)$	maximum-probability path from i to j

PSP_{ij}	shortest probabilistic path between vertices i and j
a	user-defined scaling parameter
u	loop parameter

CHAPTER 1. INTRODUCTION

In this dissertation, we investigate two famous problems in complex network and transportation. In the first problem, my research aims to identify essential proteins in protein-protein interaction networks using group centrality metrics. The main motivation for this research is finding essential proteins using computational methods which are cheaper and faster than common experimental methods. Extracting essential proteins is of paramount importance because these proteins are indispensable for growth and development of cells and identifying them is important for better understanding the minimal requirements for cell life. The proposed centrality metrics are group degree, betweenness, and closeness which are extensions of well-known nodal metrics. The structures investigated range from the "stricter" induced stars and cliques, to a "looser" definition of a representative structure, which only requires that a central node is adjacent to every other member of the structure. We derive the computational complexity for each of the newly proposed metrics. Then, we provide MILP formulations to solve the problems exactly; due to the computational complexity of the problem and the sheer size of protein-protein interaction networks, using a commercial solver with the formulations is not always a viable option. Hence, we also propose a combinatorial branch-and-bound approach to solve the problems introduced. As all proposed centrality measures here are non-monotone, we introduce monotone centrality measure to their non-monotone counterparts and take advantage of them to develop upper and lower bounds. Finally, we conclude this work with a presentation of the performance of the proposed centrality metrics in identifying essential proteins in Helicobacter Pylori and compare them to their nodal counterparts. We also compare the performance of combinatorial branch-and-bound with MILP formulation for each problem.

The second problem is finding the probabilistic minimum-cost Hamiltonian cycle problem considering arc and vertex failures. post-disaster recovery (PDR) is a key part in a disaster support system. A way to minimize the impact of a disaster on the victims is to ensure that adequate disaster supplies are available and can be delivered to the victims. Indeed, the conventional planning methods rarely account for the uncertainties that come with disasters. A major problem on delivering supplies after a disaster is that many roads may become impassable or passable in highly reduced speed. Thus, in the planning stage of a PDR system, we need to determine the location to store the supplies and design the delivery route considering vertex and/or arc failures which is associated with infrastructure damage. we propose 4 chance constraint-based MILP formulations to model this problem. The chance constrain guarantees that successful visiting each vertex starting from depot is greater than or equal to a given threshold. In the first two formulations we explicitly calculate the probability of reaching each vertex from origin, while in the next two formulations a recursive formula is developed to find those probabilities. As formulations cannot solve real-life problems, we propose a combinatorial branch-and-bound (CBB) to tackle the complexity of these problems. In order to reduce the density of original graph of the problem, we apply preprocessing which in average eliminates about 50% of arcs of original graph. Moreover, we boost performance of the CBB algorithm by proposing 5 feasibility rules to eliminate partial routes that do not result in feasible solutions. Also, we develop a lower bound based on shortest path problem with chance constraints and an upper bound to make the CBB even more efficient. Computational results using two popular benchmark instance sets indicate that CBB algorithm outperforms all formulations in terms of both number of optimally solved instances and the computational time, where CBB can solve instances with up to 231 vertices, whereas the largest instances solved by formulations has 27 vertices.

An interesting future direction for these two research is finding most central structures considering probabilistic vertices and edges. The reason of assigning probabilities to each vertex and edge is that datasets of protein-protein interaction networks are still not error-free, thus these probabilities can indicate the probability of their existence in the protein-protein interaction network.

1.1. Outline

This dissertation contains 14 chapters and is divided into four parts. In the first part, branch-and-bound is explained in detail (Chapter 2). After an introductory part about the first problem

(Chapters 3–5), we address some definitions, complexity analysis, notations, mathematical formulations, and proposed algorithms (Chapters 6–9). In the second part (Chapters 10–12) , we define the second problem and provide mathematical formulations and proposed algorithm to solve it. In the last part (Chapters 13 and 14), computational results and conclusions of both problems are discussed.

Chapter 3 starts with the motivation and application of the first problem and describes the methods available to extract protein-protein interactions with an overview of the available databases. Chapter 4 Provides an overview of all prevalent computational methods in detecting and analyzing protein complexes and functional modules within large-scale PPINs. In Chapter 5, we provide a review of protein centrality and essentiality and collect centrality and non-centrality based methods to predicting protein essentiality. Chapter 6 focuses on basic notation we will be using throughout the first problem, defines the problem, and provides its computational complexity. Then, Chapter 7 presents mathematical programming framework of the first problem. Chapter 8 explains the CBB algorithm for tackling the first problem faster. In Chapter 9, we show how group degree centrality based problems can be solved by column generation. The second problem is introduced in Chapter 10, then we provide the definition and mathematical formulations for this problem in Chapter 11. Chapter 12 explains the proposed CBB algorithm to solve the second problem. Chapter 13 presents our computational study on both problems. Finally, We conclude this research with our observations and our insights in Chapter 14.

CHAPTER 2. BRANCH-AND-BOUND ALGORITHM

2.1. Introduction

The branch-and-bound (B&B) algorithm is a widely-used methodology to find optimal solution of NP-hard optimization problems. This algorithm, which was first proposed by Land et al. (1960), implicitly enumerates all possible solutions to the problem under consideration, by storing partial solutions called subproblems in a tree structure. Each subproblem is represented by a node in the search tree. Branching and bounding are two important mechanisms that search tree works based on them. Branching is generating children for unexplored nodes by partitioning the solution space into smaller regions that can be solved recursively and bounding is pruning off regions of the search space that are provably suboptimal. Once the entire tree has been explored, the best solution found in the search is returned as the optimal solution. An overview of the core B&B algorithm was provided by Lawler et al. (1966); the solution procedure is also covered in the excellent texts by Wolsey et al. (1999), Bertsimas et al. (1997), and Papadimitriou et al. (1998).

According to the B&B framework explained above, there are three components which can have significant impacts on the performance of the algorithm:

(i) Search strategy: the order in which subproblems in the tree are explored,

(ii) Branching strategy: how the solution space is partitioned to produce new subproblems in the search tree, and

(iii) Pruning rules: rules that prevent exploration of suboptimal regions of the search tree.

Clausen (1999) gives an overview of these different components and how they affect algorithm performance for the traveling salesman problem (TSP), the graph partitioning problem (GPP), and the quadratic assignment problem (QAP). Morrison et al. (2016) discussed recent research advances in the design of B&B algorithms, particularly with regards to these three components.

2.2. Algorithm

Let $\mathcal{P} = (X, f)$ be an optimization problem, where search space X is the set of all valid solutions to the problem, and $f : X \to \mathbb{R}$ is an objective function. The goal is to find an optimal solution x^* to the following problem

$$\begin{aligned} &\mathit{minimize} \;\; f(x) \\ &s.t. \quad x \in X \end{aligned} \tag{2.1}$$

B&B algorithm solve 2.1 using an implicit enumeration search tree T of subproblems, where a subproblem S is simply a subset of the search space. The algorithm operates by iteratively selecting an active (or unexplored) subproblem S to explore from a list $\mathscr{U}$ of unexplored subproblems. Additionally, a feasible solution $\hat{x} \in X$, called the incumbent solution is stored globally. When a subproblem S is explored, if it can be proved that any $x \in S$ is no better than $\hat{x}$ ($i.e., \forall x \in \mathrm{S},\ f(x) \geq f(\hat{x})$), then S is immediately pruned (or fathomed). Otherwise, the algorithm tries to find $\bar{x} = \arg\min_{x \in S} f(x)$, called the candidate incumbent, and compare it to the $\hat{x}$. If the B&B algorithm can find $\bar{x}$, then S is terminal, and if $f(\bar{x}) < f(\hat{x})$, the global incumbent solution is replaced by the candidate incumbent. Finally, if S cannot be pruned and is not terminal, child subproblems $S_1, S_2, \ldots, S_r$ are generated from S such that $\bigcup_j S_j \supseteq S$ which ensures that all solutions contained in S are also contained in some child subproblem. $S_1, S_2, \ldots, S_r$ are then inserted into $\mathscr{U}$. Once there are no active subproblem, the best incumbent solution is returned as optimal solution. Pseudocode for the generic B&B procedure is given in Algorithm 1.

Often, an initial incumbent solution $\hat{x}$ can be found (Line 1, Algorithm 1) via a heuristic procedure (see, for example, Malaguti et al. (2011)). Algorithm 1 is guaranteed to terminate when X is finite and the partitioning procedure creates child subproblems S_i that are proper subsets of S at each subproblem S.

Algorithm 1: B&B Algorithm.

```
1  Set $\mathscr{U} = X$ and initialize $\hat{x}$
2  while $\mathscr{U} \neq \emptyset$ do
3      select $S \in \mathscr{U}$
4      if S cannot be pruned then
5          if S can be solved to find $\bar{x}$ then
6              if $f(\bar{x}) < f(\hat{x})$ then
7                  set $\hat{x} = \bar{x}$
8          else
9              Partition $S$ into $S_1, S_2, \ldots, S_r$
10             insert $S_1, S_2, \ldots, S_r$ into $\mathscr{U}$
11     remove $S$ from $\mathscr{U}$
12 return $\hat{x}$
```

The complexity of B&B algorithms is dependent to two factors: the branching factor **b**, which is the maximum number of children generated at any node in the search tree, and the search depth **d**, which is the length of the longest path from the root of T to a leaf. Hence, any B&B algorithm operates in $O(Mb^d)$ worst-case running time, where **M** is a bound on the length of time needed to explore a subproblem; however, the presence of pruning rules can substantially improve the algorithm performance.

Any B&B algorithm has two important phases: (1) search phase, a phase during which the algorithm has not yet found an optimal solution x^*. (2) verification phase, which starts when the incumbent solution is optimal, but there are still unexplored subproblems in the tree that cannot be pruned. Note that an incumbent solution cannot be proven optimal when unexplored subproblems remain; also note that the delineation between the search phase and the verification phase is not known until the algorithm terminates. A problem $\mathcal{P}$ is said to be solved if the B&B algorithm completes the verification phase. In this case, the algorithm is said to have produced a *certificate of optimality*.

The three components of B&B algorithm (search strategy, branching strategy, and pruning rules) each play a distinct role with respect to these two phases of operation. The first B&B algorithm component, search strategy, primarily impacts the search phase. For example, suppose that the pruning rules only depend on the value of the incumbent solution, e.g., a subproblem of a minimization (maximization) problem is pruned when it's lower (upper) bound is bigger (smaller) than the incumbent value. In this setting, any search strategy must explore the same set of subproblems once an optimal solution is found. The second B&B component, the branching strategy, has significant impacts on both the search phase and the verification phase. By branching appropriately at subproblems, the strategy can guide the algorithm towards optimal solutions. Once the search phase has concluded, an appropriate branching strategy can help limit the branching decisions that are made in order to prevent unnecessary work from being performed to produce a certificate of optimality. Pruning rules as the third component of B&B algorithm, often affect verification phase, particularly in the case of objective-based bounding which can be relatively weak before an optimal (or near-optimal) solution is known. In this case, if the incumbent solution has a poor objective value early in the search process the lower bounds will not be able to prune effectively, even if they are very tight. However, there are also situations in which pruning rules contribute to the search phase, such as when cutting planes in a mixed integer program (MIP) are used to identify feasible solutions.

The reason to improve performance of the B&B algorithm during the search phase is twofold. First, if the algorithm terminates before proof of optimality, the incumbent solution still can be used as a heuristic solution. An example of this behavior can be seen in Morrison et al., 2014b where B&B is used to improve upper bounds for large simple assembly line balancing instances. The second reason is that finding an optimal solution earlier in the search phase has a direct impact on the size of the search tree, because the nodes with bounds greater than the optimum value will not be explored (Guzelsoy et al., 2013).

Integer linear programming (ILP) problems form an important class of optimization problems that are often solved using B&B algorithms. An ILP problem seeks a solution to $min\{f(y) \mid Ay \leq b,\ y \in \mathbb{Z}\}$, where f is a linear objective function, A is a constraint matrix, and b is a set of constraint bounds, and y is a vector of integer variables. In B&B algorithm to solve ILP problems, bounds at a subproblem are typically computed by solving the LP relaxation, where the integrality constraints on y are relaxed. If all variables in the LP relaxation solution are integral, then the current subproblem yields a new candidate incumbent; otherwise, the standard branching rule used for integer programs selects a variable y_i with fractional value α in the LP relaxation solution. Two branches are created, one with $y_i \leq \lfloor \alpha \rfloor$, and one with $y_i \geq \lceil \alpha \rceil$.

Many different optimization problems can be formulated as integer programs, and the LP relaxation often provides tight bounds in practice. To solve integer programs using B&B techniques, some efficient software packages (both commercial and freeware) have been developed, such as CPLEX, SYMPHONY, Gurobi, LINDO, SCIP, Xpress-MP, and CBC.

2.3. B&B components

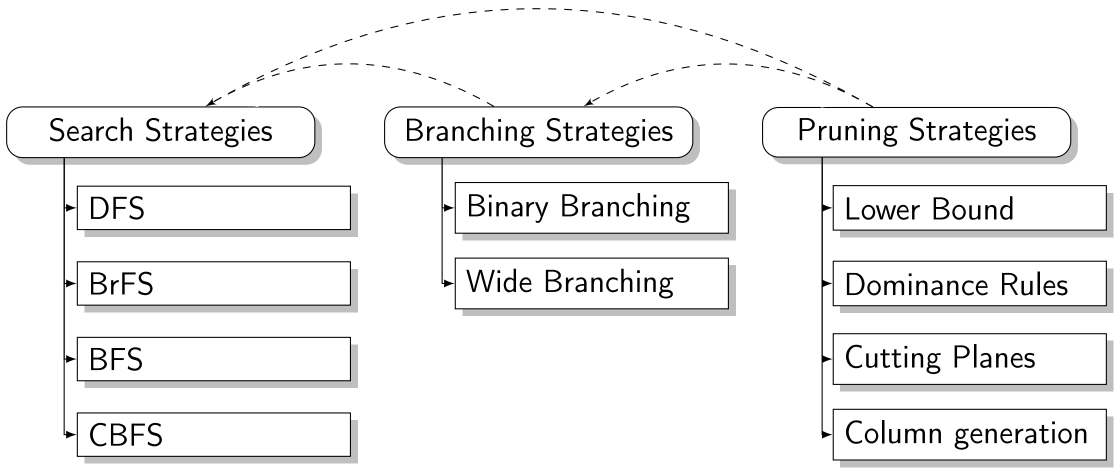


Figure 2.1: components of the B&B algorithm and their relationship.

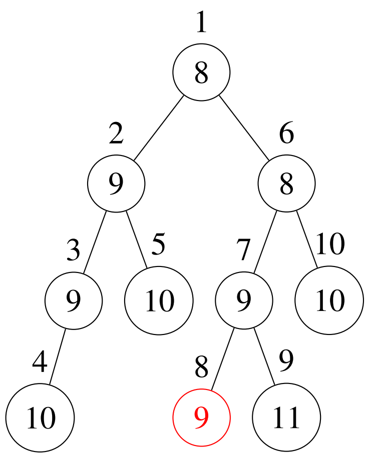
Figure 2.1 shows the different types of search, branching, and pruning strategies. The Cyclic Best-First Search (CBFS) strategy is a generalization of Depth-First Search (DFS), Breadth-First Search (BrFS), and Best-First Search (BFS) (see Section 2.4.4). Column generation techniques, while are not strictly a generalization of other techniques, are closely connected to lower bounding and cutting plane techniques (in essence, column generation adds cutting planes to the dual optimization problem to improve the computed lower bound). This figure also indicates relationships between B&B components using dashed lines. In particular, the choice of pruning rules often impacts or limits the choices that can be made in the other two areas. For example, as will discuss in Section 2.6.4, if column generation is used to improve lower bounds, the choice of branching strategies that can be used is limited. Moreover, if dominance relations are used, this may cause BrFS to become a desirable search strategy, since it has the property of never exploring a dominated subproblem. Finally, the choice of branching strategy can itself impact the choice of search strategy. For instance, if the branching strategy chosen produces a particularly unbalanced tree, the CBFS strategy can balance the search process, or variants of DFS can limit the depth explored at any stage in the algorithm.

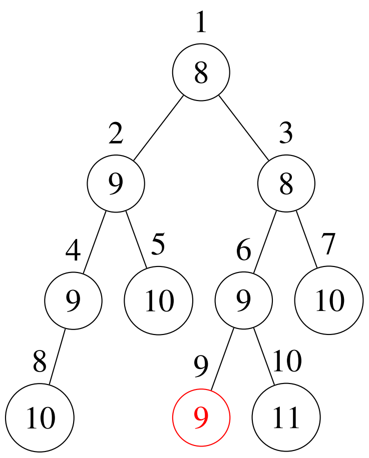
2.4. Search strategies

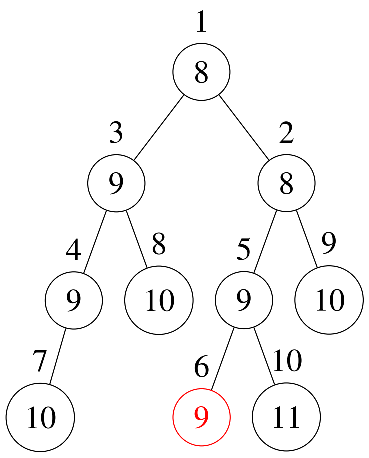
The search strategy in a B&B algorithm determines the order in which unexplored subproblems in $\mathscr{U}$ are explored. Search strategy influences the choice of subproblem to explore in Line 3 of Algorithm 1. The search strategy significantly impact computation time required for the B&B procedure, as well as the amount of memory used. In some cases, for very large or challenging problems, it may be necessary to choose a search strategy that requires low memory usage, however for problems in which memory is not a concern, other search strategies exist which may find an optimal solution very quickly, and thus explore potentially fewer subproblems. Ibaraki (1976) reviewed different search strategies. In this section, we explain different search strategies along with their strengths and weaknesses are given.

2.4.1. Depth-first search

DFS strategy (sometimes called depth-first search with backtracking, or last-in, first-out search) involves exhaustive search of all the nodes by going forward if possible or backtracking otherwise. Here, the word backtrack means that when you are moving forward and there are no more nodes along the current path, you move backwards on the same path to find nodes to traverse. This search strategy used in many different graph algorithms in addition to B&B (Golomb et al., 1965; Tarjan, 1972). This strategy uses a stack to explore subproblems in $\mathscr{U}$ in the reverse order that they are generated (Tarjan, 1972). The algorithm removes the top item from the stack as the next subproblem to explore, and push all its children on the top of $\mathscr{U}$. Thus, the next subproblem which is explored is the most recently generated subproblem. However, if the children of a subproblem can be ordered using a criteria, then DFS does not need to store all of them


(a) DFS


(b) BrFS


(c) BFS

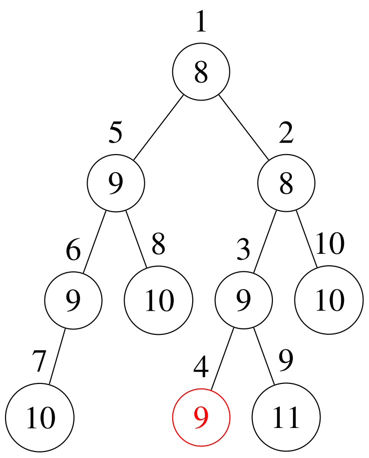

(d) CBFS

Figure 2.2: Subproblem exploration order for different search strategies. The red subproblem is optimal, numbers inside nodes are subproblem lower bounds, and numbers outside the nodes indicate exploration order. The algorithm starts with an incumbent solution of value 10. BFS and CBFS use the lower bound as the measure-of-best, with ties broken arbitrarily. Note that this requires a subproblem's lower bound is computed prior to inserting it into the list of unexplored subproblems.

(which can occupy a large memory by growing the search tree) during implementation of the algorithm. Instead, the search strategy at each subproblem only stores the path from the root of T to the current subproblem and the index of the last-explored child subproblem. Thus, at the current subproblem, the next unexplored child is selected for exploration. If no unexplored children remain, the algorithm backtracks to the closest ancestor node with unexplored children.

The next advantage of DFS is its ability to produce complete solutions early in the search process, because complete solutions are usually at larger depths. Another advantage of DFS arises when it is used to solve integer programming problems with the LP relaxations as lower bounds. In particular, when exploring a child subproblem immediately after processing its parent, it is often possible to reuse information from the parent's LP relaxation solution to speed up solving the child's LP relaxation. Reusing the optimal basis and its LU factorization when solving the child LP is known as *hot starting*, while reusing only the optimal basis is known as *warm starting* (Atamtürk et al., 2005; Chinneck, 2006). Other search strategies would need to store the appropriate information in memory in order to take advantage of these techniques. Since storing factorization is memory-intensive, hot starting is typically not used with other search strategies.

The DFS strategy has two disadvantages. The first problem is that DFS is often hindered by an inability to escape from poor regions of the search space and explore more promising areas that might enable better pruning. A related phenomenon, called *thrashing*, occurs when different regions of the search space all fail for the same or similar reasons (Kumar, 1992). The second problem occurs when search tree is extremely unbalanced. In other words, if some optimal solutions are close to the root, but there exist long paths in T that do not lead to an optimal solution, DFS can (unluckily) choose many long, bad paths before it explores a path leading to an optimal solution. However, this computation time often could be avoided via pruning rules if the search strategy instead chose to explore a short optimal path first. In fact, this behavior of DFS was first noticed on problems where the search tree had unbounded depth (Slate et al., 1983), but the same problem exists in trees with a few extremely long paths.

Some variants of the DFS strategy has been developed to overcome these limitations. One common variant is the iterative deepening DFS algorithm (Korf, 1985), which place a limit on the depth of any path explored by DFS; if the search process is not able to prove optimality using this depth limit, the depth is increased and the search is restarted from the root. This variant can decrease the amount of time DFS spends in poor regions of the search space, but requires repeated exploration of previously generated subproblems. Meseguer (1997) proposed another algorithm called interleaved DFS to overcome thrashing by performing depth-first search from multiple locations in the search tree simultaneously. This strategy can be performed by sequentially selecting exactly one subproblem to explore from each different DFS path in the search tree before returning to the first search path. DFS with complete branching is another variant of DFS which selects the next child subproblem to explore as the one with the best computed lower bound. Although this method explores the search tree more intelligently, it needs more memory space, because all child subproblems of a subproblem must be generated. However, if the tree has a relatively small branching factor, this increased memory usage is not likely to be significant.

2.4.2. Breadth-first search

BrFS explores all subproblems at the present depth prior to moving on to the subproblems at the next depth level. This stratrgy is implemented in first-in, first-out or queue manner. The advantage of BrFS strategy is finding an optimal solution that is close to the root of the tree, thus operating well on unbalanced search trees. However, BrFS is usually not used in B&B algorithms because it does not generate any complete solutions until late in the search process, which limits pruning and results in extremely high memory usage. There are two exceptions for using BrFS as search strategy in B&B. If dominance rules (which identify pairs of subproblems S_1 and S_2 that do not both need to be explored to find an optimal solution) are employed or a good (heuristic) initial solution is available (Sewell et al., 2012b).

2.4.3. Best-first search

BFS is a search strategy which explores $\mathscr{U}$ by expanding the most promising subproblem chosen according to a specified rule. This strategy estimating the promise of every unexplored subproblem by a global *"measure-of-best function"* $\mu : 2^X \to \mathbb{R}$, which attempts to predict the quality of any subproblem, and selects the next subproblem to explore as the unexplored subproblem with the best value of μ. If the measure-of-best function μ assigns to each subproblem a value that is at least as good as the value of the best complete solution that can be obtained from that subproblem (i.e., μ provides valid lower bounds when minimizing or valid upper bounds when maximizing), then BFS is sometimes called the A^* algorithm (Dechter et al., 1985); such a measure-of-best function is called admissible. When sufficient memory is available to store the entire unexplored search tree, BFS strategy is often used. BFS can easily be implemented by storing the list of subproblems in a heap data structure, using the value of μ as the key (Cormen et al., 2009).

One of the most common choices for the heuristic evaluation function for minimization (maximization) problem is a lower bound (upper bound) on the value of the best solution of the subproblem. If the lower bound is strongly correlated with the subproblem objective value, this heuristic evaluation function will encourage exploration of subproblems with better solutions. However, one difficulty with this approach is that subproblems with promising LP bounds may not actually contain good integer solutions. To remedy this, the measure-of-best function μ can use an estimate of the best integer solution that exists at a given subproblem instead of the LP bound. Such a search strategy is known as best estimate search (Linderoth et al., 1999; Achterberg, 2007).

Unlike DFS, BFS can move to different regions of the search space if the current region appears less promising with respect to μ. Under certain assumptions (the absence of ties in the measure-of-best function), BFS can be shown to explore the smallest number of subproblems of any search strategy (Dechter et al., 1985). However, the first drawback of BFS is high memory usage, since it must store all unexplored subproblems in memory at all times. The other disadvantage of BFS is delays in finding complete solutions (Sewell et al., 2012b). If there exist many subproblems in $\mathscr{U}$ for which $\mu(S) = f(x^*)$, depending on the tie-breaking rule used, BFS may spend much time in middle regions of the search tree and never explore an optimal solution (see Figure 2.2c, in which nodes 3, 4, and 5 are explored before the optimal solution is found). In this condition, BFS may be slower than DFS. To overcome these issues, usually a hybrid search strategy is developed which is a combination of BFS and DFS so as to exploit the benefits of both. In its most basic form, after exploring a subproblem, this hybrid search strategy will select one of its children to explore next, as in DFS. This DFS-like behavior continues in a process called diving , until a leaf subproblem of the search tree is generated. At this point, the measure-of-best function is queried to select a new subproblem to explore, as in BFS. The diving process then starts from the chosen subproblem. This algorithm is called BFS with diving .

2.4.4. Cyclic best-first search

The CBFS strategy, originally called distributed best-first search (Kao et al., 2009), can be viewed as a hybrid algorithm between DFS and BFS. Intuitively, DFS spends much time in intensification, due to the backtracking nature of the algorithm which leads to thrashing as explained before. In contrast, the BFS strategy potentially performs more diversification by using a measure-of-best heuristic function. The CBFS strategy attempts to diversify the search process more than BFS while retaining some of the intensification properties of DFS. Moreover, CBFS attempts to find good incumbent solutions early in the search process, as this will improve the performance of B&B both as a heuristic mechanism, and will usually enable better pruning by the solver in later iterations.

While BFS is implemented by using a single heap data structure to store all unexplored subproblems, CBFS divides the unexplored subproblems over a collection of heaps, referred to as *contours*. When a new subproblem is identified, it is inserted into one of the contours according to a set of rules (e.g., all subproblems at depth d get stored in a contour associated with depth d). To explore the search space, the CBFS strategy repeatedly iterates through all of the non-empty contours, selecting the best subproblem (according to a measure-of-best function μ) from each contour before moving on to the next one. For example, if contours are organized by depth, CBFS will explore the best subproblem at depth 0, then depth 1, and so on; upon reaching the bottom of the search tree, it will repeat the process starting from depth 0.

By separating subproblems into contours, the measure-of-best function effectively assigns to each subproblem a local ranking within a single contour instead of a global ranking. Thus, contours can be used to group subproblems that are more directly comparable. Cycling ensures that each contour is visited frequently, which serves to diversify the search and can aid in generating incumbent solutions (e.g., depth-based contours ensure that a leaf of the search tree is explored every cycle). See Morrison et al. (2017) for more detail about CBFS.

2.5. Branching strategies

Branching strategies influence how children are generated from a subproblem in Line 9 of Algorithm 1. Branching strategies can be categorized into two groups: binary branching strategies and non-binary, or wide, branching strategies.

2.5.1. Binary branching

Binary branching strategies focus on partitioning a subproblem S into two mutually-exclusive, smaller subproblems. One example for this branching strategy is the knapsack problem, where one seeks a maximum-value selection of items to fit inside a storage bin with fixed capacity. Subproblem S for an unassigned item i have two children; one in which the item i is included in the knapsack, and one in which the item i is excluded from the knapsack (Kolesar, 1967). The integer branching scheme for integer programming, described in Section 2.2, is another binary branching example.

In some cases, the mechanism for performing the partitioning is more complicated. In the graph coloring branch-and-price solver of Mehrotra et al. (1996), branching is performed by either adding edges or contracting vertices of the graph to force a pair of non-adjacent vertices to either share the same color or use different colors. Similarly, for the branch-and-price solver for the generalized assignment problem (Savelsbergh, 1997), in which a series of tasks is assigned to a group of workers to maximize profit, branching is performed by either including or excluding all schedules that assign a particular task to a worker.

2.5.2. Non-binary branching

Non-binary branching is used when more than two children can be generated for a subproblem. For example, in B&B algorithms to find maximum cliques or independent sets in a graph, a set of unused vertices is maintained for each subproblem, and each unused vertex generates a child with that vertex added to the child's set (Morrison et al., 2014a). This branching method is also known as wide branching (Babel, 1994; Held et al., 2012). Wide branching methods can reduce the size of the search tree substantially. For example, if we use binary branching strategy for maximum clique problem, each unused vertex should be considered individually which creates a long sequence of subproblems.

Special ordered sets (SOS) (Beale et al., 1970; Beale et al., 1976), is one of the important applications of wide branching strategy. There are two types of special ordered sets, denoted by SOS1 and SOS2. A SOS1 is defined to be a set of variables for which not more than one member from the set may be non-zero in a feasible solution, and a SOS2 is a set of consecutive variables in which not more than two adjacent members may be non-zero in a feasible solution. Wide branching strategies can be used in B&B to handle problems with SOS variables; for example, when an SOS1 is selected for branching, the strategy creates one branch for each element in the set. The subproblem for this set uses the chosen element and excludes all others. Finally, the branching strategy creates one subproblem which uses no elements from the set. This strategy can be generalized to handle SOS2, as well.

There are two potential problems about wide branching strategies. The first is that such strategies usually do not create mutually-exclusive branches, so it is possible to arrive at the same subproblem from several different paths. This problem can be resolved using a lexicographic ordering rule (Geoffrion, 1969) or dominance rules (Section 2.6.2). The second problem arises if the number of possible branches for a particular subproblem is very large. In this case, the algorithm could get stuck generating children at a particular subproblem and never move on to explore new regions of the search space. In addition, if the number of generated children is very large at every subproblem, the size of the search tree will grow much more quickly. There are two approaches to tackle this problem; the first one sets an arbitrary cap on the number of children that can be generated at a subproblem. If the branching factor ever exceeds this limit, any additional children are just discarded. This technique, used in the simple assembly line balancing solver of Sewell et al. (2012b), can prove optimality if the branching factor is never exceeded, otherwise it performs as a heuristic. The other method uses a delayed branching technique, where after a certain number of children have been generated, the strategy delays generation of the remaining children in the hopes that when it returns to the node, better bounds may have been computed that allow it to prune children more effectively. This method was used in a branch-and-price solver for graph coloring by Morrison et al. (2014a).

2.6. Pruning strategies

Pruning rules used to exclude regions of the search space from exploration (line 4 of Algorithm 1). Pruning rules can greatly reduce the size of the search three in both search and verification phases. Note that all active nodes that cannot be pruned by the pruning rules must be explored no matter which search strategy is used, even if an optimal solution is known before the search begins. There are many different classes of pruning rules, but they are usually problem-specific.

2.6.1. Lower bounds

In a minimization problem, lower bound on the objective function value at each subproblem is the most common pruning rule. In this strategy, subproblems with lower bound not better than the incumbent solution are pruned. In integer programming, the optimal value of the LP relaxation is the most common lower bound choice. The quality of the LP relaxation value is measured by the integrality gap of the formulation, that is, the ratio between the best integer solution and the best LP relaxation value. Since, there may be many different MILP formulations for a problem, some of these formulations may have better LP relaxations than others (higher optimal value of the LP relaxation is better and provide tighter lower bound). Thus, one technique for improving lower bounds is to derive a new formulation with a tighter integrality gap (Arora et al., 2002). For instance, Öncan et al. (2009) reported 24 asymmetric traveling salesman problem (ATSP) formulations and compared the strength of their LP relaxations and proposed a graph to show the relationships between all LP relaxations. The branch-and-cut (Sections 2.6.3) and branch-and-price algorithms (Sections 2.6.4) are common methods for exploiting integer programming formulations with tighter bounds. Reformulation-linearization technique (RLT) transforms a mathematical program with polynomial objective function and constraints into a linear program, and uses the resulting LP bound to prune in B&B algorithm to find global optimal solutions to the polynomial program (Sherali et al., 1992).

Lower bounds can be computed by relaxing various aspects of the MILP problems other than integrality constraints. **Combinatorial B&B (CBB)** algorithm is a B&B algorithm in which lower bound is computed using combinatorial methods other than LP relaxation of the original MILP problem. For example, in traveling salesman problem and its variants, in which customers must be visited by a single tour so that each customer must be visited exactly once, one common relaxation is to drop subtour elimination constraints, called AP relaxation (Carpaneto et al., 1980). In another research, Rysz et al. (2018) proposed a CBB algorithm to find clique clusters with the highest betweenness centrality in a graph. They showed that an optimal solution to this problem is either a maximal clique, or contained in a maximal clique with the same objective value and propose an analytical upper bound for the betweenness centrality of any maximal clique containing a given clique, and employed it to develop a CBB algorithm for solving this problem.

The general procedure is to attempt to prune using the easy lower bounds first, and then move on to the more complex, but tighter, lower bounds if the easy methods are unsuccessful. Another method for deriving lower bounds on integer programming problems is through duality. Though there is no strong duality theorem for integer programming, one can still arrive at a notion of weak duality. Given an integer program $min\{f(y) \mid Ay \leq b, y \in \mathbb{Z}\}$, the Lagrangian relaxation problem is $\mathcal{P}(\lambda) = min\{f(y) + \lambda(b - Ay) \mid y \in \mathbb{Z}\}$, where λ is a non-positive vector of real-valued weights called Lagrange multipliers. The optimal solution value for the Lagrangian relaxation is always bounded above by the value of the optimal solution to the original problem. Thus, the best bound possible may be computed as the solution to the Lagrangian dual problem, $\max_{\lambda \leq 0} \mathcal{P}(\lambda)$. The Lagrangian dual problem can be solved using subgradient optimization, a modification of

Newton's method for piecewise linear concave functions (Bertsimas et al., 1997). Integer programming duality methods have been used in Vila et al. (2014), Desrosiers et al. (2013), Gendron et al. (2016), and Phan (2012).

2.6.2. Dominance rules

Dominance rules prune a subproblem if it is dominated by another subproblem, in other words if subproblem S_1 dominates subproblem S_2, this means that for any solution which is a descendant of S_2, there exists an equal or better solution descending from S_1. Thus, we do not need to explore S_2 and it can be pruned. Note that, as shown by Ibaraki (1977), it is not always true that using dominance relations will improve the quality of the search process; however, there are many cases in which dominance relations will improve the search.

There are two primary types of dominance rules, memory-based and non-memory-based. Memory-based dominance rules to dominate an unexplored subproblem compare it to the explored problems that have been stored previously in the search tree (Sewell et al., 2012a). Thus, they require all explored subproblems to be stored during running algorithm. However, this may allow for additional pruning to be performed that would be otherwise impossible. On the other hand, non-memory-based dominance rules do not require the dominating state to have been previously generated in the search process, because they are able to imply the existence of a dominating subproblem, regardless of whether it has been explored or not. The advantage of these rules is that they do not require additional memory to store the generated search tree, but they may not be able to prune the same subset of problems that memory-based dominance rules can. Fischetti et al. (2010) developed non-memory-based dominance rules in a B&B solver for generic mixed integer programming problems. By solving an auxiliary problem, their solver was able to identify whether a node in the tree is dominated by some other nodes (which have not been explored yet).

Dominance rules play an important role in B&B algorithms for solving MILPs with a high degree of symmetry. In such problems, a particular solution may have many equivalent representations that appear as separate subproblems within the search tree (e.g., in finding the most degree-central clique in a graph, specific vertices forming a structure can be permuted without changing the solution). If the B&B algorithm fails to recognize this symmetry, it may end up performing redundant work that can impact the performance of the algorithm greatly. To address this difficulty, Margot (2002) and Margot (2003) introduced isomorphic pruning, which uses lexicographic inequality tests at each node in the search tree to determine if the node can be pruned. Ostrowski et al. (2011) proposed orbital branching, which identifies equivalent variables at a particular node in the search tree using the group-theoretic concept of an orbit.

2.6.3. Cutting planes

Cutting planes for the first time were proposed by Gomory (2010) as a method for solving integer programming and mixed-integer programming problems. A cutting plane is a constraint that can be added to an integer program to tighten the feasible region without removing any integer solutions. This fundamental idea was applied to B&B by Padberg et al. (1991) to develop an algorithm called branch-and-cut. In this algorithm, new cutting planes (sometimes called valid inequalities) are added to the LP relaxation at every subproblem in the search tree (note that a valid inequality is a global constraint-it must apply at the LP relaxation of the root subproblem). The algorithm of Padberg et al. (1991) was specific to the well-known traveling salesman problem, but Balas et al. (1996b) proposed a generalization of branch-and-cut for binary integer programs. Different types of cuts for general mixed integer programs were proposed by researchers (see Cornuéjols (2008) and Marchand et al. (2002)). *Gomory cuts* are very efficiently generated from a simplex tableau, whereas many other types of cuts are either expensive or even NP-hard to separate. These cuts were shown to be of both practical and theoretical interest by Balas et al. (1996a). Chvatal (1973) considered these cutting planes from a geometric perspective and showed that a finite number of such cuts can be added to a pure integer program to yield the convex hull of the integer feasible solutions. Cook et al. (1990) introduced split cuts, while Nemhauser et al. (1990) introduced mixed integer rounding (MIR) cuts; the two approaches are equivalent (Nemhauser et al., 1990). Lovász et al. (1991) and Balas et al. (1993) introduced lift-and-project cuts for 0-1 mixed integer programming. Lift-and-project operates by lifting the LP relaxation into a higher-dimensional space by adding additional variables, finding valid inequalities in this higher-dimensional space, and then projecting the valid inequalities back into the original space by deleting the extra variables. Balas et al. (2003) demonstrated the relationship between lift-and-project cuts, intersection cuts (Balas, 1971), and Gomory mixed integer cuts.

Cutting planes can also be generated by exploiting local structure in a mixed integer program. One well-known set of inequalities are *cover inequalities*, which are derived from viewing individual constraints in the MIP as separate knapsack problems (Marchand, 1998; Atamtürk, 2005). Other examples of structural cuts include flow cover inequalities (Padberg et al., 1985) and weight inequalities (Martin et al., 1997). Marchand et al. (2001) showed that many cuts based on the structural properties of a MIP can be derived as MIR cuts using only the initial set of constraints in the MIP formulation. These results were utilized by Marchand et al. (2001) in a heuristic procedure for generating cuts for arbitrary mixed integer programs.

Cutting planes for MIP have led to significant improvements in solver performance over the recent years. Crowder et al. (1983) demonstrated how cover inequalities could be used to solve large, sparse 0-1 integer programs in a reasonable amount of computing time. Balas et al. (1996a) showed the effectiveness of Gomory mixed integer cuts in a branch-and-cut framework for 0-1 mixed integer programs; as a result, Gomory cuts are included in all modern MIP solvers today. Computational results by Balas et al. (2008) have demonstrated the effectiveness of split cuts in reducing the integrality gap through experiments with the split closure on MIPLIB instances; separating such cuts in practice remains computationally difficult but progress is being made (Fischetti et al., 2013). A computational study of the various cutting plane methods available in CPLEX 12.5 can be found in Achterberg et al. (2013). The results from those experiments indicated that MIR cuts were the most effective, followed by Gomory cuts.

Another method of generating valid inequalities is through decomposition methods such as Benders' decomposition (Flippo et al., 1993). Benders decomposition is a strategy for solving large-scale optimization problems (BnnoBRs, 1962; Geoffrion, 1972). The variables of the problem are partitioned into two sets: *master problem* variables and *subproblem* (this subproblem is different from subproblem in the search tree) variables. The Benders algorithm iteratively solves a master problem, which assigns tentative values for the master problem variables, and a subproblem, obtained by fixing the master problem variables to the tentative values. In every iteration, the subproblem solution provides certain information on the assignment of master problem variables. Such information is expressed as a Benders cut, cutting off some assignments that are not acceptable. The Benders cut is then added to the master problem, narrowing down the search space of master problem variables and eventually leading to optimality. On one hand, Benders method is employed to exploit the problem structure: the problem is decomposed into a series of independent smaller subproblems, reducing the complexity of solving it (Lasdon, 2002). On the other hand, Benders method opens a dimension for "hybrid algorithms" (Wallace et al., 2002; Eremin et al., 2001) where the master problem and the subproblems can be solved with different methods.

The generation of Benders cuts is the core of Benders decomposition algorithm. Indeed, valid Benders cuts guarantee the convergence of the iterations to the optimal solution of the original problem, and also the cuts determine how fast the algorithm converges. The classic Benders decomposition algorithm (Benders, 1962) was proposed for linear programming problems, the cut generation of which is based on the strong duality property of linear programming (Lasdon, 2002). Geoffrion has extended it to a larger class of mathematical programming problems (Geoffrion, 1972).

For integer programming, however, it is difficult to generate valid integer Benders cut, due to the duality gap of integer programming in subproblems. One possible way is to use the *no-good* cut to exclude only the current tentative assignment of master problem variables that is unacceptable. Such no-good Benders cuts will result in an enumerative search and thus a slow convergence. For some specific problems, better Benders cuts can be obtained (Jain et al., 2001). For example, in the machine scheduling application, the cut that limits the incompatible jobs in the same machine is generally stronger. For more general integer programming, logic-based Benders decomposition (Hooker et al., 2003) was proposed to generate valid integer Benders cuts, but these cuts contain a large number of disjunctions (Eremin, 2004), the linearization of which leads to huge cuts with many auxiliary variables, complicating the master problem significantly. Hernández-Pérez et al. (2004) gave an example of using Benders' cuts in a branch-and-cut context to solve the traveling salesman problem with both pickups and deliveries.

One interesting question with regards to this method of pruning involves the interplay between cutting plane generation and branching. In many cases, the set of generated cuts is too large to allow all of them to be added, and it is often computationally expensive to generate new cuts, so at some point cutting planes are no longer generated and branching occurs. However, the question of when to stop generating cutting planes and start branching is an important problem when implementing a branch-and-cut algorithm (Jiinger et al., 1995; Mitchell, 2002).

2.6.4. Column generation

Branch-and-price is one of the leading solution procedures for many large-scale integer programming models (e.g., see Ropke et al., 2009; Barnhart et al., 2000; Belov et al., 2006; Hwang et al., 2008). Such a method is a branch-and-bound algorithm where the lower bounds are computed by a column generation algorithm. Column generation is an iterative procedure that is use for solving linear relaxation of the master problem which is called the linear master problem (LMP) in this context. For an overview of column generation, the reader is referred to Desaulniers et al. (2006) and Lübbecke et al. (2005). The optimal solution value of the LMP is a lower bound of its associated branch-and-bound node. To perform column generation, we use Dantzig-Wolfe decomposition (Dantzig et al., 1960) to reformulate the MILP as a set-partitioning linear master problem (LMP) that typically involves a very large number of variables. In fact, the master problem contains one variable for each extreme point and each extreme ray of the feasible domain of a so-called subproblem. This domain is defined by a subset of the constraints of the original MILP. The column generation procedure cannot directly solve the LMP because of its inability of enumerating all variables. Instead, it is an iterative procedure that alternates between solving a restricted linear relaxation of the master problem (RLMP) and a pricing subproblem. The RLMP is the LMP restricted to a subset of all variables, which can be optimally solved by the simplex algorithm. The goal of solving the pricing subproblem is to identify the columns that have negative reduced costs with respect to the dual optimal solution of the current RLMP. If no such column is found, the column generation procedure is terminated with an optimal solution to the current RLMP, which is also an optimal solution to the LMP. Otherwise, we introduce one or more columns with negative reduced costs into the current RLMP and start another column generation iteration.

Moreover, additional complexity is added when attempting to incorporate column generation with a branching strategy, because typical branching rules usually interfere with the structure of the pricing problem. In other words, once some branching decisions have been fixed, new negative-cost variables must be found that respect the branching decisions. This is often related to the kth-shortest-path problem, which is NP-hard (Hartmanis, 1982). However, despite the apparent difficulty of incorporating column generation with a branching algorithm, the Dantzig–Wolfe decomposition procedure can often substantially tighten the lower bounds produced in the tree, while at the same time reducing problem symmetry leading to thrashing behavior, and thus branch-and-price has been empirically shown to produce substantial computational gains in practice.

To avoid interfering with the pricing problem structure, most branch-and-price algorithms use alternative branching strategies that do not disrupt the structure of the pricing problem. The branching strategy for graph coloring (Mehrotra et al., 1996) or for the generalized assignment problem (Savelsbergh, 1997) are two such examples. Another general-purpose branching strategy for branch-and-price algorithms involves branching on the original (non-decomposed) problem variables (Vanderbeck, 2011). Morrison et al. (2014a) branched on the master problem variables directly, but used a wide branching strategy to limit the number of branching decisions that interfere with the pricing problem structure.

Branch-and-cut-and-price algorithm is a variant of Branch-and-price, where the linear relaxations are tightened by adding cutting planes. In branch-and-cut-and-price algorithm at each branch-and-bound node, we first optimally solve the LMP using the column generation procedure to obtain a lower bound. For the node that cannot be pruned, we next try to identify some inequalities (cuts) that are violated by the current linear solution. If such violated inequalities are found, we add them into the model and invoke the column generation procedure again to further improve the lower bound. The above procedure is repeated until either the node is pruned or no violated inequality can be found. These methods suffer from many of the same problems as branch-and-price algorithms, since now the pricing problem must respect both the branching decisions and the additional cutting planes added to the problem. However, Aragao et al. (2003) developed a new method called robust branch-and-cut-and-price which further reformulates the master problem to eliminate the interference of cuts and branching decisions with the pricing problem. This method has been used with success in a number of vehicle routing and other graph problems (Fukasawa et al., 2006; Uchoa et al., 2008).

CHAPTER 3. PROTEIN-PROTEIN INTERACTION NETWORKS

3.1. Introduction

Systems biology, or the computational and mathematical study of complex biological systems, has been an emerging field of modern computer science and engineering. One of its main objectives is to reveal fundamental, architectural principles that provide us with better insight into the underlying structure and function of all cells and microorganisms, while simultaneously mapping them onto the phenotype of an organism, often through the use of network science (Aittokallio et al., 2006). In biological systems, genes, proteins, RNA, and metabolites are referred to as biological molecules. These molecules, or entities, interact with one another forming integrated, highly complex networks, comprising cells and, consequently, organs and biological systems.

Protein-protein interaction networks (PPINs), gene regulatory networks (GRNs), and metabolic networks (MNs) are three of the most well-known types of biomolecular networks, and are commonly used to investigate biological, cellular systems (Junker et al., 2011). Genome-wide GRNs organize cell proliferation or differentiation into specific cell types, and regulate the transitions between distinct cell-fates in multi-cellular organisms. GRNs consist of thousands of genes that regulate one another (i.e., regulate gene expression) within a complex network (Huang et al., 2005). MNs, on the other hand, are fully connected biochemical networks in which all processes generating mass energy, information transfer, and specifying cell-fates are integrated within a complex network of cellular constituents and biochemical reactions (Jeong et al., 2000).

Proteins are traditionally identified based on their individual role as catalysts, signaling molecules, or building blocks in microorganisms; however, they also serve as computational molecules, necessary for the exchange and transfer of information (Bray, 1995). According to the post-genomic view, they are also responsible for a vast number of cellular processes and functions based on their roles in a network of all protein-protein interactions (Jeong et al., 2001). In such, usually dense, PPINs, proteins interact to form protein complexes and functional modules, which in turn contribute in various cellular processes. Hence, the ability to identify and characterize protein-protein interactions, as well as their organizing principles through networks, is considered fundamental for understanding the mechanisms of biological processes at the molecular level. Per Shoemaker et al. (2007), protein interactions can be classified based on strength, specificity, and location of interacting partners.

Another striking property of PPINs is their modular organization, which represents a functional association between their components. Proteins are organized in two types of cellular modules, namely protein complexes and functional modules (Han et al., 2004b). With the term protein complex we refer to a group of proteins that interacts simultaneously and at the same location, forming a physical object. As an example, consider protein transport and export complexes, or transcription factor complexes. On the other hand, functional modules are groups of proteins taking part in the same cellular process, even when interactions might happen in different times and/or places. Instances of such modules are the CDK/cyclin module (responsible for the cell-cycle progression), the yeast pheromone response pathway, etc. As proteins are fundamental entities that control numerous biological activities, information on how they bind and interact to perform said activities is an important scientific endeavor that can bring to light insight into cell mechanisms. A protein-protein interaction (PPI) is defined as the physical binding of two or more proteins. A collection of PPIs in a graph theoretic setting forms a protein-protein interaction network (PPIN). These interactions provide organism cells with the necessary adaptability to various environmental disturbances. Because of that, they are central to all biological processes and enable many different organism functionalities. Therefore, a systematic analysis of PPINs in a biological system has been prominent in our understanding of the functional organization of proteins, as well as of the cell responses to external stimuli and stresses, such as diseases and changing environmental conditions (Hao et al., 2016). PPINs are dense networks, organized in protein complexes and functional modules (groups of nodes), which, in turn, contribute to various cellular processes and activities (Han et al., 2004b).

Figure 3.1 illustrates a mapping of protein-protein interactions in *Saccharomyces cerevisiae* (yeast), based on interactions provided by the STRING database of protein-protein interactions

(Szklarczyk et al., 2014). The visualization of the largest network component of all interactions with a threshold score of 750 or more was performed using NetworkX (Hagberg et al., 2008).

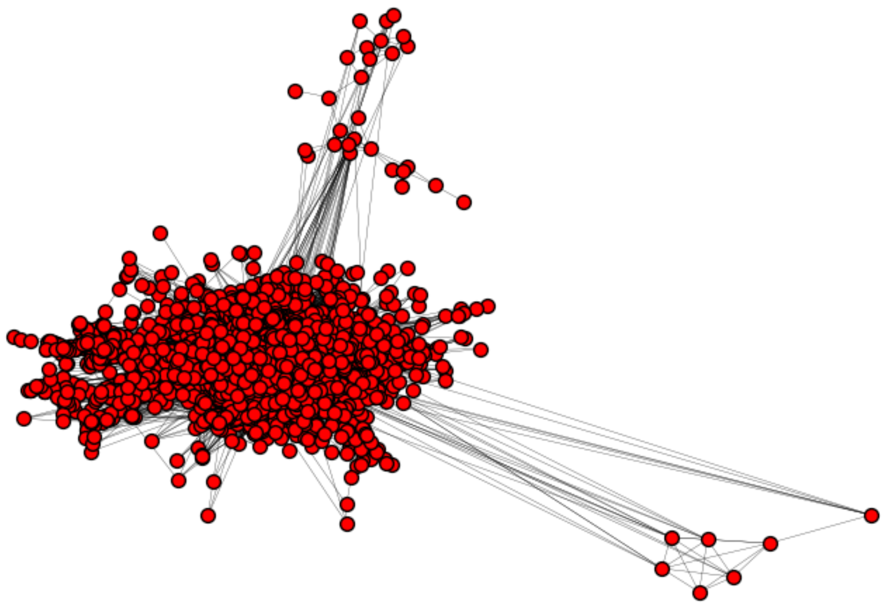


Figure 3.1: The available protein interactions for *Saccharomyces cerevisiae*, as obtained from STRING (Szklarczyk et al., 2014), and the resulting PPIN of the largest network component. All protein-protein interactions with a score of below 750 were removed for visualization purposes.

The main motivation for this research can be summarized as follows: *whether the topological prominence of a protein in a proteome may be a good predictor of its biological importance?* the term *proteome* describes the complete universe of proteins and their interactions in an organism. The challenge we are focusing on is not new, as it has attracted numerous researchers and has led to the investigation of various metrics, ranging from graph modularity (Narayanan et al., 2011) to centrality (Hahn et al., 2005). There exist numerous measures of topological prominence, called network centrality indices; local centrality indices like *degree centrality* assign centrality values based on the topology of the node's local neighborhood, whereas global centrality indices such as *closeness* and *betweenness centrality* indices assign centrality values based on the the accessibility of a network's components and node's role in maintaining the connectivity between pairs of other nodes in the network, respectively. Even though by definition degree centrality is a local measure, depending on the structure of the network, hubs may play an important role in maintaining the overall connectivity of the network. One of the first connections between the two in the context of a protein interaction network, the so-called *centrality-lethality* rule, was observed by (Jeong et al., 2001), who demonstrated that high-degree nodes or hubs in a protein interaction network of *Saccharomyces Cerevisiae* contain more essential proteins than would be expected by chance. Since then the correlation between degree and essentiality was confirmed by other studies [4]–[7], but until recently there was no systematic attempt to examine the reasons for this correlation. In particular, what is the main topological determinant of essentiality? Is it the number of immediate neighbors or some other, more global topological property that essential proteins may have in a protein interaction network?

According to (Vogiatzis et al., 2019) there are some existing caveats with nodal metrics which are covered by group centrality measures. First, assigning importance to a single protein (resp. interaction), instead of a set of proteins (resp. interactions) tends to favor those proteins that participate in large, dense complexes. Secondly, the datasets of PPINs are still not error-free (Hart et al., 2006); assuming complete information can lead to significant misattributions of importance. Last, some proteins that present low co-expression with their interacting partners would be disregarded by such metrics even though they might have a significant role in coordinating different complexes (e.g., “date" hubs). They proposed star centrality metric and showed it outperforms nodal centrality metrics (degree, betweenness, and closeness), while at the same time it alleviates all the above issues.

Also (Hart et al., 2007) demonstrated that essential proteins are distributed unevenly among groups of densely connected and functionally related proteins, which they called them COmplex BIological Modules (COBIMs). Therefore, the majority of the essential proteins lie in those COBIMs that are enriched in essential proteins, which they called them Essential COmplex BIological Modules (ECOBIMs).

Being able to use such objective metrics for studying the proteome is of importance, as it can lead us to the detection of informal groups in the interaction network (Pereira-Leal et al., 2004b). With the term “detection of informal groups" we mean the detection of sets or clusters of proteins, based only on their interactions and the topological structure, and no other externally available information. Such detection techniques would enable us with objective methods of measuring protein importance in the proteome independently of other biological experiments and could guide future experimentation. In general, topological importance (also broadly referred to as centrality) is a well-studied topic in complex networks, including protein-protein interaction networks. In our work, though, we propose a set of novel centrality metrics for each protein in the network. These metrics aims to capture both the individual interactions of every protein, as well as the interactions of its open neighborhood.

In this chapter, we discuss the experimental techniques used to obtain protein-protein interactions, and briefly describe the two most prominent such methods. We also discuss computational *in silico* methods that are often used to identify such interactions. Finally, we present a small survey on the databases currently available for obtaining this data and discuss the reliability and accuracy of the proteomic data hosted, as well as interaction accuracy improvement techniques.

3.2. Graph representations for protein interactions

In this work, we will primarily focus on binary protein interactions, which constitute the most commonly used PPIN structures. These interactions are often obtained by experiments such as the yeast 2 hybrid (Y2H) system, presented later in this chapter. In a binary protein interaction network, proteins interact in pairs and their corresponding nodes are connected by edges. Contrary to the other two representations that are used, PPINs do not contain any information on the protein complexes and how they interact, and hence computational techniques such as the ones discussed in Sect. 4 are needed.

Another possible representation is through a complex intersection graph. In a complex intersection graph, nodes now represent protein complexes with edges connecting complexes that share one or more proteins (Gavin et al., 2002). Hence, edges represent protein memberships in the same complexes, and are useful for representing common localization or architecture. Moreover, this representation allows for a different weight of the edges (other than the reliability weight that will be discussed in this research): this weight is higher when the number of common proteins between two complexes/nodes is bigger.

Finally, a third commonly used representation is with the use of protein complex hypergraphs. A hypergraph consists of sets of vertices (proteins) and sets of hyperedges (protein complexes), with each hyperedge containing a subset of the vertices. Protein complex hypergraphs can be reformulated as bipartite graphs with the complexes connected to multiple proteins depending on their memberships. In this setup, the degree of a vertex is the number of hyperedges (complexes) it belongs to, and the degree of a hyperedge is the cardinality of the vertices (proteins) it consists of. Ramadan et al. (2004) modeled the yeast protein complex data as a hypergraph and showed that the resulting network showcased small-world and power-law properties. They also proposed an algorithm to identify the core proteome, the maximum core of the protein complex hypergraph, with their results showing that the core proteome is enriched in essential proteins.

3.3. Experimental techniques

Interactions between proteins can be identified and characterized by a wide variety of genetic, biochemical, physical, and computational methods, as showcased in Table 3.1. These methods are categorized into two big groups: techniques that allow for screening large numbers of proteins in a cell (also referred to as *high-throughput* methods) and ones that are typically used to identify and characterize specific biochemical and physio-chemical properties of a single protein complex (Shoemaker et al., 2007). Examples of the first group of high-throughput methods include the well-known and commonly practiced yeast two-hybrid (Y2H) technique and affinity purification followed by mass spectrometry, which are described in subsections 3.3.1 and 3.3.2. Methods can also be divided into three categories, based on whether they can be applied in vivo, in vitro, or in silico.

There are mostly two types of methods that lead to the creation and extraction of proteomic data. Those are (a) *binary* methods, which extract direct physical interactions between proteins by testing every pair predicted to exist, and (b) *co-complex* methods, which identify physical interactions among members of a protein complex without necessarily determining the direct partners of

Table 3.1: This table presents some of the most prominently used methods for obtaining protein-protein interactions.

Technique	**High/Low throughput**	**In vivo/ vitro/silico**	**References**
Affinity chromatography	High	In vitro	Cuatrecasas (1970), Gygi et al. (2002), Peng et al. (2003)
Atomic force microscopy	Low	In vitro	Yang et al. (2003)
Co-immunoprecipitation	High	In vitro	Moresco et al. (2010), Borch et al. (2011)
Domain-pairs-based methods	High	In silico	Wojcik et al. (2001), Valente et al. (2013)
Electron microscopy	Low	In vitro	Baumeister et al. (1999)
Fluorescence resonance energy transfer	Low	In vivo	Yan et al. (2003)
Gene Neighborhood (Chromosome Proximity), Gene Fusion, and Gene Expression	High	In silico	Marcotte et al. (1999b), Enright et al. (1999), Tsoka et al. (2000), Grigoriev (2001) Pazos et al. (2008)
Ortholog-based sequence approaches	High	In silico	Chen et al. (2006a), Lee et al. (2008)
In silico-2-hybrid	High	In silico	Pazos et al. (2002)
NMR spectroscopy	Low	In vitro	Gao et al. (2004), Thompson et al. (2015)
Phage display	High	In vitro	Tong et al. (2002), Sidhu et al. (2003), Sidhu et al. (2007)
Phylogenetic Trees and Profiles	High	In silico	Pazos et al. (2001), Craig et al. (2007), Srinivas et al. (2008), Lin et al. (2013)
Protein-fragment complementation	High	In vitro	Michnick et al. (2011), Remy et al. (2015), Michnick et al. (2016)
Protein microarrays	High	In vitro	MacBeath et al. (2000), Templin et al. (2002), Hall et al. (2007)
Structure-based approaches	High	In silico	Lu et al. (2002), Aytuna et al. (2005), Hosur et al. (2011), Zhang et al. (2012), Mariano et al. (2017)
Surface plasmon resonance	Low	In vitro	Hart et al. (1999), Cooper (2003)
Synthetic lethality	High	In vivo	Bender et al. (1991), Rutherford (2000), Tong et al. (2001), Tong et al. (2004)
Tandem affinity purification – mass spectroscopy (TAP/MS)	High	In vitro	Rigaut et al. (1999), Gavin et al. (2002), Rohila et al. (2004)
Yeast 2 hybrid (Y2H)	High	In vivo	Fields et al. (1989), James et al. (1996), Uetz et al. (2000), Ito et al. (2001)

The methods are characterized by their *high* or *low throughput* capabilities, and their application *in vivo*, *in vitro*, or *in silico*. References to each of the methods are also provided.

each interaction. Co-complex methods, unlike binary methods, measure then both direct and indirect interactions between proteins. The yeast two-hybrid (Y2H) system (Fields et al., 1989), and affinity purification followed by mass spectrometry (AP/MS) (Kumar et al., 2002) are, respectively, the most commonly used binary and co-complex techniques for extracting protein interactions.

As will be shown in the following subsections, Y2H and AP/MS investigate different key characteristics of the whole interactome and both (or, similarly structured techniques) are needed for a high quality view of the full protein-protein interaction network (Yu et al., 2008). We now proceed to describe these two systems in more detail.

3.3.1. Yeast two hybrid system (Y2H)

As already mentioned, the Y2H system identifies direct physical interactions between pairs of proteins, represented as binary relationships (Ito et al., 2001; Yu et al., 2008; Venkatesan et al., 2009). Early Y2H system measurements for *Saccharomyces cerevisiae* were used to show the existence of a limited number of highly connected hub proteins (Jeong et al., 2001; Barabasi et al., 2004) that coordinate cell organization, an impressive result that will be discussed in a later chapter about essentiality and lethality.

Y2H uses the working principle that eukaryotic transcription activators commonly have two domains: one directly binding to the DNA (referred to as the *binding domain*) and another activating transcription (*activating domain*). The main idea (illustrated in Figure 3.2) is that the two proteins being tested for interaction are indicated as fusion proteins (hybrids) with a DNA binding domain and a transcriptional activating domain, with one of them playing the role of the bait. The bait is fused to the binding domain, while the other protein (playing the role of the prey) contains the activating domain. If the two proteins do indeed physically interact, then a reporter gene is activated which can, in turn, be detected and measured (Fields et al., 1989).

Expectedly, there are some advantages and disadvantages associated with the Y2H system. First, being an in vivo technique, it is quite simple, fast, and reasonably inexpensive, due to its minimal requirements, especially when compared to the high volume of purified proteins which are needed in traditional biochemical approaches. Furthermore, it does not require any previous

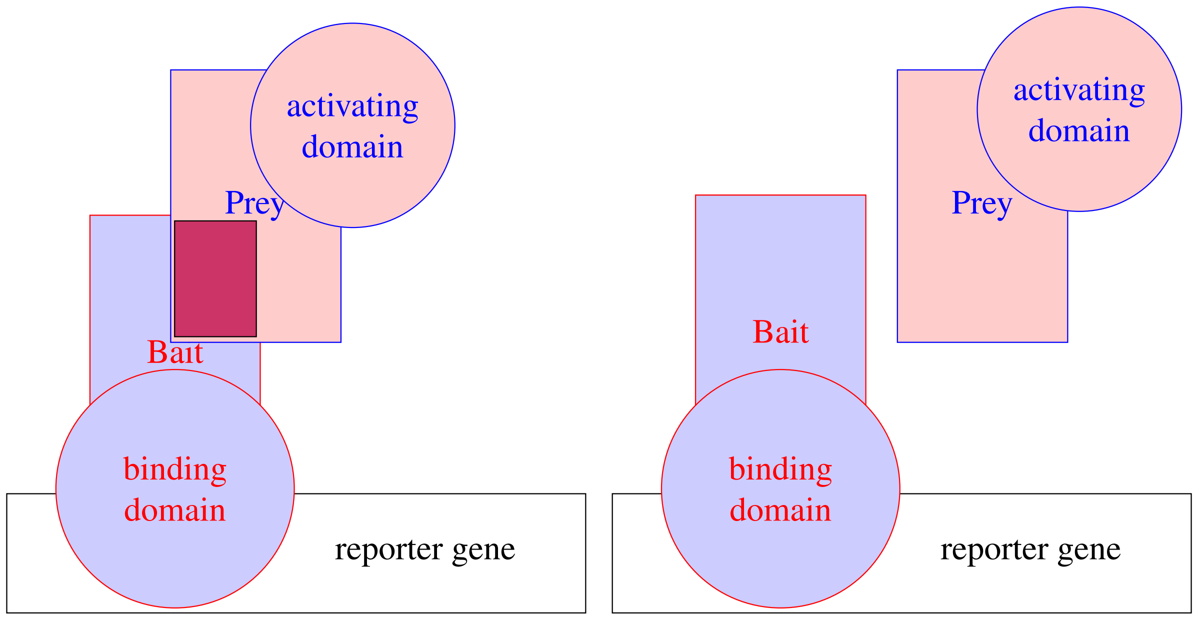


Figure 3.2: On the left, if the two (bait and prey) proteins interact, then the activating domain binds to the binding domain and the reporter gene is activated. On the other hand (right), if no interaction occurs, then the reporter gene remains disabled and is not transcribed.

knowledge of proteins and can be applied as soon as the matching genes are known. On the other hand, it can only detect binary interactions and is not able to identify cooperative binding. Another drawback is its inability to investigate certain types of proteins, such as transcription factors, as their hybrids can activate the transcription without an interaction actually taking place. Continuous use of artificial hybrids could lead to false negatives, that is, a true interaction might escape identification under the Y2H technique. Last, given two interactions between proteins A and B, and proteins B and C, Y2H does not reveal whether the two interactions do take place simultaneously or whether they are exclusive to one another. For these main reasons, extracted interactions by Y2H must be further analyzed in order to assess their biological relevance, as noted by Yu et al. (2008).

3.3.2. Affinity purification followed by mass spectrometry (AP/MS)

AP/MS is a co-complex method that detects the presence of a protein within a protein complex. In contrast to the Y2H system, AP/MS identifies physical interactions between proteins in a complex without distinguishing whether those are direct or indirect (Yu et al., 2008). Hence, the AP/MS method is dependent upon a pre-selection of a tagged protein with a molecular marker (referred to as a “bait” protein). This protein is used to extract a group of proteins associated with it (called “prey” proteins), followed by a biochemical technique to separate them through co-purification (Berggård et al., 2007).

The interaction information obtained by co-complex methods is different than the information obtained by employing binary techniques. A small example is provided in Figure 3.3, where protein 1 directly interacts with proteins 2, 5, and 6, but also participates in the same complex with every other protein. This is reflected in the interactions obtained by a co-complex method, as opposed to a binary one. It should be noted that, upon further processing, interactions that have been extracted by employing a co-complex method can be “converted” to pairwise (direct) interactions (Hakes et al., 2006).

It has been generally discussed in the literature that AP/MS more closely reflects the actual multidirectional complexity of the real PPIN, seeing as it covers more than a mere subspace of the whole interactome (Gingras et al., 2007), which is what Y2H does. On the other hand, the probability of false interactions being present in the final PPIN increases with the use of AP/MS, while the opposite is true for Y2H (Trinkle-Mulcahy et al., 2008; Boulon et al., 2010; Hubner et al., 2010). This density of interactions also leads to increased difficulty in recognizing the actual structural topology of protein complexes when using AP/MS (Gingras et al., 2007; Bensimon et al., 2012).

3.4. Protein-protein interaction databases

The interactions identified by different studies (and using a variety of the techniques mentioned earlier) are then collected, documented, and organized in curated databases. This is an invaluable process that allows scientists to characterize the entire interactome by properly combining, integrating, and leveraging information from a range of diverse experiments. It also provides non-domain scientists who are interested in developing novel quantitative methods with excellent instances on which to test their approaches. A list of the most prominently used databases for obtaining protein-protein interactions is presented in Table 3.2. Inactive databases

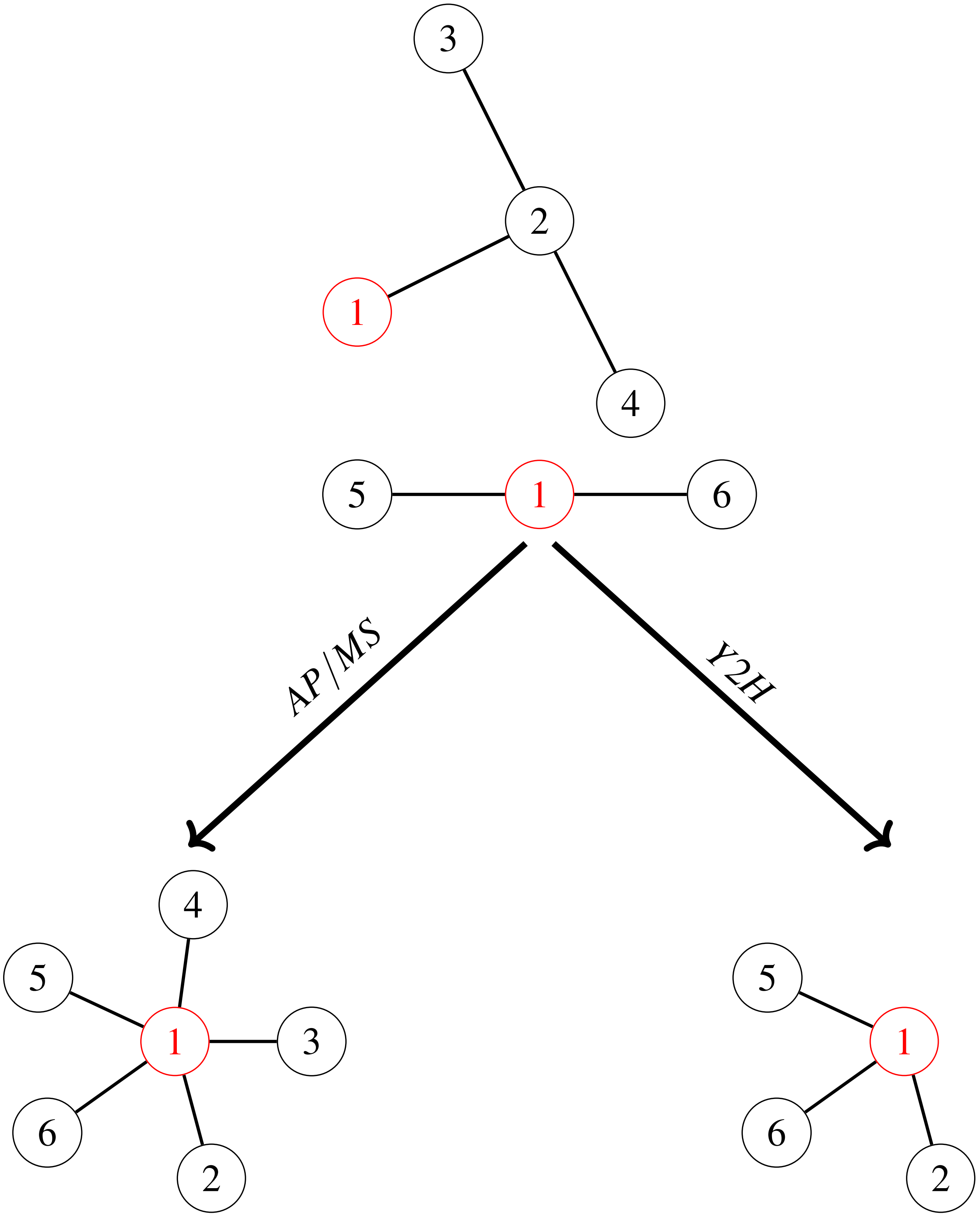


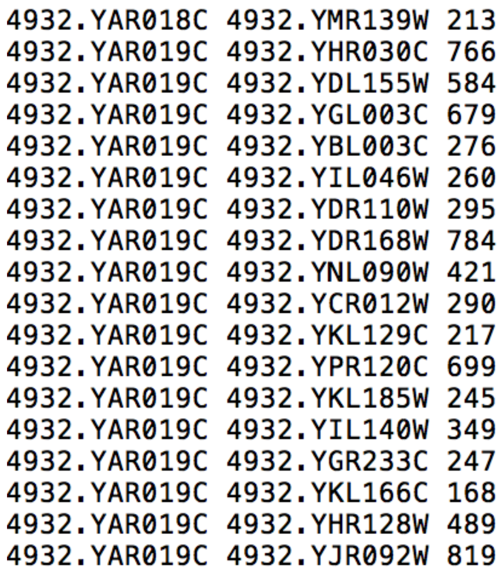
Figure 3.3: We observe here the original interactions (top), and the obtained interactions by a binary method (bottom left) and a co-complex method (bottom right).

that had attracted significant scientific interest include BIND: the biomolecular interaction network database (Bader et al., 2001; Bader et al., 2003b), which can now be downloaded from `http://download.baderlab.org/BINDTranslation` due to the efforts of Isserlin et al. (2011), who converted the material to the Proteomics Standard Initiative-Molecular Interaction 2.5 format and made it available for download. Last, an excellent work attempting to help researchers navigate the plethora of publications and data available on protein protein interactions is the web-based service of *iHOP* (`http://www.ihop-net.org/`), created by Hoffmann et al. (2002).

We also provide here a screenshot of two prominent PPIN databases. The screenshots are provided in Figures 3.4 and 3.5, and present two indicative data files that can be obtained from STRING and BioGRID. The STRING file is more intuitive in its simplest version, and presents **the two interactors** (separated by a space) along with an interaction score. The interactor field consists of a numeric code (organism identifier, e.g., 6239 in the screenshot) and the interactor itself. The BioGRID file consists of numerous fields. In the screenshots, we present all 24 fields in two parts (as they would not fit in a single screenshot). The order the fields appear in is as follows. First, a BioGRID interaction ID is provided, followed by the Entrez gene IDs for the two interactors as well as the BioGRID IDs for the interactors. **Their systematic names** are provided later, and their official symbols follow. Synonyms for the two interactors are provided immediately after, and the experimental system that led to the discovery of the interaction and the system type are presented later. The remaining fields deal with the author of the study, the PubMed identification for the study, and the organism ID for the two interactors: if they belong to the same organism as is the case here, these two numbers should be the same. Finally, the throughput (i.e., high/low) and an interaction confidence score ranging between 0 and 1 are provided. The remaining fields contain information on other information that is sometimes absent, such as qualifications, tags, and the source database. The fields in bold would help identify the same interaction between a file obtained by STRING format and by BioGRID.

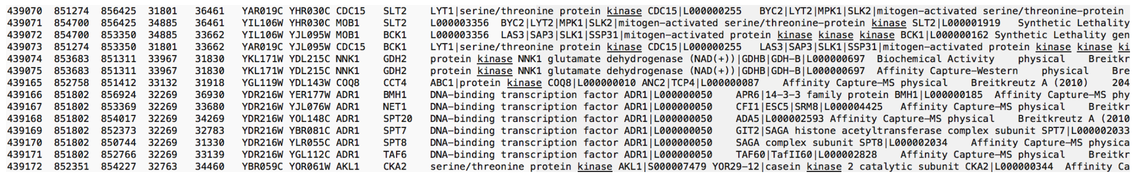
Table 3.2: A list of 21 commonly used sources of proteomic data, available online.

Database	Description	URL	References
3did	Database containing domain-domain interactions with known 3D structures. An InterpreTS z-score (Aloy et al., 2003) is provided to describe the interaction specificity.	`http://3did.irbbarcelona.org`	Mosca et al. (2013), Stein et al. (2010)
APID	Agile Protein Interactomes Data Server contains known and experimentally validated protein-protein interactions for more than 400 organisms, combining information from different sources/databases.	`cicblade.dep.usal.es:8080/APID/`	Alonso-López et al. (2016), Prieto et al. (2006)
atBioNet	Web-based network analysis tool integrating seven publicly available protein-protein interaction databases; performs protein complex and functional module detection.	`https://www.fda.gov/scienceresearch/bioinformaticstools/ucm285284.htm`	Ding et al. (2012)
BioGRID	Database that archives and makes available large amounts of protein interaction data; it completely covers the literature of *S. cerevisiae*, *S. pombe*, and *A. thaliana*.	`http://thebiogrid.org`	Stark et al. (2006), Oughtred et al. (2016), Chatr-aryamontri et al. (2017)
BioProfiling	Performs network-based interpretation analyses based on genetic and proteomic data.	`http://www.bioprofiling.de`	Antonov (2011), Antonov et al. (2009)
DIMA	Domain Interaction MAp (currently at version 3.0) contains a list of predicted (by four computational methods) and verified (from other databases) protein domain interactions.	`http://webclu.bio.wzw.tum.de/dima`	Luo et al. (2010b), Pagel et al. (2007), Pagel et al. (2004a)
DIP	This Database of Interacting Proteins, curated both manually and using computational techniques, combines a variety of sources to list experimentally determined protein interactions.	`http://dip.doe-mbi.ucla.edu/dip/`	Xenarios et al. (2000), Xenarios et al. (2002)
DOMINE	Known and predicted (by eight different computational techniques) protein domain interaction data.	`http://domine.utdallas.edu`	Raghavachari et al. (2007)
HAPPI	Contains physical and functional human protein-protein interactions provided with a confidence score.	`http://discovery.informatics.uab.edu/HAPPI/`	Chen et al. (2009), Chen et al. (2017)
HINT	High-quality interactomes for 12 organisms, including *Homo sapiens*.	`http://hint.yulab.org`	Das et al. (2012)
HPRD	Human protein-protein interaction data, curated manually after reading the relevant literature.	`http://www.hprd.org`	Peri et al. (2003), Mishra et al. (2006), Keshava Prasad et al. (2008), Goel et al. (2012)
IRView	Proteins interact through their *interacting regions*, and IRView aims to serve as a viewer and retriever of such regions for *Mus musculus* and *Homo sapiens*.	`http://ir.hgc.jp`	Fujimori et al. (2012)
MiMI	Michigan Molecular Interactions serves as a data aggregator for protein-protein interactions from different databases, properly merged to include all origin information. A Cytoscape plugin is also provided.	`http://mimi.ncibi.org`	Jayapandian et al. (2006), Tarcea et al. (2008), Gao et al. (2008)
MIntAct	Formerly two different molecular (incl. protein) interaction databases, MINT and IntAct, combined their efforts to store and disseminate interaction data.	`http://www.ebi.ac.uk/intact/`	Orchard et al. (2013), Licata et al. (2011), Kerrien et al. (2011)
MIPS Mammalian Protein-Protein Interaction Database	One of the most well-known sources for protein-protein interaction data. It is a carefully curated database consisting of only interaction data from individually performed experiments.	`http://mips.helmholtz-muenchen.de/proj/ppi/`	Pagel et al. (2004b)
PathwayLinker	Web-based service that integrates protein-protein interactions from multiple sources with statistical significance testing.	`http://PathwayLinker.org`	Farkas et al. (2012)
PINA	Protein Interaction Network Analysis is a web-based resource which includes a database of protein-protein interactions from six manually curated sources, along with tools for visualization and analysis.	`http://cbg.garvan.unsw.edu.au/pina/`	Cowley et al. (2011), Wu et al. (2009a)
PIPs	PIPs is a human protein-protein interaction prediction tool using a score to assess its plausibility.	`http://www.compbio.dundee.ac.uk/www-pips`	McDowall et al. (2008), Scott et al. (2007)
ProLinks	ProLinks offers inferred, functional links between proteins based on phylogenetic profiles, gene cluster, gene fusion, and gene expression. Protein-protein interaction data	`http://prl.mbi.ucla.edu/prlbeta/`	Bowers et al. (2004)
STRING	STRING is one of the most comprehensive sources for proteomic instances available in the literature. It includes experimentally validated and predicted protein-protein interactions, along with physical and functional links, over more than 2,000 organisms.	`http://string.embl.de`	Szklarczyk et al. (2017), Szklarczyk et al. (2014), Von Mering et al. (2005), Snel et al. (2000)
UniHI	The Unified Human Interactome project integrates data on human protein-protein interactions from different sources, and using four methods (two computational and two experimental).	`http://www.unihi.org`	Kalathur et al. (2013), Chaurasia et al. (2006), Futschik et al. (2007)

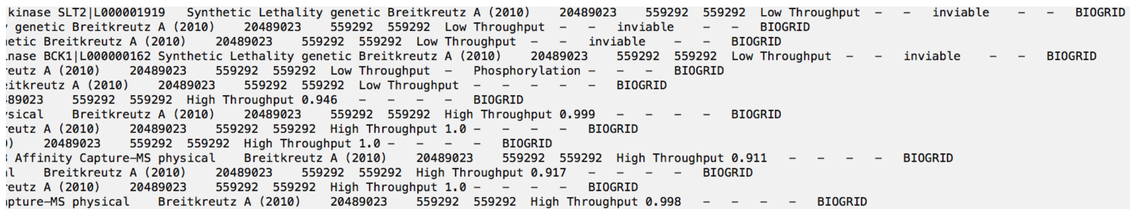


```
4932.YAR018C 4932.YMR139W 213
4932.YAR019C 4932.YHR030C 766
4932.YAR019C 4932.YDL155W 584
4932.YAR019C 4932.YGL003C 679
4932.YAR019C 4932.YBL003C 276
4932.YAR019C 4932.YIL046W 260
4932.YAR019C 4932.YDR110W 295
4932.YAR019C 4932.YDR168W 784
4932.YAR019C 4932.YNL090W 421
4932.YAR019C 4932.YCR012W 290
4932.YAR019C 4932.YKL129C 217
4932.YAR019C 4932.YPR120C 699
4932.YAR019C 4932.YKL185W 245
4932.YAR019C 4932.YIL140W 349
4932.YAR019C 4932.YGR233C 247
4932.YAR019C 4932.YKL166C 168
4932.YAR019C 4932.YHR128W 489
4932.YAR019C 4932.YJR092W 819
```

Figure 3.4: A screenshot of the S. Cerevisiae proteome file available by STRING. Each interaction is represented by a row in the file.



```
439070  851274  856425  31801   36461   YAR019C YHR030C CDC15   SLT2    LYT1|serine/threonine protein kinase CDC15|L000000255   BYC2|LYT2|MPK1|SLK2|mitogen-activated serine/threonine-protein
439071  854700  856425  34885   36461   YIL106W YHR030C MOB1    SLT2    L000003356  BYC2|LYT2|MPK1|SLK2|mitogen-activated serine/threonine-protein kinase SLT2|L000001919   Synthetic Lethality
439072  854700  853350  34885   33662   YIL106W YJL095W MOB1    BCK1    L000003356  LAS3|SAP3|SLK1|SSP31|mitogen-activated protein kinase kinase kinase BCK1|L000000162 Synthetic Lethality gen
439073  851274  853350  31801   33662   YAR019C YJL095W CDC15   BCK1    LYT1|serine/threonine protein kinase CDC15|L000000255   LAS3|SAP3|SLK1|SSP31|mitogen-activated protein kinase kinase ki
439074  853683  851311  33967   31830   YKL171W YDL215C NNK1    GDH2    protein kinase NNK1 glutamate dehydrogenase (NAD(+))|GDHB|GDH-B|L000000697  Biochemical Activity    physical    Breitkr
439075  853683  851311  33967   31830   YKL171W YDL215C NNK1    GDH2    protein kinase NNK1 glutamate dehydrogenase (NAD(+))|GDHB|GDH-B|L000000697  Affinity Capture-Western    physical    Bre
439165  852758  851412  33132   31918   YGL119W YDL143W COQ8    CCT4    ABC1|protein kinase COQ8|L000000010 ANC2|TCP4|L000000087    Affinity Capture-MS physical    Breitkreutz A (2010)    204
439166  851802  856924  32269   36930   YDR216W YER177W ADR1    BMH1    DNA-binding transcription factor ADR1|L000000050    APR6|14-3-3 family protein BMH1|L000000185  Affinity Capture-MS phy
439167  851802  853369  32269   33680   YDR216W YJL076W ADR1    NET1    DNA-binding transcription factor ADR1|L000000050    CFI1|ESC5|SRM8|L000004425   Affinity Capture-MS physical    Breitkr
439168  851802  854017  32269   34269   YDR216W YOL148C ADR1    SPT20   DNA-binding transcription factor ADR1|L000000050    ADA5|L000002593 Affinity Capture-MS physical    Breitkreutz A (2010
439169  851802  852373  32269   32783   YDR216W YBR081C ADR1    SPT7    DNA-binding transcription factor ADR1|L000000050    GIT2|SAGA histone acetyltransferase complex subunit SPT7|L000002033
439170  851802  850744  32269   31330   YDR216W YLR055C ADR1    SPT8    DNA-binding transcription factor ADR1|L000000050    SAGA complex subunit SPT8|L000002034    Affinity Capture-MS physica
439171  851802  852766  32269   33139   YDR216W YGL112C ADR1    TAF6    DNA-binding transcription factor ADR1|L000000050    TAF60|TafII60|L000002828    Affinity Capture-MS physical    Breitkr
439172  852351  854227  32763   34460   YBR059C YOR061W AKL1    CKA2    serine/threonine protein kinase AKL1|S000007479 YOR29-12|casein kinase 2 catalytic subunit CKA2|L000000344  Affinity Ca
```



```
kinase SLT2|L000001919   Synthetic Lethality genetic Breitkreutz A (2010)    20489023    559292  559292  Low Throughput  -   -   inviable    -   -   BIOGRID
 genetic Breitkreutz A (2010)    20489023    559292  559292  Low Throughput  -   -   inviable    -   -   BIOGRID
etic Breitkreutz A (2010)    20489023    559292  559292  Low Throughput  -   -   inviable    -   -   BIOGRID
nase BCK1|L000000162 Synthetic Lethality genetic Breitkreutz A (2010)    20489023    559292  559292  Low Throughput  -   -   inviable    -   -   BIOGRID
eutz A (2010)    20489023    559292  559292  Low Throughput  -   Phosphorylation -   -   -   BIOGRID
itkreutz A (2010)    20489023    559292  559292  Low Throughput  -   -   -   -   -   BIOGRID
89023    559292  559292  High Throughput 0.946   -   -   -   -   BIOGRID
sical    Breitkreutz A (2010)    20489023    559292  559292  High Throughput 0.999   -   -   -   -   BIOGRID
eutz A (2010)    20489023    559292  559292  High Throughput 1.0 -   -   -   -   BIOGRID
)    20489023    559292  559292  High Throughput 1.0 -   -   -   -   BIOGRID
 Affinity Capture-MS physical    Breitkreutz A (2010)    20489023    559292  559292  High Throughput 0.911   -   -   -   -   BIOGRID
l    Breitkreutz A (2010)    20489023    559292  559292  High Throughput 0.917   -   -   -   -   BIOGRID
eutz A (2010)    20489023    559292  559292  High Throughput 1.0 -   -   -   -   BIOGRID
pture-MS physical    Breitkreutz A (2010)    20489023    559292  559292  High Throughput 0.998   -   -   -   -   BIOGRID
```

Figure 3.5: Two screenshot sof the S. Cerevisiae proteome file available by BioGRID. Each interaction is represented by a row in the file. The first screenshot (top) presents the first 11 fields. The second screenshot (bottom) presents the remaining 13 fields.

3.5. Reliability of databases

One of the most important issues typically associated with interactome databases and their content is their reliability. A comparison of the PPINs obtained from different databases has revealed surprising discrepancies: small overlap of available interactions, large error rates, even apparent contradictions between datasets (Von Mering et al., 2002; Edwards et al., 2002; Goll et al., 2006). These discrepancies arise due to a variety of reasons, one of which is the high noise rates present in the output of the high-throughput techniques used to obtain the interactions in the first place. Under some circumstances, these discrepancies are attributed to a selection bias, where interactions that are not understood are removed (Mrowka et al., 2001). In any case, coverage is low enough and the errors are unfortunately still common throughout, which warrants new techniques to improve reliability (Hart et al., 2006). Many computational strategies have been proposed to mitigate the impact of false positives and negatives in the obtained interactions, as noted by Peng et al. (2016). Such techniques work by either predicting interactions that should be present and adding them (leading to a decrease in the number of false negatives), or by evaluating the accuracy of the obtained interactions and removing those considered noise or that stand below a certain threshold (which leads to a decrease in the number of false positives).

Methods to detect and assess PPIN reliability in the literature are categorized in three main groups:

1. experimental-based methods;
2. topology-based methods;
3. and sequence-based methods.

Experimental-based methods are similar to the ones discussed in subsection 3.3. They are used to validate and assess the reliability of the obtained interactions by using either direct bait-prey and indirect prey-prey relationships, or frequency-based techniques which assume that the higher the number of times an interaction is reported by different, independent experiments, the more reliable it is prone to be (Sprinzak et al., 2003; Chua et al., 2006; Chiang et al., 2007).

To avoid the weaknesses of using experimental-based methods, topology-based methods were introduced to either predict new and reliable interactions, or to assess existing interactions. Saito et al. (2002) and Saito et al. (2003) proposed techniques that helped identify false positives via computational measures derived from network topology, referred to as "interaction generalities". Liu et al. (2004) and Liu et al. (2008) used both local and global topological information to filter reliable interactions as well as predict new, previously unreported, ones. Neighborhood-based methods have also been used for predicting PPIs, with the average distance or the shortest path distances between all proteins as a proxy-criterion to assess and predict interactions (Li et al., 2006a; Fang et al., 2011; Birlutiu et al., 2014). Liu et al. (2009) then presented an iterative scoring method to setup weighted PPINs, with each interaction weight indicating its reliability. Another measure referred to as *functional similarity weight* was introduced by Chua et al. (2006) and is an extension of using the common neighbors of two proteins to estimate their similarity (Goldberg et al., 2003); instead, on top of this topological information it also employs a scoring scheme based on the reliability of the source. FS-Weight, along with other methods that are based on identifying common interacting partners (neighbors) between proteins, was shown to perform well, especially when noise reduction schemes are used for preprocessing.

Such a noise reduction method is the one proposed by Kritikos et al. (2011). It works by properly weighing interactions in a way that assigns higher confidence scores to more probable true positive interactions. Initially, they applied their own method, along with three other commonly used weighing techniques, to the *S. cerevisiae* interactome. Then, they used four weighted-graph clustering algorithms to cluster the graphs obtained from the first step. This computational experiment finally demonstrated that weighting schemes tend to improve protein complex prediction remarkably, compared to the same results when using the data as is without any preprocessing.

Other techniques to improve the reliability of the used protein interactions include the *PE-measure* proposed by Zaki et al. (2013) and *CAPPIC* (cluster-based assessment of protein-protein interaction confidence) introduced by Kamburov et al. (2012a). Both take advantage of the expectedly modular nature of PPINs, albeit in different ways. The PE-measure assesses the reliability of the available protein-proteins interactions by considering whether an interaction is supported or not by its neighboring interactions. On the other hand, CAPPIC uses Markov Clustering on a generated line graph of the overall PPIN (see Chapter 4 for a discussion of how using line graphs helps identify protein complexes and modules that are biologically significant, as, e.g., in the work by Pereira-Leal et al. (2004b)). This clustering allows then for the scoring of interactions depending on whether they agree (higher confidence) or not (lower confidence) with the expected modular nature of the network. An implementation of CAPPIC as well as other computational techniques to assess the reliability of the obtained PPIN can be found in `http://cpdb.molgen.mpg.de/cappic` as part of intScore, an interaction confidence score tool (Kamburov et al., 2012b).

Chen et al. (2005) hypothesized that a pair of protein that are connected by a short path of reliable interactions are likely to interact directly. With this working assumption in mind, they introduced an index, referred to as the interaction pathway reliability index (IPR), on a pair of candidates of interacting proteins as the maximum reliability of the shortest non-reducible indirect path connecting them. In subsequent work, Chen et al. (2006b) introduced a method, referred to as interaction reliability by alternative path, for *repurifying* the extracted interactions employing topology-based metrics to identify and remove false positives, as well as to detect and add highly reliable, but still unobserved, interactions. In the same logic of the underlying assumption that proteins in the same modules have to share common functional properties, a multitude of clustering methods can be used to assess interaction reliability and predict new connections (see Chapter 4 for more details on these methods).

Continuing, sequence-based methods consider the full genetic sequence information available and work under the assumption that high similarity in the sequence information of two proteins can imply interaction or shared functionality. This kind of information can also be used either to predict new interactions, as in the work by Guo et al. (2008), or to assess the confidence score of existing interactions and infer the functions of uncharacterized proteins (Deane et al., 2002). Methods that are based on gene co-expression typically use the Pearson correlation coefficient as a criterion to identify high co-expression rates in the genes that encode the interacting proteins, and use this to assess protein-protein interaction reliability (Deng et al., 2002). We keep this discussion rather brief as our focus with this survey is to present results related to PPINs, rather than delve into the, admittedly very important and vast, literature of using full gene sequences for deriving and evaluating such interactions.

Due to the general lack of available data for some of these techniques — an issue associated especially with those based on sequence information — methods that integrate information from multiple sources have been successfully proposed and applied. Deane et al. (2002) and Deng et al. (2002) were among the first contributions to propose methods that used similarity in mRNA expression profiles to estimate the reliability of perviously observed interactions. Bader et al. (2004) used additional features of interacting proteins, including their data source, the number of interacting partners, and topological features, such as network clustering, functional similarity, co-expression, as well as the presence of genetic interactions to finally assign confidence scores to protein interactions. The scheme devised by Sharan et al. (2005) assigned probabilities to interactions using a logistic regression model based on clustering, mRNA expression, and the number of times an interaction had been observed in independent experiments. Myers et al. (2005), Marcotte et al. (1999a), Jansen et al. (2003), and Von Mering et al. (2005) proposed more methods based on calculating a log-likelihood score for each interaction to predict new protein interactions, incorporating combinations of different information including functional similarity, co-essentiality, high mRNA expression correlation, and co-evolution. Last, Qi et al. (2005) and Qi et al. (2006) calculated the probability of proteins interacting by integrating direct evidence (e.g., the type of experiment used to report the interaction) and indirect evidence (e.g., gene expression, domain-domain interactions).

Moreover, as it is generally assumed that interacting proteins are expected to be localized to the same cellular component or to have a common cellular role (Sprinzak et al., 2003; Chua et al., 2006; Chen et al., 2006b), Chua et al. (2008) categorized interacting proteins into four groups. Proteins having both common cellular localization and common cellular role are considered the most reliable, while those having no common cellular role and no common cellular localization are considered least reliable, and those having a common cellular localization or a common cellular role but not both are considered to be of intermediate reliability. From information derived from multiple organisms based on the evolutionary conservation principle (Liu et al., 2005), Ramani et al. (2008) showed that a considerably higher log-likelihood ratio can be obtained by incorporating co-expression that is conserved in a second genome.Marcotte et al. (1999b), Enright et al. (1999), and Tsoka et al. (2000) used gene fusion events as a strong indication for underlying protein interactions. The fact that proteins interact through their complementary interacting domains is the central idea behind proposing approaches based on interacting domains (Sprinzak et al., 2001; Wan et al., 2002; Han et al., 2004a; Wojcik et al., 2001) or interacting motifs (Tong et al., 2002; Aytuna et al., 2005; Li et al., 2006b) to identify false negatives. Coevolution-based methods are another group of methods to predict interacting proteins(Goh et al., 2000; Pazos et al., 2001; Pazos et al., 2005). According to these methods, the interaction sites of proteins tend to coevolve, which implies that mutations in one protein must be compensated by mutations in the interacting partner-proteins (Fryxell, 1996; Pazos et al., 1997).

For a thorough review of the computational methods available for predicting interactions based on sequence, evolution, co-expression, and other structural data, we refer the interested reader to Keskin et al. (2016). To that same extent, Zahiri et al. (2013) also reviewed computational methods for predicting and assessing the confidence of protein-protein interactions categorizing them within network topology-based methods, methods based on genomic context and structural information, methods based on text mining and literature mining (or database search), and methods based on machine learning algorithms that use genomic/proteomic properties.

CHAPTER 4. PROTEIN COMPLEXES AND FUNCTIONAL MODULES

4.1. Introduction

One of the most striking and impressive topological properties of PPINs is their modular organization. In general, networks from varied sources (social, biological, information) typically possess groups of nodes that are very densely connected to one another, while simultaneously having sparser connections to the rest of the network (Girvan et al., 2002). Such groups are referred to as *communities*; for an excellent overview of community detection in graphs, including ones that arise in computational biology, we refer the interested reader to Fortunato, 2010.

PPINs are no exception to this phenomenon with such communities playing the role of protein complexes and functional modules. Moreover, it has been observed that proteins that are involved in the same biological processes do tend to interact more often with one another (Von Mering et al., 2002), as we also noted earlier in this survey.

Hence, community detection algorithms, also referred to as *network clustering algorithms*, can provide us with methods to deconstruct a given PPIN into its "communities". The general insight behind identifying such communities is that these protein groups will also correspond to structurally independent functional units in the proteome (Danon et al., 2005; Brohee et al., 2006). Clustering is, of course, not applicable to only biological networks and PPINs. However, a simple application of clustering techniques to PPINs, while suitable for other types of networks, does not always yield protein complexes that represent functional modules. This is attributed to the unique features and characteristics of PPINs that require that the employed clustering approaches be carefully tuned and specifically designed.

Some of these unique PPIN features are summarized by Bhowmick et al. (2016). Among these characteristics, we note the large-scale nature of the problem (typically including thousands of proteins and hundreds of thousands of interactions), which requires that the clustering methods used are scalable, the necessity for full coverage of the overall network, and the existence of both dense and sparse functional modules within the already identified complexes. Last, another important feature is the overlapping nature of the obtained clusters, as proteins may possess multiple functionalities (e.g., “housekeeping” proteins that participate in regular cell activities, such as cell metabolism).

4.2. Detecting protein complexes and functional modules

In this section, the computational methods for detecting protein complexes and functional modules from PPINs are organized in four general categories, each provided its own subsection:

1. identifying protein complexes based on interesting structures (e.g., cliques, clique relaxations, core-periphery, etc.);
2. identifying protein complexes based on clustering techniques;
3. identifying protein complexes through integrating multiple data sources;
4. identifying protein complexes in dynamic PPINs.

Table 4.1 presents all the approaches discussed in this section of the review.

4.2.1. Identifying protein complexes based on interesting structures

As already noted, there exists some correlation between PPIN topology and the functional role and localization of groups of proteins (Brohee et al., 2006). Proteins with similar functional properties are observed to interact with one another within relatively separated subgraphs of the original PPIN (Von Mering et al., 2002). Identifying such protein complexes will then provide us with a higher level of functional protein annotation, and a better understanding of the large-scale organization of the cell and its activities (Brohée et al., 2008). Considering the property of protein complexes to possess relatively high link-density (or, in simpler terms, more internal edges that external ones), models and approaches to identify either dense subgraphs (King et al., 2004; Kenley et al., 2011) or clique and clique-like structures (Adamcsek et al., 2006; Li et al., 2005; Cui et al., 2008) have been developed to detect modules in PPINs.

In contrast, some recent studies have been shifting towards a realization that not all protein complexes need necessarily form dense structures; instead, they can be sparsely connected as far as the average degree within the cluster is concerned (Chen et al., 2012). This realization has led

Table 4.1: A table summarizing all the methods presented in this section.

Based on structures	
Spirin et al. (2003)	Maximal clique based
Adamcsek et al. (2006)	k-clique based
Zhang et al. (2006a)	CPM based
Bader et al. (2003a)	Local neighborhood density based
Li et al. (2005)	local clique merging
Bu et al. (2003)	Quasi-clique based
Gavin et al. (2006)	Core/attachments based
Leung et al. (2009)	Core/attachments based
Wu et al. (2009b)	Core/attachments based
Liu et al. (2009)	Maximal clique-based
Nepusz et al. (2012)	Cluster cohesiveness based
Based on clustering	
Pei et al. (2007)	Subgraph refinement based
Kenley et al. (2011)	Graph entropy-based
Lian et al. (2010)	Graph entropy-based
Chen et al. (2011)	Graph entropy-based
Enright et al. (2002)	Flow simulation
Srihari et al. (2009)	Flow simulation
Satuluri et al. (2010)	Flow simulation
Shih et al. (2012)	Iterative flow simulation
Asur et al. (2007)	Ensemble clustering based
Greene et al. (2008)	Ensemble clustering based
Navlakha et al. (2010a)	Integer linear programming based
Altaf-Ul-Amin et al. (2006)	Local density and periphery based
Li et al. (2008b)	Local density and distance based
Jiang et al. (2010)	Weighted density based
Girvan et al. (2002)	Edge betweenness based
Luo et al. (2006)	Hierarchical module based
Mete et al. (2008)	Structural clustering based on common neighbors
Rivera et al. (2010)	Based on shared neighborhood
Wang et al. (2011)	Based on local metric of edge clustering
Li et al. (2008a) and Li et al. (2009)	Based on local metric of edge clustering
Rives et al. (2003)	Agglomerative average-linkage clustering technique based on shortest path
Arnau et al. (2004)	Agglomerative hierarchical clustering based on shortest path
Kim et al. (2010)	Module inference by parametriclocal modularity
Zhang et al. (2010)	Modularity density based method
Ruan et al. (2008)	Based on spectral graph partitioning and a local search subroutine
Li et al. (2010)	Based on functional similarity between the interacting proteins
Li et al. (2007)	Based on dense neighborhood extraction using connectivity and confidence features
Integrating multiple data sources	
Tanay et al. (2004)	Biclustering algorithm by integrating PPI, gene expression, transcription factor binding, and phenotypic sensitivity
Snel et al. (2002)	Based on network of gene associations
Chen et al. (2006c)	Edge betweenness based
King et al. (2004)	Local search cost based
Lubovac et al. (2006)	Neighborhood search based on functional similarity
Xu et al. (2011)	Based on GO annotation-based semantic similarity
Shi et al. (2010)	multi-layer neural network
Georgii et al. (2009)	dense module enumeration based
Luo et al. (2010a)	Baesd on detection of conditional co-regulated protein complexes
Xiong et al. (2005)	Based on hyper-clique patterns and GO annotations
Cho et al. (2007)	Simulate information Flow based using GO annotations
Cho et al. (2008)	Functional flow patterns based
Voevodski et al. (2009)	Local protein community and conductance based
Dotan-Cohen et al. (2007)	Hierarchical clustering-based method using tree-Sniping
Navlakha et al. (2010b)	Hierarchical clustering-based method using variation of certain information metrics
Segal et al. (2003)	Gene expression-based
Maraziotis et al. (2007)	Density based
Zheng et al. (2008)	Bayesian network based
Ulitsky et al. (2007)	Confidence-based methodology for detecting co-expressed sub-networks
Zhang et al. (2006b)	Kernel-based integration of PPI and protein functional annotation
Zhang et al. (2008a)	Maximal clique-based by integrating affinity scores and PPI data
Jansen et al. (2002)	Co-expression data-based
Ideker et al. (2002)	mRNA expression data-based
Dynamic PPINs	
Mucha et al. (2010)	Generalized Laplacian dynamic approach
Jin et al. (2007)	Mining algorithm
Tang et al. (2011)	Markov clustering-based
Lu et al. (2006)	Hierarchical clustering-based

to the proposal of many core-attachment approaches (Leung et al., 2009; Yu et al., 2011) and star-like structures (Chen et al., 2012; Vogiatzis et al., 2019) to identify protein complexes of interest or importance. Note here that with the term cliques we refer to fully connected subgraphs, while induced star (and star-like) structures are special cases of core-attachment structures with a center node surrounded by nodes that are pairwise not connected by an edge.

Spirin et al. (2003) proposed three algorithms to detect communities in PPINs of yeast and were able to find more than 50 real-life protein complexes, including some that were still uncharacterized at that time. Their first methodology is based on a scheme of complete enumeration of all cliques in the network, with the note that this could only be applied in practice in sparser graphs. The second proposed approach employs *superparamagnetic clustering*, a technique that was successfully applied in a series of clustering problems (Blatt et al., 1996). Last, their third method aims to maximize the edge density of the obtained cluster/subgraph. The maximization problem is then solved using a typical Monte Carlo procedure. The results from this work are very significant, as they strongly support the modular structure of PPINs.

CFinder, proposed by Adamcsek et al. (2006), is another algorithm used to locate densely connected groups of nodes. The method searches for such overlapping and densely interconnected groups of proteins using the clique percolation method (Derényi et al., 2005) to identify k-clique percolation clusters of interest, or modules. This identification is done through defining the adjacency of two cliques of size k as their sharing of exactly $k-1$ nodes. The obtained cluster then contains all nodes reachable through adjacent cliques as well as their edges. The performance of CFinder is highly dependent on the value of k (recommended between 4 and 6), with bigger values resulting in smaller sized groups. Later, Zhang et al. (2006a) also employed such a clique percolation method and coupled it with an iterative process that takes into consideration the size of the obtained modules. Based on an observation by Spirin et al. (2003) that most protein modules are of moderate size, the technique proceeds with getting the modules with a bigger size and applying the clique percolation method to them for $k' = k + 1$: this process continues until all obtained modules are smaller than a (given) module size parameter.

Seeing as the performance of computational methods in PPINs is highly dependent on the quality (accuracy and reliability) of the data (Li et al., 2010), a series of approaches have been proposed that leverage these issues with appropriately weighing vertices and edges. We begin our discussion of these methods with *molecular complex detection* (MCODE), proposed by Bader et al. (2003a). MCODE is smartly weighing proteins based on their local density, defined as the density of the highest local k-core in the neighborhood of the given protein, to provide us with a set of non-overlapping high-density clusters. After the scores for each protein have been calculated, the process starts from the protein with the highest score, and a cluster is built by recursively moving outward and including all neighboring vertices whose weight is bigger than a given threshold. Finally, MCODE refines the obtained clusters through (i) a "haircut" process to filter out complexes without 2-cores, and (ii) a "fluff" process to add extra vertices to clusters.

Another method is *local clique merging*, that was proposed by Li et al. (2005), and is also based on identifying complexes using cliques. Initially, their method generates the local neighborhood graph for each of the proteins in the PPIN, and then iteratively removes neighbors that are loosely connected. By doing so, the method "mines" local cliques for every protein. Then, the method proceeds to merge two local cliques based on their affinity, defined as the ratio between the square of the size of their intersection and the product of their cardinalities. The authors also show that their method is not sensitive to incomplete protein interaction. Bu et al. (2003) applied spectral analysis to PPINs in an effort to study their hidden topological structures. They were able to show that for every eigenvector of the adjacency matrix (with two proteins being adjacent when they interact with one another), those stemming from a positive eigenvalue reveal proteins that densely interact (forming a quasi-clique). On the other hand, those stemming from negative eigenvalues contain proteins that form a quasi-bipartite graph, and hence they interact more "outside" rather than "inside".

Requiring the presence of all possible links between nodes of a clique can prove restrictive for certain applications. This is the insight behind a series of clique relaxations that were presented by Pattillo et al. (2013). In the motivation for the work, the authors immediately provided an example from protein-protein interaction networks: they showed that several proteins form complexes that resemble clique relaxations, including k-cores and quasi-cliques. With the term γ-quasi-clique, the authors mean complexes where instead of all nodes being pairwise adjacent, instead a fraction of those edges is present. This fraction is denoted by γ. All clique relaxations offered in this study are shown to relax at least one of the main parameters that define a clique: for example, a γ-quasi-clique allows for missing edges, a k-club (a set of nodes whose induced subgraph has a diameter of k) allows for the distance between two nodes to be higher than 1, and a k-plex allows a node to be adjacent to fewer than all the other nodes in the complex.

On the other end of cliques, we find the distinction of proteins in "core" and "attachments" groups. Their definitions, due to Gavin et al. (2006), are highly co-expressed and of similar functionality proteins that serve as the central functional parts of protein complexes, and surrounding proteins with an assistive functionality, respectively. Core proteins at each protein complex are expected to possess relatively more common neighbors. Furthermore, we can also anticipate the set of all core proteins to be disjoint. With these assumptions in mind, Leung et al. (2009) developed the *CORE* method to identify protein complexes through searching for their core and attachment counterparts separately. CORE first predicts a set of disjoint candidates and calculates a p-score as a measure of how likely a candidate is to be serving as the core of a protein complex. This measure is calculated by taking into consideration the number of interactions between the potential core and the rest of the proteome. After selecting a group to serve as a core, all non-candidate proteins that are adjacent to the selected core are added as attachment proteins. Building on this idea, Wu et al. (2009b) proposed a core/attachment-based method, referred to as *COACH*, that takes a similar route in two stages: first potential cores are identified based on each protein neighborhood, and then the core is grown outwards to detect its attachments.

Continuing with proposing clustering algorithms on weighted PPINs, Liu et al. (2009) developed a clustering techniques based on iteratively merging maximal cliques to identify protein complexes. Their iterative scoring method (called *AdjstCD*) assigns weights to pairs of proteins with higher weights indicating higher confidence in the existence of their interaction. Their clustering algorithm begins by enumerating all maximal cliques in the PPIN, and follows it by assigning a score to each clique based on its weighted density. Then, the cliques are sorted in decreasing order of their scores and are iteratively merged or removed, based on their inter-connectivity scores. Clique merging to identify clusters is not new: we already mentioned the CFinder (Adamcsek et al., 2006) and local clique merging (Li et al., 2005) algorithms earlier in this subsection. However, the advantage of the method by Liu et al. (2009) is its ability to work on weighted networks as well as its success in also finding relatively low density protein complexes.

Still, one of the main concerns has to do with the overlapping nature of most protein complexes, as multiple proteins participate in more than one functions. To overcome this, Nepusz et al. (2012) proposed clustering with overlapping neighborhood expansion (*ClusterONE*), for detecting overlapping protein complexes in weighted PPINs. The method works with seeding and a greedy growth mechanism, similar to MCODE (Bader et al., 2003a). Two structural properties are used to determine how likely a group of proteins is to represent a protein complex: they are the existence of reliable interactions within the nodes of the cluster, and a good separation (lower reliability of interactions) outside the cluster. In such a way, the authors can define a cohesiveness score for a group of proteins by summing all weighted edges within and outside the cluster. Two clusters obtained by iteratively and greedily growing them can then be merged, if the resulting subgraph is still above a threshold. ClusterONE was also shown in the same study to have a superior performance in identifying overlapping complexes than MCODE, Markov clustering techniques, and maximal clique merging.

4.2.2. Identifying protein complexes by clustering techniques

Since proteins tend to exhibit strong interactions with other proteins participating in the same complex, and weak interactions with proteins outside the complex (Pereira-Leal et al., 2006), it comes as no surprise that there are many algorithms to discover subgraphs possessing such properties in PPINs (Pereira-Leal et al., 2004b; King et al., 2004). A heuristic approach to identify such densely connected inside, but sparsely connected outside clusters, was devised by Pei et al.

(2007). Therein, a “seed and refine” approach is proposed to identify promising dense and small subgraphs that can be further refined or grown. This method was able to tackle the issues inherent to overlapping cluster detection, as well as the presence of noisy edges, in an efficient way.

Graph entropy is a measure, borrowed from information theory, that reveals the structural complexity of a graph: it is defined in such a way so as a loss of entropy would signal an increase in the modularity of the graph. It was employed in multiple instances in PPINs, including the works by Kenley et al. (2011), Lian et al. (2010), and Chen et al. (2011). In the first two approaches, a random protein is selected as the seed vertex and its neighborhood as the seed cluster. The process then iteratively decides to add or remove vertices on the border of the cluster to minimize its graph entropy metric. Once a cluster of (locally) minimum graph entropy has been built, a new protein is selected as the seed and the process continues until every protein is assigned to a cluster and no more proteins can serve as a seed. In the work of Chen et al. (2011), an improved version appears where the seed cluster is more restrictive than the seed protein neighborhood: instead it can be selected as a clique of size k around the seed vertex (e.g., a triangle for a clique of size 3). The logic is that single seed proteins might be insufficient to produce complexes of any biological meaning; on the other hand, enhancing the seed cluster selection seems to improve the quality of the obtained clusters.

Markov Clustering (MCL), proposed by Enright et al. (2002), is another, very well-studied, and broadly applied clustering technique. In a comparison of four clustering methods for extracting protein complexes from an unweighted PPIN, presented by Brohee et al. (2006), MCL, restricted neighborhood search clustering, molecular complex detection (MCODE, which is described in the following subsection), and superparamagnetic clustering were put to the test in six high-throughput PPINs. The results of this analysis pointed towards the superiority of MCL. However, one of its caveats, that could potentially prove detrimental is its sensitivity to noisy input which leads to producing noisy clusters. It is for that reason, that *MCL-CA* (and its weighted counterpart, *MCL-CAw*), were proposed (Srihari et al., 2009). Computational results indicate that MCL-CAw can more accurately extract better protein complexes that simple MCL.

Seeing as Markov clustering is both fast and scalable, it is a natural fit for detecting protein complexes in weighted PPINs. However, it may generate imbalanced clusters, and hence Satuluri et al. (2010) proposed the *MLR-MCL* algorithm in an attempt to build more balanced clusters by modifying the original Markov clustering algorithm. Another extension of Markov clustering is to allow for the existence of overlapping clusters. To that extent, Shih et al. (2012) presented the *SR-MCL* algorithm that addressed the limitations of other Markov clustering-based approaches and enabled the detection of protein complexes with shared proteins. Pereira-Leal et al. (2004b) also used a Markov clustering-based technique to successfully identify overlapping functional modules.

We now continue with multiple clustering-based approaches. The insight here is that generating multiple, independent, and with varying starting conditions, clusters and then combining them to obtain a single, comprehensive, "consensus" cluster should yield better leveraged results. This is the idea behind the ensemble clustering framework by Asur et al. (2007), who combined different topology-based distance metrics with three graph clustering approaches to finally obtain six different baseline clustering methods. These methods were independently run on and a consensus method (based on principal component analysis) was employed to deliver the consensus clustering output. Allowing for overlapping clusters is important, and is allowed in the non-negative matrix factorization-based ensemble method by Greene et al. (2008). Last, the MOD-ILP algorithm (Navlakha et al., 2010a) exploits the multiple near-optimal clusterings that exist to explore the dynamics of network clusters. To do that, the problem is formulated as integer linear program. Overall, this last approach provides us with an ensemble of distinct and highly modular network partitions.

Smart edge and vertex weighting schemes can also improve several well-known clustering approaches. Such an idea was employed by Altaf-Ul-Amin et al. (2006) in the development of *DPCLus*, based on subgraph edge density and the cluster property of every node. This last notion is defined as the ratio of the cardinality of the set of edges from a node to a given cluster over the product of the cluster density and its size. The insight is that the larger the cluster property is for a given node, the closer it resembles (topologically) the rest of the cluster and thus the higher the chance it is part of it. In the same logic, *IPCA* was later proposed with a new set of rules for expanding clusters and weighing vertices (Li et al., 2008b). Subgraphs obtained by IPCA have a smaller diameter (or, a smaller average intracluster nodal distance), and hence satisfy different cluster connectivity-density properties.

With PPI data becoming bigger, it is necessary for clustering techniques to be able to handle large-scale PPINs in a scalable and efficient manner. Such an approach is *SPICi*, proposed by Jiang et al. (2010), which is able to handle the computational complexity of clustering large-scale, weighted PPINs. SPICi works by selecting a pair of seed proteins and builds a cluster by selecting, at each point, the unclustered protein with the highest weighted sum of edges towards the cluster, if that sum is above a given threshold. Moreover, the protein is only added to the cluster, if it keeps the cluster density high enough. It is clear from this description that these two checks are controlled by user parameters, namely a threshold for admissible node to cluster weighted sum, and a threshold for the lowest allowable cluster density.

We now focus the discussion towards hierarchical clustering techniques. Hierarchical clustering is prominent in PPINs, both weighted and unweighted. It is the traditional method of choice for detecting communities in a wide variety of networks, and of course, biological networks and PPINs are no exception. In the literature, we mainly use two types of hierarchical clustering, namely agglomerative and divisive algorithms. Agglomerative algorithms calculate a weight for every pair of nodes, that captures how closely connected these two nodes are. The nodes are then naturally organized into a hierarchy beginning from the stronger weight pair and iterating over all of them. In contrast, divisive algorithms work in the opposite way: the whole network is a cluster that is divided into components through the use of cuts. A good cut would be one that only affects intra-community edges, but does not touch inter-community ones. For an excellent overview we refer the interested reader to the work of Radicchi et al. (2004).

Designing a good hierarchical clustering algorithm, then, is akin to making two decisions: (i) how to define a criterion to select edges when merging or splitting clusters, and (ii) how to define a measure to assess the quality of the outputted clusters. In their fundamental work, Girvan et al. (2002) proposed a divisive algorithm to detect communities in social and biological networks. Their approach was based on *edge betweenness*. More specifically, Girvan et al. (2002) generalized the classical node betweenness metric to edges: a higher edge betweenness implies that the edge is used by multiple shortest paths, and hence it is prone to connect two distinct communities.

Luo et al. (2006) proposed an extension to the notion of degree from a single node to a subgraph and defined a new agglomerative algorithm, called *MoNet*, by combining the algorithm of Girvan et al. (2002) with a new definition of modularity. Based on their computational results, it can be concluded that MoNet complexes demonstrate stronger co-clustering of related genes; moreover, their clusters are more robust. One more note has to do with calculating betweenness in large-scale PPINs. This is an expensive calculation (with a quadratic running time), and hence impractical for many instances. To tackle that, Mete et al. (2008) proposed a linear time algorithm called *SCAN* that employs the notion of indirect connections for detecting modules. The basic idea of the algorithm is that the similarity between two proteins can stems from their shared neighbors. Last, *NeMo* (Rivera et al., 2010) is one more shared neighbor-based agglomerative hierarchical clustering technique that can be used to predict the association between two proteins.

The high presence rates of errors and noise in high-throughput PPI data are also problematic to the application of hierarchical clustering methods, seeing as they are sensitive to noisy networks. Thus, Wang et al. (2011) developed a fast hierarchical clustering algorithm by using the local metric of edge clustering. Their computational experiments revealed that applying this local metric instead of a global, more expensive, metric of edge-betweenness not only enhanced the efficiency, but also increased the robustness to increasing levels of false positives. Presenting a new definition of edge clustering coefficient and using a weighted PPIN, Li et al. (2008a) and Li et al. (2009) propose an even faster agglomerative algorithm, that is computationally efficient and that can be applied in very large-scale PPINs.

Continuing with hierarchical clustering methods, Rives et al. (2003) investigated the following three ideas. First, the shortest path between two nodes is more likely to be the one used to associate two proteins. Secondly, the vector of shortest paths from a vertex to all other vertices constitutes a unique vertex profile. Last, a protein complex would consist of vertices with similar shortest path profiles. Using the shortest path matrix and defining the association matrix as $1/d^2$ and applying an agglomerative average-linkage clustering technique revealed the modular organization of the yeast signaling network. In a similar, distance-related, note, Arnau et al. (2004) proposed a clustering software called *UVCLUSTER*, that iteratively explores distance datasets using a agglomerative hierarchical clustering. This technique converts the shortest distance between any two proteins of a PPIN into secondary distances that measure the strength of connection between each pair of proteins at the presence of all group proteins. In all the techniques discussed so far, a common strength is their capability to build a complete hierarchy of clusters corresponding to any PPIN that enables the analysis of clustering in multiple levels with a full coverage of the PPIN. However, as a deterrent to their application, these methods do not allow for overlapping clusters (Bhowmick et al., 2016).

Regarding the second important decision to be made when designing a hierarchical clustering algorithm, multiple quantitative measures have been proposed to evaluate the discovered clusters. Modularity maximization is probably the most widely used community structure detection model for networks (Newman, 2016). Modularity, defined as the number of connections within a specific community minus the expected number of connections within the same community when the connections are random (Newman et al., 2004), has been a prominent measure in evaluating hierarchical clustering. Global modularity is limited in its capability to identify small cardinality complexes in large-scale networks (Fortunato et al., 2007). Thus, a localized modularity counterpart was proposed by Muff et al. (2005). Since modularity and localized modularity portray two extreme case measures to compute connectivity in a PPIN, and it could be that neither provides us with high quality protein complexes, Kim et al. (2010) presented *miPALM* (module inference by parametric local modularity) to extract protein complexes. They predicted 138 new modules in the yeast network by applying their method.

Modularity in PPINs has been such an important property; it is for that reason that a wide variety of algorithms has been proposed over the last decade to optimize modularity. As an example, maximizing modularity via simulated annealing is common, but since the possible divisions of a network grows to be exponentially large, especially for huge networks, maximizing modularity is expected to be intractable. Hence, Zhang et al. (2010) developed a metric of modularity density as an alternative measure to evaluate the quality of clusters which were discovered earlier by Li et al. (2008c). Clusters with larger values of modular density are also higher quality communities. However, even optimizing modular density instead of modularity does not come with a tractable algorithm, hence the authors resorted again to the use of simulated annealing. The obtained results make the case for using modular density as it provides better quality protein complexes. Ruan et al. (2008) proposed a *QCUT* algorithm that combines spectral graph partitioning and a local search subroutine to optimize modularity. A second algorithm, *HQCUT*, was also developed to recursively divide communities into smaller ones. Similarly to the HQCUT algorithm, Sun et al. (2011) proposed a different iterative network partitioning algorithm.

Proteins in the same complex usually share the same or similar function properties. Hence, combining such functional annotations with topological information could prove a useful method to detect protein complexes in PPINs. One way to incorporate this functional information is by properly scoring interactions based on the functional similarity between the interacting proteins (Li et al., 2010). An example of such an integration is the *dense neighborhood extraction using connectivity and confidence features* (DECAFF) algorithm (Li et al., 2007), which is an extension of the local clique merging algorithm. DECAFF first extracts local dense neighborhoods on top of local cliques for each protein in the network using a hub-removal procedure, and then merges these neighborhoods based on their affinity to form maximal dense sub-graphs that correspond to protein complexes. By extracting both dense subgraphs and cliques (instead of only cliques) the algorithm aims to balance the possible data incompleteness (missing interactions) of the PPIN. To mitigate false complex detection, the authors model the PPIN as a weighted network, with each edge weight representing the interaction reliability. Upon detection of the protein clusters, complexes consisting of interactions of low weight are filtered out.

4.2.3. Identifying protein complexes by multiple data integration

One of the major issues that keeps coming back is the fact that there exists no single computational approach with both sensitivity and specificity at 100%, i.e., no false negatives or positives (Chatr-aryamontri et al., 2008). The intrinsic noisiness of high-throughput methods as well as the complex connectivity of PPINs requires an integration of data from multiple sources to improve the effectiveness of module detection algorithms (Tong et al., 2002). As we saw in Chapter 3, the world of high-throughput protein protein interactions does not stop at Y2H and AP/MS; instead, we have information on the reliability of these interactions (Deane et al., 2002), gene expression and protein evolution (Pazos et al., 2008), gene ontology (Ashburner et al., 2000), among other biological information.

In addition to high-throughput experimental methods such as Y2H and AP/MS there are other source of information about PPIs such as the reliability of experiments (Deane et al., 2002), the gene expression profile (Komurov et al., 2007; Li et al., 2012b; Akker et al., 2011), the gene ontology terms (Ashburner et al., 2000), and the subcellular localization annotations (Friedel et al., 2008) that can be used to evaluate the reliability of PPIs and their biological properties. Hakes et al. (2008) underlined that although there are problems with high noise rate in PPIs, the degree of biases in applying reliable data can be as problematic as the data quality, and hence alter the underlying structure of PPINs. Considering the fact that different experimental methods do not have the same reliability, one way to assess the reliability of interactions is the frequency of presence in different datasets. This implies that interactions observed at multiple datasets are more likely to be true than those that have only been observed once. Thus, weighting interactions according to the different experiments they are extracted from and the different times they are reported by a specific method, one can assess the reliability of PPIs (Chen et al., 2013). Chatr-aryamontri et al. (2008) stated that using appropriate scoring threshold for the weighted PPIN that produced by reflecting all experimental supports can led to more biologically relevant conclusions drawn from computational methods. Tan et al. (2010) proposed an algorithm to combine the scores of PPIs from different experimental methods to identify the core components of complexes.

It is widely used that genomic associations between genes lead to functional associations between the corresponding proteins. Besides, there is a direct relationship between the strength of these associations: co-occurring genes in a diverse set of species are more likely to physically interact than genes that occur together in just two species, and protein that are connected through gene fusion or gene order conservation are more likely to be member of the same complex than those that are merely encoded in the same genomes (Snel et al., 2002). Various genomic contexts such as gene expression profiles, gene fusions, phenotype data, gene co-occurrences, and transcription factor binding data have been applied to predict functional connections (Huynen et al., 2000). Tanay et al. (2004) proposed a biclustering algorithm by integrating a highly heterogeneous set of experimental data including protein interactions, gene expression, transcription factor binding, and phenotypic sensitivity. They used a weighted bipartite graph to model all genomic data with nodes representing genes on one side and properties of genes or encoded proteins by them on the other side. In this model, an edge between gene node g and property node v with a weight of w represents that gene g possesses that property v with probability w. In that work, the authors provided evidence for the hierarchical modular organization of yeast, and showed that smaller modules can be incorporated into supermodules. Finally, they also predicted the function of more than 800 uncharacterized (at that time) yeast genes.

Different kinds of associations between genes are used to identify functional associations between proteins. For example, gene fusion, conservation of gene order, and co-occurrence of genes (phylogenetic profiles) are common (Ge et al., 2001; Dittrich et al., 2008). Snel et al. (2002) used conserved gene order to connect genes together and form a network of gene associations in which nodes are orthologous groups and edges represent the genomic relationships between them. In order to identify such orthologous groups of genes, they clustered the network and found functional categories for groups by comparing them to the clusters of an orthologous groups (COG) database. Among different types of genomic information used, gene expression profiles are the most common complementary information for data integration. Chen et al. (2006c) used microarray expression profiles to add weights to PPINs: they introduced the idea of "non-redundance" into the calculation of edge-betweenness to resolve the imbalanced partitioning observed.

Cost-based methods, that assign costs to a partition in the network by defining a proper cost function, are prominent, too. As an example, the *restricted neighborhood search clustering* (RNSC) algorithm combines both topology-based and gene ontology information in order to detect protein clusters (King et al., 2004). The method is randomized and resembles the well-known tabu search methodology. The method is initialized randomly and then proceeds to transfer nodes between clusters with the goal of minimizing the total clustering cost. Clearly, such moves can lead restricted neighborhood search clustering to converge to a local minimum – to avoid that, the authors recommend diversification, multiple start points, as well as prohibiting certain moves. As the resulting complexes can be of average quality, a series of post-processing techniques are used, ranging from the discarding of smaller sized clusters as well as lower density clusters, to using known biological data from protein functionality to calculate the functional homogeneity of the obtained clusters (and discard those falling below a cut-off threshold). However, it is unclear how the method could be extended to account for the overlapping nature of some protein clusters or the existence of interaction weights.

The Gene Ontology (GO) database and informatics resource (Gene Ontology Consortium, 2001; Gene Ontology Consortium, 2004) contains information that spans molecular functions, biological processes, and cellular components. The GO is used as a de facto standard for annotating gene products (Gene Ontology Consortium, 2001; Deng et al., 2004). Some methods have been proposed to predict the function of proteins from PPI graphs using GO (Karaoz et al., 2004; Lord et al., 2003). These GO-based functional annotations are another source of information which can be used to assign weights to PPINs (Ashburner et al., 2000). Because of the significant correlation between GO-based semantic similarity of gene products and their sequence similarity, GO was used for measuring the similarity between gene products. Combining new functional properties with topological information, Lubovac et al. (2006) proposed two network weighted measures to analyze PPINs referred to as weighted average nearest-neighbors degree and weighted clustering coefficient. To calculate these measures, they developed a GO-driven semantic similarity method that used interaction weights as a measure of interaction strength. Then, to prove the usefulness of their measures they proposed their SWEMODE (*Semantic WEights for MODule Elucidation*) algorithm, as a method to identify modules containing functionally similar proteins. SWEMODE begins by ranking proteins according to their weighted neighborhood cohesiveness, and considers the nodes with the highest rank as seeds for candidate modules. Then, the algorithm proceeds to search the neighborhood of each seed protein iteratively, to find densely connected proteins which have high functional similarity. Xu et al. (2011) developed a GO annotation-based semantic similarity method to assign reliability values to every interaction. The OIIP algorithm uses the vertex weight as a measure for selecting seeds. It sorts vertices in a non-increasing order and then starts with the most highly ranked vertex and grows it to a cluster; the nodes that participate in that cluster are then removed from consideration and then next highly ranked vertex is picked. OIIP is similar to IPCA with the difference that it computes vertex weight based on an updated weighted network, while IPCA calculates weight of a vertex based on the original network.

Another approach that uses a GO-based similarity method to assign weights to interaction to boost the reliability of PPINs is due to Shi et al. (2010). Considering the fact that modules may overlap with each other and protein complexes are not necessarily dense subgraphs, they proposed a multi-layer neural network by integrating both topological and biological properties of proteins to identify protein complexes. A dense module enumeration algorithm was developed by Georgii et al. (2009). It enumerates all densely connected modules that satisfy a density criterion in a weighted PPIN. This enumeration approach has the ability to integrate additional constraints from phenotypic profiles, evolutionary conservation, and tissue-specific modules, improving its performance. Recent studies indicate that there is a strong correlation between gene expression, transcription regulation, and protein interactions (Luo et al., 2010a). In that same work, Luo et al. (2010a) integrated gene expression data, transcription regulation data, and PPI data to detect successfully a specific kind of protein complexes, referred to as conditional co-regulated protein complexes.

Xiong et al. (2005) proposed an approach based on hyper-clique patterns to extract functional modules from protein complexes. Exploiting the GO annotations, they found that proteins within the same pattern are more likely to perform the same function and to participate in the same biological process. Furthermore, a 3-D structural view of proteins within a detected patten revealed that a hyper-clique can be shared by different protein complexes performing various functions and several hyper-cliques with different functions can participate in the same protein complex. Cho et al. (2007) proposed a flow-based approach to identify overlapping modules in a weighted PPIN. They developed novel semantic similarity and semantic interactivity metrics that used GO annotations to assign reliability values to protein interactions. Given this weighted PPIN, their method starts with selecting a small number of informative proteins based on the weighted degree, and then simulates the information flow from each informative protein through the whole network to determine proteins that are functionally influenced by that protein. In a follow-up study, Cho et al. (2008) extended the informative protein-based approach by introducing functional flow patters as a sequence of the functional influence of a source protein to a set of target proteins. Then, they exploited a pattern-based clustering algorithm (Wang et al., 2002) to identify their final modules. Voevodski et al. (2009) developed local protein community finder which used local clustering algorithms called Nibble and PageRank-Nibble (Andersen et al., 2006; Spielman et al., 2008). Starting from a queried protein, their method looks for a cluster of good conductance in the protein's neighborhood. To evaluate the functional coherence of identified clusters, they used functional distances that are derived using the GO classification scheme.

Another group of hierarchical clustering methods, similar to the ones discussed earlier, is annotation-driven clustering. To improve the biological significance of the computationally obtained clusters, Dotan-Cohen et al. (2007) proposed a novel partitioning scheme, called Tree-Sniping, that partitions the hierarchical tree by snipping-cutting edges at different levels so as to maximize the consistency of induced clusters. To achieve that, background data such as GO annotations and functional classifications were used. Navlakha et al. (2010b) also presented *VI-Cut* to decompose a hierarchical tree into clusters with the objective to match the resulting clusters to a set of known annotations based on their variation of certain information metrics.

In the work by Segal et al. (2003), gene expression and protein interaction data were integrated within a unified probabilistic model to detect group of genes that are co-expressed and whose proteins interact in the PPIN. Their model was based on the assumption that genes participating in the same pathway are activate at the same time and thus should exhibit similar gene expression profiles and, thus, the proteins of genes that coordinate to perform a specific function often interact in PPIN. Using gene expression profiles, Maraziotis et al. (2007) created a weighted PPIN and then applied their *DMSP* (detect module from seed protein) method to construct functional modules starting from a “seed” protein. The method then expands the neighborhood, and the algorithm detects functional modules in two steps. In the first step, and based on the density of the kernel and its weighted internal and external degrees, a subset of seed neighbors is selected as the kernel neighborhood. In the second step, adjacent proteins are added to the selected kernel iteratively based on a specified criterion. Zheng et al. (2008) used a Bayesian network data integration approach to integrate seven genomic features and four experimental interaction datasets to generate a more reliable PPIN. Then they used Markov clustering to identify protein complexes. Another gene expression-based approach Ulitsky et al. (2009) developed CEZANNE (co-expression zone analysis using networks, a part of the MATISSE software by Ulitsky et al. (2007)) a novel confidence-based methodology for detecting co-expressed sub-networks.

Zhang et al. (2006b) used a kernel-based integration of PPI data and protein functional annotation data to develop a protein protein relationship network. They then applied MCODE on both the original PPIN and the relationship network, and they showed that using functional annotation data improved the complex detection abilities of MCODE. Zhang et al. (2008a) introduced a multi-step method to identify protein complexes from Mass Spectrometry (MS) data: the main assumption of their method is that the similarity of co-purification patterns of two proteins derived from MS can serve as an indicator for evaluating their interaction affinity. They proceeded to create a PPIN by integrating affinity scores and a knowledge-guided information theoretic thresholding method to find all maximal cliques as possible candidates of protein complexes. Finally, they proposed an algorithm to merge highly overlapped cliques into a single protein complex and divide the set of proteins of each complex to core and attachment components.

Last, studying known protein complexes, Jansen et al. (2002) found that proteins participating in a complex exhibit significant co-expression, both in terms of similarities of absolute mRNA levels and their expression profiles. After dividing the available protein complexes to permanent and transient, it was concluded that, contrary to transient, permanent complexes have higher co-expression levels. Usine mRNA expression data (and integrating the PPIN topology), Ideker et al. (2002) proposed a method to detect connected regions of the network that indicate considerable changes of expression over a specific set of conditions. Such subnetworks are identified based on a scoring mechanism that relied on the integration of network topology with mRNA expression data using simulated annealing.

4.2.4. Identifying protein complexes in dynamic PPINs

Modeling PPINs using static information is a simple and efficient methodology to understand how proteins interact. However, such static networks hide temporal information which are crucial to our understanding of the functional properties of proteins and complexes (Pereira-Leal et al., 2006; Zhang et al., 2016d; Ideker et al., 2001; Przytycka et al., 2010). All protein complex detection methods mentioned up to now focused on static PPINs. PPINs are dynamic and change over time, depening on surrounding environmental conditions, biotic and abiotic stresses, and the different stages of cell cycle (Chen et al., 2013; Tang et al., 2011; Terentiev et al., 2009). In order to fully understand the intricacies of how cellular life functions and has been shaped during evolution, it is important to study the dynamics and evolution of interactions both at the atomic and network level (Levy et al., 2008).

Currently available high-throughput PPI data are usually unable to reflect the dynamic properties of PPINs (Hegde et al., 2008). Researchers try to elucidate the dynamic structure of PPINs by projecting additional information such as gene expression data (Terentiev et al., 2009) and sub-cellular localization annotations (Lichtenberg et al., 2005) into PPINs. Lichtenberg et al. (2005) integrated PPIs with the timing of the transcription of genes within the cell cycle. They filtered interactions in two stages: assigning topology-based confidence scores to each interaction and selecting high-confidence interactions, and using sub-cellular localization to remove interactions between proteins annotated to incompatible sections. They constructed a network containing both constitutively expressed (static) and periodically expressed (dynamic) proteins that interacted with one another, and found that most protein complexes contain both kinds of proteins.

In their work, Lu et al. (2007) integrated a biological interaction network with gene expression data to investigate the relationship between the network and the differentially expressed genes. Their results demonstrated that genes with low connectivity tend to have a higher level of change in gene expression, whereas hub proteins (nodes with high connectivity) and superhubs (nodes that link hubs) are more likely to have a lower level of change in expression. Using gene ontology data to analyze the correlation between topology and molecular functions, they found that hubs and superhubs have crucial biological functions, when compared to more peripheral nodes. Dynamic features of PPINs can be extracted by integrating gene co-expression information with PPIs (Han et al., 2004b; Ekman et al., 2006). Lin et al. (2010) integrated PPIN, gene expression profiles, and GO annotations to construct co-expressed PPINs under specific conditions in order to discover dynamic functional modules. Their results indicated that proteins encoded by differentially expressed genes under specific biological conditions are also topologically significant proteins.

Han et al. (2004b) studied the biological roles of hub proteins in dynamic PPINs as far as both time and space are concerned. In their work, two types of hubs are distinguished: "party" hubs, that interact with their partner proteins simultaneously, and "date" hubs, that bind their partners at different times or locations. The authors then used gene expression and protein localization data to investigate the temporal characteristics of hub proteins and their partners. Both additional sources of data indicated towards a diversity in time and spatial distribution for the partner proteins of "date" hubs, more than partners of "party" hubs. In summary, "date" hubs were shown to connect modules, whereas "party" hubs function inside modules.

Later, Komurov et al. (2007) proposed the existence of “family” hubs, as proteins that tend to interact with their neighbors, but are not highly co-expressed with them. Thus, “family” and “party” hubs form static and dynamic modules, respectively, while “date” hubs organize them into a network. The conclusions drawn from this study point that static modules increase the robustness of the cell against genetic perturbations or protein expression noise, whereas dynamic modules regulate the cell behavior under varying conditions. The study of the dynamic structure of human interactome to determine whether and which organizational changes can lead to oncogenesis was done by Taylor et al. (2009). Their work showed that intermodular hub proteins have expression correlation with their neighbors in a tissue-restricted manner, while intramodular hub proteins are co-expressed with their interacting partners in all, or most, tissues.

When it comes to combining the community detection approaches discussed earlier and the dynamic nature of PPINs, the problems becomes complicated. Hence, Mucha et al. (2010) proposed a generalized Laplacian dynamic approach, which allows for the simultaneous study of quality functions related to community structures across multiple time points. This method works on multislice networks, which are obtained by combining multiple adjacency matrices, and uses a quality function to determine the underlying community structures. Similarly to the methods applied in static networks, they proposed comparing the number of intra-community edges to what could be expected in a random network as a measure to quantify communities. Three types of edges are considered in their generalization: intra-slice, inter-slice between only neighboring slices, and inter-slice among all available slices. The network slices can represent variations across time or even the same network at different scales.

Dynamic modules were defined by Jin et al. (2007) as sets of proteins that form connected components in PPINs and whose expression profiles in the temporal domain possess certain properties or form specific structures. In their work, they proposed a mining algorithm to identify dynamic modules in a temporal network. They also showed that, using this mining algorithm, most of the discovered dynamic modules are functionally homogeneous. Tang et al. (2011) proposed time course protein interaction networks (TC-PINs) by integrating time series gene expression into

PPINs. After that, they employed Markov clustering to identify functional modules in three types of networks: TC-PINs, static PPINs and pseudorandom networks. Finally, GO enrichment was performed in order to further biologically analyze of modules. The results showcased that extracted functional modules from TC-PINs have clearer biological interpretations than those obtained from static PPINs and pseudorandom networks. Last, Lu et al. (2006) developed a hierarchical clustering algorithm in protein interaction data coupled with sub-cellular localization data, and expression profile data, to distinguish between protein complexes and functional modules. Then, they employed spatial and temporal information ensuring that co-localized and co-expressed protein groups are to be clustered first.

CHAPTER 5. ESSENTIALITY AND CENTRALITY

5.1. Protein essentiality

The *importance* of a protein can be qualitatively assessed by considering the consequences to the organism phenotype upon protein deletion or disruption, as also noted in the study by Song et al. (2013). This leads to the definition of *essential proteins*, which are ones that upon their corruption cause the cell to die or cease its fundamental processes (Kamath et al., 2003). Information about essential proteins have been collected over years of biological experiments and are now organized in such databases like the *Database of Essential Genes* (Zhang et al., 2004; Zhang et al., 2008b).

Another reason we are interested in protein essentiality, is that the set of essential proteins can help us understand the minimal requirements for cell life (Glass et al., 2006; Glass et al., 2009), as well as disease study and drug discovery (Clatworthy et al., 2007; Lu et al., 2014). Moreover, essential proteins provide insight in human gene morbidity: Wilson et al. (1977) showed that proteins encoded by essential genes evolve slower than their non-essential counterparts, while Kondrashov et al. (2004) proved the existence of similarities between human morbid genes and essentiality in Drosophila melanogaster (fruit fly). Identifying essential proteins experimentally is usually performed via gene knockouts, conditional knockouts, RNA inference. Such experiments are time and resource consuming. However, the advent of high throughput techniques for extracting PPI data has enabled us to use computational methods for identifying essential proteins using network analysis.

5.2. Identification of essential proteins

In this section, we discuss the network topology and other features that essential proteins possess, and that help us distinguish them from non-essential ones. We classify these methods in the following categories, each with its own subsection:

1. Topology-based;

2. Integrating multiple sources.

We now proceed with the discussion on methods that are based on studying the topology of PPINs.

5.2.1. Topology-based methods

Finding the relationship between the topological properties of an organism's PPIN and the functional features of its cells is one of the main goals of computational biology. Numerous studies have confirmed that topological prominence of a protein in a PPIN may be a good predictor of its biological importance (Giaever et al., 2002; Winzeler et al., 1999; Fraser et al., 2002; Yu et al., 2004; Batada et al., 2006; Wang et al., 2009; Hahn et al., 2004a; Yu et al., 2007). In topology-based methods, every protein in a PPIN is scored using different centrality measures; the highest ranking proteins are then declared to be essential.

Jeong et al. (2001) were among the first to recognize that there is a correlation between the number of interactions of a protein and the phenotypic effects of its removal from the network. They referred to this relation between essentiality and the number of direct neighbors in the PPIN as the *centrality-lethality rule*. Their computational results revealed that 93% of the proteins in Saccharomyces cerevisiae participate in 5 or fewer interactions, and only 21% of them are essential. On the contrary, 0.7% of the proteins have more than 15 connections, but simple removal of 62% of them proves lethal to the organism. This corroborates the insight that topologically central proteins in the network with a high number of connections are three times more likely to be essential than proteins that participate in only a small number of interactions.

The notion of marginal essentiality was later proposed by Yu et al. (2004); it was found that proteins with greater marginal essentiality are more likely to be network hubs. A multitude of topological characteristics and their correlation to essentiality were studied by Coulomb et al. (2005) in interaction networks. They showed that essentiality in yeast is only weakly related to their local connectivity, and that the physiological effects of removal are unrelated to their average degrees or to their relative distances. Besides, the lack of a clear relationship between clustering coefficients of essential and non-essential genes prevented them to draw a general conclusion about the relationship between the essentiality of a gene and its cliquishness in PPINs. Liang et al. (2007)

investigated the human and mouse proteomes, and concluded that essentiality was, as expected, positively correlated with protein connectivity in the PPIN. However, no correlation was identified in the two mammalian PPINs studied between gene duplication and gene essentiality, even though duplicate genes have higher degrees on average.

At that point, it started becoming clearer that essentiality is not usually identified by querying a local network characteristic; instead, groups of proteins should be queried. To that extent, Lin et al. (2009) defined the essentiality of a subnetwork as the ratio of essential proteins to the subnetwork cardinality. Using this ratio, the authors found that larger cliques tend to have a bigger number of essential proteins in the PPINs of Escherichia coli and Saccharomyces cerevisiae. They also showed that the ratio of essential proteins to all proteins in the maximum clique of Escherichia coli (with size 7) and Saccharomyces cerevisiae (with size 10) were equal to 86% and 90%, respectively. In a follow-up computational experiment, they also expanded the maximum clique found by merging neighboring cliques, and used it as an essential core of the PPIN that contains the higher ratio of essential proteins.

A similar question between the difference of essentiality as a local versus a global network characteristic was studied by Wuchty (2002). That study also helped recognize that degree, while a useful proxy, can be a naive classifier of essentiality. In a subsequent study by Wuchty et al. (2003), *excentricity* was used to discriminate between essential and non-essential proteins; surprisingly, in three PPINs, essential proteins did not possess more "central" locations as far as their excentricity was concerned. Estrada et al. (2005) introduced subgraph centrality in an effort to characterize the participation of every node in each subgraph of the network. It calculates the subgraph centrality of a node as the sum of closed walks, starting and ending at that node, with different lengths, and then weighting walks based on their length, such that shorter walks are more influential on the centrality of the node. Computational results indicated a superiority of this measure when compared to other, nodal measures such as degree, closeness, betweenness, and eigenvector centralities. To determine which centrality measure is more accurate to identify essential proteins in the yeast PPIN, Estrada (2006) compared degree, closeness, betweenness, eigenvector, information (Stephenson et al., 1989), and subgraph centrality. Their results corroborated that centrality metrics perform substantially better than random selection method in identifying essential proteins in yeast, with subgraph centrality outperforming other centrality measures in detecting essentiality.

At the same time, Hahn et al. (2004b) discussed evolutionary constraints in the importance of a gene. They also reported that using only topological information from the PPINs to identify highly central nodes can be misleading, as the impact of absence of even the most central nodes was limited. However, this does not mean that topological studies on PPINs are doomed: it merely implies that the results need to be further synthesized with other biological data. In a different study, three eukaryotes (yeast, fly, and worm) PPINs were examined and a relationship between the position of a node/protein in the network and its evolution rate was uncovered by Hahn et al. (2004a). What was found was that three of the most standard centrality measures (degree, betweenness, and closeness) were negatively correlated to its evolutionary rate, with betweenness having the strongest correlation. Moreover, the authors found that evolutionary rates are reduced for essential genes to approximately 70% the rate of their non-essential counterparts. Hence, this provides evidence of effects, albeit small, that protein location and centrality has on its probability of being essential.

In 2005, Joy et al. (2005) found that there exist a higher number of proteins with high betweenness and low degree in the yeast interactome, than in computer generated random scale-free networks (Barabási et al., 1999). This finding is surprising in that essential proteins have a higher average betweenness than they possess degree. When calculating the average betweenness for all proteins in the proteome versus the average betweenness of only the essential proteins an increase of 82% is observed; the frequency of essential proteins in the higher betweenness ones is also greater. However, when looking at the lower degree nodes with higher betweenness, which appear numerously in the yeast proteome, they appear to be serving as important links between different modules, another result that provides insight into the modular organization of PPINs.

In another work trying to contrast betweenness and degree in PPINs, the notion of bottleneckness versus hubness was studied as a tool to discover essential proteins (Yu et al., 2007).

Due to the observed correlation between betweenness of a node and its degree (Goh et al., 2003), they divided the proteins of a network into four categories to investigate whether bottlenecks are likelier to be essential because they have high betweenness or because they tend to be hubs (have high degree). They observed that bottlenecks (both non-hub and hub alike) are more likely to be essential than hub proteins that do not serve as bottlenecks in regulatory networks. In interaction networks, a substantial difference between bottlenecks that are not categorized as hub (they are less likely to be essential) and hub proteins that are not bottlenecks (more likely to be essential) seemed to indicate that degree is a better determinant of essentiality than betweenness.

Wang et al. (2014) developed a new measure to characterize structurally dominant proteins in PPINs based on a multivariate statistical analysis integrating multiple centrality metrics, such as degree, betweenness, closeness, the clustering coefficient, eigenvector centrality, semi-local centrality, and further considering the appearances of nodes in network "motifs". Their results illustrated that structurally dominant proteins in yeast tend to be more conserved from an evolutionary standpoint than other proteins. Moreover, PPINs were shown to be robust against random mutations, while fragile when mutations appear to be targeted. In the same sense of network "motifs", Vogiatzis et al. (2019) defined a star centrality metric and proposed two approximation algorithms to calculate it in large-scale PPINs. Their results also indicate towards it serving as a good proxy for identifying essentiality.

Recently, Wuchty (2014) and Wuchty et al. (2017) investigated network controllability in PPINs and employed minimum dominating sets to identify important proteins to serve as drug targets. Their study showed that the proteins obtained through minimum dominating sets contained multiple that were classified as essential. However, there can be multiple minimum dominating sets, especially in PPINs that are of formidable size. Using the definitions from Ishitsuka et al. (2016) on *critical* (proteins that appear in all minimum dominating sets), *intermittent* (proteins that appear in at least one minimum dominating set), and *redundant* (proteins that appear in none of the minimum dominating sets), it was shown that essential proteins appear often within sets of critical proteins.

Due to the high number of false positives in PPINs (as discussed in Chapter 3 because of the high-throughput methods in use) and the sensitivity of centrality-based network topology methods for identifying essential proteins, Li et al. (2013) proposed a new weighting method for evaluating the confidence of interactions and demonstrated the improvements in identifying essential proteins. Li et al. (2015) proposed *topology potential* (a new centrality metric in which every node is treated as a particle which creates a "field" around it) to identify essential proteins in PPINs. Calculating the topology potential of all proteins and ranking the proteins in descending order revealed that the top ones are more likely to be essential. Their results further indicated that this new ranking scheme outperforms other centrality metrics when identifying essentiality.

As discussed earlier, PPINs exhibit a modular structure. In their work, Hart et al. (2007) also demonstrated that essentiality is a modular property (rather than an individual one) and that many essential proteins are concentrated in a smaller number of complexes. They reached that conclusion using Markov clustering in a high-confidence interaction network, extracted by combining data from independent biological assays. The complexes that were identified using Markov clustering had high correlation with existing annotations.

We have already noted that there is a discussion between the essentiality (or not) of hub proteins: to that extent, Zotenko et al. (2008) concluded that albeit removing hubs, in general, destroys the overall network connectivity, this is not necessarily due to hub protein essentiality. They defined complex biological modules (COBIMs) as dense and functionally related protein complexes, and their essential counterparts (ECOBIMs), which possess the previous characteristics, but contain more essential proteins. In an experiment on six PPINs, they found that essential proteins are more important locally (as degree captures) rather than enabling global connectivity. Moreover, as in the work by Hart et al. (2007), they also concluded that there are complexes with higher portions of essential proteins: as per their definitions, those would be ECOBIMs.

Since multiple centrality-based essential discovery methods just consider the importance of a single protein in PPINs, Wang et al. (2012a) proposed an essential protein discovery method based on an edge clustering coefficient approach, effectively binding the characteristics of nodes and edges. They demonstrated that the number of essential proteins correctly classified by their method exceeds the number of discovered proteins by six other, commonly used centrality metrics.

Last, Ning et al. (2010) proposed the placement of proteins in reverse nearest neighbor topology as an indication to identify their essentiality. Reverse nearest neighbor is a topology where every node is the head of weighted edges emanating from its nearest neighbors. Their results revealed that hubs in reverse nearest neighbor topology are more likely to be essential proteins, which is a finding similar to the one in original PPINs. In addition, essential proteins are more commonly found in reverse nearest neighbor clusters. Based on these two findings for essential proteins, they proposed a new *reverse nearest neighbor cluster centrality* metric that was shown to outperform other centrality measures for identifying essential proteins.

Even though centrality has been traditionally applied to quantitatively assess individual node importance, in certain situations and applications we are interested in groups of nodes. This realization motivated a series of works that extend standard centrality metrics to groups and classes (see, e.g., Everett et al. (1999), Everett et al. (2005), and Borgatti (2006), Veremyev et al. (2017)). This further motivated research in groups that induce specific structures or cohesiveness requirements, as in the works by Vogiatzis et al. (2015), Rysz et al. (2018), Vogiatzis et al. (2019), Nasirian et al. (2020), and Zhong et al. (2020). Vogiatzis et al. (2019) propose a novel centrality metric, referred to as star centrality to cover the drawback of degree centrality in misclassifying protein importance. They developed a mathematical formulation, and proposed two approximation algorithms. Using protein-protein interaction networks of several organisms, they predicted protein essentiality and showed this new metric significantly outperform other nodal centrality metrics at detecting essential proteins. They also analyzed the average and worst case performance of the two approximation algorithms in practice, and showed that they are viable options for computing star centrality in very large-scale protein-protein interaction networks, such as the human proteome, where exact methodologies are bound to be time and memory intensive.

5.2.2. Integrating multiple sources

Identifying essential proteins based on their PPIN topological properties alone is very challenging and often inconclusive, as discussed in the previous subsection. The main shortcomings associated with topology-based methods are:

1. the lower accuracy due to the plethora of false positive and false negative interactions present in PPINs (Mrowka et al., 2001);
2. the poor performance of PPINs with lower connectivity; and
3. the lack of information tying topology to the biological significance and meaning of essential proteins (Li et al., 2016a).

Hence, researchers have proposed a series of methods that use information from other sources and tie them to topology-based techniques. To begin with, a group of methods use sequence-based information and features. These are intrinsic features of individual proteins determined by their genomic sequences. According to these features, there are three main groups of methods that can be employed to identify essential proteins, namely sub-cellular localization, evolutionary conservation, and gene expression.

Sub-cellular localization-based methods assume that essential proteins appear more frequently in specific sub-cellular locations, as compared to nonessential proteins. In these methods, proteins are scored and ranked based on their sub-cellular locations and then these scores are used to tell essential proteins apart from nonessential ones. The main idea behind evolutionary conservation-based methods is that essential proteins evolve more conservative than nonessential proteins (Jordan et al., 2002). Gene expression-based methods hypothesize that the expression level of mRNA is closely associated to essentiality. In other words, the expression level of essential genes is higher than that of nonessential ones.

Integrating the above-mentioned principles with network topology is one of the most common approaches used to improve the accuracy of essential protein detection methods in PPINs.

Based on the hypothesis that hub proteins tend to also share certain relevant functional properties that not only enable them to participate in multitudes of protein interactions, but can also be used for the discovery of such hub proteins without prior knowledge of the corresponding PPINs, Hsing et al. (2008) developed a hub predictor using the available interaction data and gene ontology annotations for proteins. In a similar integration, Li et al. (2012a) proposed a new centrality measure, named *PeC*, using protein-protein interaction and gene expression data. Contrary to other centrality measures, PeC determines protein essentiality based not only on its degree in the PPIN, but leveraging whether it has a high probability to be co-clustered and co-expressed with its neighbors. Experimental results showed that higher precision of PeC to identify essential proteins in comparison with a series of another fifteen network topology based centrality measures.

Zhang et al. (2013) proposed a new essential protein detection method named *Co-Expression Weighted by Clustering Coefficient* (CoEWCC). In that method two sources of data are integrated: protein-protein interactions and gene expressions. To determine essentiality, their measure considers two factors: (i) whether it has a high probability to be co-expressed with its neighbors, and (ii) whether its neighbors participate in densely connected clusters. In contrast with the PeC method, CoEWCC focuses more on the clustering property of the protein neighborhood, rather than the protein itself. Experimental results in their work revealed that this method outperforms PeC on the PPI network of Saccharomyces cerevisiae. Considering that most species have already a number of known essential proteins (from other sources and/or biological experimentation), Li et al. (2014) proposed two essential proteins discovery algorithms, referred to as *CPPK* and *CEPPK*. CPPK identifies essential proteins using PPIN topology, while CEPPK integrates PPIN topology with gene expressions. Their computational results verify that the integration of data from different parts helps: both methods had superior performance over 10 other typically used centrality measures, while CEPPK even outperformed other measures outside of nodal centrality metrics, such as PeC and CoEWCC.

In their work, Zhao et al. (2014) proposed a method based on overlapping essential modules for predicting essential proteins. To overcome the inherent PPIN noise and error rates, and to improve the prediction accuracy of their methods, they integrated information from gene expression profiles with the network topological features investigate. In their method, proteins are categorized into three categories: nonessential proteins, date hubs, and party hubs, according to their frequencies in the predicted essential modules. The experimental results showed the higher performance of their method, when comparing it to the existing methods for the prediction of essential proteins.

As discussed earlier, Pereira-Leal et al. (2004a) showcased that essential proteins are, on average, more connected to other essential proteins, rather than nonessential proteins. Based on that, Peng et al. (2012) proposed an iterative method to predict essential proteins by integrating biological data with a PPIN, named *ION*. ION combines three different features to assign a ranking score to each protein using an iterative method. The three features are the connections between proteins, the orthologous properties, and the features of their neighbors. Computational results revealed a 42% improvement compared to the edge clustering coefficient centrality (NC) method that was the best one among seven centrality methods. It was also shown to perform better than PeC. Peng et al. (2015) and Wang et al. (2013) proposed the *UDoNC* method by integrating protein domain features and other topological properties of a PPIN to predict essentiality. UDoNC uses both the number and frequency of protein domain types to evaluate the essentiality of proteins, as well as the edge clustering coefficient to measure essentiality of adjacent proteins. Tang et al. (2014), taking advantage of the Pearson correlation coefficient, aimed to bridge the gap between PPINs and gene expression profiles. To do so, they used an edge clustering coefficient and proposed a weighted degree centrality measure. They were able to show that their metric outperforms other methods including typical nodal centrality metrics and PeC.

Integrating PPIN topology and protein complex information, Ren et al. (2015) proposed a weighted average of complex centrality and subgraph centrality, referred to as *harmonic centrality* (HC). HC showed better performance against standard centrality measures. More importantly, the improvements observed were even more pronounced for low centrality values and PPINs with incomplete data. Seeing as this is a common theme, this was a great advancement. Additionally, they generated weighted PPINs by integrating cellular localization and biological processes data to the existing PPIN, which improved the performance of HC by 5%. More recently, Li et al. (2017) integrated protein complexes information with topological features of PPINs and proposed a *united complex centrality* (UC) metric to identify essential proteins. Analyzing the relationship between essential proteins and known protein complexes of Saccharomyces cerevisiae and Homo sapiens, they found that proteins in complexes are more likely to be essential, as opposed to proteins that do not appear in any complexes. Moreover, proteins participating in multiple complexes are more likely to be essential than proteins that only take part in a single protein complex. They finally showed that their proposed method outperforms 8 other network-based essential protein discovery methods in terms of a series of metrics including prediction precision, sensitivity, specificity, and accuracy rate.

He et al. (2006) proposed the concept of essential protein-protein interactions as interactions that are indispensable to the survival or reproduction of an organism, similar to the notion of essential proteins. An essential interaction renders both interacting proteins essential by definition, as deletion of either interacting protein may lead to lethality or infertility due to disruption of the interaction. In their work, they argued that essential interactions can explain the essentiality-lethality rule regardless of PPIN structure: since a hub protein has more interactions than a non-hub, it also possesses a higher probability to be involved in essential interactions. In addition, they explained that essential interactions are evolutionary more conserved that non-essential ones. Batada et al. (2006) compared more reliable, literature-curated datasets of protein interactions with high-throughput protein interaction datasets in Saccharomyces cerevisiae and observed that although there is a positive correlation between local connectivity and essentiality in both cases, the first results appear more robust. Further analysis of the data revealed that hubs do not evolve slower than non-hubs and prior results about correlation between connectivity and rate of evolution were a product of biases inherent in the high throughput data. Applying *k*-core decomposition in PPINs, Wuchty et al. (2005) found that the probability of a protein being both essential and evolutionally conserved successively increases as we move towards the innermost cores. They then categorized proteins as globally or locally central depending on their appearance in inner or outer cores, respectively, and hypothesized that globally central proteins serve as the backbone of the proteome.

Other methods, based on machine learning, build models from inputs to predict and improve predictions through learning algorithm. Many of these methods integrate topological features such as degree, betweenness, closeness, clustering coefficient, subgraph centrality with biological features such as gene sequence, gene expression, or functional annotation-related features. These features are then used as learning attributes to increase the prediction rates of essential proteins and genes (Zhang et al., 2016c). Chen et al. (2004) combined protein evolutionary rate, gene expression cooperativity, protein-protein interaction connectivity, and gene duplication data, in an attempt to predict essentiality. They then devised a neural network and a support vector machine framework to extract features of essential versus non-essential genes in the training process, and used the trained models to predict protein dispensability. Their results indicated that all the above mentioned features are important factors with protein evolution rates and protein-protein interaction connectivity being the most important ones. A different attempt at integrating protein-protein interaction data with genome sequencing data is due to Gustafson et al. (2006). In this work, the authors proposed a machine learning algorithm based on a set of features including topological ones (such as the protein degree in the PPIN), and biological ones (e.g., the open reading frame length, the codon adaptation index, etc.) as learning attributes. They utilized a Naive Bayes classifier to predict essential genes in both the proteomes of Saccharomyces cerevisiae and Escherichia coli. Hwang et al. (2009) also combined topological features in PPINs such as degree, betweenness, and closeness, with sequence properties such as ORF length and phyletic retention to develop a SVM classifier capable of predicting essential proteins.

Silva et al. (2008) developed *NTPGE* (Network Topology-based Prediction of Gene Essentiality), that combines the network features of a PPIN, a metabolic network (MN), and a transcriptional regulatory network (TRN), and uses the network topology features of all three networks as learning attributes to estimate essentiality. Constructing the integrated molecular network for a given organism comprising the above mentioned networks, a decision tree-based machine learning algorithm was devised using known essential and non-essential genes for training. Then, the generated decision tree classifier is applied to predict essentiality. Using supervised machine learning based methods, Acencio et al. (2009) also integrated the PPIN, MN, and TRN of Saccharomyces cerevisiae, and used the combination of topological features of the resulting network, along with data on cellular localization and biological process, as learning attributes to increase the predictability of essential genes. They also trained a decision tree based meta-predictor with individual and group attributes to create a series of predictors of essentiality. Their experiments showed that this integration of information increases the predictability of essentiality.

In the work of Seringhaus et al. (2006), 12 different different network topological features (including the well-known degree, betweenness, and closeness) were integrated with biological features, such as cellular localization and biological process data. They then developed several decision tree based classifiers, trained them, and put them to the test. Zhong et al. (2013) proposed a gene expression programming method for predicting essentiality; their work also combined topological features with biological features, along with other properties computed by using the PeC, WDC, and ION methods. Gene expression programming is a learning algorithm that "learns" the relationships between all variables in the data, and consequently, builds models to describe them. The experimental results indicated that this method outperforms other methods using one of the features at a time to predict essentiality. Additionally, this machine learning approach led to improved results than other machine learning methods, such as SVM, Naive Bayes, Random Tree. Zhong et al. (2015) collected 26 essentiality-related features and constructed a feature space, including topological features, biological features, and other composed features from ION, PeC, and WDC. Then, they used SVM and recursive feature elimination (SVM-RFE) and the PCC method to isolate a subset of features from the original space. Their method selected 6 features as the optimal subset and they used them to compare their method with other machine learning methods, such as simple SVM, Naive Bayes, NBTree. Their results proved the effectiveness of their feature selection method and the better performance when using their optimal subset of features, when compared to other methods.

Using experimental data of genome-wide knock-out screens of one bacterial organism, Plaimas et al. (2010) developed a SVM method to infer essentiality of a different bacterial organism. They considered a wide variety of features associated with essentiality, including topological features, genomic and transcriptomic features, and using Pearson correlation coefficient, they selected those with a higher correlation to essentiality. In order to be able to predict essentiality of a new organism, they performed a cross-validation analysis across the organisms of Escherichia coli and Pseudomonas aeruginosa. Based on their prediction results, and the experimental knock-out screening, they proposed 35 enzymes as drug targets for Salmonella typhimurium, 23 of which had been described previously as drug targets in other micro-organisms. Deng et al. (2010) developed a machine learning integrative approach to transfer gene essentiality annotations between distantly related bacterial species. They trained the classifier to learn by essential genes in one organism, and applied it to make predictions in the other, and evaluated the predictions based on their agreement with known essential genes in the target organism. They achieved that using features from gene sequencing, features derived from genomic sequence, and experimental functional genomics data they created. Four classifiers were used to train and test the model, namely a logistical regression model, a Naive Bayes classifier, the CN2 rule, and a C4.5 decision tree. The best performance was obtained by combining the outputs of these diverse classifiers using an unweighted average approach.

Similarly, Cheng et al. (2013) collected 16 widely used features(including topological features) and developed a novel method called the feature based weighted Naive Bayes model, which can address the issues with multicollinearity among gene features as well as feature divergence between species. They found that because of correlation between features there are two issues:

1. features might share no or little correlation with essentiality;

2. the relationship between gene features and essentiality might be different and even contrasting in different species.

To address those issues, they investigated the correlation between gene features and essentiality and used a stepwise regression model combined with forward selection and a genetic algorithm to determine a weight vector signaling the extent of contribution of each feature to essentiality. Then, they predicted essential genes in target organism by weighted Naive Bayes classifier. Their results showed that their approach can significantly improve the accuracy and robustness of essentiality prediction.

Staying in Naive Bayes classifiers to predict essentiality, Cheng et al. (2014) proposed a method to predict reciprocally essential genes. They found that the prediction accuracy is significantly influenced by the training set used, and thus they defined four criteria for training set selection:

1. essential genes in the selected training set should be reliable;
2. training species and target organisms should be closely related;
3. the growth conditions of essential genes should be consistent in both training and prediction sets; and
4. training organisms and prediction organisms should exhibit similar phenotypes or lifestyles.

Comparing the performance of incomplete and integrated training sets in different organisms, they found that the size of the training set should be at least equal to 10% of the total genes available to yield accurate predictions as well as improve robustness and accuracy.

Li et al. (2016a) integrated the information available from a variety of sources, like sub-cellular localization, orthologous proteins, and PPINs and proposed a novel method, called *SON*. They counted the number of essential and non-essential proteins in sub-cellular locations, and found that the ratio of essential proteins is higher than that of non-essential ones. Moreover, they found that for orthologous proteins, for at least 80 species, the ratio of essential proteins is 51% but for non-orthologous proteins this ratio falls dramatically to 22%. Computational results showed the better performance of SON when compared to nodal centrality metrics, subgraph centrality, as well as PeC and ION. They also showed that SON is able to perform well in identifying lower connectivity essential proteins, which is a common issue with centrality metrics exclusively dependent on PPIN topology. Integrating gene expression, orthology, and sub-cellular localization, Li et al. (2016b) proposed a novel neighborhood-based algorithm to predict essentiality. Due to the significant impact of unreliable neighbors and interactions on the prediction precision and accuracy of neighborhood-based methods, they combined gene expression and sub-cellular localization data to determine reliable neighbors in the PPIN. Conservative features of proteins were analyzed by orthologous features, and for integrating topological characteristics, a random walk model was proposed. The experimental results showed that their method outperformed ten of the existing methods.

In their work, Qin et al. (2017) combined topological properties, subcellular localization and orthologous protein information, and proposed the *CoTB* algorithm to predict essential proteins. Introducing several traditional topological properties of PPINs and integrating them with new measures of orthologous and subcellular localization scoring, they obtained a probability score for the proteins being essential using a random forest model. They showed their method outperforms traditional computational methods as well as SON.

Co-expressed genes are more probable to encode interacting proteins (Von Mering et al., 2002). Hence, two connected essential proteins are likelier to be co-expressed. This is the idea that Zhang et al. (2016a) used to create their *GEG* method to predict essential proteins. To achieve that, they integrated gene expression profiles and gene ontology annotation data. They then used a Pearson correlation coefficient to calculate the gene expression similarity of two proteins. In addition, since two connected essential proteins are also likely to share biological function (Krogan et al., 2006), they measure gene ontology similarity between two connected proteins (Wang et al., 2007). Finally, the defined the GEG measure of a given protein by combining gene expression and gene functionality similarities; their numerical experiments showed both the better performance of GEG, but also, more importantly, its robustness under perturbation.

In order to decrease the impact of false positives, and improve the accuracy of identifying essential proteins, Xiao et al. (2015) proposed a new framework based on PPINs constructed by integrating static protein-protein interactions with dynamic gene expressions. Then, they applied common centrality measures in the new PPIN, including degree, local average connectivity, and subgraph centrality among others. Their results are encouraging in the fact that dynamic interactions can now be captured and translated to essentiality, an important characteristic that is missed when dealing with, usually static, traditional PPINs.

Last, Zhang et al. (2016b) integrated PPIN and gene expression data to boost the prediction accuracy of other centrality-based methods and proposed an ensemble framework. Based on the assumption that various protein-protein interactions might contribute differently to determining protein essentiality, they generated a series of m PPINs based on the original PPIN using a gene expression correlation-guided strategy. Then, five centrality measures (degree, betweenness, closeness, eigenvector, and subgraph centrality) were used to calculate m different scores for each protein (in each PPIN). Continuing, for each protein, they combined the scores using a weighted voting strategy and derived an overall score. Finally, proteins were ranked and the top ranked proteins were considered potentially essential. The results showed higher prediction accuracy of this ensemble framework as opposed to individual centrality measures; they also showed that the ensemble framework is able to find lower degree, and easier to disregard, essential proteins.

CHAPTER 6. NOVEL GROUP CENTRALITY METRICS FOR STUDYING ESSENTIALITY IN PROTEIN-PROTEIN INTERACTION NETWORKS

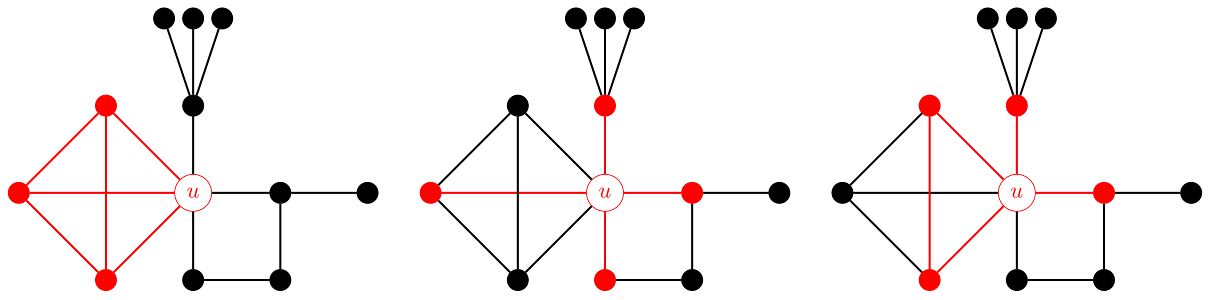
This chapter discusses the necessary notation and terminology that we will need throughout this problem. We then proceed to formally define group centrality as well as the problems we are studying. Examples of the problems and the calculations involved are presented afterward. We finish the chapter with discussion about computational complexity of the problems.

6.1. Notation

Let $G(V,E)$ be a simple, undirected, and unweighted graph, with V representing a set of $|V| = n$ nodes, and E a set of $|E| = m$ edges. For each node $i \in V$, let $N(i) = \{j \in V : (i,j) \in E\}$ be the open neighborhood of node i. Given a subset of nodes $S \subseteq V$, the subgraph induced by S in G is denoted as $G[S]$ and defined by $G[S]$ as the graph whose set of vertices is $V[S] = S$ and its set of edges is $E[S] = (S, (S \times S) \cap E)$. A node belonging to a group/set of nodes S can also be referred to as a group node/member, while nodes in $V \setminus S$ are non-group nodes. We may also generalize the notion of open neighborhood to sets of nodes $S \subseteq V$ as $N(S) = \{j : j \notin S, (i,j) \in E, i \in S\}$.

Also define a set P_{ij} that includes all shortest (geodesic) paths connecting nodes i and j. We let $\gamma_{ij} \geq 1$ represent the number of shortest paths between nodes i and j, that is $|P_{ij}| = \gamma_{ij}$. The length of a shortest path between i and j is the distance between i and j, and we use $d(i,j)$ to represent it. With a slight alteration in notation, we will use $\gamma_{ij}(S)$ as the number of shortest paths between two nodes $i, j \in V$ that intersect set S. The largest distance between any two nodes in the graph is referred to as the diameter of the graph: we have $diam(G) = max_{i,j \in V} d(i,j)$. Finally, we extend the definition of distance as $d(i,S) = min_{j \in S} d(i,j)$ for distances between a node i and a set S. Clearly, if $i \in S$, then $d(i,S) = 0$.

We will write that $S \subseteq V$ forms a *clique* (or, a complete subgraph on the set of nodes S) when $S \times S \subseteq E$. A set of nodes $S \subseteq V$ is said to form an induced *star* centered at some node $i \in S$, if we have that $(i,j) \in E, \forall j \in S \setminus \{i\}$ and every two other nodes in $S \setminus \{i\}$ are not adjacent, that is, $(k,\ell) \notin E, \forall k,\ell \in S \setminus \{i\}$. Last, we define here a *representative* set centered at some node $i \in S$ if $(i,j) \in E, \forall j \in S \setminus \{i\}$. It is clear from the definitions that a clique containing a node i as well as an induced star centered at i are both representatives of i, and hence the representative structure introduced is a relaxation of both cliques and induced stars. Examples for each of the structures of interest to us in this work are presented in Figure 6.1. As mentioned, the structures in Figures 6.1a and 6.1b could also serve as examples of possible representatives of u.



(a) A clique containing u. (b) An induced star centered at u. (c) A representative set of u.

Figure 6.1: An example of the three structures we are investigating in this work.

6.2. Group centrality

We define group degree, closeness, and betweenness centrality as follows (Everett et al., 1999; Everett et al., 2005).

Definition 1 (Group Degree Centrality)**.** *Given a graph $G(V,E)$, the degree centrality of a set of nodes $S \subseteq V$, $C^d(S)$ is the number of non-group nodes that are adjacent to S:*

$$C^d(S) = |N(S)|.$$

Note that multiple links to the same node are only counted once.

Definition 2 (Group Average Closeness Centrality)**.** *Given a graph $G(V,E)$, the (average) closeness centrality of a set of nodes $S \subseteq V$, $C^C(S)$, is the average distance of all nodes in $V \setminus S$ to S:*

$$C^c(S) = \frac{1}{|V| - |S|} \sum_{i \in V \setminus S} d(S,i),$$

where $C^c(S) = 0$, when $S = V$.

Definition 3 (GROUP BETWEENNESS CENTRALITY). *Given sets $S \subseteq V$ and $S' \subseteq V$, the betweenness centrality of S with respect to S' is defined as:*

$$C^b_{S'}(S) = \frac{\sum\limits_{i,j \in S' \setminus S, i<j} \gamma_{ij}(S)}{\sum\limits_{i,j \in S' \setminus S, i<j} \gamma_{ij}}. \tag{6.1}$$

If $S' = V \setminus S$, then the $C^b_{S'}(S)$ is referred to as the betweenness centrality of set S and we write it as $C^b(S)$.

6.3. Problem definitions

In this work, we are investigating the group degree, closeness, and betweenness centrality for a specific node $u \in V$. We only allow groups that induce a specific "motif", that is they form cliques, induced stars, and representatives (see Section 6.1). Combining the three different centrality metrics with the three different studied structures, this results in nine different problem setups:

- Clique
- Induced Star
- Representative

- Degree
- Closeness
- Betweenness

For the sake of brevity, we only define three of the above problems; the rest of them can be defined similarly. We also provide three examples to showcase how these metrics work. The problem definitions follow.

Definition 4 (REPRESENTATIVE DEGREE CENTRALITY). *The representative degree centrality of a node $u \in V$ is the degree centrality of the induced representative $S \subseteq V$ centered at u that produces the maximum open neighborhood size.*

As an example, suppose we are interested in the representative degree centrality of node $u \in V$ in the graph of Figure 6.2. The set that produces the maximum degree centrality is $\{u, 1, 5, 8\}$, with a value of 10. Observe that adding in S any of the adjacent to u nodes leads to a decrease in the cardinality of the open neighborhood by one.

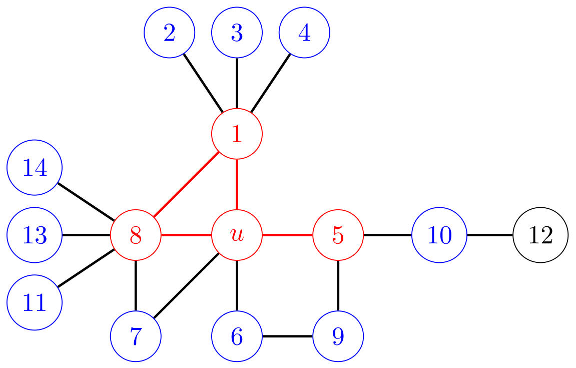


Figure 6.2: An example of representative degree centrality. The representative set S of u of maximum degree centrality is marked in red, and the open neighborhood is marked in blue.

Definition 5 (Clique Betweenness Centrality). *The clique betweenness centrality of a node $u \in V$ is the betweenness centrality of the clique $S \subseteq V$ with $u \in S$ that maximizes the fraction of shortest paths between $i, j \in V \setminus S$ passing through S.*

A different pictorial example of finding the clique betweenness centrality of a node u is given in Figure 6.3.

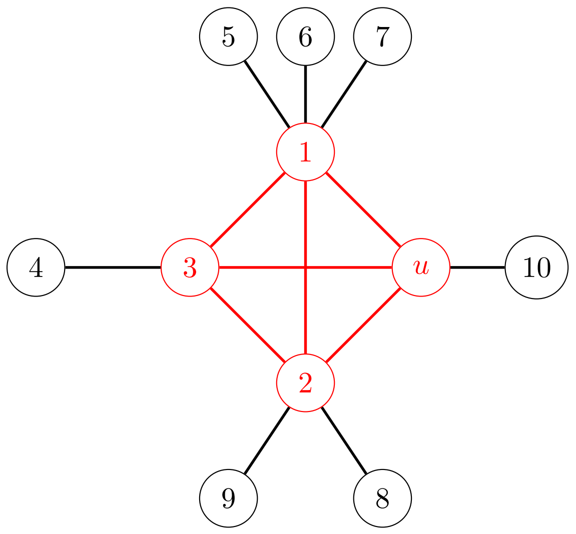


Figure 6.3: An example of clique betweenness centrality. The induced clique S of vertex u with maximum betweenness is marked in red.

To determine the clique betweenness centrality of node $u \in V$, we find the induced clique $S = \{u, 1, 2, 3\}$ (red nodes) including node u ($u \in S$) that maximizes the fraction of shortest paths passing through S. The betweenness centrality of this clique is 0.952.

Definition 6 (STAR CLOSENESS CENTRALITY)**.** *The star closeness centrality of a node $u \in V$ is the closeness centrality of the induced star $S \subseteq V$ centered at u that minimizes the average distance from S to every other node in $V \setminus S$.*

As an example, consider the graph in Figure 6.4. If we are interested in calculating the star closeness centrality of node u, then we need to consider all induced stars centered at u, and pick the one with the smallest average closeness value.

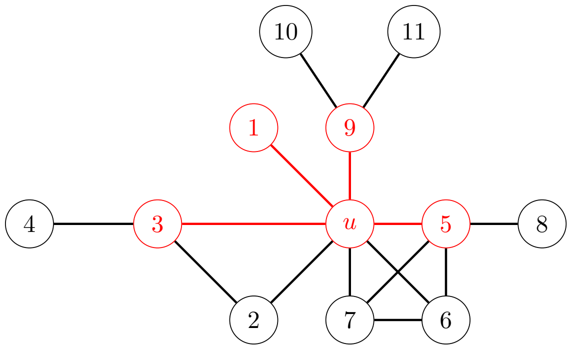


Figure 6.4: An example of the star closeness centrality. The induced star S with minimum average distance to nodes in $V \setminus S$ is marked in red.

For example, $S = \{u\}$ would be a trivial induced star centered at u and would lead to average distance equal to $\frac{15}{11}$ (as there are 7 nodes at a distance of 1, and 4 nodes at a distance of 2, for a total distance of 15). Another example would be induced star $S = \{u, 3, 6, 9\}$ with a total distance of 9 (7 nodes at a distance of 1 and 1 node at a distance of 2) leading to an average distance of $\frac{9}{8}$. An optimal one is the star denoted ($S = \{u, 1, 3, 5, 9\}$) with average distance equal to 1. Another optimal solution could remove node 1 from the star. Note the close relationship between this problem and the problem of finding a dominating set given a structural constraint.

6.4. Complexity

In this section, we derive the computational complexity of the problems. We divide the discussion in three parts, one per centrality metric investigated.

6.4.1. Structure degree centrality

We begin with degree centrality. In general, the problems we are tasked with solving involve identifying a set of nodes inducing a specific structure with maximum degree. Formally, we have the following problem definition.

Definition 7 (STRUCTURE DEGREE CENTRALITY OF A NODE $u \in V$)**.** *Given $G(V,E)$, a node $u \in V$, and an integer number k, does there exist an induced structure subgraph $S \subseteq V$ of u such that $N(S) \geq k$?*

Before we proceed to the complexity result for this problem, let us introduce the SET COVER problem, which has been shown to be $\mathscr{NP}$-complete (Karp, 1972).

Definition 8 (SET COVER)**.** *Given a universe of elements $\mathscr{U}$, a collection of sets $\mathscr{S}$ whose union is equal to $\mathscr{U}$, and an integer ℓ, does there exist a subset $S \subseteq \mathscr{S}$ such that $\bigcup\limits_{C \in S} C = \mathscr{U}$ and $|S| \leq \ell$?*

Theorem 1. STRUCTURE DEGREE CENTRALITY OF A NODE $u \in V$ *is $\mathscr{NP}$-complete.*

Proof of Theorem 1. First of all, we note that the problem is in $\mathscr{NP}$, as given a solution $S \subseteq V$, we can verify in polynomial time whether $N(S) \geq \ell$, whether u belongs to S, and whether S satisfies the structure property we are interested in (i.e., u is adjacent to every node, S induces a clique, S is an induced star).

Now, consider an instance of the SET COVER problem, with a universe of elements $\mathscr{U}$, a collection of sets $\mathscr{S}$ of cardinality δ, and an integer ℓ. Now, construct a graph $G(V,E)$ as follows. First, add a node u, a node for every set $C \in \mathscr{S}$ ($V_{\mathscr{S}}$), and a node for every element in the universe $\mathscr{U}$ ($V_{\mathscr{U}}$). Secondly, in the case of an induced star, connect by an edge u to every node representing a set with $E_{\mathscr{S}} = \bigcup\limits_{v \in V_{\mathscr{S}}} \{(u,v)\}$, and every node v_C representing a set $C \in \mathscr{S}$ to every node v_w representing element $w \in \mathscr{U}$ whenever $w \in C$, or $E_{\mathscr{U}} = \{(v_C, v_w) : v_C \in V_{\mathscr{S}}, v_w \in V_{\mathscr{U}}, w \in C, \forall w \in \mathscr{U}, \forall C \in \mathscr{S}\}$. In the case of an induced clique, on top of the previous edge sets, also connect by an edge all nodes v_C representing a set $C \in \mathscr{S}$, as in $E_{\mathscr{S}\times\mathscr{S}} = \{(v_{C_1}, v_{C_2}) : v_{C_1} \in V_{\mathscr{S}}, v_{C_2} \in \mathscr{S}\}$. Formally, we have the node set in (6.2) and the edge set in (6.3) (for an induced star) or in (6.4) (for a clique).

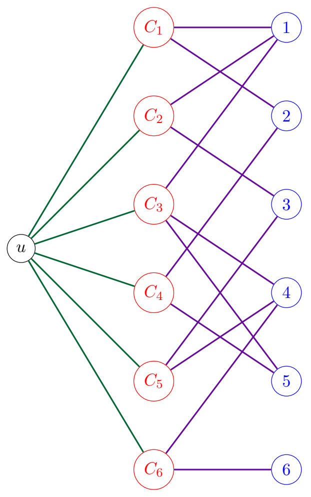
$$V = \{u\} \bigcup V_{\mathscr{S}} \bigcup V_{\mathscr{U}} \tag{6.2}$$

$$E = E_{\mathscr{S}} \bigcup E_{\mathscr{U}} \tag{6.3}$$

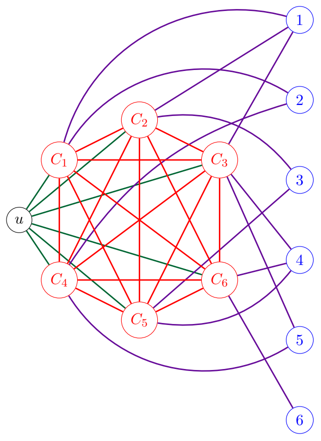
$$E = E_{\mathscr{S}} \bigcup E_{\mathscr{U}}. \tag{6.4}$$

For the representative structure, *any* of the two gadgets, as well as any gadget with any connections between two nodes corresponding to sets in $\mathscr{S}$ would work. An example of the gadgets used are shown in Figure 6.5.

$$\left.\begin{array}{l} \mathscr{U} = \{1,2,3,4,5,6\} \\ \mathscr{S} = \{(1,2),(1,3),(1,4,5),(2,5),(3,4),(4,6)\} \\ \delta = 6 \end{array}\right\} \Rightarrow$$



(a) The gadget used for STAR DEGREE CENTRALITY OF A NODE $u \in V$. The nodes in red are the ones in $V_{\mathscr{S}}$, while the ones in blue are in $V_{\mathscr{U}}$. The edges in green are in $E_{\mathscr{S}}$ and the ones in purple are in $E_{\mathscr{U}}$.



(b) The gadget used for CLIQUE DEGREE CENTRALITY OF A NODE $u \in V$. On top of the nodes and edges from the gadget of Figure 6.5a, we also show here the edges in $E_{\mathscr{S} \times \mathscr{S}}$ in red.

Figure 6.5: An example of the gadget used for transforming an instance of SET COVER to an instance of STRUCTURE DEGREE CENTRALITY OF A NODE $u \in V$.

Finally, to end the transformation from SET COVER, let $k = |\mathscr{U}| + \delta - \ell$. We will now show that a set cover of cardinality smaller than or equal to ℓ exists if and only if there exists in $G(V,E)$ a representative/induced star/clique $S \subseteq V$ of u, such that $|N(S)| \geq k$. For the first direction, assume there exists a set cover S, such that $|S| \leq \ell$. Consider the subgraph consisting of u itself and the nodes $v_C \in V_{\mathscr{S}}$ such that $C \in S$. Its open neighborhood size then is equal to $|\mathscr{U}| + \delta - \ell = k$ as all elements of U are adjacent to the nodes in the representative, as well as all nodes in $V_{\mathscr{S}}$ that are not in the set.

For the second direction, assume, for a contradiction, that there exists no set cover of cardinality smaller than or equal to ℓ; yet, there exists a subgraph $S \subseteq V$ of u in $G(V,E)$ such that $|N(S)| \geq k$. We distinguish between two cases:

1. the set is adjacent to all elements in $V_{\mathscr{U}}$, or

2. the set is not adjacent to every element in $V_{\mathscr{U}}$.

In the former case, since $k = |\mathscr{U}| + \delta - \ell$, this implies that there exists at most ℓ nodes in $V_{\mathscr{S}}$ that are in the set. However, by construction, this also implies that taking the sets that correspond to the nodes in $S \cap V_{\mathscr{S}}$ we would end up with a set cover of cardinality at most ℓ, which contradicts our original assumption.

In the latter case, assume only $\overline{U} \subset \mathscr{U}$ elements are adjacent to our set, S. Seeing as $|N(S)| \geq k = |\mathscr{U} + \delta - \ell$, and given that we are adjacent to $|\overline{U}|$ nodes in $V_{\mathscr{U}}$, this implies that we have at most $\ell - |\mathscr{U}| + |\overline{U}|$ nodes from $V_{\mathscr{S}}$ in the representative. From the definition of the set cover problem, the union of all sets in $\mathscr{S}$ is equal to $\mathscr{U}$: hence, in the sets that do not correspond to nodes belonging in S, there exists a subset such that their union with the sets already in S they cover all of $\mathscr{U}$. In the worst case, each of those sets covers exactly one new element, leading to the addition of $|\mathscr{U}| - |\overline{U}|$ new sets from $\mathscr{S}$. Overall, we have constructed a set cover of cardinality at most $\ell - |\mathscr{U}| + |\overline{U}| + |\mathscr{U}| - |\overline{U}| = \ell$, a contradiction. This concludes the proof.

□

6.4.2. Structure closeness centrality

We now proceed to show the complexity for closeness centrality. Prior to the complexity derivations, we begin with the definition of the decision version. We also restate here the CLIQUE and INDEPENDENT SET problems, which are well-known to be $\mathscr{N}\mathscr{P}$-complete (Karp, 1972).

Definition 9 (STRUCTURE CLOSENESS CENTRALITY OF A NODE $u \in V$). *Given $G(V,E)$, a node $u \in V$, and a real number k, does there exist an induced structure subgraph $S \subseteq V$ of u such that the average distance to all other nodes in $V \setminus S$ is less than or equal to k?*

Definition 10 (CLIQUE). *Given a graph $G(V,E)$ and an integer number ℓ, does there exist a subset of nodes $S \subseteq V$ such that they induce a complete subgraph of G, $G[S]$, and $|S| \geq \ell$?*

Definition 11 (INDEPENDENT SET). *Given a graph $G(V,E)$ and an integer number ℓ, does there exist a set of nodes $S \subseteq V$ such that no two nodes in S are adjacent, and $|S| \geq \ell$?*

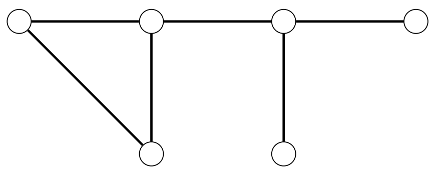
Theorem 2. CLIQUE CLOSENESS CENTRALITY OF A NODE $u \in V$ *and* STAR CLOSENESS CENTRALITY OF A NODE $u \in V$ *are both $\mathscr{NP}$-complete.*

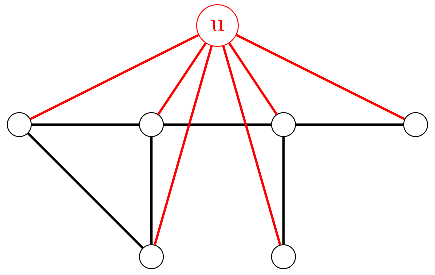
Proof of Theorem 2. Once more, we begin by showing that both problems belong to $\mathscr{NP}$. Consider a solution $S \subseteq V$: we can verify in polynomial time whether $G[S]$ is a complete subgraph or whether it induces a star. Moreover, it is easy to check the cardinality of the set.

Consider an instance of CLIQUE $< G(V,E), \ell >$ (resp,. INDEPENDENT SET $< G(V,E), \ell >$). Construct an instance of CLIQUE CLOSENESS CENTRALITY OF A NODE $u \in V$ (resp., STAR CLOSENESS CENTRALITY OF A NODE $u \in V$) as follows. First, add a node u to G, and connect it with an edge to every other node in V ($E_u = \{(u,i), \forall i \in V\}$). Then, for every node in V create a chain of δ nodes (let them be $V_i, \forall i \in V$), such that from the originating node in V, one node is at a distance of δ hops, one node is at a distance of $\delta - 1$ hops, and so on (let it be $E_i = (i, i_0) \bigcup \left(\bigcup\limits_{j=1}^{\delta} (i_{j-1}, i_j) \right)$). Figure 6.6 presents and example of the gadget used. Formally, we can describe the nodeset as in (6.5) and the edgeset as in (6.6).

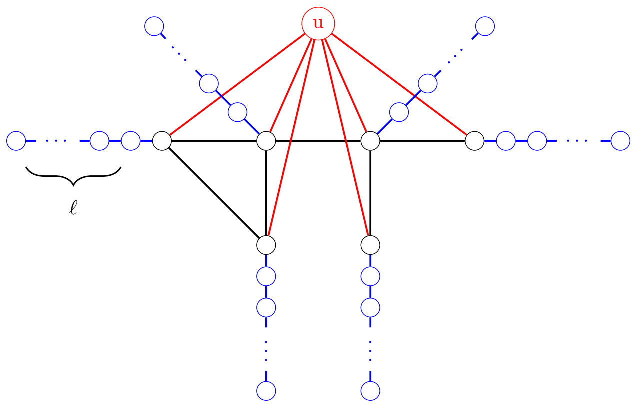
$$\hat{V} = V \bigcup \{u\} \bigcup \left(\bigcup_{i \in V} V_i \right), \tag{6.5}$$

$$\hat{E} = E \bigcup E_u \bigcup \left(\bigcup_{i \in V} E_i \right). \tag{6.6}$$


(a) The original graph $G(V,E)$ on which CLIQUE or INDEPENDENT SET are to be solved, given an integer ℓ.



(b) The graph after adding u and the corresponding edge set, E_u (shown in red).



(c) The graph after adding node set $V_i, \forall i \in V$, and edge set $E_i, \forall i \in V$ (shown in blue).

Figure 6.6: An example of the transformation from CLIQUE (resp., INDEPENDENT SET) to the CLIQUE CLOSENESS CENTRALITY OF A NODE $u \in V$ (resp., STAR CLOSENESS CENTRALITY OF A NODE $u \in V$).

We will now show that $G(V,E)$ has a clique (resp., independent set) of size k if and only if the clique (resp., star) closeness of node $u \in \hat{V}$ in $\hat{G}(\hat{V},\hat{E})$ is less than or equal to $\overline{k} = \frac{n \cdot \ell + 2(n-k)}{2n}$, where n is the number of nodes in G ($n = |V|$). For the first part of the proof, assume that $G(V,E)$ has a clique (resp., independent set) S of size k. Note that $S \cup \{u\}$ forms a clique (resp., star) in $\hat{G}$, too. The closeness of that clique (resp., star) then is equal to:

$$C(\hat{S}) = \frac{(\ell + \ell - 1 + \ell - 2 + \ldots + 1) \cdot k + (\ell + 1 + \ell + \ell - 1 + \ldots + 1) \cdot (n-k)}{n \cdot (\ell + 1)} =$$
$$= \frac{\frac{(\ell+1) \cdot \ell}{2} \cdot k + \frac{(\ell+2) \cdot (\ell+1)}{2} \cdot (n-k)}{n \cdot (\ell+1)} = \frac{k \cdot \ell + (\ell+2) \cdot (n-k)}{2n} = \frac{n \cdot \ell + 2 \cdot (n-k)}{2n}.$$

For the other direction, assume that there exists a clique (resp., star) $\hat{S}$ in $\hat{G}$ such that $C(\hat{S}) \leq \bar{k}$. Further assume, for a contradiction, that there exists no clique (resp., independent set) of size k in G. We then have that:

$$\begin{aligned}
C(\hat{S}) &= \frac{\frac{(\ell+1)\cdot\ell}{2}\cdot\left(|\hat{S}|-1\right)+\frac{(\ell+2)(\ell+1)}{2}\cdot\left(n-|\hat{S}|+1\right)}{n\cdot(\ell+1)} = \frac{\ell\cdot\left(|\hat{S}|-1\right)+(\ell+2)\cdot\left(n-|\hat{S}|+1\right)}{2n} = \\
&= \frac{\ell\cdot n+2\cdot\left(n-|\hat{S}+1\right)}{2n} \leq \bar{k} \implies \frac{\ell\cdot n+2\cdot\left(n-|\hat{S}+1\right)}{2n} \leq \frac{\ell\cdot n+2\cdot(n-k)}{2n} \implies \\
&\implies n-|\hat{S}|+1 \leq n-k \implies k \leq |\hat{S}|-1 \implies |\hat{S}| \geq k+1. \qquad (6.7)
\end{aligned}$$

Observe that $S=\hat{S}\setminus\{u\}$ is a clique (resp., independent set) in G of size $|S|=|\hat{S}|-1$. From (6.7), though, we have that $|S|\geq k$, which contradicts the initial assumption, finishing the proof.

□

We can now also derive the complexity of the representative problem.

Theorem 3. REPRESENTATIVE CLOSENESS CENTRALITY OF A NODE $u \in V$ *is* $\mathscr{NP}$*-complete.*

Proof of Theorem 3. The problem is clearly in $\mathscr{NP}$, since it is easily verifiable in polynomial time whether a set of nodes is all connected to u and hence forms a representative, and whether its average closeness is smaller than or equal to a real value k.

Consider an instance of SET COVER with a universe $\mathscr{U}$, a collection of sets $\mathscr{S}$, and an integer ℓ. First, create a node for every set $C \in \mathscr{S}$ ($V_{\mathscr{S}}$), and two nodes u_i and v_i for every element $i \in \mathscr{U}$ (V_u and V_v). Also add one node u. Formally, the nodeset can be described as $\hat{V} = V_{\mathscr{S}} \bigcup V_u \bigcup V_v \bigcup \{u\}$.

Connect u to every node corresponding to a set $C \in \mathscr{S}$: let this be edgeset $E_{\mathscr{S}}$. Now, connect with an edge every node u_i to a node in $V_{\mathscr{S}}$ if element $i \in \mathscr{U}$ belongs to the set $C \in \mathscr{S}$ the node corresponds to, forming edgeset $E_{\mathscr{S}\times\mathscr{U}}$. Last, connect every node u_i to v_i, $\forall i \in \mathscr{U}$ ($E_{\mathscr{U}} = \{(u_i, v_i), \forall i \in \mathscr{U}\}$). Finally, we have the edgeset of the constructed graph be $\hat{E} = E_{\mathscr{S}} \bigcup E_{\mathscr{S}\times\mathscr{U}} \bigcup E_{\mathscr{U}}$. We provide an example of the reduction in Figure 6.7.

$$\left.\begin{array}{l}\mathscr{U} = \{1,2,3,4,5,6\} \\ \mathscr{S} = \{(1,2),(1,3),(1,4,5),(2,5),(3,4),(4,6)\} \\ \delta = 6\end{array}\right\} \Rightarrow$$

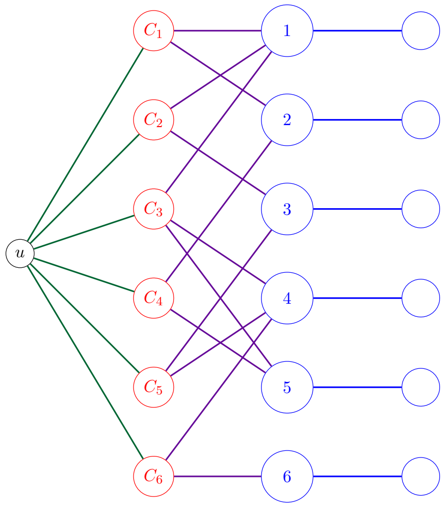


Figure 6.7: An example of the gadget used for transforming an instance of SET COVER to an instance of REPRESENTATIVE CLOSENESS CENTRALITY OF A NODE $u \in V$. The nodes in red are the ones in $V_{\mathscr{S}}$, while the ones in blue are in V_u and V_v. The edges in green are in $E_{\mathscr{S}}$, the ones in purple are in $E_{\mathscr{S}\times\mathscr{U}}$, and the ones in blue are in $E_{\mathscr{U}}$.

We will show that a set cover of cardinality ℓ exists if and only if $\hat{G}(\hat{V},\hat{E})$ has a representative set of u such that its average closeness is $k = 1 + \frac{|\mathscr{U}|}{2|\mathscr{U}|+(n-\ell)}$. For the first direction, assume there exists a set cover of cardinality ℓ. Consider the representative of u that consists of u itself and the ℓ nodes that correspond to the sets in the set cover. The total distance of this set of nodes to every node in the graph is equal to:

$$|\mathscr{U}| \cdot 1 + |\mathscr{U}| \cdot 2 + (n-\ell) \cdot 1 = 3 \cdot |\mathscr{U}| + (n-\ell)\,. \tag{6.8}$$

This is easy to see as all elements in V_u are covered in a distance of 1 from one element in the representative, all elements in V_v are within a distance of 2, and the remaining nodes (which correspond to sets that were not in the set cover) are at a distance of 1 from u (by construction). The total number of nodes that are not in the representative are $2|\mathscr{U}| + (n-\ell)$, leading to an average closeness of:

$$\frac{3\cdot|\mathscr{U}|+(n-\ell)}{2\cdot|\mathscr{U}|+(n-\ell)} = 1+\frac{|\mathscr{U}|}{2\cdot|\mathscr{U}|+(n-\ell)} = k. \tag{6.9}$$

For the other direction, assume no set cover of size ℓ exists; yet, there exists a representative of u set $\hat{S}\cup\{u\}$ with closeness less than or equal to k. We separate the proof in two parts:

a) $\hat{S}$ is adjacent to all nodes in V_u. Let $|\hat{S}| = \hat{\ell}$. Then, we have that:

$$1+\frac{|\mathscr{U}|}{2\cdot|\mathscr{U}|+\left(n-\hat{\ell}\right)} \leq k \implies 1+\frac{|\mathscr{U}|}{2\cdot|\mathscr{U}|+\left(n-\hat{\ell}\right)} \leq 1+\frac{|\mathscr{U}|}{2\cdot|\mathscr{U}|+(n-\ell)} \implies$$
$$\implies (n-\ell) \leq \left(n-\hat{\ell}\right) \implies \hat{\ell} \leq \ell. \tag{6.10}$$

The last inequality at (6.10) implies that there exists a set cover of cardinality $\hat{\ell} \leq \ell$, which contradicts our initial assumption.

b) $\hat{S}$ is adjacent to a strict subset of the nodes in V_U. Let this subset be $V_{\hat{u}} \subset V_u$ with the corresponding universe elements $\hat{\mathscr{U}} \subset \mathscr{U}$ and let $|\hat{S}| = \hat{\ell}$. Its closeness is:

$$\frac{3\cdot|\mathscr{U}\setminus\hat{\mathscr{U}}|+2\cdot|\mathscr{U}\setminus\hat{\mathscr{U}}|+2\cdot|\hat{\mathscr{U}}|+|\hat{\mathscr{U}}|+\left(n-\hat{\ell}\right)}{2\cdot|\mathscr{U}|+\left(n-\hat{\ell}\right)} = 1+\frac{3\cdot|\mathscr{U}|-2\cdot|\hat{\mathscr{U}}|}{2\cdot|\mathscr{U}|+\left(n-\hat{\ell}\right)} =$$
$$= 1+\frac{|\mathscr{U}|}{2\cdot|\mathscr{U}|+\left(n-\hat{\ell}\right)}+\frac{2\cdot|\mathscr{U}\setminus\hat{\mathscr{U}}|}{2\cdot|\mathscr{U}|+\left(n-\hat{\ell}\right)} \leq \hat{k} = 1+\frac{|\mathscr{U}|}{2\cdot|\mathscr{U}|+(n-\ell)} \implies \tag{6.11}$$
$$\implies 1+\frac{|\mathscr{U}|}{2\cdot|\mathscr{U}|+\left(n-\hat{\ell}\right)} < 1+\frac{|\mathscr{U}|}{2\cdot|\mathscr{U}|+(n-\ell)}. \tag{6.12}$$

From (6.12), we get that $\hat{\ell} < \ell$. Now, if we show that $|\mathscr{U}\setminus\hat{\mathscr{U}}| \leq \left(\ell-\hat{\ell}\right)$, this will imply that we can add (in the worst case) the $|\mathscr{U}\setminus\hat{\mathscr{U}}|$ nodes in $V_{\mathscr{S}}$ to the representative and still have a representative of cardinality less than or equal to ℓ. From (6.11), we have that:

$$
\begin{aligned}
&\frac{|\mathscr{U}|+2\cdot|\mathscr{U}\setminus\hat{\mathscr{U}}|}{2\cdot|\mathscr{U}|+\left(n-\hat{\ell}\right)}+\leq\frac{|\mathscr{U}|}{2\cdot|\mathscr{U}|+(n-\ell)}\implies\frac{|\mathscr{U}|+2\cdot|\mathscr{U}\setminus\hat{\mathscr{U}}|}{2\cdot|\mathscr{U}|+(n-\ell)+\left(\ell-\hat{\ell}\right)}\leq\frac{|\mathscr{U}|}{2\cdot|\mathscr{U}|+(n-\ell)}\\
&\implies\frac{2\cdot|\mathscr{U}\setminus\hat{\mathscr{U}}|}{\left(\ell-\hat{\ell}\right)}\leq\frac{|\mathscr{U}|}{2\cdot|\mathscr{U}|+(n-\ell)}\implies 2\cdot|\mathscr{U}\setminus\hat{\mathscr{U}}|\cdot(2\cdot\mathscr{U}+(n-\ell))\leq|\mathscr{U}|\cdot\left(\ell-\hat{\ell}\right)\\
&\implies 4\cdot|\mathscr{U}|\cdot|\mathscr{U}\setminus\hat{\mathscr{U}}|\leq\mathscr{U}\cdot\left(\ell-\hat{\ell}\right)\implies|\mathscr{U}\setminus\hat{\mathscr{U}}|\leq\frac{1}{4}\cdot\left(\ell-\hat{\ell}\right)\implies|\mathscr{U}\setminus\hat{\mathscr{U}}|\leq\left(\ell-\hat{\ell}\right).
\end{aligned}
\tag{6.13}
$$

The inequality at (6.13) finishes the proof, as adding the (at most) $|\mathscr{U}\setminus\hat{\mathscr{U}}|$ sets from $\mathscr{S}$ needed to cover the remaining elements from $\mathscr{U}$, and whose vertex counterparts in $\hat{G}$ were not in $\hat{S}$, creates a set cover of cardinality at most ℓ, which is a contradiction.

□

6.4.3. Structure betweenness centrality

Similarly to the previous subsections, we begin by providing the definition of the decision version of the problem.

Definition 12 (STRUCTURE BETWEENNESS CENTRALITY OF A NODE $u \in V$)**.** *Given $G(V,E)$, a node $u \in V$, and a real number k, does there exist an induced structure subgraph $S \subseteq V$ of u such that the betweenness centrality (defined as the proportion of geodesics between every two nodes $i, j \in V \setminus S$ that pass through S) is greater than or equal to k?*

We are now ready to show the $\mathscr{NP}$-completeness of the problem when the structure containing $u \in V$ is enforced to be a clique or an induced star.

Theorem 4. CLIQUE BETWEENNESS CENTRALITY OF A NODE $u \in V$ *and* STAR BETWEENNESS CENTRALITY OF A NODE $u \in V$ *are both $\mathscr{NP}$-complete.*

Proof of Theorem 4. We see again that both problems are in $\mathscr{NP}$ as it can be verified in polynomial time whether the structure obtained forms a clique containing u or an induced star centered at u with a betweenness greater than or equal to k. As a reminder, the betweenness of a group of nodes can be calculated in polynomial time (see, e.g., Puzis et al. (2007a)).

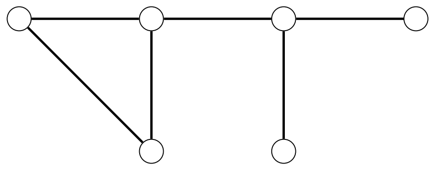
Once more we put the CLIQUE and INDEPENDENT SET problems to use. Consider an instance of CLIQUE (resp., INDEPENDENT SET) $< G(V,E), \ell >$. We will construct an instance of our problem as follows. First, introduce two new nodes u, u^*: connect u with an edge to every node $i \in V$ (forming $E_u = \{(u,i), \forall i \in V\}$). Moreover, add a new set V_i, for every node i, containing M nodes: each of them is connected with an edge to their originating node i (i.e., $E_i = \left\{ \left(v_i^{(j)}, i\right), \forall j = 1, \ldots, M \right\}, \forall i \in V$) and to u^* (i.e., $E_{u^*} = \left\{ \left(v_i^{(j)}, u^*\right), \forall i \in V, \forall j = 1, \ldots, M \right\}$). Formally, the node set and edge set of the constructed graph are given in (6.14) and (6.15): an example of the transformation is given in Figure 6.8.

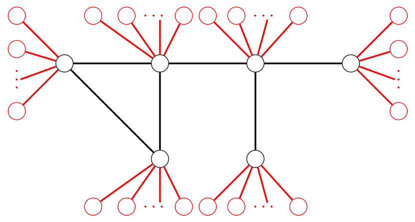
$$\hat{V} = V \bigcup \left(\bigcup_{i \in V} V_i \right) \bigcup \{u\} \bigcup \{u^*\} \tag{6.14}$$

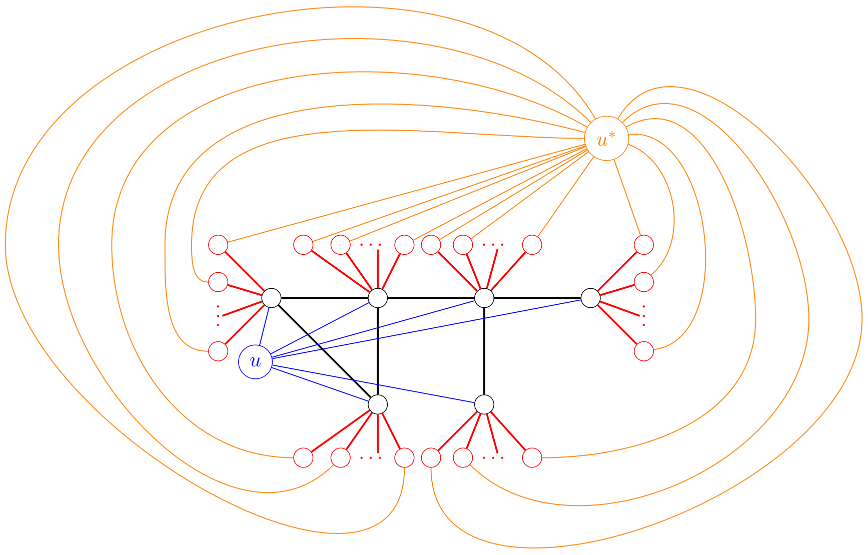
$$\hat{E} = E \bigcup \left(\bigcup_{i \in V} E_i \right) \bigcup E_u \bigcup E_{u^*} \tag{6.15}$$

We will show that a clique (resp., independent set) of size ℓ exists in $G(V,E)$ if and only if there exists a clique containing (resp., induced star centered at) u in $\hat{G}(\hat{V}, \hat{E})$ such that its betweenness is greater than or equal to $k = \frac{1}{2} \frac{\ell}{n^2}$.

For the first direction of the proof, assume there exists a clique S in $G(V,E)$ such that $|S| = \ell$. Clearly, the set $S \cup \{u\}$ forms a clique in $\hat{G}$. Its betweenness can be shown to be greater than or equal to k as follows. Note that, by construction, all shortest paths from a node in V_i to a node in V_j or to u^* does not use any of the nodes in the clique containing (resp., induced star centered at) u. However, shortest paths connecting two nodes in V_i do have a fraction that uses one node (i) in the clique: more specifically half of those shortest paths pass through the clique, with the other half using the other shortest path that uses u^*. Hence, this gives us a total of $\frac{1}{2} \ell \frac{M \cdot (M-1)}{2}$ shortest paths. Moreover, there are at least $\frac{1}{2} nM (n - \ell)$ shortest paths passing through the link to connect a node in V_i to a node in $V \setminus S$ and at most $nM (n - \ell)$. Similarly, there are at least 0 and at


(a) The original graph $G(V,E)$ on which CLIQUE or INDEPENDENT SET are to be solved, given an integer ℓ.


(b) The graph after adding a total of $n \cdot M$ leaves ($V_i, \forall i \in V$) and their corresponding edge set, $E_i, \forall i \in V$ (shown in red).



(c) The graph after adding nodes u and u^*, as well as edge sets E_u, E_{u^*} (shown in blue and orange, respectively).

Figure 6.8: An example of the transformation from CLIQUE (resp., INDEPENDENT SET) to the CLIQUE BETWEENNESS CENTRALITY OF A NODE $u \in V$ (resp., STAR BETWEENNESS CENTRALITY OF A NODE $u \in V$).

most $\frac{(n-\ell)\cdot(n-\ell-1)}{2}$ shortest paths between two nodes in $V \setminus S$. Finally, there are no shortest paths using a node in the clique (resp., independent set) that connect a node in $V \setminus S$ to u^*. This gives us that there are at least a total number of shortest paths using a node in S:

$$\frac{1}{2}\ell\frac{M\cdot(M-1)}{2}+\frac{1}{2}nM(n-\ell)=\frac{\frac{1}{2}\cdot\ell\cdot M^2+\frac{1}{2}\left(n^2-\ell\cdot n+\ell\right)\cdot M}{2}. \tag{6.16}$$

The total number of shortest paths is equal to

$$\frac{(n\cdot(M+1)-\ell+1)\cdot(n\cdot(M+1)-\ell)}{2} = \frac{n^2\cdot M^2+\left(2n^2-2\ell n+n\right)\cdot M+n^2+\ell^2-2\ell n+n-\ell}{2}. \tag{6.17}$$

Combining (6.16) and (6.17) as well as the fact that M can be made suitably big, we finally get that the betweenness of $S\cup\{u\}$ is at least equal to $\frac{1}{2}\frac{\ell}{n^2}=k$.

Now, let us assume that there is no clique (resp., independent set) of size ℓ in $G(V,E)$ and yet there exists a clique containing (resp., induced star centered at) u in $\hat{G}(\hat{V},\hat{E})$ with a betweenness greater than or equal to k. Let it be $\hat{S}\cup\{u\}$ and assume that $|\hat{S}|=\hat{\ell}$.

From the earlier discussion, we have that by construction the number of shortest paths using at least one node in $\hat{S}\cup\{u\}$ is at most equal to:

$$\begin{gathered}\frac{1}{2}\hat{\ell}\frac{M\cdot(M-1)}{2}+nM\left(n-\hat{\ell}\right)+\frac{\left(n-\hat{\ell}\right)\cdot\left(n-\hat{\ell}-1\right)}{2}= \\ \frac{\frac{1}{2}\hat{\ell}\cdot M^2+\left(2n^2-2\hat{\ell}n-\frac{1}{2}e\hat{l}l\right)\cdot M+n^2-2\hat{\ell}n+\hat{\ell}^2-n+\hat{\ell}}{2}.\end{gathered} \tag{6.18}$$

Combining (6.17) and (6.18) with the fact that M can be made suitably big, we have that the betweenness of $\hat{S}\cup\{u\}$ cannot be more than $\frac{1}{2}\frac{\hat{\ell}}{n^2}$. Hence, we have:

$$\frac{1}{2}\frac{\hat{\ell}}{n^2}\geq k \implies \frac{1}{2}\frac{\hat{\ell}}{n^2}\geq\frac{1}{2}\frac{\ell}{n^2} \implies \hat{\ell}\geq\ell. \tag{6.19}$$

This contradicts the original assumption, as $\hat{S}$ forms a clique and has cardinality $|\hat{S}|=\hat{\ell}\geq\ell$.

□

For the case of a representative set of a node $u\in V$, we can use the MAXIMUM BETWEENNESS CENTRALITY problem, as this was introduced by (Puzis et al., 2007b): the authors also derived the $\mathcal{NP}$-hardness of the problem by reduction from the well-known VERTEX COVER.

Definition 13 (MAXIMUM BETWEENNESS CENTRALITY)**.** *Given a graph $G(V,E)$ and an integer ℓ which subset of nodes $S \subseteq V$ of cardinality $|S| = \ell$ possesses maximum betweenness centrality?*

CHAPTER 7. MATHEMATICAL FORMULATIONS

In this chapter, we develop mathematical programming formulations for all problems, as those were defined in the previous chapter. The current chapter is organized in three main sections (one per centrality metric studied, i.e., degree, closeness, and betweenness), and each section contains formulations for all structures (i.e., representative, induced star, and clique).

7.1. Structure degree centrality

In this section, we discuss the formulations for the degree centrality version of the structures studied. Each of them is described in its own subsection, beginning with the broader representative structure of a node $u \in V$.

7.1.1. Representative structure

As discussed in Chapter 6, a set of nodes S forms a representative of $u \in V$ if in the induced subgraph of S, u has a degree of $|S|-1$. The problem of identifying the representative set of a node u with maximum degree centrality can be formulated as in (7.1). Before we proceed, let us describe the decision variables needed.

$$x_i^0 = \begin{cases} 1, & \text{if node } i \in S, \\ 0, & \text{otherwise.} \end{cases}$$

$$x_i^1 = \begin{cases} 1, & \text{if node } i \in V \text{ is adjacent to a node in } S \text{ and not in } S, \\ 0, & \text{otherwise.} \end{cases}$$

We use x_i^ℓ for convenience, as it denotes whether i is found ℓ hops away from S nodes i is. For degree, we are only interested in –at most– one hop away, leading to $\ell = 0, 1$. We proceed to give the full formulation in (7.1).

$$
\begin{aligned}
\textbf{RD(u):} \quad & \max \quad \sum_{i \in V} x_i^1 && && \text{(7.1a)} \\
& \text{s.t.} \quad x_i^0 + x_i^1 \le 1, && \forall i \in V, && \text{(7.1b)} \\
& x_i^0 \le a_{iu}, && \forall i \in V \setminus \{u\}, && \text{(7.1c)} \\
& x_i^1 \le \sum_{j:(i,j) \in E} x_j^0, && \forall i \in V, && \text{(7.1d)} \\
& x_u^0 = 1, && && \text{(7.1e)} \\
& x_i^0, x_i^1 \in \{0,1\} && \forall i \in V. && \text{(7.1f)}
\end{aligned}
$$

The objective is to maximize the cardinality of the open neighborhood, as shown in (7.1a). Then, constraint family (7.1b) ensures that no node can be both in the set and its open neighborhood. Constraints (7.1c) enforce that only nodes that are adjacent to u are considered, as otherwise the set would not be a representative of u. Further, with constraints (7.1d) we guarantee that a node belongs to the open neighborhood of set S if at least one of its neighbors is in S. Last, in (7.1e) and (7.1f) we force u to be part of the set, and restrict all variables to be binary.

7.1.2. Induced star structure

Once more, as described in Chapter 6, a set of nodes S is said to form an induced star centered at $u \in V$ if the induced subgraph of S has exactly one node (u) with degree $|S|-1$, while the remaining $|S|-1$ nodes have degrees equal to 1. The formulation is identical as the one in (7.1) with the addition of the constraint that ensures the required star structure of the set, which is added in (7.2c). The constraint does not allow two adjacent nodes to both serve as leaves in S.

$$
\begin{aligned}
\textbf{SD(u):} \quad & \max \quad \sum_{i \in V} x_i^1 && && \text{(7.2a)} \\
& \text{s.t.} \quad \text{(7.1b)–(7.1f)}, && && \text{(7.2b)} \\
& x_i^0 + x_j^0 \le 1, && \forall (i,j) \in E : i \ne u, j \ne u. && \text{(7.2c)}
\end{aligned}
$$

7.1.3. Clique structure

Finally, as a reminder from Chapter 6, a set of nodes S forms a clique if the induced subgraph of S has $|S|$ nodes of degree $|S|-1$. The formulation, with the addition of the well-known clique constraint in (7.3c), is given in (7.3).

$$\textbf{CD(u):} \quad \max \sum_{i\in V} x_i^1 \tag{7.3a}$$

$$\text{s.t.} \quad (7.1b)\text{–}(7.1f), \tag{7.3b}$$

$$x_i^0 + x_j^0 \le 1, \qquad \forall (i,j) \notin E. \tag{7.3c}$$

7.1.4. A different formulation for structure degree centrality

We can take advantage of the local considerations in degree centrality and hence, we may propose a different formulation. First, define $\overline{N}(u) = \bigcup_{v\in N(u)}(N(v)\setminus N(u))$ and $\hat{N}(u) = N(u)\cup u$. Then, we can reformulate the representative degree problem as follows:

$$\textbf{[IP2]} \quad \max \sum_{i\in\overline{N}(u)} x_i^1 - \sum_{i\in\hat{N}(u)} x_i^0 + |N(u)| \tag{7.4a}$$

$$x_i^1 \le \sum_{j\in(N(i)\cap\hat{N}(u))} x_j^0, \qquad \forall i\in\overline{N}(u), \tag{7.4b}$$

$$x_i^0, x_i^1 \in \{0,1\} \qquad \forall i \in V. \tag{7.4c}$$

The two other problems (clique and star) can be accommodated by adding

$$x_i^0 + x_j^0 \le 1 + a_{ij}, \qquad \forall i,j\in N(u), \tag{7.5}$$

and

$$x_i^0 + x_j^0 \le 2 - a_{ij}, \qquad \forall i,j\in N(u), \tag{7.6}$$

respectively.

7.2. Structure closeness centrality

We now proceed to provide the decision variables and formulations for the three structures and closeness centrality.

7.2.1. Representative structure

Following in the footsteps of Vogiatzis et al. (2015), we introduce binary variables $x_i^\ell, \forall i \in V, \forall \ell = 1, \ldots, diam(G)$ to help us calculate the distance of every node $i \in V \setminus S$ to a node in the representative set S. Formally, we have:

$$x_i^\ell = \begin{cases} 1, & \text{if node } i \in V \text{ is at most at a distance of } \ell = 0, \ldots, diam(G) \text{ from a node in } S, \\ 0, & \text{otherwise.} \end{cases}$$

Note that x_i^0 and x_i^1 are then identically defined as in the case of degree centrality. We also note that based on this definition, the number of nodes that are found at *exactly* a distance of ℓ from S is equal to $\sum_{i\in V} x_i^\ell - \sum_{i\in V} x_i^{\ell-1}$. A first formulation can now be found in (7.7).

$$\begin{aligned}
\min \quad & \frac{\sum_{\ell=1}^{diam(G)} \ell \left(\sum_{i\in V} x_i^\ell - \sum_{i\in V} x_i^{\ell-1} \right)}{|V| - \sum_{i\in V} x_i^0} && && (7.7a) \\
\text{s.t.} \quad & x_i^0 \le a_{iu}, && \forall i \in V \setminus \{u\}, && (7.7b) \\
& x_i^\ell \le \sum_{j:(i,j)\in E} x_j^{\ell-1} + x_i^{\ell-1}, && \forall i \in V, \ell = 1, \ldots, diam(G), && (7.7c) \\
& x_i^\ell \ge \frac{1}{(1+|N(i)|)} \left(\sum_{j:(i,j)\in E} x_j^{\ell-1} + x_i^{\ell-1} \right), && \forall i \in V, \ell = 1, \ldots, diam(G), && (7.7d) \\
& x_u^0 = 1, && && (7.7e) \\
& x_i^\ell \in \{0,1\}, && \forall i \in V, \ell = 0, \ldots, diam(G). && (7.7f)
\end{aligned}$$

In the formulation, the objective in (7.7a) is to minimize the average distance from S to all other nodes in $V \setminus S$. Similarly to the degree formulation, only nodes that are adjacent to u are considered candidates for the representative: this is enforced in (7.7b). Constraints (7.7c) and (7.7d) together ensure the proper and correct definition of the x_i^ℓ variables. This happens because x_i^ℓ is not allowed to be equal to 1 unless itself or a neighbor are at a distance of $\ell-1$ from S and it is strictly greater than 1 when at least one of its neighbors or itself are at a distance of $\ell-1$ hops from S. Last, (7.7e) forces u to be in the set, and (7.7f) restricts all variables to be binary.

It is easy to see that this formulation is not linear, because of the fractional nature of the objective function. We provide a linearization as follows. First, we introduce a new variable in the form of $w = 1/\Big(|V| - \sum_{i\in V} x_i^0\Big)$: by definition, $0 \leq w \leq 1$. Hence, the objective function can now be expressed as in (7.8).

$$\sum_{\ell=1}^{diam(G)} \ell\left(\sum_{i\in V} x_i^\ell w - \sum_{i\in V} x_i^{\ell-1} w\right). \tag{7.8}$$

Luckily, because $w \in [0,1]$, we can define new variables $y_i^\ell = x_i^\ell \cdot w$ and hence, the objective function is rendered as in (7.9).

$$\sum_{\ell=1}^{diam(G)} \ell\left(\sum_{i\in V} y_i^\ell - \sum_{i\in V} y_i^{\ell-1}\right). \tag{7.9}$$

Of course, y_i^ℓ and w need to be properly defined in the constraint set of the problem. This leads to the addition of the following 4 constraints:

$$y_i^l \leq x_i^l, \qquad \forall i \in V, \forall \ell = 1,\ldots,diam(G), \tag{7.10}$$

$$y_i^l \leq w, \qquad \forall i \in V, \forall \ell = 1,\ldots,diam(G), \tag{7.11}$$

$$y_i^l \geq x_i^l + w - 1, \qquad \forall i \in V, \forall \ell = 1,\ldots,diam(G), \tag{7.12}$$

$$w|V| - \sum_{i\in V} y_i^0 = 1. \tag{7.13}$$

Finally, we have the mixed integer linear program shown in (7.14).

$$\begin{aligned} \min \quad & (7.9) && && (7.14a) \\ \text{s.t.} \quad & (7.7b)\text{–}(7.7f) && && (7.14b) \\ & (7.10)\text{–}(7.13) && && (7.14c) \\ & 0 \le w \le 1 && && (7.14d) \\ & y_i^\ell \ge 0, && \forall i \in V, \forall \ell = 1, \ldots, diam(G). && (7.14e) \end{aligned}$$

7.2.2. Star and clique structures

Identically to the previous subsection on degree centrality, we need only ensure that the proper star and clique constraints are enforced. They correspond to

$$x_i^0 + x_j^0 \le 1, \forall (i,j) \in E : i \ne u, j \ne u, \tag{7.15}$$

and

$$x_i^0 + x_j^0 \le 1, \forall (i,j) \notin E, i \ne j. \tag{7.16}$$

respectively.

Before ending this subsection, we note that in this work, we consider the average distance from all other nodes to the nodes in the clique, as opposed to the minimizing the total and maximum distances that were considered in Vogiatzis et al. (2015).

7.3. Structure betweenness centrality

Prior to getting to the formulations for the different structures investigated for betweenness, we provide some necessary properties.

In the case of standard betweenness, for all three structures, an interesting property holds. Let again u be the node that will serve as the center of the star/representative, or belong to the clique we are looking for. Moreover, let γ_{ij}^k, γ_{ij}^{kl}, and γ_{ij}^{klm} be the number of shortest paths connecting i and

j that intersect a structure node $k, k \neq i, j$ (with $\gamma_{ij}^{i} = \gamma_{ij}^{j} = 0$), structure nodes $k, l, k < l, k, l \neq i, j$ (with $\gamma_{ij}^{ij} = 0, \gamma_{ij}^{ik} \neq 0$, and $\gamma_{ij}^{kj} \neq 0$), and structure nodes $k, l, m, k < l < m$, respectively. Also, let $\gamma_{ij}(S, S')$ denote the total number of shortest paths between i and j whose intersection with set S is equal to set S'. We then have the following Proposition 1, based on Rysz et al. (2018).

Proposition 1 (Based on Rysz et al. (2018)). *Given a structure $S \subseteq V$ (when S is a induced representative, star, or clique of a node u), vertices $i, j \in V \setminus S$, and $k, l, m \in S, k \neq l \neq m$, we have:*

1. $\gamma_{ij}(S, \{k, l, m\}) = \gamma_{ij}(\{k, l, m\}, \{k, l, m\}) = \gamma_{ij}^{klm}$.

   Proof. Assume for a contradiction that the left and right hand side are not equal. Then, there is some vertex $k' \in S$, such that there is a shortest path between i and j that passes through k, l, m, as well as k'. Hence, this path is counted in $\gamma_{ij}(\{k, l, m\}, \{k, l, m\})$ and is not counted in $\gamma_{ij}(S, \{k, l, m\})$ which means that $\gamma_{ij}(S, \{k, l, m\}) < \gamma_{ij}(\{k, l, m\}, \{k, l, m\})$. We know that (according to the definitions of representative, star and clique) the shortest path between any two non-structure vertices will contain at most 3 structure vertices (for representative and star, this number is at most 3, and for a clique it is at most 2). Thus, it is a contradiction and the proof is complete. □

2. $\gamma_{ij}(S, \{k, l\}) = \gamma_{ij}^{kl} - \sum_{m \in S \setminus \{k, l\}} \gamma_{ij}(\{k, l, m\}, \{k, l, m\})$.

   Proof. The set of shortest paths between i and j that contain $k, l \in S$ can be partitioned into $|S| - 1$ sets. Namely, the set of shortest paths between i and j whose intersection with S is $\{k, l\}$ and the set of shortest paths between i and j whose intersection with S is $\{k, l, m\}$ for some other vertex $m \in S, m \neq k, l$. Thus, we have that:

$$\gamma_{ij}^{kl} = \gamma_{ij}(S, \{k, l\}) + \sum_{m \in S \setminus \{k, l\}} \gamma_{ij}(S, \{k, l, m\}).$$

Using the first part of Proposition 1, we also have:

$$\gamma_{ij}(S,\{k,l\}) = \gamma_{ij}^{kl} - \sum_{m\in S\setminus\{k,l\}} \gamma_{ij}(\{k,l,m\},\{k,l,m\}),$$

which finishes this part of the proof. $\square$

3. $\gamma_{ij}(S,\{k\}) = \gamma_{ij}^{k} - \sum_{l\in S\setminus\{k\}} \gamma_{ij}^{kl} + \sum_{l,m\in S\setminus\{k\},l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\})$.

Proof. The set of shortest paths between i and j that contain $k \in S$ can be partitioned into $|S| + \binom{|S|}{2}$ sets. Namely, the set of shortest paths between i and j whose intersection with S is $\{k\}$, the set of shortest paths between i and j whose intersection with S is $\{k,l\}$ for some other $l \in S, l \neq k$, and the set of shortest paths between i and j whose intersection with S is $\{k,l,m\}$ for some other $l,m \in S, l,m \neq k, l < m$. Thus, we now have:

$$\gamma_{ij}^{k} = \gamma_{ij}(S,\{k\}) + \sum_{l\in S\setminus\{k\}} \gamma_{ij}(S,\{k,l\}) + \sum_{l,m\in S\setminus\{k\},l<m} \gamma_{ij}(S,\{k,l,m\}).$$

This last equality can be written as:

$$\gamma_{ij}(S,\{k\}) = \gamma_{ij}^{k} - \sum_{l\in S\setminus\{k\}} \gamma_{ij}(S,\{k,l\}) - \sum_{l,m\in S\setminus\{k\},l<m} \gamma_{ij}(S,\{k,l,m\}).$$

Using parts 1 and 2 of Proposition 1, we have:

$$\begin{aligned}\gamma_{ij}(S,\{k\}) = \gamma_{ij}^{k} - \sum_{l\in S\setminus\{k\}} \gamma_{ij}^{kl} + \sum_{l\in S\setminus\{k\}} \sum_{m\in S\setminus\{k,l\}} \gamma_{ij}(\{k,l,m\},\{k,l,m\}) \\ - \sum_{l,m\in S\setminus\{k\},l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\}).\end{aligned}$$

We also note that:

$$\sum_{l\in S\setminus\{k\}} \sum_{m\in S\setminus\{k,l\}} \gamma_{ij}(\{k,l,m\},\{k,l,m\}) = 2 \sum_{l,m\in S\setminus\{k\},l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\}).$$

Therefore:

$$\gamma_{ij}(S,\{k\}) = \gamma_{ij}^{k} - \sum_{l\in S\setminus\{k\}} \gamma_{ij}^{kl} + \sum_{l,m\in S\setminus\{k\},l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\}),$$

which concludes this part of the proof. □

4. $\gamma_{ij}(S) = \sum_{k\in S} \gamma_{ij}^{k} - \sum_{k,l\in S,k<l} \gamma_{ij}^{kl} + \sum_{k,l,m\in S,k<l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\})$.

Proof. As shown during the proof of part 1 of this proposition, any shortest path between $i, j \in V \setminus S$ intersects at most 3 vertices of structure S. The set of shortest paths between i and j that intersect structure S can be partitioned into three groups or $|S| + \binom{|S|}{2} + \binom{|S|}{3}$ sets. Namely, the set of shortest paths between i and j whose intersection with S is $\{k\}$ for some vertex $k \in S$, the set of shortest paths between i and j whose intersection with S is $\{k,l\}$ for some pair of vertices $k,l \in S, k < l$, and the set of shortest paths between i and j whose intersection with S is $\{k,l,m\}$ for some $k,l,m \in S, k < l < m$. Overall, we have that:

$$\gamma_{ij}(S) = \sum_{k\in S} \gamma_{ij}(S,\{k\}) + \sum_{k,l\in S,k<l} \gamma_{ij}(S,\{k,l\}) + \sum_{k,l,m\in S,k<l<m} \gamma_{ij}(S,\{k,l,m\}).$$

By parts 1, 2, and 3 of Proposition 1, we have that:

$$\begin{aligned}\gamma_{ij}(S) = &\sum_{k\in S}\left[\gamma_{ij}^{k} - \sum_{l\in S\setminus\{k\}} \gamma_{ij}^{kl} + \sum_{l,m\in S\setminus\{k\},l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\})\right] + \\ &\sum_{k,l\in S,k<l}\left[\gamma_{ij}^{kl} - \sum_{m\in S\setminus\{k,l\}} \gamma_{ij}(\{k,l,m\},\{k,l,m\})\right] + \\ &\sum_{k,l,m\in S,k<l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\}).\end{aligned}$$

This can be also written as:

$$\gamma_{ij}(S) = \sum_{k\in S}\gamma_{ij}^{k} - \sum_{k\in S}\sum_{l\in S\setminus\{k\}}\gamma_{ij}^{kl} + \sum_{k\in S}\sum_{l,m\in S\setminus\{k\},l<m}\gamma_{ij}(\{k,l,m\},\{k,l,m\})+$$

$$\sum_{k,l\in S,k<l}\gamma_{ij}^{kl} - \sum_{k,l\in S,k<l}\sum_{m\in S\setminus\{k,l\}}\gamma_{ij}(\{k,l,m\},\{k,l,m\})+$$

$$\sum_{k,l,m\in S,k<l<m}\gamma_{ij}(\{k,l,m\},\{k,l,m\}).$$

Further, note that:

$$\sum_{k\in S}\sum_{l\in S\setminus\{k\}}\gamma_{ij}^{kl} = 2\sum_{k,l\in S,k<l}\gamma_{ij}^{kl}.$$

$$\sum_{k\in S}\sum_{l,m\in S\setminus\{k\},l<m}\gamma_{ij}(\{k,l,m\},\{k,l,m\}) = 3\sum_{k,l,m\in S,k<l<m}\gamma_{ij}(\{k,l,m\},\{k,l,m\}).$$

$$\sum_{k,l\in S,k<l}\sum_{m\in S\setminus\{k,l\}}\gamma_{ij}(\{k,l,m\},\{k,l,m\}) = 3\sum_{k,l,m\in S,k<l<m}\gamma_{ij}(\{k,l,m\},\{k,l,m\})$$

In conclusion, we have:

$$\begin{aligned}\gamma_{ij}(S) = &\sum_{k\in S}\gamma_{ij}^{k} - 2\sum_{k,l\in S,k<l}\gamma_{ij}^{kl} + 3\sum_{k,l,m\in S,k<l<m}\gamma_{ij}(\{k,l,m\},\{k,l,m\})+\\ &\sum_{k,l\in S,k<l}\gamma_{ij}^{kl} - 3\sum_{k,l,m\in S,k<l<m}\gamma_{ij}(\{k,l,m\},\{k,l,m\})+\\ &\sum_{k,l,m\in S,k<l<m}\gamma_{ij}(\{k,l,m\},\{k,l,m\})\\ &= \sum_{k\in S}\gamma_{ij}^{k} - \sum_{k,l\in S,k<l}\gamma_{ij}^{kl} + \sum_{k,l,m\in S,k<l<m}\gamma_{ij}(\{k,l,m\},\{k,l,m\}).\end{aligned}$$

□

As mentioned earlier, in the case of a clique, the calculation is simplified as any shortest path between $i, j \in V\setminus S$ intersects at most 2 vertices of clique S.

7.3.1. Representative structure

The formulation for finding a representative S of a given node $u \in V$ as the center of the representative, while maximizing the betweenness centrality can be written as in (7.17).

$$\max \quad \frac{\sum\limits_{i,j \in V} \left(\sum\limits_{k \in V} \gamma_{ij}^{k} x_k^0 - \sum\limits_{k,l \in V, k<l} \gamma_{ij}^{kl} x_k^0 x_l^0 + \sum\limits_{k,l,m \in V, k<l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\}) x_k^0 x_l^0 x_m^0 \right)}{\sum\limits_{i,j \in V} \gamma_{ij}(1-x_i^0)(1-x_j^0)} \tag{7.17a}$$

$$\text{s.t.} \quad x_i^0 \leq a_{iu}, \qquad \forall i \in V \setminus \{u\} \tag{7.17b}$$

$$x_u^0 = 1, \tag{7.17c}$$

$$x_i^0 \in \{0,1\}, \qquad \forall i \in V \tag{7.17d}$$

Constraints (7.17b) ensure that only the nodes that are adjacent to u are considered as candidates to be in S. Constraint (7.17c) enforces that node u is in the representative, and finally binary variable restrictions are given in (7.17d). We now claim that the objective function in (7.17a) correctly calculates the betweenness of a set S as this was defined in (6.1). We formally show this in Proposition 2.

Proposition 2. *The objective function in* (7.17a) *correctly calculates the value of betweenness centrality for a representative S, i.e.,*

$$\begin{aligned} C^b(S) = & \frac{\sum\limits_{i,j \in V \setminus S} \gamma_{ij}(S)}{\sum\limits_{i,j \in V \setminus S} \gamma_{ij}} = \\ = & \frac{\sum\limits_{i,j \in V} \left(\sum\limits_{k \in V} \gamma_{ij}^{k} x_k^0 - \sum\limits_{k,l \in V, k<l} \gamma_{ij}^{kl} x_k^0 x_l^0 + \sum\limits_{k,l,m \in V, k<l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\}) x_k^0 x_l^0 x_m^0 \right)}{\sum\limits_{i,j \in V} \gamma_{ij}(1-x_i^0)(1-x_j^0)}. \end{aligned} \tag{7.18}$$

Proof. Using Proposition 1 and the definition of binary variables x_i^0, equation (6.1) can be written as follows:

$$C^b(S) = \frac{\sum\limits_{i,j\in V\setminus S} \gamma_{ij}(S)}{\sum\limits_{i,j\in V\setminus S} \gamma_{ij}} = \frac{\sum\limits_{i,j\in V\setminus S} \left(\sum\limits_{k\in S} \gamma_{ij}^k - \sum\limits_{k,l\in S,k<l} \gamma_{ij}^{kl} + \sum\limits_{k,l,m\in S,k<l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\}) \right)}{\sum\limits_{i,j\in V\setminus S} \gamma_{ij}}$$
$$= \frac{\sum\limits_{i,j\in V\setminus S} \left(\sum\limits_{k\in V} \gamma_{ij}^k x_k^0 - \sum\limits_{k,l\in V,k<l} \gamma_{ij}^{kl} x_k^0 x_l^0 + \sum\limits_{k,l,m\in V,k<l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\}) x_k^0 x_l^0 x_m^0 \right)}{\sum\limits_{i,j\in V\setminus S} \gamma_{ij}}. \tag{7.19}$$

To show that (7.18) is equal to (7.19), we need to show that:

$$\sum_{k\in V} \gamma_{ij}^k x_k^0 - \sum_{k,l\in V,k<l} \gamma_{ij}^{kl} x_k^0 x_l^0 + \sum_{k,l,m\in V,k<l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\}) x_k^0 x_l^0 x_m^0 = 0,\ \forall i \in S, j \in V \tag{7.20}$$

and that

$$\gamma_{ij}(1-x_i^0)(1-x_j^0) = 0,\ \forall i \in S, j \in V. \tag{7.21}$$

It is obvious that the expression in the left hand side of (7.21) is equal to zero. The first expression in (7.20) can be rewritten as:

$$\begin{aligned} &\sum_{k\in V\setminus S} \gamma_{ij}^k x_k^0 - \sum_{k,l\in V\setminus S,k<l} \gamma_{ij}^{kl} x_k^0 x_l^0 + \sum_{k,l,m\in V\setminus S,k<l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\}) x_k^0 x_l^0 x_m^0 \\ &\quad + \sum_{k\in S} \gamma_{ij}^k - \sum_{k,l\in S,k<l} \gamma_{ij}^{kl} + \sum_{k,l,m\in S,k<l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\}) = 0 \quad \forall i \in S, j \in V \end{aligned} \tag{7.22}$$

Since $x_i^0 = 0, \forall i \in V \setminus S$, we can write (7.22) as:

$$\sum_{k\in S} \gamma_{ij}^k - \sum_{k,l\in S,k<l} \gamma_{ij}^{kl} + \sum_{k,l,m\in S,k<l<m} \gamma_{ij}(\{k,l,m\},\{k,l,m\}) = 0 \quad \forall i \in S, j \in V \tag{7.23}$$

Hence, we simply need to show that the left hand side of (7.23) is equal to zero. We distinguish between two cases for any node j:

1. node j belongs to the representative: $j \in S$.

   Since S is a representative there are three cases:

   (a) i or j is the center of the representative: In this case, there is only one shortest path between i and j and it goes through edge $(i,j) \in E_s$. Thus, $\gamma_{ij}^{k} = 0$ for all $k \in S$, $\gamma_{ij}^{kl} = 0$ for all $k,l \in S, k < l$, and $\gamma_{ij}^{klm} = 0$ for all $k,l,m \in S, k < l < m$ and (7.23) is true.

   (b) both i and j are leaf nodes of the representative S and they are not connected. In this case there are two possible subcases: 1) the shortest paths between i and j go through other structure nodes $p \in S$ so that $p \in N(i) \cap N(j)$. We suppose that the number of these shortest paths are $n_1, 1 \le n_1 \le |S| - 2$. Thus, $\sum_{k\in S} \gamma_{ij}^{k} = n_1, \sum_{k,l\in S,k<l} \gamma_{ij}^{kl} = 2n_1$, and $\sum_{k,l,m\in S,k<l<m} \gamma_{ij}^{klm} = n_1$ and (7.23) is true. 2) in addition to the shortest paths between i and j that go through structure nodes (at least 1 path), there are other shortest paths that pass through nodes $q \in V \setminus S$ so that $q \in N(i) \cap N(j)$. We suppose that the number of these shortest paths are $n_2, 1 \le n_2 \le |V| - |S|$. Therefore, $\sum_{k\in S} \gamma_{ij}^{k} = n_1, \sum_{k,l\in S,k<l} \gamma_{ij}^{kl} = 2n_1$, and $\sum_{k,l,m\in S,k<l<m} \gamma_{ij}^{klm} = n_1$ and (7.23) is true.

   (c) both i and j are leaf nodes of the representative S and they are connected through edge $(i,j) \in E_S$. Hence,there is only one shortest path between i or j and it goes through edge $(i,j) \in E_s$, so $\sum_{k\in S} \gamma_{ij}^{k} = 0, \sum_{k,l\in S,k<l} \gamma_{ij}^{kl} = 0$, and $\sum_{k,l,m\in S,k<l<m} \gamma_{ij}^{klm} = 0$ and (7.23) is true.

2. node j does not belong to the representative: $j \in V \setminus S$.

   There are four possible cases:

   (a) If geodesic path between i and j does not intersect other representative nodes, then $\sum_{k\in S} \gamma_{ij}^{k} = 0, \sum_{k,l\in S,k<l} \gamma_{ij}^{kl} = 0$, and $\sum_{k,l,m\in S,k<l<m} \gamma_{ij}^{klm} = 0$ and (7.23) is true.

(b) if the shortest path between i and j intersects only one other structure node which is connected to node i, then $\sum_{k\in S}\gamma_{ij}^{k}=1, \sum_{k,l\in S,k<l}\gamma_{ij}^{kl}=1$, and $\sum_{k,l,m\in S,k<l<m}\gamma_{ij}^{klm}=0$ and (7.23) is true.

(c) if the shortest path between i and j intersects one other structure node p which is not connected to node i, in this case the shortest path between i and j goes through node $t, t\in N(i)\cap N(p) : t\in V\setminus S$. Thus, $\sum_{k\in S}\gamma_{ij}^{k}=3, \sum_{k,l\in S,k<l}\gamma_{ij}^{kl}=4$, and $\sum_{k,l,m\in S,k<l<m}\gamma_{ij}^{klm}=1$ and (7.23) is true.

(d) if the shortest path between i and j intersects only two nodes of the representative S, then $\sum_{k\in S}\gamma_{ij}^{k}=2, \sum_{k,l\in S,k<l}\gamma_{ij}^{kl}=3$, and $\sum_{k,l,m\in S,k<l<m}\gamma_{ij}^{klm}=1$ and (7.23) is true.

This concludes the proof. □

The formulation is nonlinear due to the objective function in (7.17a). We proceed to show the linearization of this formulation in the remainder of the subsection. Let us consider new variable $w=\frac{1}{\sum\limits_{i,j\in V}\gamma_{ij}(1-x_i^0)(1-x_j^0)}$, then we have that $0\le w\le 1$, and we can write the objective function as:

$$\sum_{i,j\in V}\left(\sum_{k\in V}\gamma_{ij}^{k}x_k^0w-\sum_{k,l\in V,k<l}\gamma_{ij}^{kl}x_k^0x_l^0w+\sum_{k,l,m\in V,k<l<m}\gamma_{ij}(\{k,l,m\},\{k,l,m\})x_k^0x_l^0x_m^0w\right).$$

If we introduce new decision variables $y_k=x_k^0w, z_{kl}=x_k^0x_l^0w$ and $r_{klm}=x_k^0x_l^0x_m^0w$, then the objective function will become:

$$\sum_{i,j\in V}\left(\sum_{k\in V}\gamma_{ij}^{k}y_k-\sum_{k,l\in V,k<l}\gamma_{ij}^{kl}z_{kl}+\sum_{k,l,m\in V,k<l<m}\gamma_{ij}(\{k,l,m\},\{k,l,m\})r_{klm}\right). \tag{7.24}$$

We need to add the following constraints to properly define decision variable w in the model:

$$\begin{aligned}
&w = \frac{1}{\sum\limits_{i,j \in V} \gamma_{ij}(1-x_i^0)(1-x_j^0)} \\
&w \sum_{i,j \in V} \gamma_{ij}(1-x_i^0)(1-x_j^0) = 1 \\
&\sum_{i,j \in V} \gamma_{ij}(w - y_i - y_j + z_{ij}) = 1 \qquad (7.25)
\end{aligned}$$

Thus, the final formulation for maximizing betweenness will be as in (7.26).

$$\begin{aligned}
[BCRP] \quad & \max \quad (7.24) && && (7.26a)\\
& \text{s.t.} \quad (7.17b)\text{–}(7.17d), (7.25) && && (7.26b)\\
& y_i \le x_i^0, y_i \le w, y_i \ge x_i^0 + w - 1, && \forall i \in V && (7.26c)\\
& z_{ij} \le x_i^0, z_{ij} \le x_j^0, z_{ij} \le w, z_{ij} \ge x_i^0 + x_j^0 + w - 2, && \forall i, j \in V && (7.26d)\\
& r_{ijk} \le x_i^0, r_{ijk} \le x_j^0, r_{ijk} \le x_k^0, r_{ijk} \le w, r_{ijk} \ge x_i^0 + x_j^0 + x_k^0 + w - 3, && \forall i, j, k \in V && (7.26e)\\
& y_i, z_{ij}, r_{ijk}, w \in [0,1], && \forall i, j, k \in V, && (7.26f)
\end{aligned}$$

7.3.1.1. Star and clique structures

Once again, we may add the same structure constraints as in the previous corresponding subsections (see, e.g., subsection 7.2.2) in the formulation of (7.26).

CHAPTER 8. COMBINATORIAL BRANCH-AND-BOUND

In this chapter, we propose a CBB algorithm to solve the problems defined. We will later (in Chapter 13) compare its performance against using a commercial solver to solve the formulations from Chapter 7.

For the rest of this chapter, we distinguish between the terms vertex and node, which are usually considered the same and are used interchangeably earlier; we specify that vertex refers to a vertex in a given graph G, and node refers to a node in the search tree. Once more, we break down the discussion in subsections depending on the centrality metrics. In the first subsection, we focus on betweenness centrality. For each centrality metric, we discuss the general search tree structure, and then the specifics for each of the structures: representatives, induced stars, and cliques.

8.1. Interesting properties

Prior to presenting the CBB approaches for each of the problem, we provide some interesting properties.

Theorem 5 (Hereditary property of cliques)**.** *Let S_i^* be the clique of maximum centrality containing node i, and let $j \neq i$ belong to S_i^*. Then, if $i \in S_j^*$, we must have $S_i^* = S_j^*$.*

Proof. Assume for a contradiction that $i \in S_j^*$ and $j \in S_i^*$, but $S_i^* \neq S_j^*$. Then, we distinguish between two cases: (i) $\exists k : k \in S_i^* \setminus S_j^*$ and (ii) $\exists k : k \in S_j^* \setminus S_i^*$. Finally, let $C(S)$ be the centrality value of clique S. We will show that we can build S_j^* starting from S_i^* and, hence, S_i^* cannot be optimal.

In the first case, observe that $S_i^* \setminus k$ forms a clique, as we are simply removing a node from the set. Similarly, in the second case, $S_i^* \cup \{k\}$ satisfies all requirements for a clique, as both $k \in S_j^*$ and $i \in S_j^*$, and hence $(i,k) \in E$. Finally, adding and removing every node in the cases above leads to transforming S_i^* to S_j^*, which, by assumption, has a higher centrality value $C(S_j^*)$. This contradicts the optimality of S_i^*.

□

Theorem 6 (Lower and upper bound for the representative problem)**.** *Let C_i^*, S_i^*, and R_i^* be the clique, star, and representative of maximum centrality containing node i, respectively. Then, for the*

MCSP(i) problems we have that $C^c(R_i^*) \leq C^c(C_i^*)$ and $C^c(R_i^*) \leq C^c(S_i^*)$. Moreover, for MDSP$(i)$ and MBSP(i) problems we have $C^d(R_i^*) \geq C^d(C_i^*)$ and $C^b(R_i^*) \geq C^b(S_i^*)$.

Proof. Based on the definitions of a clique, star, and representative, any clique or star of vertex $i \in V$ is also a representative of node i; the opposite does not necessarily hold. $\square$

8.2. Structure betweenness centrality

8.2.1. Properties of the betweenness centrality measure

Based on Rysz et al. (2018), we propose an important observation for updating the betweenness centrality of a structure, when its cardinality is increased by one. This result is used in the CBB approach to calculate the betweenness centrality of a child node given the betweenness of its parent, as well as facilitate the calculation of its upper bound. We now state the proposition.

Proposition 3. *Given a structure $S \subseteq V$, and a vertex $p \in V \setminus S$ such that $S \cup \{p\}$ satisfies the properties of the structure, we have that:*

$$C^b(S \cup \{p\}) = \frac{\sum_{i,j \in V \setminus S} \gamma_{ij}(S) - \sum_{j \in V \setminus (S \cup \{p\})} \left[\sum_{k \in S} \gamma_{pj}^k - \sum_{k,l \in S, k<l} \gamma_{pj}^{kl} \right]}{\sum_{i,j \in V \setminus S} \gamma_{ij} - \sum_{j \in V \setminus (S \cup \{p\})} \gamma_{pj}} + \frac{\sum_{i,j \in V \setminus (S \cup \{p\}), i<j} \left[\gamma_{ij}^p - \sum_{l \in S} \gamma_{ij}^{pl} + \sum_{l,m \in S, l<m} \gamma_{ij}^{plm} \right]}{\sum_{i,j \in V \setminus S} \gamma_{ij} - \sum_{j \in V \setminus (S \cup \{p\})} \gamma_{pj}}. \tag{8.1}$$

Proof. Adding p to S, we have a new structure $S \cup \{p\}$. The new structure's betweenness differs from S because the number of the shortest paths that intersect the new structure $S \cup \{p\}$ (numerator) and the total number of shortest paths between $i, j : i, j \in V \setminus (S \cup \{p\})$ (denominator) are both changed. Thus, we need to update the betweenness of $S \cup \{p\}$ using the betweenness of S in a way that captures these changes.

The total number of shortest paths between p and $j, j \in V \setminus (S \cup \{p\})$ that intersect S is subtracted from the numerator. On the other hand, the total number of shortest paths between $i, j : i, j \notin (S \cup \{p\})$ whose intersection with $S \cup \{p\}$ is p is added to the numerator. Finally, the total number of shortest paths between $p, j : j \in V \setminus (S \cup \{p\})$ is subtracted from the denominator. The following equation of (8.2) shows these changes.

$$C^b(S \cup \{p\}) = \frac{\sum\limits_{i,j \in V \setminus S} \gamma_{ij}(S) - \sum\limits_{j \in V \setminus (S \cup \{p\})} \gamma_{pj}(S) + \sum\limits_{i,j \in V \setminus (S \cup \{p\}), i<j} \gamma_{ij}(S \cup \{p\}, \{p\})}{\sum\limits_{i,j \in V \setminus S} \gamma_{ij} - \sum\limits_{j \in V \setminus (S \cup \{p\})} \gamma_{pj}}. \tag{8.2}$$

Using Parts 3 and 4 from Proposition 1, we also have that:

$$C^b(S \cup \{p\}) = \frac{\sum\limits_{i,j \in V \setminus S} \gamma_{ij}(S) - \sum\limits_{j \in V \setminus (S \cup \{p\})} \left[\sum\limits_{k \in S} \gamma^k_{pj} - \sum\limits_{k,l \in S, k<l} \gamma^{kl}_{pj} + \sum\limits_{k,l,m \in S, k<l<m} \gamma^{klm}_{pj} \right]}{\sum\limits_{i,j \in V \setminus S} \gamma_{ij} - \sum\limits_{j \in V \setminus (S \cup \{p\})} \gamma_{pj}}$$
$$+ \frac{\sum\limits_{i,j \in V \setminus (S \cup \{p\}), i<j} \left[\gamma^p_{ij} - \sum\limits_{l \in S} \gamma^{pl}_{ij} + \sum\limits_{l,m \in S, l<m} \gamma^{plm}_{ij} \right]}{\sum\limits_{i,j \in V \setminus S} \gamma_{ij} - \sum\limits_{j \in V \setminus (S \cup \{p\})} \gamma_{pj}}. \tag{8.3}$$

We note that since vertex p forms a structure with set S, then $\sum_{k,l,m \in S, k<l<m} \gamma^{klm}_{pj} = 0$ for any $j \in V \setminus (S \cup \{p\})$. Therefore, because of (8.3) we have:

$$C^b(S \cup \{p\}) = \frac{\sum\limits_{i,j \in V \setminus S} \gamma_{ij}(S) - \sum\limits_{j \in V \setminus (S \cup \{p\})} \left[\sum\limits_{k \in S} \gamma^k_{pj} - \sum\limits_{k,l \in S, k<l} \gamma^{kl}_{pj} \right]}{\sum\limits_{i,j \in V \setminus S} \gamma_{ij} - \sum\limits_{j \in V \setminus (S \cup \{p\})} \gamma_{pj}}$$
$$+ \frac{\sum\limits_{i,j \in V \setminus (S \cup \{p\}), i<j} \left[\gamma^p_{ij} - \sum\limits_{l \in S} \gamma^{pl}_{ij} + \sum\limits_{l,m \in S, l<m} \gamma^{plm}_{ij} \right]}{\sum\limits_{i,j \in V \setminus S} \gamma_{ij} - \sum\limits_{j \in V \setminus (S \cup \{p\})} \gamma_{pj}}. \tag{8.4}$$

As a note, when structure S is a clique, then $\sum_{k,l \in S, k<l} \gamma^{kl}_{pj} = 0$ for any $j \in V \setminus (S \cup \{p\})$ and $\sum_{l,m \in S, l<m} \gamma^{plm}_{ij} = 0$ for any $i, j \in V \setminus (S \cup \{p\}), i, j$. Therefore, by (8.4) we have:

$$C^b(S \cup \{p\}) = \frac{\sum\limits_{i,j \in V \setminus S} \gamma_{ij}(S) - \sum\limits_{j \in V \setminus (S \cup \{p\})} \left[\sum\limits_{k \in S} \gamma^k_{pj} \right] + \sum\limits_{i,j \in V \setminus S \cup \{p\}, i<j} \left[\gamma^p_{ij} - \sum\limits_{l \in S : (p,l) \in E} \gamma^{pl}_{ij} \right]}{\sum\limits_{i,j \in V \setminus S} \gamma_{ij} - \sum\limits_{j \in V \setminus (S \cup \{p\})} \gamma_{pj}}. \tag{8.5}$$

□

Lemma 1. *Given structure $S \subseteq V$ of vertex u, $i, j \in V \setminus S$, and $k \in S$, if $d(i,j) < d(u,j)$, then $\gamma_{ij}(S) = 0$.*

Proof. Suppose there exists a shortest path between i and j that passes through S. There are two cases:

1. this shortest path goes through u, then $d(i,j) > d(u,j)$, which is a contradiction.
2. this shortest path between i and j does not pass through u and passes through k, we have $d(i,j) > d(k,j)$ and since u is reachable from k in one hop, $d(u,j) = d(k,j)+1$. So, we have $d(u,j)-1 < d(i,j)$ or $d(u,j) \leq d(i,j)$ which is also a contradiction.

□

Based on Lemma 1 we can rewrite Proposition 3 as follows:

$$C^b(S \cup \{p\}) = \frac{\sum_{i,j \in V \setminus S} \gamma_{ij}(S) - \sum_{j \in V \setminus (S \cup \{p\})} \left[\sum_{k \in S} \gamma_{pj}^{k} - \sum_{k,l \in S, k<l} \gamma_{pj}^{kl} \right]}{\sum_{i,j \in V \setminus S} \gamma_{ij} - \sum_{j \in V \setminus S \cup \{p\}} \gamma_{pj}} + \frac{\sum_{i,j \in V \setminus (S \cup \{p\}), i<j, d(i,j) \geq d(u,j)} \left[\gamma_{ij}^{p} - \sum_{l \in S} \gamma_{ij}^{pl} + \sum_{l,m \in S, l<m} \gamma_{ij}^{plm} \right]}{\sum_{i,j \in V \setminus S} \gamma_{ij} - \sum_{j \in V \setminus (S \cup \{p\})} \gamma_{pj}}. \tag{8.6}$$

8.2.2. Search tree structure

For the CBB approach, we propose the following tree structure to identify feasible solutions for the $MBSP(u)$ problem that are candidates to become an optimal solution. Let each node l of the search tree consist of three sets of nodes: (i) a set of structure vertices denoted by S_l, (ii) a set of candidate vertices denoted by C_l, and (iii) a set B_l containing the betweenness centrality of all vertices $j \in C_l$ with respect to set $V \setminus (S_l \cup \{j\})$. The set of structure vertices S_l contains all vertices that are in the structure currently in construction along with the unique search path originating at the root node of the search tree and ending at the present node l. The set of candidate vertices C_l contains all vertices in $N(u) \setminus S_l$ with each member able to be added to S_l and still satisfy the corresponding structure constraint (star, or clique) to form a larger structure (for representative we do not have any constraint).

Now, the reason for having set B_l in a tree node l is twofold. First, it serves to calculate a valid upper bound on the betweenness centrality of any solution within the subtree rooted at node l (by Proposition 6), as we need $C^b_{V\setminus(S_l \cup \{j\})}(\{j\}), \forall j \in C_l$. Secondly, we use it to find the branching vertex at search node l, which is the vertex $i = \text{argmax}_{j\in C_l}\{C^b_{V\setminus(S_l\cup\{j\})}(\{j\})\}$. To avoid building the same structure, while branching on a tree node l, elements of C_l associated with already created children nodes of node l will not be in the candidate sets of all other children nodes of node l.

A depth-first-search (DFS) strategy is utilized for creating nodes to quickly obtain a good feasible solution. To create a child node of a node l, namely node $l' \in ch(l)$, a previously unselected branching vertex $i \in C_l$ that is associated with the maximum value of $C^b_{V\setminus(S_l\cup\{i\})}(\{i\})$ is chosen. Next, the set $S_{l'}$ is formed by adding vertex i to S_l, and the set $C_{l'}$ is constructed by removing vertex i from C_l. In the clique problem, all vertices in C_l that are not adjacent to vertex i, and in the star problem all the vertices in C_l that are adjacent to vertex i, are also removed from C_l. The set $B_{l'}$ is updated using equation (8.21). Since set $S_l \cup \{i\}$ forms a structure, Proposition 3 can be used to calculate $C^b(S_{l'})$ by updating $C^b(S_l)$. A valid upper bound on the betweenness centrality of node l' is calculated using Proposition 6.

In the clique problem of vertex u, we take advantage of the Theorem 5 to make the searching procedure more efficient. While branching on vertex $i \in C_l$, if we know the optimal clique of vertex i, C^*_i and $u \in C^*_i$, then tree node l is fathomed and the incumbent solution is updated by $C^b(C^*_i)$ if necessary. For this purpose, for each vertex $i \in V$ the optimal clique C^*_i and it's betweenness centrality $C^b(C^*_i)$ are stored in the $optimalcliques$ set.

As our centrality measure C^b is not monotone, after creating each child node the incumbent solution is updated, if necessary. A tree node l is fathomed by feasibility if C_l is empty and the incumbent solution is updated if necessary. Otherwise, it can be fathomed by bound. Once all nodes of the search tree have been explored, the betweenness centrality of the incumbent solution is equal to the maximum betweenness centrality of a structure of node u. The proposed CBB algorithm for *most betweenness-central clique of vertex u problem, MBCP(u),* is presented in Algorithm 2. For the star and representative cases, we cannot take advantage of Theorem 5. For the *most betweenness-central representative of vertex u problem, MBRP(u),* we can improve the performance of the algorithm using Theorem 6. Hence, for a given vertex u, first we solve MBCP(u) and MBSP(u), and later, in the root node of the MBRP(u), if $C^b(S^*_u) > C^b(C^*_u)$ holds, then the incumbent is set as the S^*_u; otherwise, it is set as the set C^*_u, where S^*_u is the optimal star of vertex u.

8.2.3. Upper bound

An important property of a centrality measure is monotonicity. According to the definition, a centrality measure is monotone if its value cannot deteriorate by including additional vertices to the set over which it is determined. Namely, a centrality measure $\mathcal{C}(\cdot)$ is monotone if and only if $\mathcal{C}(S') \leq \mathcal{C}(S)$ for any sets $S', S \subseteq V$ such that $S' \subseteq S$. Note that the inequality direction is switched if lower values of centrality are “better”. In this respect, Proposition (8.2.3) shows that group betweenness centrality of the Definition 3 is not monotone and Definition 14 propose a monotone betweenness centrality measure with a bigger betweenness value than our centrality measure.

Proposition 4. *The group betweenness centrality presented in Definition 3 is not monotone.*

Proof. The graph shown in Figure 6.3 showcases a counterexample to the monotonicity of the betweenness centrality measure presented in Definition 3. In this example, $C^b(\{u,1\}) = 0.766$ and if we increase the group size by including vertex 2, its betweenness centrality decreases to $C^b(\{u,1,2\}) = 0.755$. Thus, we can say this betweenness centrality measure is not monotone. $\square$

Algorithm 2: CBB algorithm for MBCP(u).

```
$optimalcliques = \emptyset$;
for $u \in V$ do
    Create root node: $S_0 \leftarrow u; C_0 \leftarrow N(u); B_0 \leftarrow \{(k, C^b_{V\setminus(S_l\cup\{k\})}(\{k\}), \forall k \in C_0\}; stack \leftarrow$
      $(u, S_0, C_0, B_0); C^*_u \leftarrow S_0; C^{b^*} \leftarrow C^b(S_0)$;
    while $stack \neq \emptyset$ do
        $l \leftarrow stack.pop()$;
        if $C_l \neq \emptyset$ then
            if $Upperbound(S_l) > C^{b^*}$ then
                $i = \text{argmax}_{j\in C_l}\{C^b_{V\setminus(S_l\cup\{j\})}(\{j\})\}$;
                $C_l \leftarrow C_l \setminus \{i\}$;
                if $i \in optimalcliques\ and\ u \in C^*_i$ then
                    if $C^b(C^*_i) > C^{b^*}$ then
                        $C^{b^*} \leftarrow C^b(C^*_i)$ ;
                        $C^*_u \leftarrow C^*_i$ ;
                    end
                else
                    $S_{l'} \leftarrow S_l \cup \{i\}$;
                    $C_{l'} \leftarrow \{j \in C_l : \{j\} \cup S_{l'}$ forms a new clique$\}$;
                    $B_{l'} \leftarrow$ updated $B_l$ using equation (8.21);
                    $stack.append(S_{l'})$;
                    if $C^b(S_l) > C^{b^*}$ then
                        $C^{b^*} \leftarrow C^b(S_l)$ ;
                        $C^*_u \leftarrow S_l$ ;
                    end
                end
            end
        else
            if $C^b(S_l) > C^{b^*}$ then
                $C^{b^*} \leftarrow C^b(S_l)$ ;
                $C^*_u \leftarrow S_l$ ;
            end
        end
    end
    return $C^{b^*}, C^*_u$;
    $optimalcliques \leftarrow (u, C^*_u, C^{b^*})$;
end
```

Definition 14. *Given sets $S \subseteq V$ and $S' \subseteq V$, the betweenness centrality of S with respect to S' defined as follows is monotone:*

$$\mathcal{C}^b_{S'}(S) = \frac{\sum\limits_{i,j \in S', i<j} \gamma_{ij}(S)}{\sum\limits_{i,j \in S', i<j} \gamma_{ij}}. \tag{8.7}$$

If $S' = V$, then $\mathcal{C}^b_{S'}(S)$ is referred to as the monotone betweenness centrality of set S and we write it as $\mathcal{C}^b(S)$.

Lemma 2. *Given sets $S \subseteq V, S' \subseteq V$, and $S' \subseteq S$, we have:*

$$\mathcal{C}^b(S') \leq \mathcal{C}^b(S) \tag{8.8}$$

which means, centrality measure $\mathcal{C}^b(\cdot)$ is monotone.

Proof. The denominator of both $\mathcal{C}^b(S')$ and $\mathcal{C}^b(S)$ is $\sum\limits_{i,j \in V, i<j} \gamma_{ij}$, so we need to prove that $\sum\limits_{i,j \in V, i<j} \gamma_{ij}(S') \leq \sum\limits_{i,j \in V, i<j} \gamma_{ij}(S)$. Suppose that $\sum\limits_{i,j \in V, i<j} \gamma_{ij}(S') > \sum\limits_{i,j \in V, i<j} \gamma_{ij}(S)$, which means $\exists\, i, j \in V$ such that the shortest path between them does path S' but not S. This last statement is a contradiction to $S' \subseteq S$, which completes the proof. □

Proposition 5. *Given set $S \subseteq V$, we have*

$$C^b(S) \leq \mathcal{C}^b(S) \tag{8.9}$$

Proof. Note that the set of pairs of vertices $i, j \in V (i < j)$ can be partitioned into three sets: s_1 : pairs of vertices in S, s_2 : pairs of vertices outside S, and s_3 : pairs of vertices with one vertex in S and another vertex in $V \setminus S$. Thus we can write $\mathcal{C}^b(S)$ as follows:

$$\mathcal{C}^b(S) = \frac{\sum\limits_{i,j \in V, i<j} \gamma_{ij}(S)}{\sum\limits_{i,j \in V, i<j} \gamma_{ij}} = \frac{\sum\limits_{i,j \in S, i<j} \gamma_{ij}(S) + \sum\limits_{i,j \in V \setminus S, i<j} \gamma_{ij}(S) + \sum\limits_{i \in V \setminus S, j \in S, i<j} \gamma_{ij}(S)}{\sum\limits_{i,j \in S, i<j} \gamma_{ij} + \sum\limits_{i,j \in V \setminus S, i<j} \gamma_{ij} + \sum\limits_{i \in V \setminus S, j \in S, i<j} \gamma_{ij}} \tag{8.10}$$

Therefore, we have

$$\begin{aligned}\mathcal{C}^b(S) - C^b(S) &= \frac{\sum\limits_{i,j\in S, i<j}\gamma_{ij}(S) + \sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij}(S) + \sum\limits_{i\in V\setminus S, j\in S, i<j}\gamma_{ij}(S)}{\sum\limits_{i,j\in S, i<j}\gamma_{ij} + \sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij} + \sum\limits_{i\in V\setminus S, j\in S, i<j}\gamma_{ij}} - \frac{\sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij}(S)}{\sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij}} \\ &= \frac{\sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij}\left(\sum\limits_{i,j\in S, i<j}\gamma_{ij}(S) + \sum\limits_{i\in V\setminus S, j\in S, i<j}\gamma_{ij}(S)\right) - \sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij}(S)\left(\sum\limits_{i,j\in S, i<j}\gamma_{ij} + \sum\limits_{i\in V\setminus S, j\in S, i<j}\gamma_{ij}\right)}{\left(\sum\limits_{i,j\in S, i<j}\gamma_{ij} + \sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij} + \sum\limits_{i\in V\setminus S, j\in S, i<j}\gamma_{ij}\right)\sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij}}\end{aligned} \tag{8.11}$$

Since $\sum\limits_{i,j\in S, i<j}\gamma_{ij}(S) = \sum\limits_{i,j\in S, i<j}\gamma_{ij}$ and $\sum\limits_{i\in V\setminus S, j\in S, i<j}\gamma_{ij}(S) = \sum\limits_{i\in V\setminus S, j\in S, i<j}\gamma_{ij}$, we have

$$\mathcal{C}^b(S) - C^b(S) = \frac{\left(\sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij} - \sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij}(S)\right)\left(\sum\limits_{i,j\in S, i<j}\gamma_{ij} + \sum\limits_{i\in V\setminus S, j\in S, i<j}\gamma_{ij}\right)}{\left(\sum\limits_{i,j\in S, i<j}\gamma_{ij} + \sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij} + \sum\limits_{i\in V\setminus S, j\in S, i<j}\gamma_{ij}\right)\sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij}} \geq 0 \tag{8.12}$$

(8.19) holds, because $\sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij} \geq \sum\limits_{i,j\in V\setminus S, i<j}\gamma_{ij}(S)$, which complete the proof. □

Corollary 1. *Let S be the optimal solution of $MBSP(u)$. Using Proposition 5, we have*

$$C^b(S) \leq \mathcal{C}^b(S) \tag{8.13}$$

As centrality measure $\mathcal{C}^b(\cdot)$ is monotone, the monotone betweenness centrality of any structure of vertex u is less than or equal to the betweenness centrality of the maximal structure containing node u. Thus, considering $\hat{S}$ as the maximal structure of node u

$$\mathcal{C}^b(S) \leq \mathcal{C}^b(\hat{S}) \tag{8.14}$$

and from constraints (8.13) *and* (8.14)*, we conclude that*

$$C^b(S) \leq \mathcal{C}^b(\hat{S}) \tag{8.15}$$

Based on Corollary 1, a valid upper bound on the centrality measure of Definition 3 for any structure containing vertex u is the monotone centrality measure of the maximal structure containing vertex u.

Proposition 6. *For a search tree node t with structure set S_t and nonempty candidate set C_t, the betweenness centrality of any structure containing S_t is less than or equal to*

$$\frac{\left(|S_t|+|C_t|\right)\left(\frac{|S_t|+|C_t|-1}{2}\right)+\sum\limits_{i\in V\setminus H_t, j\in H_t, i<j}\gamma_{ij}+\sum\limits_{i,j\in V\setminus S_t, i<j}\gamma_{ij}(S_t)+\sum\limits_{k\in C_t}\sum\limits_{i,j\in V\setminus(S_t\cup\{k\})}\gamma_{ij}(k)}{\sum\limits_{i,j\in V, i<j}\gamma_{ij}}, \tag{8.16}$$

where $H_t = S_t \cup C_t$.

Proof of Proposition 6. Let M be a maximal structure containing S_t. Then, since $M \subseteq S_t + C_t$, based on Lemma 2 $\mathcal{C}^b(M) \leq \mathcal{C}^b(S_t + C_t)$. Note that the numerator of the centrality $\mathcal{C}^b(H_t)(H_t = S_t + C_t)$ equals to the sum of the number of pairs of vertices in H_t, plus sum of the number of shortest paths between pairs of vertices with one vertex in H_t and another vertex in $V \setminus H_t$, plus sum of the number of shortest paths between pairs of vertices in $V \setminus H_t$ passing through H_t:

$$\begin{aligned}\mathcal{C}^b(H_t) &= \frac{\sum\limits_{i,j\in H_t, i<j}\gamma_{ij}(H_t)+\sum\limits_{i\in V\setminus H_t, j\in H_t, i<j}\gamma_{ij}(H_t)+\sum\limits_{i,j\in V\setminus H_t, i<j}\gamma_{ij}(H_t)}{\sum\limits_{i,j\in H_t, i<j}\gamma_{ij}+\sum\limits_{i\in V\setminus H_t, j\in H_t, i<j}\gamma_{ij}+\sum\limits_{i,j\in V\setminus H_t, i<j}\gamma_{ij}} \\ &\leq \frac{\left(|S_t|+|C_t|\right)\left(\frac{|S_t|+|C_t|-1}{2}\right)+\sum\limits_{i\in V\setminus H_t, j\in H_t, i<j}\gamma_{ij}+\sum\limits_{i,j\in V\setminus H_t, i<j}\gamma_{ij}(H_t)}{\sum\limits_{i,j\in V, i<j}\gamma_{ij}}\end{aligned} \tag{8.17}$$

based on Lemma 1.2 in Rysz et al., 2018,

$$\begin{aligned}\sum_{i,j\in V\setminus H_t, i<j}\gamma_{ij}(H_t) &\leq \sum_{i,j\in V\setminus H_t, i<j}\gamma_{ij}(S_t)+\sum_{i,j\in V\setminus H_t, i<j}\gamma_{ij}(H_t\setminus S_t) \\ &\leq \sum_{i,j\in V\setminus S_t, i<j}\gamma_{ij}(S_t)+\sum_{k\in C_t}\sum_{i,j\in V\setminus(S_t\cup\{k\})}\gamma_{ij}(k)\end{aligned} \tag{8.18}$$

By (8.17) and (8.18) we have the following expression

$$\mathcal{C}^b(H_t) \leq \frac{\left(|S_t|+|C_t|\right)\left(\frac{|S_t|+|C_t|-1}{2}\right)+\sum\limits_{i\in V\setminus H_t, j\in H_t, i<j}\gamma_{ij}+\sum\limits_{i,j\in V\setminus S_t, i<j}\gamma_{ij}(S_t)+\sum\limits_{k\in C_t}\sum\limits_{i,j\in V\setminus(S_t\cup\{k\})}\gamma_{ij}(k)}{\sum\limits_{i,j\in V, i<j}\gamma_{ij}} \tag{8.19}$$

Based on Rysz et al., 2018, for the clique problem we can use the graph coloring algorithm of Tomita et al., 2006 to find the maximal cliques of $G[C_t]$. If we show maximal clique of $G[C_t]$ by C_t', then the upper bound for clique problem is as follows:

$$\frac{\left(|S_t|+|C_t'|\right)\left(\frac{|S_t|+|C_t'|-1}{2}\right)+\sum\limits_{i\in V\setminus H_t, j\in H_t, i<j}\gamma_{ij}+\sum\limits_{i,j\in V\setminus S_t, i<j}\gamma_{ij}(S_t)+\sum\limits_{k\in C_t}\sum\limits_{i,j\in V\setminus(S_t\cup\{k\})}\gamma_{ij}(k)}{\sum\limits_{i,j\in V, i<j}\gamma_{ij}} \tag{8.20}$$

which is smaller than (8.19), because $|C_t'| \leq |C_t|$. □

The value of $\sum\limits_{i,j\in V\setminus S_t, i<j}\gamma_{ij}(S_t)$ is the numerator of $C^b(S_t)$ which in each tree node t is calculated by updating the value of $C^b(S_P)$, where p is the parent node of node t in the search tree, using Proposition 3. To calculate the value of $\sum\limits_{k\in C_t}\sum\limits_{i,j\in V\setminus(S_t\cup\{k\})}\gamma_{ij}(k)$ in each tree node t, we update the value of $C^b_{V\setminus(S_t\cup\{k\})}(\{k\}), \forall k\in C_t$ using the following equation

$$C^b_{V\setminus(S_t\cup\{k\})}(\{k\}) = \frac{\sum\limits_{i,j\in V\setminus(S_p\cup\{k\})}\gamma_{ij}^k - \sum\limits_{j\in V\setminus(S_t\cup\{k\})}\gamma_{lj}^k}{\sum\limits_{i,j\in V\setminus(S_p\cup\{k\})}\gamma_{ij} - \sum\limits_{j\in V\setminus(S_t\cup\{k\})}\gamma_{lj}}, \qquad \forall k\in C_t \tag{8.21}$$

where l is the branching vertex in child node t. Then, the value of $\sum\limits_{i,j\in V\setminus(S_t\cup\{k\})}\gamma_{ij}(k), \forall k\in C_t$ is the numerator of (8.21).

Note that the values of $\gamma_{ij}^k, \gamma_{ij}^{kl}$ and γ_{ij}^{klm} for all $i,j\in V, (i\neq j), k,l,m\in V, (k\neq l\neq m)$ are calculated and stored before starting the proposed CBB algorithm.

8.3. Structure closeness centrality

8.3.1. Properties of the closeness centrality measure

In this section, we are presenting the CBB algorithm for closeness centrality. The search tree structure is similar to that of betweenness centrality, which was described in detail in the previous subsection. To update the closeness centrality of a child using the closeness of its parent, we use the following proposition.

Proposition 7. *Given a structure $S \subseteq V$, and a vertex $p \in V \setminus S$ which forms a structure with S, we have that:*

$$C^c(S \cup \{p\}) = \frac{C^c(S)(|V| - |S|) - 1 - \sum\limits_{i \in V \setminus S \cup \{p\}} \min\{1, \gamma_{iS}^p\}}{|V| - |S| - 1}. \tag{8.22}$$

Proof of Proposition 7. Adding p to S, alters both the numerator and denominator of $C^c(S)$. Defining γ_{iS}^p as follows:

$$\gamma_{iS}^p = \gamma_{ij}^p : \quad d(i,j) = \min_{k \in S} d(i,k) \tag{8.23}$$

The numerator decreases by $1 + \sum_{i \in V \setminus S \cup \{p\}} \min\{1, \gamma_{Si}^p\}$, because p moves from $N(S)$ to the $S \cup \{p\}$ and its distance with S decreases from 1 to 0. Also, the distance of all non-group nodes whose shortest paths with S intersect p decrease by 1. Note that, since γ_{Si}^p could be more than 1, we use the $\min\{1, \gamma_{Si}^p\}$ to count these paths just once. The denominator decreases by 1 because the number of non-group nodes is decreased by 1. $\square$

Proposition 8 (Greedy rule for selecting the branching vertex)**.** *In node t of the search tree with structure set S_t and candidate set C_t, we select the branching vertex i as follows:*

$$i = \underset{j}{\operatorname{argmax}} \{D_{S_t}(j)\}, \quad j \in C_t, \tag{8.24}$$

where $D_{S_t}(j)$ is the decrease in the distance of $S_t \cup \{j\}$ from $V \setminus (S_t \cup \{j\})$ when vertex j is added to the structure S_t. To calculate the value of $D_{S_t}(j)$, $\forall j \in C_t$ in each tree node t, we update the value of $D_{S_p}(j)$ using the following equation

$$D_{S_t}(j) = D_{S_p}(j) - \sum_{i \in F(j)} \left(1 - \min\{\gamma_{li}^j, 1\}\right) \qquad \forall j \in C_t, \tag{8.25}$$

where p is the parent node of node t in the search tree, l is the branching vertex in child node t, and $F(j) = \{i \in V \setminus \{u, j\} \mid \gamma^j_{iu} > 0\}$. For $S_t = \{u\}$, since $S_p = \emptyset$, we use the following equation

$$D_{\{u\}}(j) = 1 + \sum_{i \in V \setminus \{u,j\}} \min\{\gamma^j_{ui}, 1\}, \qquad \forall j \in C_t \tag{8.26}$$

8.3.2. Search tree structure

In group closeness-based search trees, set B_l contains the decrease in the distance of $S_l \cup \{j\}$ from $V \setminus (S_l \cup \{j\})$ when vertex j is added to the structure S_l for all $j \in C_l$. We need set B_l in a tree node l to find the branching vertex i at search node l. The set $B_{l'}$ is updated using Proposition 8. Since set $S_l \cup \{i\}$ forms a structure, Proposition 7 can be used to calculate $C^c(S_{l'})$ by updating $C^c(S_l)$. A valid lower bound on closeness centrality of node l' is calculated using Proposition 11.

The proposed CBB algorithms for *MCCP(u), MCSP(u), and MCRP(u)* are similar to their counterparts of betweenness centrality presented in Section 8.2.2.

8.3.3. Lower bound

Proposition 9. *The group average closeness centrality presented in Definition 2 is not monotone.*

Proof. The graph shown in the Figure 6.4 is a counter example for the monotonicity of average closeness centrality measure of the Definition 2. In this example, $C^c(\{u\}) = 1.36$ and if we increase the group size by including vertex 2, its average closeness centrality increases to $C^c(\{u, 2\}) = 1.4$. Thus, we can say this average closeness centrality measure is not monotone. □

Definition 15. *Given a graph $G(V,E)$ and set $S \subseteq V$, the average-distance-closeness centrality of S defined as follows is strictly monotone:*

$$\mathcal{C}^c(S) = \frac{1}{|V|} \sum_{i \in V \setminus S} d(S, i) \tag{8.27}$$

Lemma 3. *Given sets $S \subseteq V$ and $S' \subset S$, we have:*

$$\mathcal{C}^c(S) < \mathcal{C}^c(S') \tag{8.28}$$

which means, centrality measure $\mathcal{C}^c(\cdot)$ is strictly monotone.

Proof. The denominator of both $\mathcal{C}^c(S')$ and $\mathcal{C}^c(S)$ is $|V|$, so we need to prove that $\sum_{i \in V \setminus S} d(S,i) < \sum_{i \in V \setminus S'} d(S',i)$. The set $V \setminus S'$ can be partitioned into two sets $V \setminus S$ and $S \setminus S'$. Hence, we can write

$$\sum_{i \in V \setminus S'} d(S',i) = \sum_{i \in V \setminus S} d(S',i) + \sum_{i \in S \setminus S'} d(S',i) \tag{8.29}$$

Since $S \setminus S' \neq \emptyset$, $\sum_{i \in S \setminus S'} d(S',i) > 0$ and we have

$$\sum_{i \in V \setminus S'} d(S',i) > \sum_{i \in V \setminus S} d(S',i) \geq \sum_{i \in V \setminus S} d(S,i) \tag{8.30}$$

The last inequality in (8.30) holds, because $d(S',i) \geq d(S,i), \forall i \in V$. □

Proposition 10. *Given set $S \subseteq V$, we have*

$$\mathcal{C}^c(S) \leq C^c(S) \tag{8.31}$$

If $S = V, \mathcal{C}^c(S) = C^c(S)$, otherwise $\mathcal{C}^c(S) < C^c(S)$, because $|v| > |V| - |S|$.

Corollary 2. *Let S be the optimal solution of MCSP(u). Using Proposition 10 we have*

$$\mathcal{C}^c(S) \leq C^c(S) \tag{8.32}$$

As centrality measure $\mathcal{C}^c(\cdot)$ is strictly monotone, the monotone closeness centrality of any structure of vertex u is greater than or equal to the closeness centrality of the maximal structure containing node u. Thus, considering $\hat{S}$ as the maximal structure of node u

$$\mathcal{C}^c(S) \geq \mathcal{C}^c(\hat{S}) \tag{8.33}$$

Equality holds when $S = \hat{S}$. From constraints (8.32) *and* (8.33)*, we conclude that*

$$C^c(S) \geq \mathcal{C}^c(\hat{S}) \tag{8.34}$$

Based on Corollary 2, a valid lower bound on centrality measure of Definition 2 for any structure containing vertex u is the strictly monotone average-closeness centrality measure of the maximal structure containing vertex u.

To calculate the closeness centrality of a set $S \subseteq V$, we need to find the distance of S from each vertex $i \in V \setminus S$. In this respect, in Lemma 4 we present a useful property related to the distance of a vertex from a clique that will be used to establish the relationship between the maximum- distance-closeness centrality of a clique and one of its members in Proposition 2.

Lemma 4. *Given structure $S \subseteq V$ and $i \in V \setminus S$, we have:*

- *if S is a clique:*
  1. $|d(i,j) - d(i,k)| \leq 1, \ \forall j,k \in S$ *(Ahuja et al., 1988)*
  2. $d(i,j) - 1 \leq d(S,i) \leq d(i,j), \ \forall j \in S$
- *if S is a star or representative:*
  1. $|d(i,j) - d(i,k)| \leq 2, \ \forall j,k \in S$
  2. $d(i,j) - 2 \leq d(S,i) \leq d(i,j), \ \forall j \in S$
  3. $d(i,j) - 1 \leq d(S,i) \leq d(i,j),$ *if j is the center of the star or representative*

Proposition 11. *In the MCSP(u), For a search tree node t with structure set S_t and nonempty candidate set C_t, the closeness centrality of any structure containing S_t is greater than or equal to*

$$\frac{1}{|V|}\Big(C^c(u)(|V|-1)-|V|+1\Big) \tag{8.35}$$

Proof. Let $\hat{S}$ be a maximal structure containing S_t. Then, since $\hat{S} \subseteq H_t, (H_t = S_t \cup C_t)$, based on Lemma 3 $\mathcal{C}^c(\hat{S}) \geq \mathcal{C}^c(H_t)$:

$$\mathcal{C}^c(H_t) = \frac{1}{|V|}\sum_{i\in V\setminus H_t} d(H_t,i) \tag{8.36}$$

By applying Lemma 4, we have

$$\begin{aligned}\frac{1}{|V|}\sum_{i\in V\setminus H_t} d(H_t,i) &\geq \frac{1}{|V|}\sum_{i\in V\setminus H_t}(d(i,u)-1)\\ &= \frac{1}{|V|}\Big(C^c(u)(|V|-1)-\sum_{i\in H_t\setminus\{u\}} d(i,u)-|V|+|H_t|\Big)\\ &= \frac{1}{|V|}\Big(C^c(u)(|V|-1)-|V|+1\Big)\end{aligned} \tag{8.37}$$

□

8.4. Structure degree centrality

In this section, we are presenting the CBB algorithm for degree centrality.

8.4.1. Properties of the degree centrality measure

Proposition 12. *Given a structure $S \subseteq V$, and a vertex $p \in V \setminus S$ which forms a structure with S, we have that:*

$$C^d(S\cup\{p\}) = C^d(S)-1+\sum_{i\in N(p)\setminus S}\left(1-\max_{j\in S}\{a_{ij}\}\right) \tag{8.38}$$

Proof. Adding p to S, degree of S is decreased by one unit, because we know that if p is connected to S through multiple links, they are counted just once. On the other hand, those non-group neighbors of p which do not have connection with other structure nodes are added to the degree of S. □

Proposition 13 (Greedy rule for selecting the branching vertex)**.** *In node t of the search tree with structure set S_t and candidate set C_t, we select the branching vertex i as follows:*

$$i = \underset{j}{\operatorname{argmax}}\{E_{S_t}(j)\}, \quad j \in C_t, \tag{8.39}$$

where $E_{S_t}(j)$ is the increase in the degree of $S_t \cup \{j\}$ when vertex j is added to the structure S_t. To calculate the value of $E_{S_t}(j)$, $\forall j \in C_t$ in each tree node t, we update the value of $E_{S_p}(j)$ using the following equation

$$E_{S_t}(j) = E_{S_p}(j) - \sum_{i \in N(j) \setminus S_t} a_{il} \qquad \forall j \in C_t \tag{8.40}$$

where p is the parent node of node t in the search tree and l is the branching vertex in child node t. For $S_t = \{u\}$, since $S_p = \emptyset$, we use the following equation

$$D_{\{u\}}(j) = N(j) - 1 - \sum_{i \in N(j) \setminus \{u\}} a_{iu} \qquad \forall j \in C_t \tag{8.41}$$

We can use the following Lemmas to create a lower bound for the degree problems:

Lemma 5. *Let S_i^* be the most connected clique of vertex i and vertex $j \in S_i^*$, then we have $C^d(S_j^*) \geq C^d(S_i^*)$.*

Lemma 6. *Let S_i^* be the most connected star of vertex i and vertex $j \in S_i^*$, then we have $C^d(S_j^*) \geq C^d(S_i^* \setminus \{i\})$.*

Lemma 7. *Let S_i^* be the most connected representative of vertex i and vertex $j \in S_i^*$, then we have $C^d(S_j^*) \geq C^d(S_i^*)$.*

8.4.2. Search tree structure

In the search tree of group degree-based algorithms, set B_l contains the increase in the degree of $S_l \cup \{j\}$ when vertex j is added to the structure S_l for all $j \in C_l$. After creation root node, we use Lemma 5, Lemma 6, and Lemma 7 for clique, star, and representative problems, respectively, to create a good initial incumbent based on previously identified optimal structures. The proposed CBB algorithms for *MDCP(u) and MDSP(u)* are similar to their counterparts of betweenness centrality. In the CBB algorithm of *MDRP(u)*, we take advantage of Proposition 17 to initialize the C_l at root node.

8.4.2.1. Upper bound

Proposition 14. *The group centrality presented in Definition 1 is not monotone.*

Proof. The graph shown in the Figure 6.2 is a counter example for the monotonicity of degree centrality measure of Definition 1. In this example,$C^d(\{u,8\}) = 7$ and if we increase the group size by including vertex 7, its degree centrality decreased to $C^d(\{u,7,8\}) = 6$. Thus, we can say this degree centrality measure is not monotone. □

Definition 16. *Given a graph $G(V,E)$ and set $S \subseteq V$, the degree centrality of S defined as follows is monotone:*

$$C^d(S) = |N(S)| + |S| \tag{8.42}$$

Lemma 8. *Given sets $S \subseteq V$ and $S' \subseteq S$, we have:*

$$\mathcal{C}^d(S) \geq \mathcal{C}^d(S') \tag{8.43}$$

which means, centrality measure $\mathcal{C}^d(\cdot)$ is monotone.

Proof. Using the monotone degree centrality definition, we can write

$$\mathcal{C}^d(S) - \mathcal{C}^d(S') = |N(S)| + |S| - |N(S')| - |S'| \tag{8.44}$$

Note that the open neighborhood of S, $N(S)$, is equal to the sum of the open neighborhood of $S', N(S')$ and the open neighborhood of the $S \setminus S'$ which is not included in the open neighborhood of $S', N(S \setminus S')$ excluding set $S \setminus S'$:

$$\mathcal{C}^d(S) - \mathcal{C}^d(S') = |N(S')| + |N(S \setminus S')| - |S \setminus S'| + |S| - |N(S')| - |S'| \tag{8.45}$$

Clearly, we have that $S = S' \cup (S \setminus S')$. Thus, we get that

$$\begin{aligned}\mathcal{C}^d(S) - \mathcal{C}^d(S') &= |N(S')| + |N(S \setminus S')| - |S \setminus S'| + |S'| + |S \setminus S'| - |N(S')| - |S'| \\ &= |N(S \setminus S')| \geq 0 \end{aligned} \tag{8.46}$$

□

Proposition 15. *Given set $S \subseteq V$, we have*

$$\mathcal{C}^d(S) > C^d(S) \tag{8.47}$$

Since $|S| > 0$, $\mathcal{C}^d(S) > C^d(S)$.

Corollary 3. *Let S be the optimal solution of MDSP(u). Using Proposition 8.47 we have*

$$\mathcal{C}^d(S) > C^d(S) \tag{8.48}$$

As centrality measure $\mathcal{C}^d(\cdot)$ is monotone, the monotone degree centrality of any structure of vertex u is less than or equal to the degree centrality of the maximal structure containing node u. Thus, considering $\hat{S}$ as the maximal structure of node u

$$\mathcal{C}^d(S) \leq \mathcal{C}^d(\hat{S}) \tag{8.49}$$

From constraints (8.48) *and* (8.49), *we conclude that*

$$C^d(S) \leq \mathfrak{C}^d(\hat{S}) \tag{8.50}$$

Based on Corollary 3, a valid upper bound on centrality measure of Definition 1 for any structure containing vertex u is the monotone degree centrality measure of the maximal structure containing vertex u.

Proposition 16. *In the MDSP(u), For a search tree node t with structure set S_t and nonempty candidate set C_t, the degree centrality of any structure containing S_t is less than or equal to*

$$|N(S_t \cup C_t)| + |S_t| + |C_t| \tag{8.51}$$

Proof. Let in the node t of the search tree $\hat{S}$ be the maximal structure containing S_t and S be the optimal solution of $MDSP(u)$. By Corollary 3 we have

$$C^d(S) \leq \mathfrak{C}^d(\hat{S}) \tag{8.52}$$

Since $\hat{S} \subseteq (S_t \cup C_t)$, by Lemma 8 we have

$$\mathfrak{C}^d \hat{S}) \leq \mathfrak{C}^d(S_t \cup C_t) \tag{8.53}$$

By Inequalities (8.52) and (8.53) the following expression holds

$$C^d \hat{S}) \leq \mathfrak{C}^d(S_t \cup C_t) = |N(S_t \cup C_t)| + |S_t| + |C_t| \tag{8.54}$$

Similar to betweenness centrality, If we show maximal clique of $G[C_t]$ by C'_t, then the upper bound for clique problem is as follows:

$$|N(S_t \cup C'_t)| + |S_t| + |C'_t| \tag{8.55}$$

which is smaller than or equal to (8.51), because $|C_t'| \leq |C_t|$. □

Proposition 17. *Let S_i^* be the most connected representative of vertex i. Then, we have $(S_i^* \setminus \{i\}) \subseteq SC^*(i)$. Where $SC^*(i)$ is the optimal solution of the set cover problem of vertex i.*

The Set Cover formulation for vertex u can be given by:

$$\min \quad \sum_{i \in N(u)} x_i \tag{8.56a}$$

$$\sum_{i \in N(u)} x_i a_{ij} \geq 1 \qquad \forall j \in NN(u). \tag{8.56b}$$

$$x_i \in \{0,1\} \qquad \forall i \in N(u). \tag{8.56c}$$

Proof. Suppose S_i^* is the most connected representative of vertex i and $j \in S_i^*$ and $j \notin SC^*(i)$. Since $j \notin SC^*(i)$, there exist set $D \subseteq SC^*(i)$ in a way that $N(j) \subset \bigcup_{k \in D}(N(k)$. By adding D and excluding vertex j from S_i^* the degree centrality of S_i^* is changed by at lest $1 - |D| + |D|$. The reason is that, because $D \subseteq SC^*(i)$, for each vertex $k \in D$ there is at least one vertex $l \in N(k) \setminus \hat{N}(u)$ which is not covered by other vertex in $N(u)$. Since $1 - |D| + |D| \geq 1$, it is in contradiction with optimality of S_i^* which completes the proof. □

Taking advantage of Proposition 17 in the CBB algorithm for MDRP(u), we finally have that $C_0 = SC^*(u)$.

CHAPTER 9. COLUMN GENERATION

Due to the specific property of Group degree-based problems (proposition 18), in this chapter we proposed a column generation (CG) algorithm to solve them.

9.1. Dantzig-Wolfe decomposition

We can directly apply Gurobi to handle the MILP formulations. Nevertheless, after some preliminary experiments, we find that the size of the instances optimally solved by Gurobi is quite limited. To achieve optimal solutions for the instances of practical size, we reformulate each problem into a master problem through Dantzig-Wolfe decomposition (Dantzig et al., 1960) and then develop a Column generation algorithm to solve it.

Dantzig-Wolfe decomposition is a technique that reformulates a linear program into a so-called linear master problem that typically involves a very large number of variables. In fact, the master problem contains one variable for each extreme point and each extreme ray of the feasible domain of a so-called subproblem. This domain is defined by a subset of the constraints of the original linear program. Once the reformulation is performed, the master problem is solved using an iterative column generation algorithm that solves at each iteration a restricted master problem (that is, the master problem restricted to a subset of the variables that varies from one iteration to another) and the subproblem (with a different objective function at each iteration). Furthermore, the Dantzig-Wolfe decomposition principle can also be applied to certain integer programs. In this case, the master problem is solved by a branch-and-bound procedure, where at each branch-and-bound node a lower bound is obtained by a column generation procedure.

Here, we propose a decomposition of each problem and provide the formulations of the resulting master problem and subproblem.

9.1.1. Master problem

To present the master problem, we define the following additional notations:

Parameters

- $\mathcal{R}^v$: set of all possible representatives of node $v, v \in V$.
- $\mathcal{C}^v$: set of all possible cliques of node $v, v \in V$.

- $\mathcal{S}^v$: set of all possible stars of node $v, v \in V$.

- D_s : degree centrality of structure $s, s \in \mathcal{R}^v \cup \mathcal{C}^v \cup \mathcal{S}^v$.

- C_s : closeness centrality of structure $s, s \in \mathcal{R}^v \cup \mathcal{C}^v \cup \mathcal{S}^v$.

- B_s : betweenness centrality of structure $s, s \in \mathcal{R}^v \cup \mathcal{C}^v \cup \mathcal{S}^v$.

- a_{ir} : binary parameter equal to 1 if node $i, i \in V$ is in the structure $j, j \in \mathcal{R}^v \cup \mathcal{C}^v \cup \mathcal{S}^v$ and 0 otherwise.

Decision Variables

- x_s : binary variables equal to 1 if structure $s \in \mathcal{R}^v \cup \mathcal{C}^v \cup \mathcal{S}^v$ is selected and 0 otherwise.

With the above notations, the master problem (MP) is given as

$$
\begin{aligned}
[RDMP_v] \quad \max \quad & \sum_{i \in \mathcal{R}^v} D_i x_i && && (9.1a) \\
\text{s.t.} \quad & \sum_{j \in \mathcal{R}^v} a_{ij} x_j \leq 1, && \forall i \in N(v), && (9.1b) \\
& \sum_{i \in \mathcal{R}^v} x_i = 1, && && (9.1c) \\
& x_i \in \{0, 1\}, && \forall i \in \mathcal{R}^v && (9.1d)
\end{aligned}
$$

Constraints (9.1a) maximizes the degree centrality. Constraints (9.1b) ensure that in the optimal structure each neighbor of node v is considered at most once.Constraints (9.1c) impose that the optimal solution is just one structure. According to constraints (9.1d) decision variable x_i is binary.

In the case of clique and star, we need to substitute $\mathcal{R}^v$ with $\mathcal{C}^v$ and $\mathcal{S}^v$, respectively.

9.1.2. Pricing problem

Given a dual solution to the LMP, the pricing subproblem is used to find a master variable x_s that has the least reduced. Then, we check whether this minimum is negative or not. Recall that every column of the constraint matrix of MP corresponds to a structure, and every entry of the column says whether the corresponding node is in the structure or not. Solving the pricing subproblem is essentially equivalent to enumerating all feasible structures. We define the following dual variables:

- π_i: nonnegative (nonpositive) dual variable of constraint (9.1b) for node $i \in N(v)$
- γ: dual variable of constraint (9.1c)

In the case of closeness centrality π_i is nonpositive.

To find the representative with least reduced cost we need to solve the following MILP formulation:

$$
\begin{aligned}
[RDPP_u] \quad \min \quad & -\sum_{i\in V} x_i^1 - \sum_{i\in N(u)} x_i^0 \pi_i - \gamma && && (9.2a)\\
\text{s.t.} \quad & x_i^0 + x_i^1 \leq 1, && \forall i \in V, && (9.2b)\\
& x_i^0 \leq a_{iu}, && \forall i \in V \setminus \{u\}, && (9.2c)\\
& x_i^1 \leq \sum_{j:(i,j)\in E} x_j^0, && \forall i \in V, && (9.2d)\\
& x_u^0 = 1, && && (9.2e)\\
& x_i^1, x_i^0 \geq 0 && \forall i \in V, && (9.2f)
\end{aligned}
$$

The objective function (9.2a) aims to achieve the minimal reduced cost of all feasible representatives centered at u. Explanations for all constraints are same as constraints of representative degree formulation. In the case of clique and star we need to add constraints (7.3c) and (7.2c), respectively.

9.2. Column generation

Column generation is an iterative procedure that we use for solving the linear relaxation of the master problem (i.e., the linear relaxation of the formulation (9.1b)-(9.1b)). For an overview of column generation, the reader is referred to and The optimal solution value of the LMP is a lower bound of its associated branch-and-bound node. The column generation procedure cannot directly solve the LMP because of its inability of enumerating all variables x_s. Instead, it is an iterative procedure that alternates between solving a restricted linear relaxation of the master problem (RLMP) and a pricing problem (PP). The RLMP is the LMP restricted to a subset of all variables x_s, which can be optimally solved by the simplex algorithm. Then, pricing problem (9.2a)-(9.2f) is solved in the hope of finding negative reduced cost columns (variables) with respect to the dual optimal solution of the current RLMP. If no such column is found, the column generation procedure is terminated with an optimal solution to the current RLMP, which is also an optimal solution to the LMP. Otherwise, we introduce one or more columns with negative reduced costs into the current RLMP and start another column generation iteration (See for further details).

9.2.1. Initial solution

To use column generation, we need a subset of all variables x_s, where each x_s is n structure. To create these initial structures, we use a greedy algorithm as follows:

Algorithm 3: Greedy algorithm to generate initial structures.

```
for $u \in V$ do
    for $i \in N(u)$ do
        $S_l \leftarrow S_l \cup \{i\}; S^* \leftarrow S_l; C^{d^*} \leftarrow C_l^d$;
        for $j \in N(u) \setminus \{i\}$ do
            if $S_l \cup \{j\}$ is a structure then
                $S_{l'} \leftarrow S_l \cup \{j\}$;
                if $C^d(S_{l'}) > C^{d^*}$ then
                    $C^{d^*} \leftarrow C^d(S_{l'})$ ;
                    $S^* \leftarrow S_{l'}$ ;
                end
            end
        end
        return $C^{d^*}, S^*$;
    end
end
```

To update the degree centrality in the greedy algorithm 3 we use the following proposition.

Proposition 18. *For MDRP(u), the column generation gives the optimal solution. For MDCP(u) and MDSP(u), if the optimal solution of the column generation is a clique or star, respectively it is the optimal solution of the original problem, otherwise it contains the optimal solution of the original problem.*

9.2.2. Representative

To find the optimal solution of the $MDRP(u)$, we use the algorithm 4.

Algorithm 4: Column Generation Algorithm for $MDRP(u)$.

```
1  for $u \in V$ do
2      generate initial representatives using algorithm 3 ;
3      solve RLMP, find dual variables and update the objective function of PP;
4      solve PP and put the optimal objective value in $obj(PP^*)$;
5      while $obj(PP^*) \leq 0$ do
6          update RLMP using generated column by RPP;
7          solve RLMP, find dual variables and update the objective function of PP;
8          solve PP and put the optimal objective value in $obj(PP^*)$;
9      end
10     return optimal solution of RLMP, $RLMP^*$ as optimal representative of vertex u
11             and $obj(RLMP^*)$ as its degree;
12 end
```

9.2.3. Clique and star

For clique and star problems of node u, $MDCP(u)$ and $MDSP(u)$, respectively, first we solve the problem using column generation. For clique problem, If the optimal solution of the column generation is a clique, then it is the optimal solution, otherwise we use the CBB to find the optimal clique of node u using the output structure of column generation.

CHAPTER 10. ASYMMETRIC PROBABILISTIC MINIMUM-COST HAMILTONIAN CYCLE PROBLEM CONSIDERING ARC AND VERTEX FAILURES

10.1. Introduction

A Hamiltonian cycle of directed graph $G = (V,A)$ is a cycle in which each vertex of V is visited exactly once, and thereafter it is called a "tour" (Laporte et al., 2007). Most problems consisting of finding Hamiltonian cycles are to find the minimum-cost (e.g., distance, time, flow, etc.) tours with certain constraints. We call such problems minimum-cost Hamiltonian cycle problem (MCHCP).

One of the most famous MCHCPs is the traveling salesman problem (TSP) (Gutin et al., 2002). Given a finite number of cities (vertices) and travel costs between them, the TSP is to find the cheapest tour of visiting each city exactly once and returning to the starting vertice. In terms of graph theory, this is equivalent to finding the minimum-cost Hamiltonian cycle in a complete graph. For some different varieties of TSP, see Gutin et al., 2006.

There are two significant motivations for studying stochastic MCHCP (Henchiri et al., 2014). The first is to define and analyze models which are more appropriate with reality where randomness is a major source of concern. For example, for many delivery companies, only a part of their customers require a daily delivery. Ideally, we can solve the MCHCP for each day, but this will be very time consuming. It is therefore necessary to adopt a model that takes random scenarios into account (Bertsimas et al., 1993; Bertsimas, 1988). The second motivation is the possibility to analyze the stability of optimal solutions to deterministic problems when the instances are disturbed by the absence of certain data. The most common sources of uncertainty considered in literature are: (a) demand volumes (Bertsimas, 1992; Laporte et al., 2002), (b) the presence of customers (Jaillet, 1985; Bertsimas, 1988), and (c) travel and/or service time (Laporte, 1992; Kenyon et al., 2003; Lei et al., 2012; Taş et al., 2013; Adulyasak et al., 2015).

Three main modeling strategies are widely applied for stochastic problems: (i) stochastic programs with recourses, where a "priori" solution and a modification strategy (recourse) are provided in the first stage and then in the second stage after disclosing random variables the recourse is applied to the first stage solution (Dror et al., 1989), (ii) chance-constrained problems, where one seeks a priori plan together with a bound on the probability that the given plan will be feasible once all the uncertainty is disclosed (Errico et al., 2018), and (iii) multi-stage dynamic reoptimization, where the problem is reoptimized as new information becomes available (Jaillet et al., 2008; Larsen et al., 2008; Ritzinger et al., 2016).

To the best of our knowledge, the probabilistic TSP was introduced for the first time in Jaillet Jaillet, 1985 in which only a subset of vertices (i.e., active set, determined by a known probability distribution) are required to be visited on any given instance of the problem. As re-optimization (i.e. to find an optimal solution for any instance) is time-and-resource-consuming and the complete information is usually unavailable at the beginning, the goal is to find a *priori* tour through all vertices with minimum expected length. The order of visiting vertices in any instance will be same as they appear in the priori tour. This problem has been solved using exact algorithms (Berman et al., 1988; Laporte et al., 1994) and heuristics (Bertsimas et al., 1990; Bertsimas et al., 1993; Birattari et al., 2008; Bertsimas, 1988). Some important works in the area of probabilistic version of TSP are as follows. Jula et al., 2006 consider the TSP with hard-time windows and stochastic travel and service time, where the objective is to find the minimum-cost tour such that for each customer the arrival-on-time tour probability is greater than or equal to a given value. The authors develop a heuristic algorithm based on dynamic programming and on an estimation of the mean and variance of the arrival-time distribution at the customers. Campbell et al., 2008 study a TSP with customer deadlines and uncertainty about the customer presence, where customers must be visited before given deadlines. The authors present two recourse models and a chance constrained model for the problem. A simple heuristic approach is proposed, and experiments comparing the deterministic and the stochastic solutions are performed. Chang et al., 2009 investigate stochastic dynamic pickup/delivery traveling salesman problem with hard time windows, where travel and service time follow normal distributions. They apply the convolution-propagation approach to approximate the arrival time of vertices by a normal distribution and propose a heuristic method to solve the problem.

In this research, we consider the following chance constraint in the deterministic MCHCP:

$$P\{\text{successfully visiting vertex } i \text{ starting from the origin}\backslash\text{depot}\} \geq \beta_i, \forall i \in V, \tag{10.1}$$

where β_i's are constant, and the vertices and arcs of the graph are possible to fail with independent success probabilities. We call it asymmetric probabilistic minimum-cost Hamiltonian cycle problem (APMCHCP), as both cost and success probability matrices can be asymmetric. By the definition of Hamiltonian cycle, an important special case of constraint (10.1) is the following success-tour chance constraint:

$$P\{\text{the Hamiltonian cycle is successful}\} \geq \beta. \tag{10.2}$$

Indeed, APMCHCP associated with this type of chance constraints have many applications in problems associated with Hamiltonian path or cycle.

For example, post-disaster recovery (PDR) is a key part in a disaster support system (Daud et al., 2016). A way to minimize the impact of a disaster on the victims is to ensure that adequate disaster supplies are available and can be delivered to the victims. Indeed, there have been a lot of research in the past few years on stock pre-positioning for disasters (Campbell et al., 2011; Duran et al., 2011; Lodree Jr et al., 2012). However, the conventional planning methods rarely account for the uncertainties that come with disasters. A major problem on delivering supplies after a disaster is that many roads may become impassable. Thus, in the planning stage of a PDR system, we need to determine the location to store the supplies and design the delivery route considering vertex and/or arc failures which is associated with infrastructure damage. As another example, in electronic circuit design, multi-threshold CMOS (MCMOS) is a popular technique which can reduce the power leakage by turning off inactive circuit domains. To avoid large peak current consumption during its sleep-to-active mode transition, a solution is to turn on power switches one after another to reduce the peak current during the mode transition. This problem is equivalent to find a minimum cumulative peak current Hamiltonian cycle to set up the power switches (i.e., to choose “arcs”) and connect all gates in the circuit (Wang et al., 2012b). However, in a complex circuit design, switches between some gates could be relatively easy to fail. Hence, it is important to take the chance constraint (10.2) into consideration in the design stage to avoid choosing unreliable switches. The third application is in the security maintenance of wireless sensor networks (WSN). According to (Tague et al., 2007; Lin et al., 2016), one of the attackers’ most common strategies is to invade the wireless nodes one by one along the path with minimum time-cost until the whole WSN is hacked, which is equivalent to find a minimum-cost Hamiltonian path. A problem of this strategy is that in a practical WSN, attacking success rates between node pairs are different, due to factors such as connection failure rates and effectiveness of defense systems. Thus, to the attacker, the target is not only to finish the attack in the shortest time, but also to guarantee that the selected attack sequence (i.e., the Hamiltonian path) has a large probability to be successful so that the whole WSN can be hacked. It is important to consider this point for the security of WSN.

CHAPTER 11. PROBLEM DEFINITION AND MATHEMATICAL FORMULATIONS

In this chapter, we provide notation definitions and terminology that will be used throughout this problem. We proceed to formally define probabilistic route and the modified Miller-Tucker-Zemlin (MTZ) constraints. Then, four mathematical formulations are proposed for the APMCHCP.

11.1. Notations

Consider a directed graph $G := (V, A)$ on n vertices. With each arc $(i, j) \in A$, an arc cost $c_{ij} > 0$ and an arc success probability $p_{ij} \in (0, 1]$ are associated. Furthermore, assume that for each vertex $i \in V$, a vertex success probability $p_i \in (0, 1]$ and a success arriving probability threshold $\beta_i \in [0, 1]$ are given (see (10.1)). The interval $[\beta_i, 1]$ is called the *probability interval* of vertex i, and the width of the probability interval is given by $1 - \beta_i$. The probability interval for vertex $i \in V$ is called *active*, if $\beta_i > 0$. The problem is to find a sequence of the vertices (starting at one of the vertices, visiting each vertex exactly once and ending at the same vertex) with minimal cost such that for every vertex $i \in V$ its arrival probability P^i lies within the given probability interval $[\beta_i, 1]$. The APMCHCP is NP-hard, as it reduces to the asymmetric TSP if $\beta_i = 0$ for every $i \in V$.

For notation convenience, a route $\mathscr{R}$ consisting of the ordered arc set

$$\{(v_1, v_2), (v_2, v_3), \cdots, (v_{k-1}, v_k)\}$$

is denoted by $\mathscr{R} = (v_1, v_2, \ldots, v_k)$.

11.2. The probabilistic route and the modified MTZ constraints

Given a route $\mathscr{R} = (v_1, \ldots, v_k)$, the arrival probability P^i at vertex $v_i (i = 1, \ldots, k)$ along $\mathscr{R}$ is computed by

$$P^{v_1} := p_{v_1} \tag{11.1a}$$

$$P^{v_i} := P^{v_{i-1}} p_{v_{i-1}, v_i} p_{v_i}, \qquad \text{for } i = 2, \ldots, k \tag{11.1b}$$

We denote by $p_{\mathscr{R}}$ the probability of a route $\mathscr{R} = (v_1, v_2, \dots, v_k)$, namely the probability that one can arrive at the last vertex of $\mathscr{R}$, v_k, through the route $\mathscr{R}$. Thus, we have

$$p_{\mathscr{R}} = \left(\prod_{j\in\mathscr{R}} p_j\right)\left(\prod_{i=1}^{k-1} p_{v_i,v_{i+1}}\right) . \tag{11.2}$$

Clearly, if the origin (or equivalently, the depot) and the route are given, one can get the probability of the route and check the chance constraint (10.1) by applying (11.2). Note that if the route $\mathscr{R} = (v_1, v_2, \dots v_n, v_1)$ is a cycle, we only use the probability of the origin, p_{i_0}, for once in (11.2).

A Hamiltonian cycle (or equivalently, tour) $T := (v_1, v_2, \dots, v_n, v_1)$ of G starting at node v_1 is called *feasible* if each vertex is visited within its probability interval, i.e., $P^i \geq \beta_i$ for $i = 1, \dots, n$. A rote $\mathscr{R} = (v_1, ..., v_k)$ with $2 \leq k \leq n$ is said to be *infeasible* if it cannot be included in any feasible tour. Easily checkable and obvious sufficient conditions for infeasibility are given in the following lemma.

Lemma 9. *A given route $\mathscr{R} = (v_1, \dots, v_k)$ is infeasible, if (at least) one of the following conditions holds:*

(i) $\mathscr{R}$ violates the chance constraint of at least one of its vertices, i.e., $\exists i \in \mathscr{R} : P^i < \beta_i$,.

(ii) There exists a vertex u not included in $\mathscr{R}$ such that the maximum-probability path from u to v_1, denoted by $MPP(u, v_1)$, violates the chance constraint of at least one of the vertices included in $\mathscr{R}$ and the maximum-probability path from v_k to u, denoted by $MPP(v_k, u)$, violates the chance constraint of u.

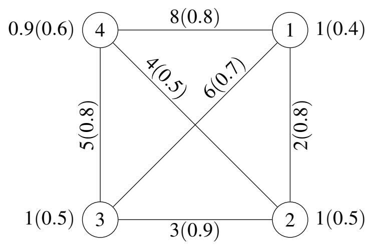
The proof of Lemma 9 is obvious. Algorithm 5 is used to find the maximum-probability path from i to j, i.e., $MPP(i, j)$, for all pairs of vertices in G.

In many deterministic MCHCP's (for example, the TSP), it is equivalent to consider any vertex in the tour to be the origin. However, in a APMCHCP, the origin will affect the probability of reaching each vertex and hence it is critical to determine the origin. Let c_{ij} be the cost of choosing arc (i, j) and consider the TSP showed in Figure 11.1a with chance constraint (10.1),

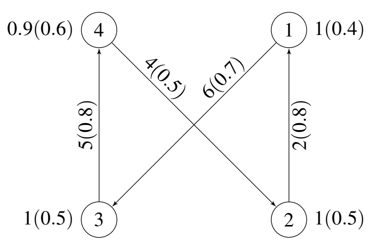
Algorithm 5: Finding the maximum-probability path.

```
Data: Digraph G
Result: A path with maximum probability between any two given vertices
for $i \in V$ do
    $prob[i] \leftarrow p_i$                          // determine source vertex
    $Q.push(i, prob[i])$                              // Q is a priority queue
    while $Q \neq \emptyset$ do
        $u \leftarrow$ pop(Q)          // pop(Q) extracts vertex in Q with max prob[]
        for each successor $v$ of $u$ do
            $alt = prob[u]p_{uv}p_v$
            if $alt \geq \beta_v$ then
                if $v \notin prob$ or $alt > prob[v]$ then
                    $prob[v] \leftarrow alt$
                    $Q.push(v, alt)$
    return $prob[]$
```

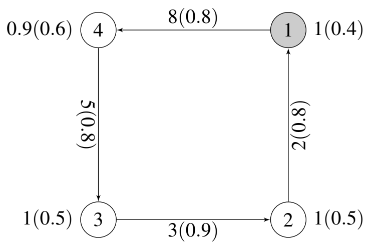
where the numbers on each arc represent $c_{ij}(p_{ij})$ and at each vertex stand for $p_j(\beta_j)$. It is easy to verify that the optimal tour of the TSP, without considering the chance constraint, is $(1,3,4,2,1)$ with total cost 17, as showed Figure 11.1b. But this tour violates the chance constraint at vertex 4, as $P^4 = 0.504 < \beta_4 = 0.6$. Actually, the optimal tour of this problem is $\mathscr{R}^* = (1,4,3,2,1)$ with total cost 18 (see Figure 11.1c), where arrival probabilities to vertices 4, 3, 2, 1 are 0.72, 0.576, 0.5184, and 0.41472, rescpecitvely. However, if we start the optimal tour of the APMCHCP from vertex 2, i.e., $\mathscr{R}' = (2,1,4,3,2)$, the probabilities of reaching vertices 4 and 3 are changed to be $0.576(< \beta_4 = 0.6)$ and $0.4608(< \beta_3 = 0.5)$, which makes it no longer a feasible tour.



(a) A APMCHCP with 4 vertices.



(b) Optimal MCHCP tour.



(c) Optimal APMCHCP tour.

Figure 11.1: An example of APMCHCP. The gray vertex is starting point.

MTZ constraints (Miller et al., 1960), which have a polynomial cardinality, are widely used for eliminating subtours and meanwhile satisfying the demand of each vertex. Later, Desrochers et al., 1991 propose the following strengthened MTZ (SMTZ) constraints which is more efficient and has a stronger LP relaxation:

$$u_i - u_j + (n-1)x_{ij} + (n-3)x_{ji} \leq n-2, \qquad \forall i,j \in V \setminus \{1\}, i \neq j \tag{11.3a}$$

$$u_i \geq 1 + (n-3)x_{i1} + \sum_{j \in V \setminus \{1\}, i \neq j} x_{ji} \qquad \forall i \in V \setminus \{1\} \tag{11.3b}$$

$$u_i \leq n-1-(n-3)x_{1i} - \sum_{j \in V \setminus \{1\}, i \neq j} x_{ij} \qquad \forall i \in V \setminus \{1\} \tag{11.3c}$$

where vertex 1 is assumed to be the origin; binary variable $x_{ij} = 1$, if arc $(i,j) \in A$ is used and 0 otherwise; and MTZ variable u_i represents the order of vertex i in the tour. It is showed that both MTZ and SMTZ constraints can be adopted to a number of MCHCP formulations, such as TSP, VRP, and their varieties (Toth et al., 2014).

To include the selection of the origin to SMTZ (which is necessary in the APMCHCP), we modify (11.3a)-(11.3c) by defining a new binary variable r_i ($r_i = 1$, iff vertex i is the origin), as showed in the following proposition.

Proposition 19. *The constraints*

$$u_j - u_i \geq (n-1-nr_j)x_{ij} + (n-3+nr_i)x_{ji} - n + 2, \qquad \forall i,j \in V, i \neq j \tag{11.4a}$$

$$\textstyle\sum_{i=1}^{n} r_i = 1 \tag{11.4b}$$

$$r_i \in \{0,1\}, \qquad \forall i \in V \tag{11.4c}$$

with $u_i, u_j \in [0, n-1]$ are valid inequalities for the APMCHCP.

Proof. The validation of (11.4b) and (11.4c) is obvious. In the following, we show the validation of (11.4a) by discussing three cases:

1. Neither i nor j is the origin, i.e., $r_i = r_j = 0$. Then (11.4a) becomes (11.3a).

2. either i or j is the origin and they are not adjacent. In this case, (11.4a) can be written as

$$u_j - u_i \geq -n+2, \qquad \forall i,j \in V, i \neq j.$$

3. either i or j is the origin and they are adjacent. There are two possible sub-cases:

   (a) $j = i_0$ and $i = i_{n-1}$ (or $i = i_0$ and $j = i_{n-1}$), where i_0 and i_{n-1} are the first and last vertices in the tour. Then (11.4a) can be written as

   $$u_{i_0} - u_{i_{n-1}} \geq -n+1 \quad \text{(or, } u_{i_{n-1}} - u_{i_0} \geq n-1\text{)}, \qquad \forall i,j \in V, i \neq j.$$

   (b) $j = i_0$ and $i = i_1$ (or $i = i_0$ and $j = i_1$). Then (11.4a) can be written as

   $$u_{i_0} - u_{i_1} \geq -1 \quad \text{(or, } u_{i_1} - u_{i_0} \geq 1\text{)}, \qquad \forall i,j \in V, i \neq j.$$

□

Comparing (11.4a) – (11.4c) with (11.3a) – (11.3c), one can see that the proposed subtour elimination constraints is a generalization of SMTZ (Desrochers et al., 1991). Taking advantage of (11.2) and variable u_i defined in (11.4), one can get the following proposition.

Proposition 20. *The probability of reaching vertex j from the origin vertex i along route $\mathscr{R} = (i, \cdots, j)$ can be calculated as follows:*

$$p_{\mathscr{R}} = \begin{cases} \prod\limits_{k \in V: u_k \leq u_j} p_k \prod\limits_{(l,m) \in A: u_l < u_j} p_{lm}, & \textit{if } j \in V \setminus \{i\} \\ \prod\limits_{k \in V: u_k \geq u_j} p_k \prod\limits_{(l,m) \in A: u_l \geq u_j} p_{lm}, & \textit{if } j = i = \textit{the origin} \end{cases} \tag{11.5}$$

11.3. The formulations

In this section, we develop four exact mixed integer programming (MIP) formulations for the APMCHCP.

11.3.1. IP 1 (intuitive direct formulations for the chance constraint)

To explicitly formulate the chance constraint, an intuitive idea is to define a new variable to indicate the order relationship between the MTZ variables u_i and u_j,

$$\delta_{ij} := \begin{cases} 1, & \text{if } u_i \le u_j; \\ 0, & \text{otherwise} \end{cases}, \quad \forall i, j \in V, \tag{11.6}$$

with the following constraints

$$\frac{1}{n}(u_j - u_i) \le \delta_{ij} \le 1 + \frac{1}{n}(u_j - u_i), \qquad \forall i, j \in V \tag{11.7a}$$

$$\delta_{ii} = 1, \qquad \forall i \in V \tag{11.7b}$$

$$\delta_{ij} \in \{0, 1\}, \qquad \forall i, j \in V \tag{11.7c}$$

It is worth mentioning that $\forall j \in V \setminus \{i_0\}$ we have $\delta_{i_0 j} \equiv 1$ and $\delta_{j i_0} \equiv 0$, where vertex i_0 is the origin of the tour.

Theorem 7. *Using δ_{ij} defined in* (11.7)*, the chance constraint* (10.1) *for the APMCHCP can be explicitly formulated as follows:*

$$\left(\prod_{j \in V} p_j^{(1-r_i)\delta_{ji}}\right)\left(\prod_{\substack{j \in V, \\ j \neq i}} \prod_{l:(j,l) \in A} p_{jl}^{(1-r_i)x_{jl}\delta_{ji}}\right) \ge \beta_i, \qquad \forall i \in V \tag{11.8a}$$

$$\left(\left(\prod_{j \in V} p_j\right)\left(\prod_{j \in V} \prod_{l:(j,l) \in A} p_{jl}^{x_{jl}}\right)\right)^{r_i} \ge \beta_i, \qquad \forall i \in V \tag{11.8b}$$

which are equivalent to

$$\sum_{j\in V}(\delta_{ji}-\sigma_{ij})\log p_i + \sum_{\substack{j\in V,\\ j\neq i}}\sum_{l:(j,l)\in A}(\zeta_{ijl}-\psi_{ijl})\log p_{jl} \geq \log\beta_i, \qquad \forall i\in V \qquad (11.9a)$$

$$r_i\sum_{j\in V}\log p_j + \sum_{j\in V}\sum_{l:(j,l)\in A}\phi_{ijl}\log p_{jl} \geq \log\beta_i, \qquad \forall i\in V \qquad (11.9b)$$

where $\sigma_{ij}=r_i\delta_{ji}$, $\zeta_{ijl}=x_{jl}\delta_{ji}$, $\psi_{ijl}=r_ix_{jl}\delta_{jl}$, *and* $\phi_{ijl}=r_ix_{jl}$.

Proof. Consider two possible cases for vertex $i\in V$.

1. $r_i=0$: Constraint (11.8b) becomes $1\geq\beta_i$ and constraints (11.8a) define the chance constraints for all vertices other than the origin. The first big-parentheses of (11.8a) calculates the success probabilities of all vertices which are visited before vertex i from the origin, including vertex i itself. The second big-parentheses of (11.8a) calculates the probabilities of all arcs passed to reach vertex i from the origin, which can be proved by discussing the following four combinations of i, j, and l. All these cases can be figured out using schematic APMCHCP tour presented in Figure 11.2.

   (a) $l=i: \prod_{\substack{j\in V,\\ j\neq i}} p_{ji}^{(1-r_i)x_{ji}\delta_{ji}} = p_{\tilde{j}i}$, where vertex $\tilde{j}$ is right before vertex i (i.e., $x_{\tilde{j}i}=1$). Thus, (11.8a) calculates the success probability of the incoming arc to vertex i by Proposition 20.

   (b) $l=$ *origin*: $\prod_{\substack{j\in V,\\ j\neq i}} p_{jl}^{(1-r_i)x_{jl}\delta_{ji}} \equiv 1$. This is because, if $x_{\tilde{j}l}=1$ (i.e., $\tilde{j}$ being the last vertex of the tour), then $\delta_{\tilde{j}i}\equiv 0$.

   (c) $l\neq i, l\neq$ *origin*, $j=$ *origin*: $\prod_{l:(j,l)\in A} p_{jl}^{(1-r_i)x_{jl}\delta_{ji}} = p_{j\tilde{l}}$, getting the success arc probability of reaching the first vertex (i.e., vertex $\tilde{l}$) from the origin, where $x_{j\tilde{l}}=1$.

   (d) $l\neq i, l\neq$ *origin*, $j\neq$ *origin*: if $u_j>u_i$ (i.e., $\delta_{ji}=0$), then (11.8a) does not count the probability of arc (j,l) as we expect, as it is not in the route from the origin to vertex i. Otherwise, (11.8a) calculates the success arc probability of all arcs between the origin and vertex i except for the first (discussed in case (c)) and last (discussed in case (a)) arcs.

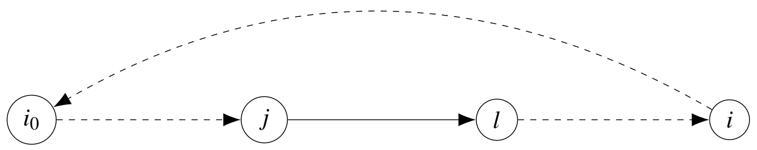


Figure 11.2: An example of a APMCHCP tour when i_0 is the starting vertex.

2. $r_i = 1$: Constraint (11.8a) becomes $1 \geq \beta_i$ and constraint (11.8b) defines the chance constraint for the origin.

$\square$

11.3.2. IP 2 (efficient direct formulations for the chance constraint)

We provide a more efficient formulation, in which the δ_{ij} is defined to be "dynamic":

$$\delta_{ij} = \begin{cases} \begin{cases} 1, & \text{if } u_i < u_j \\ 0, & \text{otherwise} \end{cases}, & \text{if } r_j = 0 \\ 1, & \text{if } r_j = 1 \end{cases} \tag{11.10}$$

Proposition 21. *In the APMCHCP, constraints*

$$r_j + \frac{1}{n}(u_j - u_i) \leq \delta_{ij} \leq 1 + r_j + \frac{1}{n}(u_j - u_i), \qquad \forall i, j \in V \tag{11.11a}$$

$$\delta_{ii} = r_i, \qquad \forall i \in V \tag{11.11b}$$

$$\delta_{ij} \in \{0, 1\}, \qquad \forall i, j \in V \tag{11.11c}$$

determine the value of δ_{ij} defined in (11.10)*, using MTZ variables u_i define in* (11.4).

Proof. Consider the following two cases for any vertex $j \in V$:

1. j is not the origin, i.e., $r_j = 0$: (11.11a) becomes $\frac{1}{n}(u_j - u_i) \leq \delta_{ij} \leq 1 + \frac{1}{n}(u_j - u_i), \ \forall i, j \in V$, which gives $\delta_{ij} = 1$, if $u_i < u_j$, and $\delta_{ij} = 0$, otherwise.

2. j is the origin, i.e., $r_j = 1$ and $u_j = 0$, (11.11a) becomes $1 - \frac{u_i}{n} \leq \delta_{ij} \leq 2 - \frac{u_i}{n}, \ \forall i, j \in V$, which specifies $\delta_{ij} = 1, \forall i$.

$\square$

Then, we can explicitly write the chance constraint (10.1) with only one formulation and an easier structure.

Theorem 8. *Using δ_{ij} defined in* (11.11), *the chance constraint* (10.1) *for the APMCHCP can be written as:*

$$\left(p_i^{(1-r_i)}\right)\left(\prod_{j\in V} p_j^{\delta_{ji}}\right)\left(\prod_{j\in V}\prod_{l:(j,l)\in A} p_{jl}^{x_{jl}\delta_{ji}}\right) \geq \beta_i, \qquad \forall i \in V \tag{11.12}$$

which is equivalent to

$$(1-r_i)\log p_i + \sum_{j\in V}\delta_{ji}\log p_j + \sum_{j\in V}\sum_{l:(j,l)\in A} z_{jil}\log p_{jl} \geq \log\beta_i, \qquad \forall i \in V \tag{11.13}$$

where $z_{jil} = x_{jl}\delta_{ji}$.

Proof. We discuss two cases for $i \in V$ to prove (11.12).

1. $r_i = 0$: the first and second big-parentheses of (11.12) count the multiplication of probabilities of all vertices which are passed to reach vertex i from the origin, including vertex i itself. The third big-parentheses counts the multiplication of probabilities of all arcs which are traversed to reach vertex i from the origin. To prove the third big-parentheses, one more case needs to be discussed in addition to the four cases for $r_i = 0$ showed in the proof of Theorem 7 :

   (e) $j = i$: $\delta_{ji} \equiv 0$ (by (11.11b)) and hence $\prod_{l:(j,l)\in A} p_{jl}^{x_{jl}\delta_{ji}} \equiv 1$.

2. $r_i = 1$: the first big-parentheses become trivial, and the second big-parentheses of (11.12) counts the multiplication of probabilities of all vertices in the tour. Note that the probability of the origin vertex is also counted by the second big-parentheses, as $\delta_{ii} = 1$. The third big-parentheses counts the multiplication of probabilities of all arcs in the tour. We prove the third part by discussing two possible combinations of i, j and l:

(a) $l = i$: based on (11.11a) $\delta_{ji} \equiv 1$, and hence $\prod_{j \in V} \prod_{l:(j,l) \in A} p_{jl}^{x_{jl} \delta_{ji}} = p_{\tilde{j}l}$, where vertex $\tilde{j}$ is the last vertex in the tour (i.e., $x_{\tilde{j}l} = 1$). Thus, the third big-parentheses of (11.12) calculates the success probability of the incoming arc to the origin.

(b) $j = i$: $\delta_{ji} = \delta_{ii} = 1$, and hence $\prod_{l:(i,l) \in A} p_{il}^{x_{il} \delta_{ii}} = p_{i\tilde{l}}$, where vertex $\tilde{l}$ is the second vertex in the tour (i.e.,, $x_{i\tilde{l}} = 1$). Thus, the third big-parentheses of (11.12) gets the probability of the outgoing arc from the origin.

□

11.3.3. IP 3 (intuitive recursive formulations for the chance constraint)

In this formulation, we apply the recursive property of calculating probability of reaching each vertex by its direct predecessor. For this purpose, new decision variable y_{ij} is defined as follows:

$$y_{ij} = \text{the probability of reaching node } j \text{ from } i, \quad \forall i, j \in V \qquad (11.14)$$

Theorem 9. *Using y_{ij} defined in* (11.14)*, the chance constraint* (10.1) *for the APMCHCP can be written as:*

$$y_{ij} \leq \sum_{k \in V, k \neq i,j} p_j p_{kj} x_{kj} y_{ik} + p_i p_j x_{ij} p_{ij}, \qquad \forall i, j \in V, i \neq j \qquad (11.15a)$$

$$y_{ii} \leq \sum_{k \in V, k \neq i} p_{ki} x_{ki} y_{ik}, \qquad \forall i \in V, \qquad (11.15b)$$

$$y_{ij} \geq r_i \beta_j, \qquad \forall i, j \in V \qquad (11.15c)$$

which is equivalent to

$$y_{ij} \leq \sum_{k\in V, k\neq i,j} p_j p_{kj} q_{ikj} + p_i p_j p_{ij} x_{ij}, \qquad \forall i,j \in V, i \neq j \tag{11.16a}$$

$$y_{ii} \leq \sum_{k\in V, k\neq i} p_{ki} q_{iki}, \qquad \forall i \in V \tag{11.16b}$$

$$y_{ij} \geq r_i \beta_j, \qquad \forall i,j \in V \tag{11.16c}$$

where $q_{ikj} = x_{kj} y_{ik}$.

Proof. We discuss two cases for i and j to prove (11.15).

1. $i = j$: this case which is shown in constraint (11.15b), calculates the probability of cycle starting from vertex $i, \forall i \in V$. Based on constraint (11.15c), the chance constraint for y_{ii} is active only when i is the origin.

2. $i \neq j$: this case which is indicated in constraint (11.15a), calculates the probability of reaching vertex j from vertex i. We prove this case by discussing two possible subcases:

   (a) j is the vertex immediately after vertex i, i.e. $x_{ij} = 1$: based on (11.15a), $\sum_{k\in V, k\neq i,j} p_j p_{kj} x_{kj} y_{ik} = 0$ and $p_i p_j x_{ij} p_{ij}$ correctly calculates the probability of reaching j from i.

   (b) There are at least one intermediate vertex between i and j, i.e. $x_{ij} \neq 1$: in this case $p_i p_j x_{ij} p_{ij} = 0$ and $\sum_{k\in V, k\neq i,j} p_j p_{kj} x_{kj} y_{ik}$ calculated the probability of reaching vertex j from i using the probability of reaching vertex k from i, where k is the vertex immediately before j along route $\mathscr{R}$.

□

11.3.4. IP 4 (efficient recursive formulations for the chance constraint)

In this formulation, we define decision variable y_i as follows:

$$y_i = \text{the probability of reaching node } i \text{ from the origin}, \qquad \forall i \in V \tag{11.17}$$

Theorem 10. *Using y_i defined in* (11.17)*, the chance constraint* (10.1) *for the APMCHCP can be written as:*

$$p_i p_{ij} p_j r_i + (1 - r_i - r_j) y_i p_{ij} p_j + y_i p_{ij} r_j - y_j \geq x_{ij} - 1, \qquad \forall i, j \in V, i \neq j \tag{11.18a}$$

$$y_i \geq \beta_i, \qquad \forall i \in V \tag{11.18b}$$

which is equivalent to

$$p_i p_{ij} p_j r_i + (y_i - s_{ii} - s_{ji}) p_{ij} p_j + s_{ji} p_{ij} - y_j \geq x_{ij} - 1, \qquad \forall i, j \in V, i \neq j \tag{11.19a}$$

$$y_i \geq \beta_i, \qquad \forall i \in V \tag{11.19b}$$

where $s_{ij} = y_j r_i$.

Proof. We discuss three cases for i and j to prove (11.18).

1. i or j is not the origin, i.e. $r_i = r_j = 0$: in this case if $x_{ij} = 1$ then constraint (11.18a) becomes $y_i p_{ij} p_j \geq y_j$ which correctly calculates y_j using y_i. On the other hand, if $x_{ij} = 0$, (11.18a) becomes $y_i p_{ij} p_j - y_j \geq -1$ which is correct because the upper bound of y_i and y_j is 1.

2. i is the origin, i.e. $r_i = 1$: in this case if $x_{ij} = 1$ then constraint (11.18a) becomes $p_i p_{ij} p_j \geq y_j$ which correctly calculates the probability of reaching first vertex after origin ($r_j = i$ because of constraint (11.4b)). On the other hand, if $x_{ij} = 0$, (11.18a) becomes $p_i p_{ij} p_j - y_j \geq -1$ which is correct.

3. j is the origin, i.e. $r_j = 1$: in this case if $x_{ij} = 1$ then constraint (11.18a) becomes $y_i p_{ij} \geq y_j$ which correctly calculates the probability of reaching origin using the probability of last vertex of the cycle. On the other hand, if $x_{ij} = 0$, (11.18a) becomes $y_i p_{ij} - y_j \geq -1$ which is correct.

□

11.3.5. Complete exact MIP formulations for the APMCHCP

Techniques discussed in Sections 11.3.1-11.3.4 can be used to develop complete formulations of APMCHCP.

11.3.5.1. Efficient direct APMCHCP formulation

The complete model of APMCHCP with efficient chance constraint formulation (proposed in Section 11.3.2) is as follows.

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{11.20a}$$

$$s.t. \quad (11.4a)-(11.4c),(11.11a)-(11.11c),(11.13)$$

$$\sum_{i:(i,j)\in A} x_{ij} = 1, \qquad \forall j \in V \tag{11.20b}$$

$$\sum_{i:(j,i)\in A} x_{ji} = 1, \qquad \forall j \in V \tag{11.20c}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A \tag{11.20d}$$

In this formulation, the objective function (11.20a) is to minimize the total cost of the tour. Assignment constraints (11.20b) and (11.20c) are degree constraints. The APMCHCP can finally be written as the following MIP.

$$\textbf{[IP2]} \quad \min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{11.21a}$$

$$s.t. \quad (11.4b),(11.4c),(11.11a)-(11.11c),(11.13),$$

$$(11.20b)-(11.20d)$$

$$u_j - u_i \geq (n-1)x_{ij} - n\theta_{ij} + (n-3)x_{ji} + n\theta_{ji} - n + 2, \qquad \forall i,j \in V, i \neq j \tag{11.21b}$$

$$z_{jil} \leq x_{jl},\ z_{jil} \leq \delta_{ji},\ z_{jil} \geq x_{jl} + \delta_{ji} - 1, \qquad \forall j,i,l \in V, j \neq l \tag{11.21c}$$

$$\theta_{ij} \leq r_j,\ \theta_{ij} \leq x_{ij},\ \theta_{ij} \geq r_j + x_{ij} - 1, \qquad \forall i,j \in V \tag{11.21d}$$

$$z_{jil} \in \{0,1\}, \qquad \forall (j,l) \in A; i \in V \tag{11.21e}$$

$$\theta_{ij} \in \{0,1\}, \qquad \forall i,j \in V; i \neq j \tag{11.21f}$$

11.3.5.2. Intuitive direct APMCHCP formulation

The complete APMCHCP model with intuitive chance constraint formulation (proposed in Section 11.3.1) is as follows.

$$\textbf{[IP1]} \quad \min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{11.22a}$$

$$s.t. \quad (11.4b),(11.4c),(11.21b),(11.7a)-(11.7c),(11.9a),(11.9b),$$

$$(11.20b)-(11.20d),(11.21d),(11.21f)$$

$$\sigma_{ij} \le r_i,\ \ \sigma_{ij} \le \delta_{ji},\ \ \sigma_{ij} \ge r_i + \delta_{ji} - 1, \qquad \forall i,j \in V \tag{11.22b}$$

$$\zeta_{ijl} \le x_{jl},\ \ \zeta_{ijl} \le \delta_{ji},\ \ \zeta_{ijl} \ge x_{jl} + \delta_{ji} - 1, \qquad \forall i \in V; \forall (j,l) \in A \tag{11.22c}$$

$$\psi_{ijl} \le r_i,\ \ \psi_{ijl} \le x_{jl},\ \ \psi_{ijl} \le \delta_{ji},\ \ \psi_{ijl} \ge r_i + x_{jl} + \delta_{ji} - 2, \qquad \forall i \in V; \forall (j,l) \in A \tag{11.22d}$$

$$\phi_{ijl} \le r_i,\ \ \phi_{ijl} \le x_{jl},\ \ \phi_{ijl} \ge r_i + x_{jl} - 1, \qquad \forall i \in V; \forall (j,l) \in A \tag{11.22e}$$

$$\sigma_{ij}, \zeta_{ijl}, \psi_{ijl}, \phi_{ijl} \in \{0,1\}, \qquad \forall i,j,l \in V \tag{11.22f}$$

where (11.22b) – (11.22f) are for linearizing (11.9) by defining new binary variables $\sigma_{ij} = r_i\delta_{ji},\ \zeta_{ijl} = x_{jl}\delta_{ji},\ \psi_{ijl} = r_i x_{jl}\delta_{jl}$, and $\phi_{ijl} = r_i x_{jl}$.

11.3.5.3. Intuitive recursive APMCHCP formulation

The complete APMCHCP model with the first recursive formulation (proposed in Section 11.3.3) is as follows.

$$\textbf{[IP3]} \quad \min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{11.23a}$$

$$s.t. \quad (11.4b),(11.4c),(11.21b),(11.20b)-(11.20d),$$

$$(11.16a)-(11.16c),(11.21d),(11.21f)$$

$$q_{ikj} \le x_{kj},\ \ q_{ikj} \le y_{ik},\ \ q_{ikj} \ge x_{kj} + y_{ik} - 1, \qquad \forall i,j \in V, i \ne j; \forall k \in V, k \ne i,j \tag{11.23b}$$

$$y_{ij}, q_{ikj} \in [0,1], \qquad \forall i,j,k \in V \tag{11.23c}$$

where (11.23b) is for linearizing (11.15).

11.3.5.4. Efficient recursive APMCHCP formulation

The complete APMCHCP model with the second recursive formulation (proposed in Section 11.3.4) is as follows.

$$
\textbf{[IP4]} \quad \min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{11.24a}
$$

$$
\begin{aligned}
s.t.\quad & (11.4b),(11.4c),(11.21b),(11.20b)-(11.20d),\\
& (11.19a),(11.19b),(11.21d),(11.21f)\\
& s_{ij} \le r_i,\;\; s_{ij} \le y_j,\;\; qs_{ij} \ge r_i + y_j - 1, && \forall i,j \in V, && (11.24b)\\
& y_i, s_{ij} \in [0,1], && \forall i,j \in V && (11.24c)
\end{aligned}
$$

where (11.24b) is for linearizing (11.15b).

CHAPTER 12. COMBINATORIAL BRANCH-AND-BOUND

In this chapter, we propose a CBB to solve the APMCHCP.

12.1. Data preprocessing

The main aim of data preprocessing is to construct "tighter" formulations of a problem such that no optimal solution of the original problem is lost and each optimal solution of the tighter problem corresponds to an optimal solution of the original problem. To the best of our knowledge, this kind of preprocessing was firstly suggested by Desrosiers et al. (1995).

For the APMCHCP, the preprocessing includes three main stages: (i) tightening the probability intervals, (ii) constructing precedences among the vertices, and (iii) eliminating arcs. Note that the computation of these three steps should be repeated until no further modifications can be made, as the probability intervals (see Section 12.1.1), the precedence relations (see Section 12.1.2), and the reduced arc set (see Section 12.1.3) mutually affect each other. More details for such repeated data preprocessing procedures can be found in Desrosiers et al. (1995).

12.1.1. Tightening of the probability intervals

This approach allows us to increase the values of β_i's. The key idea here is that if the smallest arrival probability at vertex i from any of its predecessors is bigger than β_i, then β_i can be set to that value. In other words, we can tighten β_i by

$$\beta_i \leftarrow \max\left\{\beta_i, \min_{j\in V, j\neq i}\{\beta_j p_j p_{ji}\}\right\}, \quad \forall i \in V. \tag{12.1}$$

12.1.2. Construction of precedences

When the maximum probability path from vertices j to i is smaller than β_i, clearly, i must precede j in all feasible solutions. For all $j \in V$, let π_j be the set of all vertices that must precede j, namely,

$$\pi_j := \{i \in V \setminus j \mid LAP(j,i) < \beta_i\}, \tag{12.2}$$

where $LAP(j,i)$ is the largest arrival probability at vertex i from vertex j. It is not difficult to see that

$$LAP(i,j) \equiv MPP(i,j) \ . \tag{12.3}$$

Let $i \prec j$ denote the relationship that vertex i must precede vertex j in all feasible solutions. Let $\mathscr{P} = (V, \mathscr{A})$ be the precedence digraph defined on V, where an arc $(i,j) \in \mathscr{A}$ represents a precedence relationship $i \prec j$. Clearly, $\mathscr{P}$ must be acyclic and can be assumed to be transitively closed.

12.1.3. Elimination of arcs

Elimination of arcs is done through three steps described as follows.

Step1: By definition, if (i,j) is in the arc set $\mathscr{A}$ of the precedence digraph, then arc (j,i) cannot be contained in any feasible Hamiltonian path and hence we can delete all these arcs from the original arc set A. Furthermore, for all vertices $i,j,k \in V$ with $(i,j) \in \mathscr{A}$ and $(j,k) \in \mathscr{A}$, we can conclude that arc (i,k) cannot be used in any feasible Hamiltonian path as vertex j has to be sequenced between i and k. Therefore, those (i,k) arcs can also be eliminated from A.

Step2: For each arc $(i,j) \in A$, if $p_i p_{ij} p_j < \beta_j$ then arc (i,j) cannot be contained in any feasible Hamiltonian path. So we can delete all these arcs from A.

Before showing Step 3, we define *concatenation* which is the key of Step 3.

Definition 17 (Concatenation)**.** *Suppose we are given a family $\mathscr{F} := \{\mathscr{R}_1, \mathscr{R}_2, \ldots, \mathscr{R}_k\}$ of simple routes such that $\mathscr{R}_i \cap \mathscr{R}_j = \emptyset$, $\forall i,j \in \{1,2,\ldots,k\}, i \neq j$, and let ω be any permutation of the indices of $\mathscr{F}$. The route $\mathscr{R} = (\mathscr{R}_{\omega(1)}, \mathscr{R}_{\omega(2)}, \ldots, \mathscr{R}_{\omega(k)})$ is called a concatenation of the routes in $\mathscr{F}$ (Ascheuer et al., 2001).*

Note that it is possible that routes $\mathscr{R}_1, \mathscr{R}_2, \ldots, \mathscr{R}_k$ are individually feasible but there is no way to concatenate them to get a feasible route.

Step3: Consider the new arc set A after steps 1 and 2. For all arcs $(i,j) \in A$, we start with route $\mathscr{R} = (i,j)$ and concatenate $\mathscr{R}$ with routes constructed by vertices from a given vertex set $\mathscr{V} := \{v_1, \ldots, v_k\}$. If all these concatenations result in infeasible routes, we can conclude that arc (i,j) cannot be used in any feasible solution and it will be eliminated from A. Due to the fact that the total number of such concatenations increases exponentially with respect to $|\mathscr{V}|$, for the sake of computational efficiency, in our implementation we only consider the case for $|\mathscr{V}| \leq 2$.

In the remaining part of Chapter 12, A is considered to be the feasible arc set after the above mentioned data preprocessing stages.

12.2. The search tree structure

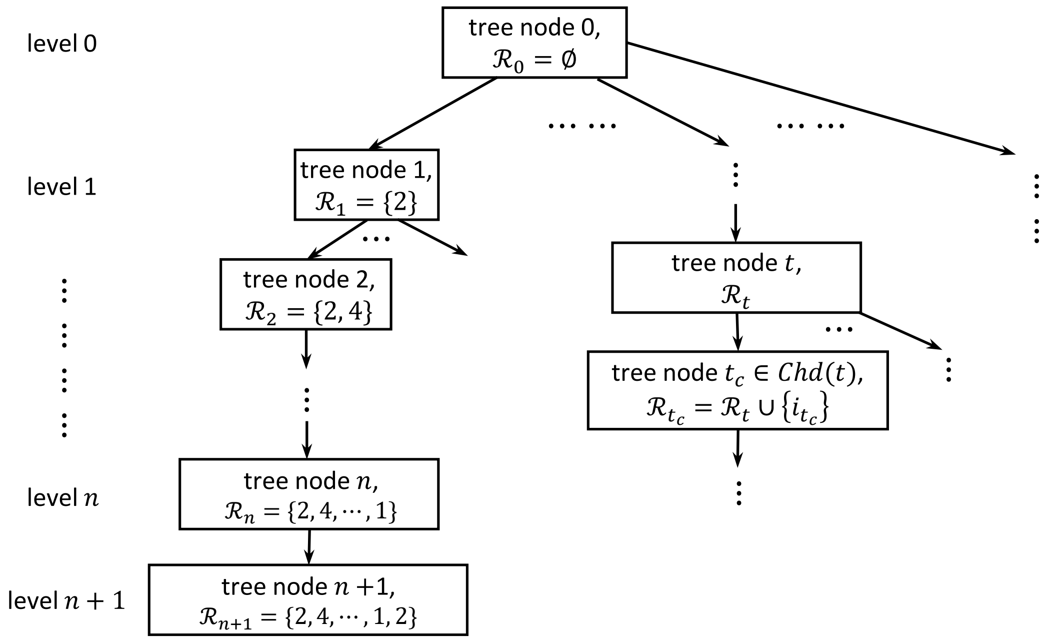
We begin by describing the search scheme used to identify feasible solutions that are candidates to become an optimal solution to APMCHCP.

Let node 0 be the root node of the search tree. For each search tree node t, we define two sets: set of visited vertices and set of candidate vertices. Let $\mathscr{R}_t$ denote the ordered set of visited vertices, which contains all vertices that have been visited in order along the search path that starts from tree node 0 and ends at tree node t. Let $\mathscr{C}_t$ denote the set that contains all candidate vertices in $V \setminus \mathscr{R}_t$ that can be added to set $\mathscr{R}_t$. In addition, let $par(t)$ and $Chd(t)$ be the parent node and the set of child nodes of tree node t, respectively. By the definition of APMCHCP, a vertex $i \in \mathscr{C}_t$ can be a candidate vertex to create a tree node $t_c \in Chd(t)$, in which

$$\mathscr{C}_t := \begin{cases} \left\{ v \in V \mid v \notin \mathscr{R}_t,\ \ p_{\mathscr{R}_t} p_v p_{fv} \geq max\{\beta_v, \beta_s\} \right\}, & \text{if } |\mathscr{R}_t| \leq n-1 \\ \{s\}, & \text{if } |\mathscr{R}_t| = n \text{ and } p_{\mathscr{R}_t} p_{f,s} \geq \beta_s\,, \\ \emptyset, & \text{otherwise} \end{cases} \tag{12.4}$$

where f and s are the last and the first vertices in $\mathscr{R}_t$. We set $\mathscr{R}_0 = \emptyset$ and the initial $\mathscr{C}_0 = V$. Figure 12.1 gives an example of the structure of search tree of an APMCHCP.

Clearly, each tree node t has up to $|V \setminus \mathscr{R}_t|$ child nodes. We use depth-first search (DFS) for branching, as the number of nodes in each level of the search tree increases exponentially. If



level 0
level 1
level n
level $n+1$
tree node 0, $\mathcal{R}_0 = \emptyset$
tree node 1, $\mathcal{R}_1 = \{2\}$
tree node 2, $\mathcal{R}_2 = \{2, 4\}$
tree node n, $\mathcal{R}_n = \{2, 4, \cdots, 1\}$
tree node $n+1$, $\mathcal{R}_{n+1} = \{2, 4, \cdots, 1, 2\}$
tree node t, $\mathcal{R}_t$
tree node $t_c \in Chd(t)$, $\mathcal{R}_{t_c} = \mathcal{R}_t \cup \{i_{t_c}\}$

Figure 12.1: Illustration on the structure of the search tree of an APMCHCP.

we use breadth-first search, when $|V|$ is large, the memory is likely to get exhausted before finding a solution for the problem, because we cannot have a solution until the last level of the search tree.

For branching tree node 0, we select vertex $i_{t_1} \in \mathscr{C}_0$ and create a new first-level node t_1 by

$$i_{t_1} = \underset{i \in \mathscr{C}_0}{\arg\min}\ \beta_i \ , \tag{12.5}$$

because starting from a vertex $i \in V$ with a small β_i is more likely to find a feasible tour. If multiple vertices satisfy (12.5), then one of them is selected at random. Next, we update $\mathscr{C}_0 = \mathscr{C}_0 \setminus \{i_{t_1}\}$, setting $\mathscr{R}_{t_1} = \{i_{t_1}\}$, and initiate $\mathscr{C}_{t_1}$ by (12.4). To branch a tree node t in levels $l = 1, 2, \ldots, n$ (if exists, i.e., $\mathscr{C}_t \neq \emptyset$) and create a new node $t_c \in \mathscr{C}hd(t)$, we select vertex $i_{t_c} \in \mathscr{C}_t$ using the following *greedy rule* :

$$i_{t_c} = \underset{i \in \mathscr{C}_t}{\arg\max}\ p_i p_{f,i} \ , \tag{12.6}$$

where f is the last vertex in $\mathscr{R}_t$. If multiple vertices satisfy (12.6), then the one with the smallest $c_{f,i}$ is selected. Next, we set $\mathscr{R}_{t_c} = \mathscr{R}_t \cup \{i_{t_c}\}$ (i.e., to add element i_{t_c} to the end of $\mathscr{R}_t$ and get $\mathscr{R}_{t_c}$), updating $\mathscr{C}_t = \mathscr{C}_t \setminus \{i_{t_c}\}$, and initiate $\mathscr{C}_{t_c}$ by (12.4). In each search path, level $n+1$, if exists, returns to the starting vertex to create a Hamiltonian cycle.

The cost of a tree node t, z_t, can be calculated by :

$$z_t = \begin{cases} z_{par(t)} + c_{f,i_t} \ , & \text{if } t \neq 0, t \notin Chd(0) \\ 0 \ , & \text{otherwise} \end{cases} \ , \tag{12.7}$$

where f and i_t are the last vertices in $\mathscr{R}_{par(t)}$ and $\mathscr{R}_t$, respectively.

Obviously, if $\mathscr{C}_t = \emptyset$, then tree node t will be fathomed. Otherwise, the algorithm backtracks by DFS to find a new tree node. In the following, we provide several feasibility rules for the APMCHCP which provide more criteria to fathom a tree node or detect the infeasibility of a given problem.

12.3. Feasibility rules

For the TSP, the most important and difficult constraint is to visit all vertices (Dumas et al., 1995). When probability intervals and precedence constraints are present, they impose partial orders of the vertices. Computational efficiency can be improved from eliminating partial routes that do not satisfy such partial orders. The feasibility rules presented here detect partial routes that cannot be extended to form a feasible Hamiltonian cycle, thereby allowing elimination of such routes.

To describe those rules, let $SDP(i,j)$ be the *smallest departure probability* from vertex i such that β_j is satisfied. This can be calculate as follows:

$$SDP(i,j) = \frac{\beta_j p_i}{MPP(i,j)} \ . \tag{12.8}$$

Rule 1. (feasibility check) A partial route $\mathscr{R}$ ended at vertex i cannot be extended to a feasible solution, if we have

$$p_{\mathscr{R}} < \max_{k \in V \setminus \mathscr{R}} \{SDP(i,k)\} \ . \tag{12.9}$$

Based on this (global) rule, if extending partial route $\mathscr{R}$ to an unvisited vertex is infeasible, then route $\mathscr{R}$ can be eliminated. Also, a route $\mathscr{R}'$ can be eliminated, if it visits exactly the same vertices as $\mathscr{R}$ and $p_{\mathscr{R}'} \leq p_{\mathscr{R}}$ (the dominance rule).

Rule 2. (precedence relationship check) A partial route $\mathscr{R}$ ended at vertex i is infeasible if we have

$$\pi(i) \not\subset \mathscr{R} \,. \tag{12.10}$$

This rule states that all predecessors of vertex i must be visited before visiting i.

Rule 3. (extension check) Given partial route $\mathscr{R}$ ending at vertex i, if $\mathscr{R}$ can be extended to a vertex $k \notin \mathscr{R}$ but cannot be further extended to some $l \notin \mathscr{R}$, i.e., $\exists k, l \notin \mathscr{R}, k \neq l$ such that $p_{\mathscr{R}} \geq SDP(i,k)$ but $p_{\mathscr{R}} p_{ik} p_k < SDP(k,l)$, then $\mathscr{R}$ cannot be extended toward k. Also, a route $\mathscr{R}'$ ending with i cannot be extended toward k, if it includes exactly the same vertices as $\mathscr{R}$ and $p_{\mathscr{R}'} \leq p_{\mathscr{R}}$ (the dominance rule).

Rule 4. (probabilistic Hamiltonian cycle (PHC) check) For a given origin vertex i, the feasibility of building a Hamiltonian cycle can be checked using the following proposition.

Proposition 22. *An APMCHCP starting at vertex i is infeasible if the following restricted APMCHCP (RAPMCHCP) is infeasible.*

$$\textbf{\textit{[RAPMCHCP(i)]}} \quad \min\ 0 \tag{12.11a}$$

$$s.t. \quad (11.3a)-(11.3c), (11.20b)-(11.20d)$$

$$\prod_{j\in V} p_j \prod_{(j,l)\in A} p_{jl}^{x_{jl}} \geq \beta_i \,, \qquad \forall i \in V \tag{12.11b}$$

Proof. If (12.11) is infeasible, we have $\prod_{j\in V} p_j \prod_{(j,l)\in A} p_{jl}^{x_{jl}} < \beta_i$, which makes the original APMCHCP infeasible due to Lemma 9. □

We apply Proposition 22 because in general, an RAPMCHCP can be solved much faster than an APMCHCP, and it is useful to recognize the infeasibility at the first level of the CBB search tree. In real applications, the RAPMCHCP can be linearized by logarithmizing both sides of (12.11b) and easily solved as a binary linear program.

Rule 5. (return-to-the-origin check) At each tree node t in levels $l = 1, 2, \ldots, n$, we check the feasibility of returning to the origin by finding the maximum possible probability for the remaining arcs. In tree node t associated with route $\mathscr{R}_t$, the chance constraint of the origin will be violated if we have

$$p_{\mathscr{R}_t} \prod_{j \in V \setminus \mathscr{R}_t} p_j \prod_{i \in (V \setminus \mathscr{R}_t) \cup \{f\}} \max_{k \in (V \setminus \mathscr{R}_t) \cup \{s\}} \{P_{ik}\} < \beta_s \ . \tag{12.12}$$

12.4. Lower bound

Proposition 23. *Given a search tree node t associated with a nonempty candidate set $\mathscr{C}_t$, the objective value of any Hamiltonian cycle containing route $\mathscr{R}_t = (s, v_2, \ldots, v_{k-1}, f)$ has a lower bound*

$$z_{\mathscr{R}_t} + \sum_{i \in (V \setminus \mathscr{R}_t) \cup \{f\}} \min_{k \in (V \setminus \mathscr{R}_t) \cup \{s\}} \{PSP_{ik}\} \ , \tag{12.13}$$

where PSP_{ik} is the shortest probabilistic path between vertices i and k. The value of $PSP_{ik}, \forall (i,k) \in A$, can be calculated by Algorithm 6.

Proof. Let $\mathscr{R}_t^*$ be the optimal Hamiltonian cycle containing $\mathscr{R}_t$. We have

$$z_{\mathscr{R}_t^*} = z_{\mathscr{R}_t} + z_{\bar{\mathscr{R}}_t} \ , \tag{12.14}$$

where $\bar{\mathscr{R}}_t := (f, v_{k+1}, v_{k+2}, \ldots, v_n, s)$. $\bar{\mathscr{R}}_t$ can be found by solving an elementary shortest path problem with resource constraints (ESPPRC) (see Irnich et al. (2005)), where probability as a resource is accumulated along the route. We call this problem elementary shortest path problem with chance constraints (ESPPCC). However, ESPPRC is proved to be strongly NP-hard (Dror, 1994), and hence the same for ESPPCC. If we relax some of the constraints of ESPPCC, then the problem becomes easier to be solved. In particular, we can change the problem to the shortest path problem with chance constraints (SPPCC) by allowing cycles in the tour. Let $\hat{\bar{\mathscr{R}}}_t$ be the SPPCC that visits all the vertices of $\bar{\mathscr{R}}_t$ for at least once. Clearly, we must have

$$z_{\bar{\mathscr{R}}_t} \geq z_{\hat{\bar{\mathscr{R}}}_t} \ . \tag{12.15}$$

By relaxing assignment constraints and subtour elimination constraints, one can apply Algorithm 6 for all vertices in $(V \setminus \mathscr{R}_t) \cup f$ to find the lower bound of $z_{\hat{\bar{\mathscr{R}}}_t}$, because

$$z_{\hat{\bar{\mathscr{R}}}_t} \geq \sum_{i \in (V \setminus \mathscr{R}_t) \cup \{f\}} \min_{k \in (V \setminus \mathscr{R}_t) \cup \{s\}} \{PSP_{ik}\} \ . \tag{12.16}$$

By inequalities (12.14), (12.15), and (12.16), we get

$$z_{\mathscr{R}_t^*} \geq z_{\mathscr{R}_t} + \sum_{i \in (V \setminus \mathscr{R}_t) \cup \{f\}} \min_{k \in (V \setminus \mathscr{R}_t) \cup \{s\}} \{PSP_{ik}\}, \tag{12.17}$$

which completes the proof. □

12.5. Upper bound

To find the upperbound of the APMCHCP, we solve the following problem.

$$[UB] \quad \max \sum_{(i,j) \in A} \log(p_{ij}) x_{ij} - au \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{12.18a}$$

$$s.t. \quad (11.3a) - (11.3c), (11.20b) - (11.20d)$$

In the objective function (12.18a), the first part is to maximize the cumulative arc probability $\prod_{(i,j) \in A} p_{ij}^{x_{ij}}$, and the second part is an adjustment to get a relatively small total cost $\sum_{(i,j) \in A} c_{ij} x_{ij}$,

Algorithm 6: Finding probabilistic shortest path (PSP).

```
Data: Digraph G
Result: The probabilistic shortest path between any two given vertices
for $i \in V$ do
    $dist[i] \leftarrow 0, prob[i] \leftarrow p_i$                // determine source vertex
    $Q.push(i, dist[i], prob[i])$                                // Q is a priority queue
    while $Q \neq \emptyset$ do
        $u \leftarrow$ pop(Q)        // pop(Q) extracts vertex in Q with min dist[]
        for each successor $v$ of $u$ do
            if $prob[u]p_{uv}p_v \geq \beta_v$ then
                $alt = dist[u] + c_{uv}$
                if $v \notin dist$ or $alt < dist[v]$ then
                    $dist[v] \leftarrow alt$
                    $Q.push(v, alt, prob[u]p_{uv}p_v)$
    return $dist[]$
```

where a is a user-defined scaling parameter for balancing the weight of $\log(p_{ij})$ and c_{ij} in the objective function, and u is a loop parameter used in Algorithm 7 to adjust the scaling level.

Proposition 24. *Given digraph G corresponding with an APMCHCP, Algorithm 7 finds the upper bound.*

12.6. The complete CBB algorithm

The complete CBB algorithm for solving the APMCHCP is given in Algorithm 8.

Algorithm 7: Greedy algorithm to find the upper bound.

```
Data: Digraph G
Result: Upper bound for the APMCHCP corresponding to G
1  $upperbound \leftarrow \infty$
2  $u \leftarrow 0$
3  while $u \leq |\frac{1}{a}|$ do
4      $update \leftarrow$ "no"
5      solve UB and put the optimal tour in $t^*$
6      for $i \in V$ do
7          if there is a feasible APMCHCP starting from vertex i in $t^*$ then
8              $upperbound \leftarrow z_{t^*}$                    // $z_{t^*}$ is the objective value of $t^*$
9              $update \leftarrow$ "yes"
10             break
11     if $update$ ="no" then
12         break
13     $u = u + 1$
14 return $upperbound$
```

Algorithm 8: The complete CBB algorithm for APMCHCP.

```
1  CALL data preprocessing
2  CALL Algorithm 7
3  if An upper bound can be found using Algorithm 7 then
4      $\mathscr{R}^* \leftarrow \mathscr{R}^*_{ub}$, $z^* \leftarrow z^*_{ub}$
5  else
6      $\mathscr{R}^* \leftarrow \emptyset$, $z^* \leftarrow \infty$
7  Create root node: $l \leftarrow 0; \mathscr{R}_0 \leftarrow \emptyset; \mathscr{C}_0 \leftarrow V; stack \leftarrow (l, \mathscr{R}_0, \mathscr{C}_0)$
8  while $stack \neq \emptyset$ do
9      node $t$=stack.pop()
10     if $l = 0$ then
11         determine $i_{t_1}$ using (12.5)
12         if $\pi(i_{t_1}) = \emptyset$ and $i_{t_1} \notin \bigcup_{j \in V \setminus \{i_{t_1}\}} \pi(j)$ then
13             if RAPMCHCP(i) is feasible then
14                 if Feasibility Check (Rule 1) then
15                     $\mathscr{R}_{t_1} \leftarrow \{i_{t_1}\}$; initiate $\mathscr{C}_{t_1}$ using (12.4); reduce $\mathscr{C}_{t_1}$ using Rule 3; $l \leftarrow l+1$;
                         calculate $z_{t_1}$ by (12.7) and $lb$ using (12.13); $stack \leftarrow (l, \mathscr{R}_{t_1}, \mathscr{C}_{t_1})$
16     else
17         if $\mathscr{C}_t \neq \emptyset$ then
18             if $lb_t < z^*$ then
19                 determine $i_{t_c}$ using the greedy rule described in (12.6)
20                 if $\pi(i_{t_c}) \subseteq \mathscr{R}_t$ then
21                     if Returning to Origin Check (Rule 5) then
22                         if Feasibility Check (Rule 1) then
23                             $\mathscr{R}_{t_c} \leftarrow \mathscr{R}_t \cup \{i_{t_c}\}$; initiate $\mathscr{C}_{t_c}$ using (12.4); reduce $\mathscr{C}_{t_1}$ using Rule 3;
                                 $l \leftarrow l+1$;  calculate $z_{t_c}$ by (12.7) and $lb$ using (12.13);
                                 $stack \leftarrow (l, \mathscr{R}_{t_c}, \mathscr{C}_{t_c})$
24         else
25             if $|\mathscr{R}_t| = n+1$ then
26                 if $z_t < z^*$ then
27                     $\mathscr{R}^* \leftarrow \mathscr{R}_t$, $z^* \leftarrow z_t$
28 return $\mathscr{R}^*, z^*$
```

CHAPTER 13. COMPUTATIONAL RESULTS

In this chapter we present the computational results for both problems presented in Chapters 6 and 10.

13.1. Novel group centrality metrics for studying essentiality in protein-protein interaction networks

In this section, we present our experimental setup. Our goal is to portray how different group centrality metrics behave and perform when put to the test against each other as well as against nodal centrality metrics in PPIN analysis.

13.1.1. Experimental setup

We coded all the algorithms in Python and conducted the experiments on a quad-core Intel i7 at 2.8 GHz with 16 GB of RAM. We also used Gurobi Optimizer 8.0 to solve all MILP formulations. Data on protein interactions for different organisms was obtained by STRING v. 10.0 (Szklarczyk et al., 2014). More specifically, we used the data set for helicobacter pylori (stylized hereafter as HP). Essential proteins for the organism were found using the curated database in DEG 10 (Luo et al., 2013).

The experiment is organized as follows: in subsection 13.1.2, we describe how PPINs are generated by creating a graph vertex for each protein; the vertex is connected to all other proteins-vertices that share an interaction. Each interaction comes with an interaction score (out of 1000). To simplify the generation of networks to apply our group centrality metrics on, we select a threshold score below which all interactions are removed. The threshold scores selected for presentation in this study were 600 (60% interaction score), 700 (70% interaction score), and 800 (80% interaction score). In this fashion, we were able to create three networks where all known centrality metrics can be captured given the available computational power. The networks were further broken down into their connected components with each component being independently analyzed, without loss of generality.

For each centrality metric, the experiment is performed in three sections. First, the performance of the proposed CBB approaches are compared to the MILP formulations, both as far as computational runtime and solution quality are concerned. Secondly, we measure the number of essential proteins that appear among the top and bottom *k* ranked proteins (for different values of *k* as shown in Table 13.23) using each of the approaches. Lastly, We define the membership measure as the number of times that a protein appears in the optimal structures of all proteins for the combination of each centrality measure and structure. The goal of defining this measure is to know is there any relation between the protein essentiality and membership. Here, we also contrast this metric to nodal metrics of centrality (degree, closeness, betweenness). All nodal metrics are computed with a Python implementation, using networkX version 2.4 (Hagberg et al., 2008).

13.1.2. Protein-protein interaction network

Table 13.23 shows the number of vertices and also the number of edges for the HP PPIN for four given thresholds, namely 0 (all interactions present), 600 (only interactions with a threshold score of 60% and above), 700 (70% threshold), and 800 (80% threshold). We can also see the number of essential proteins and top and bottom *k* proteins for HP in this table. We opted for $k = 500$ seeing as helicobacter pylori has 323 essential proteins reported in DEG.

Table 13.1: Details of the PPIN of the selected organism.

Organism	**|V|**	**|E|**				**Essential Proteins**	**Top k**	**Bottom k**
		0	600	700	800			
Helicobacter pylori	1,570	89,648	17,792	12,822	7,859	323	500	500

13.1.3. Degree centrality

13.1.3.1. Comparing the CBB algorithm to the mathematical formulations

The computational results when solving the MDRP, MDCP, and MDSP using the MILP formulations and proposed CBB and CG algorithms for Helicobacter pylori are presented in this section. Specifically, we compare the performance of the CBB algorithm, CG algorithm, IP1 and IP2 in practice, using the obtained PPINs. All results are summarized in Table 13.24.

The first three columns provide general information for each problem instance (organism name, chosen threshold, and structure which are indicated by Org, Thr, and Str, respectively).

Table 13.2: Results of the experiments for degree centrality on the Helicobacter pylori. IP1 is the linear IP developed by Vogiatzis et al., 2015, IP2, CBB and CG are our proposed linear ILP, CBB algorithm, and CG algorithm, respectively. Gap (%) represents the optimality gap, and time(s) is the runtime in second.

Org	**Thr**	**Str**	**IP1**		**IP2**		**CBB**		**CG**	
			Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)	Gap (%)	Time (s)
HP	600	Clique	0.00	644.89	0.00	324.37	0.17	510.23	0.43	2547.85
HP	700	Clique	0.00	458.50	0.00	210.92	0.17	409.16	0.29	1428.07
HP	800	Clique	0.00	435.19	0.00	116.93	0.17	297.34	0.10	673.15
HP	600	Star	0.00	629.47	0.00	321.72	0.38	568.54	0.77	2684.90
HP	700	Star	0.00	465.77	0.00	206.45	0.10	314.82	0.38	1531.57
HP	800	Star	0.00	303.98	0.00	113.74	0.04	145.85	0.10	704.29
HP	600	Rep	0.00	768.96	0.00	268.01	0.00	909.09	0.00	1607.88
HP	700	Rep	0.00	430.37	0.00	170.93	0.00	630.20	0.00	1088.72
HP	800	Rep	0.00	317.64	0.00	104.53	0.00	299.09	0.00	502.72

The next sets of columns show the performance of the formulation in the literature, our new formulation, and our proposed CBB algorithm on the chosen problem, respectively. Notice that the optimality gap reported is the average of all optimality gaps for the proteins that their optimal structure cannot be found. The gap for each protein is calculated by dividing the difference between the optimal degree centrality and best degree centrality found in 1 second by the optimal degree centrality. The runtime of our CBB algorithm and Gurobi is limited to 1 second per protein.

We make the following observations. As shown in Table 13.24, for all instances, IP2 has a better performance than IP1 in terms of runtime. This observation might be due to the fact that IP2 uses the idea that the group degree centrality is a local measure and isolates the local neighborhood of each protein to find its most-connected structure. In this way, IP2 decreases the number of variables and constraints, which results in a much shorter runtime.

When it comes to the CBB, it can find the optimal solution for representative-based problems as it takes advantage of the set cover problem to limit the number of branching vertices in the root node of the search tree. In terms of runtime for representative-based problems, we can see that the CBB is faster than IP1 when threshold is 800, but slower for 600 and 700 (where the instances are of bigger size). For the clique and star-based problems, CBB is faster than IP1 but not IP2, and it fails to always find an optimal solution. In clique-based problems, CBB can find the optimal clique for 94, 95, and 97 percent of proteins in the given time limit when the threshold is set to 600, 700, and 800, respectively. In star-based problems, CBB can find the optimal clique for 92, 98, and 99 percent of proteins in the given time limit when the threshold is 600, 700, and 800, respectively. The average gap for all proteins is reported in the gap column of Table 13.24.

CG algorithm can find the optimal solution for representative-based problems based on proposition 18. For the clique and star-based problems, CG fails to always find an optimal solution. In clique-based problems, CG can find the optimal clique for 89, 94, and 98 percent of proteins in the given time limit when the threshold is set to 600, 700, and 800, respectively. In star-based problems, CG can find the optimal clique for 85, 93, and 98 percent of proteins in the given time limit when the threshold is 600, 700, and 800, respectively. In terms of runtime, we can see that the CG is slower than all other methods. Comparing the average gap for all proteins, we can see that although both CBB and CG algorithms have a small gaps, CBB is better than CG in this regard.

13.1.3.2. Analysis of top ranked proteins per metric

After calculating the centrality metrics of all proteins for each PPIN, we may compute the ratio of essential proteins in the top *k* and the bottom *k* proteins (as ranked by each centrality metric). For Helicobacter pylori, the number of essential proteins found in the PPIN is 323, and hence the top 500 proteins are investigated.

For each threshold we provide two Figures and one Table. The first figure represents the performance over the top *k* proteins; the second pair is for the bottom *k* proteins. The table, on the other hand, summarizes numerically the same results for both top and bottom *k* proteins. Note that for the first representation, the higher the ratio, the better that metric is said to perform. The opposite is true for the second representation as a metric is said to perform better, when the ratio is smaller. In this section, we only provide the interpretation of the figures and tables when thresholds is 600. For the other two thresholds 700 and 800, the explanations are similar.

The results for a threshold of 600 are shown in Figures 13.5 and 13.6. Clique-CBB and Clique-IP achieve a final score of detecting 49.85% essential proteins within the top 500 proteins, as opposed to 46.44% for degree and 46.13% for closeness. Considering the performance over the least well ranked proteins, we can see that Clique-CBB, Clique-IP, and simple closeness are best at not ranking highly non-essential proteins, achieving a final score of 19.20%, while the score for the other competitive metric is 22.60% for degree. It is also worth mentioning that the CBB and IP have the same performance in determining essential proteins, but the CBB is faster.

Table 13.3: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using group degree centrality and a threshold score of 800.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	12.07	21.05	28.48	34.98	43.65	4.02	7.12	11.15	16.72	22.29
Clique-IP	12.07	21.05	28.48	34.98	43.65	4.02	7.12	11.15	16.72	22.29
Star-CBB	11.76	20.12	28.79	36.22	43.34	2.48	7.12	11.46	17.96	23.53
Star-IP	11.76	20.12	28.79	36.22	43.34	2.48	7.12	11.46	17.96	23.53
Rep-IP-CBB	11.46	21.05	29.72	36.22	43.65	2.48	7.12	11.46	17.03	22.91
Degree	11.46	18.58	26.32	36.53	40.56	2.48	7.12	12.07	16.10	21.05
Closeness	3.41	6.50	14.86	24.77	33.75	5.26	10.22	14.24	19.50	24.15
Betweenness	8.98	15.48	22.29	29.72	35.60	3.10	7.74	13.00	18.27	25.39

Table 13.4: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using group degree centrality and a threshold score of 700.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	10.22	21.36	31.27	39.94	47.68	3.10	7.12	11.46	14.24	18.27
Clique-IP	5.57	12.07	18.89	24.77	32.20	8.05	13.31	19.20	25.70	30.96
Star-CBB	6.50	15.17	23.22	27.55	31.89	4.33	8.36	17.34	23.22	30.34
Star-IP	6.50	15.17	23.22	27.55	31.89	4.33	8.36	17.34	23.22	30.34
Rep-IP-CBB	6.81	14.86	22.29	26.93	33.13	4.33	8.36	17.03	24.15	30.03
Degree	14.24	21.98	30.03	39.32	44.58	3.10	8.98	12.07	15.79	21.05
Closeness	3.41	14.55	23.53	33.75	43.96	2.17	6.50	9.29	12.07	16.72
Betweenness	10.22	15.79	25.39	32.51	38.70	5.26	8.98	16.41	19.81	27.24

13.1.3.3. Analysis of top ranked proteins per metric using membership measure

After identifying the optimal structure of all proteins for each PPIN using each approach, we can calculate the number of times each protein appears in the all optimal structures. For example, if in the MDCP, protein p1 is in the optimal cliques of proteins p2, p10 and p13, then the membership of P1 is 4 (because it appears in it's own clique, also). Then, we may compute the

Table 13.5: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using group degree centrality and a threshold score of 600.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	11.46	21.05	29.72	39.94	49.85	4.02	7.74	10.22	13.31	19.20
Clique-IP	11.76	21.05	29.72	39.94	49.85	4.02	7.74	10.22	13.31	19.20
Star-CBB	8.67	15.48	20.43	26.63	34.37	5.57	14.24	19.81	27.55	33.13
Star-IP	8.67	15.79	20.43	26.63	34.37	5.57	14.24	19.81	27.55	33.13
Rep-IP-CBB	7.43	15.17	21.67	27.24	32.51	5.57	12.69	20.74	26.93	32.82
Degree	14.24	23.53	30.03	37.46	46.44	4.02	9.29	12.07	18.27	22.60
Closeness	7.12	18.58	26.93	35.60	46.13	3.72	6.81	9.91	14.86	19.20
Betweenness	5.57	11.76	17.03	24.15	31.58	4.33	12.69	18.89	26.32	31.58

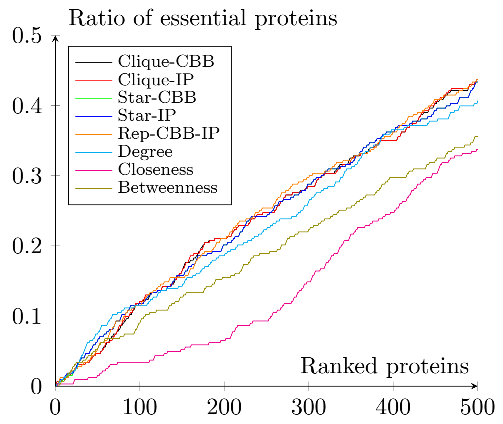


Figure 13.1: The ratio of essential proteins found in the top k ranked proteins with a threshold score of 800. The group centrality metric used is based on degree.

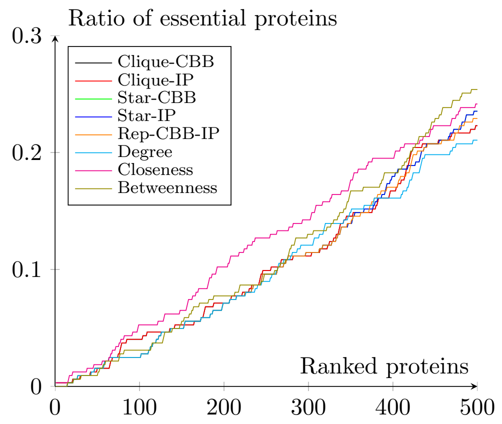


Figure 13.2: The ratio of essential proteins found in the bottom k ranked proteins with a threshold score of 800. The group centrality metric used is based on degree.

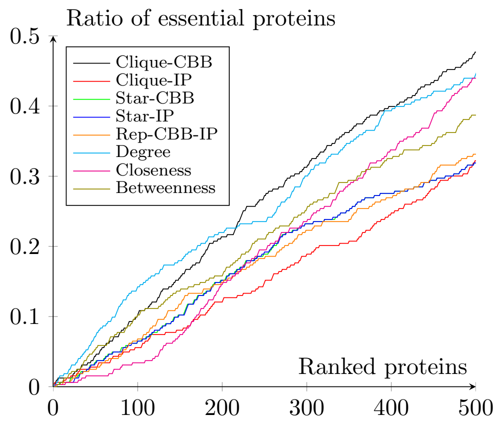


Figure 13.3: The ratio of essential proteins found in the top k ranked proteins with a threshold score of 700. The group centrality metric used is based on degree.

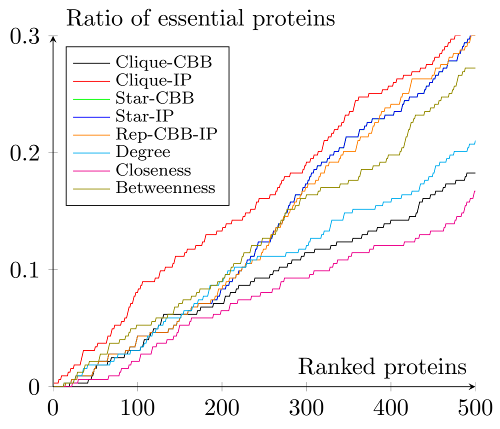


Figure 13.4: The ratio of essential proteins found in the bottom k ranked proteins with a threshold score of 700. The group centrality metric used is based on degree.

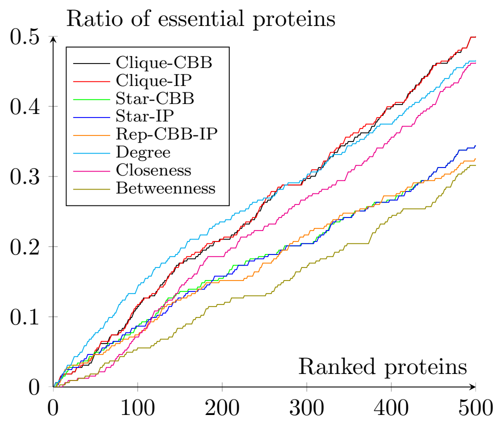


Figure 13.5: The ratio of essential proteins found in the top k ranked proteins with a threshold score of 600. The group centrality metric used is based on degree.

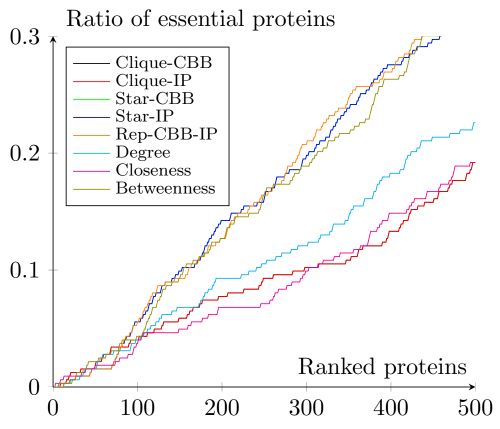


Figure 13.6: The ratio of essential proteins found in the bottom k ranked proteins with a threshold score of 600. The group centrality metric used is based on degree.

ratio of essential proteins in the top *k* and the bottom *k* proteins (as ranked by each membership measure).

For each threshold we provide two Figures and one Table. The explanation of figures and tables are like what we had in the previous section. In this section, we only provide the interpretation of the figures and tables when thresholds is 600. For the other two thresholds 700 and 800, the explanations are similar.

The results for a threshold of 600 are shown in Figures 13.11 and 13.11. Degree and Closeness achieve a final score of detecting 46.44% and 46.13% essential proteins within the top 500 proteins, respectively, as opposed to 44.27% for Clique-CBB and Clique-IP. Considering the performance over the least well ranked proteins, we can see that simple Closeness and Degree are best at not ranking highly non-essential proteins, achieving a final scores of 19.20% and 22.60%, respectively, while the score for the other competitive metric is 26.01% for Clique-CBB. It is also worth mentioning that for clique the CBB and IP have the same performance and for star IP is better than CBB in determining essential proteins, but the CBB is faster.

Table 13.6: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using membership measure for group degree centrality and a threshold score of 800.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	10.53	19.20	25.70	33.44	39.01	9.60	15.17	19.50	26.01	33.44
Clique-IP	10.53	17.96	26.01	33.13	39.32	3.10	7.74	13.00	19.81	26.63
Star-CBB	8.67	15.17	22.60	29.10	33.75	3.10	8.05	13.31	19.50	25.08
Star-IP	8.98	15.79	22.91	28.17	34.98	3.10	8.05	13.31	19.50	25.08
Rep-IP-CBB	9.91	18.89	26.01	31.89	39.63	3.10	7.74	13.31	17.96	23.84
Degree	11.46	18.58	26.32	36.53	40.56	2.48	7.12	12.07	16.10	21.05
Closeness	3.41	6.50	14.86	24.77	33.75	5.26	10.22	14.24	19.50	24.15
Betweenness	8.98	15.48	22.29	29.72	35.60	3.10	7.74	13.00	18.27	25.39

Table 13.7: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using membership measure for group degree centrality and a threshold score of 700.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	9.29	21.67	29.41	37.15	43.96	6.81	11.46	17.03	23.53	28.48
Clique-IP	9.91	21.36	29.41	36.22	44.27	5.26	10.84	15.79	21.67	26.32
Star-CBB	9.29	17.65	25.08	31.58	38.08	5.26	10.53	15.48	21.36	26.93
Star-IP	9.60	17.03	23.84	33.75	39.01	5.26	9.91	14.55	20.43	26.01
Rep-IP-CBB	10.84	18.58	29.72	37.15	42.72	5.26	9.60	14.86	20.43	25.39
Degree	14.24	21.98	30.03	39.32	44.58	3.10	8.98	12.07	15.79	21.05
Closeness	3.41	14.55	23.53	33.75	43.96	2.17	6.50	9.29	12.07	16.72
Betweenness	10.22	15.79	25.39	32.51	38.70	5.26	8.98	16.41	19.81	27.24

Table 13.8: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using membership measure for group degree centrality and a threshold score of 600.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	12.69	19.81	29.10	36.84	44.27	5.26	9.29	14.55	20.74	26.01
Clique-IP	12.07	19.50	28.79	36.53	44.27	5.57	10.84	16.10	20.43	26.63
Star-CBB	8.67	17.96	24.77	30.34	37.77	4.33	9.91	17.65	22.91	29.10
Star-IP	8.98	18.27	25.08	30.96	38.70	4.64	10.22	17.34	22.29	27.24
Rep-IP-CBB	10.84	19.20	26.63	34.37	40.56	4.64	9.60	17.34	23.22	29.10
Degree	14.24	23.53	30.03	37.46	46.44	4.02	9.29	12.07	18.27	22.60
Closeness	7.12	18.58	26.93	35.60	46.13	3.72	6.81	9.91	14.86	19.20
Betweenness	5.57	11.76	17.03	24.15	31.58	4.33	12.69	18.89	26.32	31.58

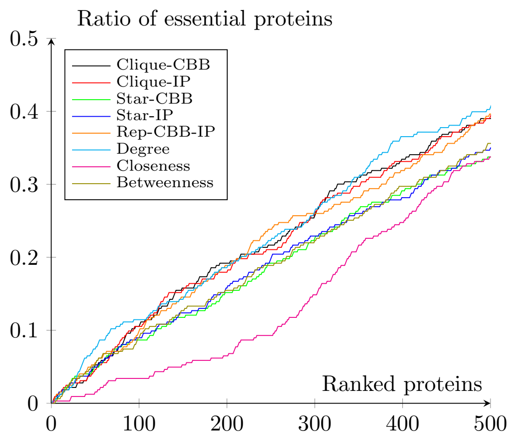


Figure 13.7: The ratio of essential proteins found in the top k ranked proteins using membership measure with a threshold score of 800. The group centrality metric used is based on degree.

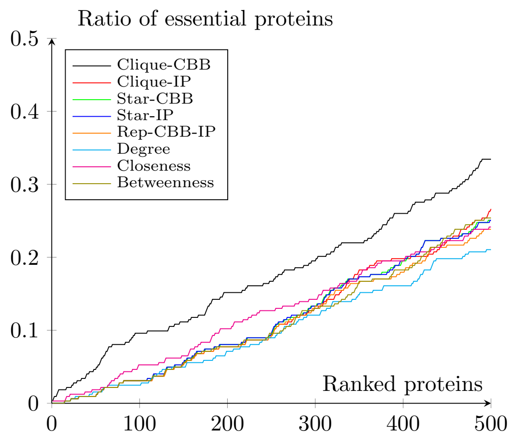


Figure 13.8: The ratio of essential proteins found in the bottom k ranked proteins using membership measure with a threshold score of 800. The group centrality metric used is based on degree.

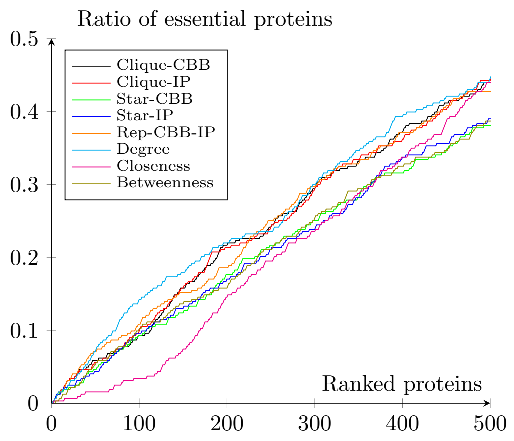


Figure 13.9: The ratio of essential proteins found in the top k ranked proteins using membership measure with a threshold score of 700. The group centrality metric used is based on degree.

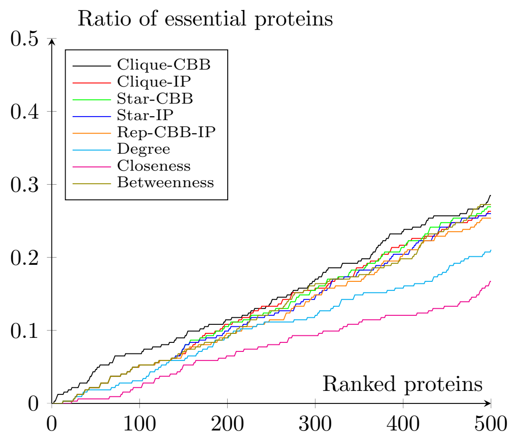


Figure 13.10: The ratio of essential proteins found in the bottom k ranked proteins using membership measure with a threshold score of 700. The group centrality metric used is based on degree.

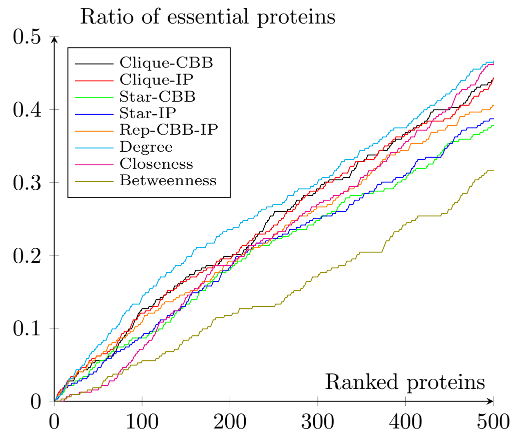


Figure 13.11: The ratio of essential proteins found in the top *k* ranked proteins using membership measure with a threshold score of 600. The group centrality metric used is based on degree.

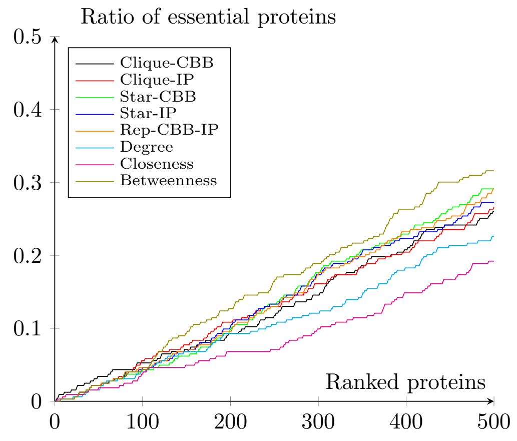


Figure 13.12: The ratio of essential proteins found in the bottom *k* ranked proteins using membership measure with a threshold score of 600. The group centrality metric used is based on degree.

13.1.4. Closeness centrality

13.1.4.1. Comparing the CBB algorithm to the mathematical formulations

Table 13.9: Results of the experiments for closeness centrality on the Helicobacter pylori. IP and CBB are our proposed linear IP and algorithm, respectively. Gap (%) represents the optimality gap, and time(s) is the runtime in seconds.

Org	**Thr**	**Str**	**IP**		**CBB**	
			Gap (%)	Time (s)	Gap (%)	Time (s)
HP	600	Clique	0.01	105795.86	0.15	25667.99
HP	700	Clique	0.00	91971.92	0.08	14301.00
HP	800	Clique	0.00	78147.98	0.03	5939.34
HP	600	Star	0.01	33659.90	0.59	42847.92
HP	700	Star	0.02	33913.44	0.11	21966.91
HP	800	Star	0.00	29251.38	0.03	5996.58
HP	600	Rep	0.00	32556.21	0.11	64064.42
HP	700	Rep	0.00	33364.86	0.11	45438.32
HP	800	Rep	0.00	29136.98	0.09	23519.65

We perform the same analysis, but for closeness centrality in this section. As shown in Table 13.9, in terms of the optimality gap for all instances although IP and CBB algorithm both have a small gap, IP has a better performance than CBB algorithm. However, considering runtime CBB algorithm is about two times faster than IP. For clique-based problems this gap is even bigger, where CBB algorithm is 6 times faster than the IP. This observation might be due to the fact that in the CBB algorithm we take advantage of the Theorem 5 which decreases the number of active nodes of the search tree.

In clique-based problems, CBB algorithm can find the optimal clique for 81, 87, and 92 percent of proteins in the given time limit when threshold is 600, 700, and 800, respectively. In star-based problems, CBB algorithm can find the optimal clique for 64, 82, and 95 percent of proteins in the given time limit when threshold is 600, 700, and 800, respectively. Finally, In representative-based problems, CBB algorithm can find the optimal clique for 49, 63, and 83 percent of proteins in the given time limit when threshold is 600, 700, and 800, respectively. The average gap for all proteins in 60 seconds time limit is reported in the gap column of the Table 13.9.

13.1.4.2. Analysis of top ranked proteins per metric

The results for threshold of 600 are shown in Figures 13.17 and 13.18. Rep-IP, Rep-CBB, Clique-CBB and Clique-IP achieves a final score of detecting 48.61% within the top 500 proteins, as opposed to 46.44% for degree, 46.13% for closeness, 45.82% Star-CBB, 45.51% and Star-IP, and 31.58% for betweenness method. Considering the performance over the least well ranked proteins, we can see that Clique-CBB and Clique-IP are best at not ranking highly non-essential proteins, achieving a final score of 16.41%, while the scores for the other centrality metrics are 19.20% for Rep-CBB, Rep-IP and closeness, 21.05% star-CBB and star-IP, 22.60% degree, and 31.58% betweenness method. It is worth mentioning that the CBB and IP have the same performance in determining essential proteins, although the CBB is much faster than IP.

Table 13.10: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using group closeness centrality and a threshold score of 800.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	12.07	19.50	29.10	36.84	43.65	3.41	8.05	11.15	15.48	22.29
Clique-IP	11.76	20.12	29.10	36.22	43.65	4.02	7.74	11.15	15.48	22.29
Star-CBB	11.15	19.81	28.79	36.84	41.18	4.33	8.05	11.15	16.72	21.98
Star-IP	11.15	19.81	28.79	36.84	41.18	4.02	7.74	11.15	16.72	21.98
Rep-CBB	10.84	21.05	29.10	37.77	43.96	3.10	8.05	11.15	15.79	21.98
Rep-IP	12.07	20.74	28.48	37.46	43.96	4.02	7.74	11.15	15.79	21.98
Degree	11.46	18.58	26.32	36.53	40.56	2.48	7.12	12.07	16.10	21.05
Closeness	3.41	6.50	14.86	24.77	33.75	5.26	10.22	14.24	19.50	24.15
Betweenness	8.98	15.48	22.29	29.72	35.60	3.10	7.74	13.00	18.27	25.39

Table 13.11: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using group closeness centrality and a threshold score of 700.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	8.98	20.74	31.27	38.70	47.37	4.95	6.81	11.46	13.93	16.41
Clique-IP	8.67	21.36	31.58	39.01	47.37	4.95	6.81	11.46	13.93	16.41
Star-CBB	10.22	20.43	30.34	38.70	46.75	4.33	7.12	11.46	14.55	17.03
Star-IP	10.22	20.43	29.72	38.70	46.75	4.64	7.12	11.46	14.86	17.34
Rep-CBB	12.07	19.81	30.96	41.49	47.99	4.95	6.81	11.46	14.55	17.65
Rep-IP	12.69	20.74	31.27	41.80	47.99	4.64	6.81	11.46	14.55	17.65
Degree	14.24	21.98	30.03	39.32	44.58	3.10	8.98	12.07	15.79	21.05
Closeness	3.41	14.55	23.53	33.75	43.96	2.17	6.50	9.29	12.07	16.72
Betweenness	10.22	15.79	25.39	32.51	38.70	5.26	8.98	16.41	19.81	27.24

Table 13.12: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using group closeness centrality and a threshold score of 600.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	11.76	20.12	30.03	39.63	48.61	4.02	7.74	9.29	13.31	16.41
Clique-IP	11.76	21.05	30.03	39.94	48.61	4.02	7.74	9.29	13.00	16.10
Star-CBB	8.67	18.58	28.79	39.01	45.82	4.02	7.43	10.84	14.86	21.05
Star-IP	8.98	19.81	29.41	38.39	45.51	4.02	7.43	10.84	14.86	21.05
Rep-CBB	12.07	20.43	29.10	39.63	48.61	4.02	7.74	9.91	13.93	19.20
Rep-IP	12.38	20.74	28.79	39.32	48.61	4.02	7.74	9.91	13.93	19.20
Degree	14.24	23.53	30.03	37.46	46.44	4.02	9.29	12.07	18.27	22.60
Closeness	7.12	18.58	26.93	35.60	46.13	3.72	6.81	9.91	14.86	19.20
Betweenness	5.57	11.76	17.03	24.15	31.58	4.33	12.69	18.89	26.32	31.58

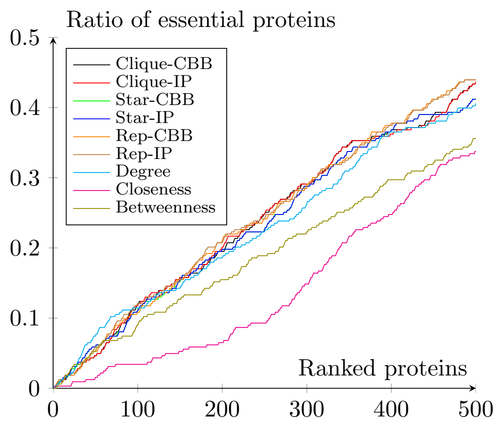


Figure 13.13: The ratio of essential proteins found in the top *k* ranked proteins with a threshold score of 800. The group centrality metric used is based on closeness.

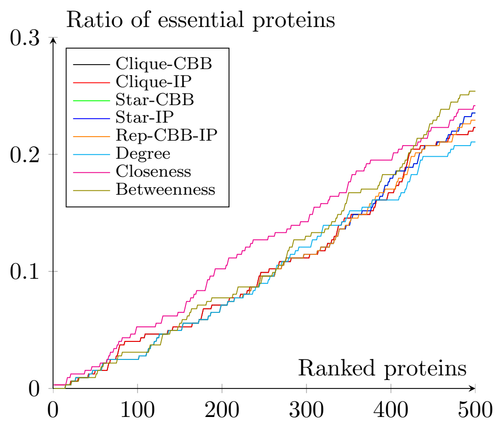


Figure 13.14: The ratio of essential proteins found in the bottom *k* ranked proteins with a threshold score of 800. The group centrality metric used is based on closeness.

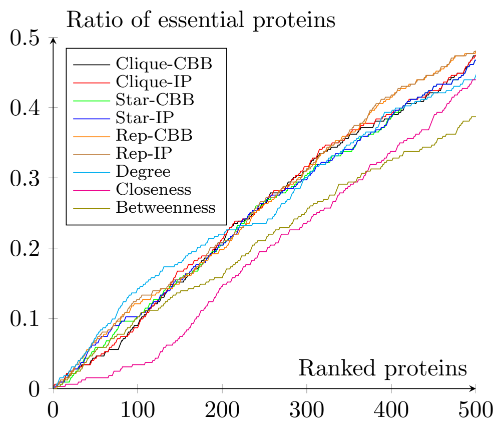


Figure 13.15: The ratio of essential proteins found in the top *k* ranked proteins with a threshold score of 700. The group centrality metric used is based on closeness.

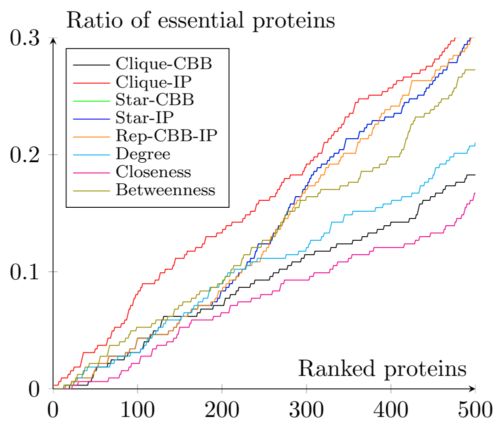


Figure 13.16: The ratio of essential proteins found in the bottom *k* ranked proteins with a threshold score of 700. The group centrality metric used is based on closeness.

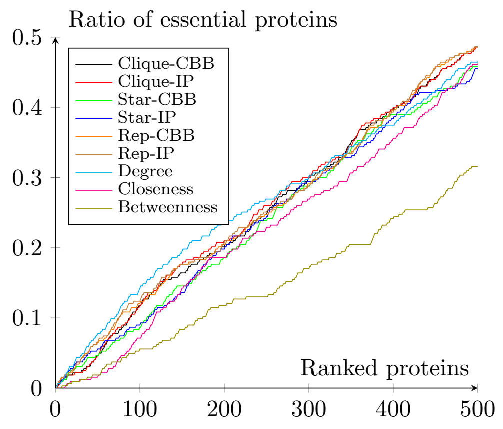


Figure 13.17: The ratio of essential proteins found in the top k ranked proteins with a threshold score of 600. The group centrality metric used is based on closeness.

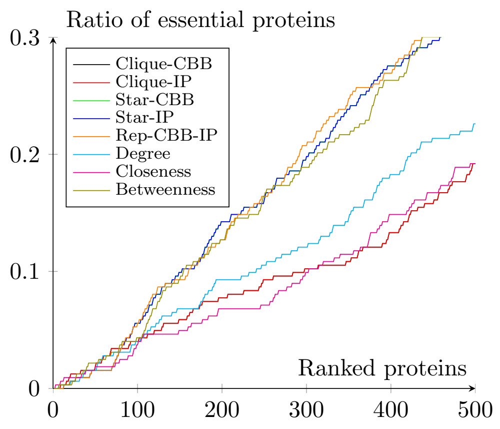


Figure 13.18: The ratio of essential proteins found in the bottom k ranked proteins with a threshold score of 600. The group centrality metric used is based on closeness.

13.1.4.3. Analysis of top ranked proteins per metric using membership measure

The results for threshold of 600 are shown in Figures 13.23 and 13.24. Nodal Degree and Closeness achieve a final score of detecting 46.44% and 46.13%, respectively within the top 500 proteins, as opposed to 45.82% for Rep-CBB, 43.34% for Clique-CBB and Clique-IP, 45.41% REP-IP, 36.53% Star-IP, 35.91% Star-CBB and 31.58% for betweenness method. Considering the performance over the least well ranked proteins, we can see that simple Closeness and Degree are best at not ranking highly non-essential proteins, achieving a final score of 19.20% and 22.60%, respectively, while the scores for the other centrality metrics are 23.53% for Rep-CBB, 26.01% Clique-CBB and Clique-IP, star-IP 28.17%, 28.48% Star-CBB, and 31.58% betweenness method. It is worth mentioning that the CBB and IP have the same performance in determining essential proteins, although the CBB is much faster than IP.

Table 13.13: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using membership measure for group closeness centrality and a threshold score of 800.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	11.76	21.05	27.24	34.98	40.25	2.79	7.43	13.93	19.81	23.22
Clique-IP	11.76	18.89	26.63	32.20	39.01	2.48	6.50	13.31	18.89	24.46
Star-CBB	7.43	16.72	22.60	27.86	34.67	2.79	8.67	14.24	19.81	28.48
Star-IP	8.05	15.79	21.98	28.79	35.91	2.48	6.50	13.93	20.12	25.70
Rep-CBB	11.46	18.27	27.86	33.44	40.56	2.48	7.12	11.76	19.50	23.53
Rep-IP	8.98	17.34	25.08	30.65	37.77	2.48	6.50	13.62	19.81	23.84
Degree	11.46	18.58	26.32	36.53	40.56	2.48	7.12	12.07	16.10	21.05
Closeness	3.41	6.50	14.86	24.77	33.75	5.26	10.22	14.24	19.50	24.15
Betweenness	8.98	15.48	22.29	29.72	35.60	3.10	7.74	13.00	18.27	25.39

Table 13.14: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using membership measure for group closeness centrality and a threshold score of 700.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	10.53	21.36	29.41	38.08	45.20	3.10	9.91	15.17	20.74	26.63
Clique-IP	9.91	21.05	27.55	36.53	42.41	7.74	11.15	16.41	22.91	27.24
Star-CBB	9.29	16.41	23.84	32.51	38.08	3.72	8.36	13.93	20.74	28.17
Star-IP	8.98	16.72	24.77	32.51	38.70	3.10	10.53	16.41	22.29	27.24
Rep-CBB	14.55	21.98	30.03	39.01	46.44	3.10	6.50	11.76	16.72	21.67
Rep-IP	5.57	11.76	17.03	24.15	31.58	4.33	12.69	18.89	26.32	31.58
Degree	14.24	21.98	30.03	39.32	44.58	3.10	8.98	12.07	15.79	21.05
Closeness	3.41	14.55	23.53	33.75	43.96	2.17	6.50	9.29	12.07	16.72
Betweenness	10.22	15.79	25.39	32.51	38.70	5.26	8.98	16.41	19.81	27.24

Table 13.15: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using membership measure for group closeness centrality and a threshold score of 600.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	12.07	21.98	29.72	36.84	43.34	4.95	10.22	15.17	21.36	26.01
Clique-IP	10.84	20.74	30.34	36.53	43.34	5.26	10.22	16.10	20.74	26.01
Star-CBB	11.15	17.03	24.15	31.27	35.91	4.02	8.36	13.62	21.05	28.48
Star-IP	8.98	17.96	26.01	30.65	36.53	4.33	10.84	17.65	22.29	28.17
Rep-CBB	14.24	23.22	30.03	37.77	45.82	3.10	8.36	11.15	18.27	23.53
Rep-IP	11.46	19.81	26.93	35.29	42.41	4.33	9.91	15.79	21.67	27.55
Degree	14.24	23.53	30.03	37.46	46.44	4.02	9.29	12.07	18.27	22.60
Closeness	7.12	18.58	26.93	35.60	46.13	3.72	6.81	9.91	14.86	19.20
Betweenness	5.57	11.76	17.03	24.15	31.58	4.33	12.69	18.89	26.32	31.58

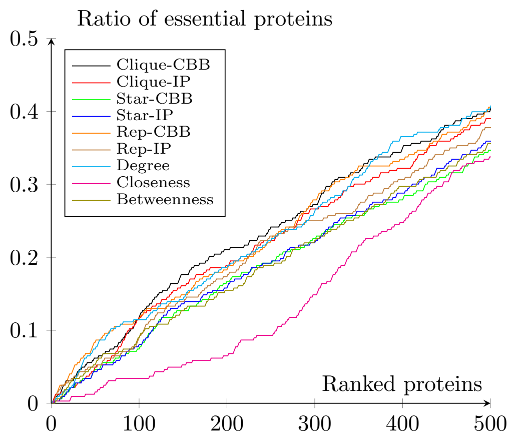


Figure 13.19: The ratio of essential proteins found in the top k ranked proteins using membership measure with a threshold score of 800. The group centrality metric used is based on closeness.

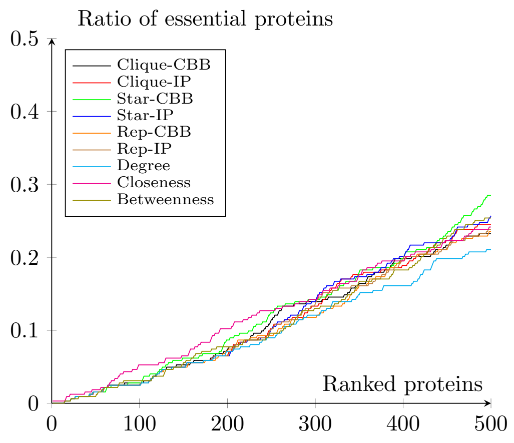


Figure 13.20: The ratio of essential proteins found in the bottom k ranked proteins using membership measure with a threshold score of 800. The group centrality metric used is based on closeness.

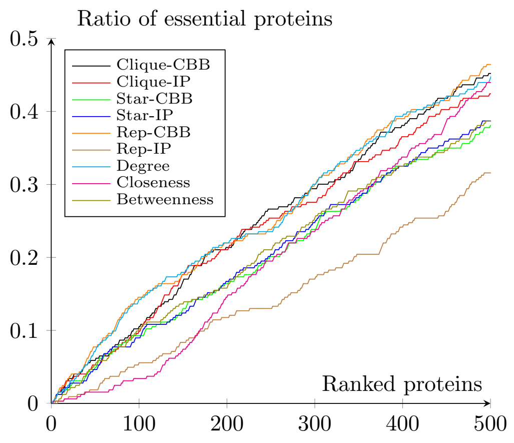


Figure 13.21: The ratio of essential proteins found in the top k ranked proteins using membership measure with a threshold score of 700. The group centrality metric used is based on closeness.

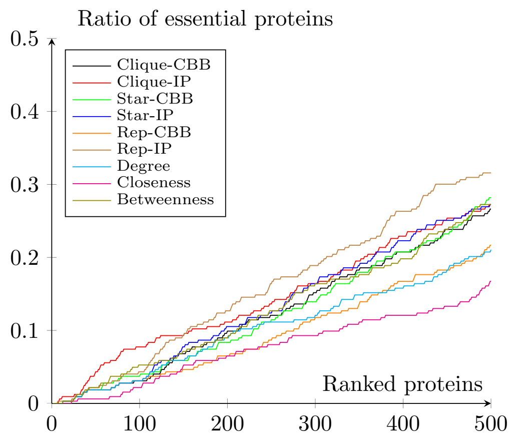


Figure 13.22: The ratio of essential proteins found in the bottom k ranked proteins using membership measure with a threshold score of 700. The group centrality metric used is based on closeness.

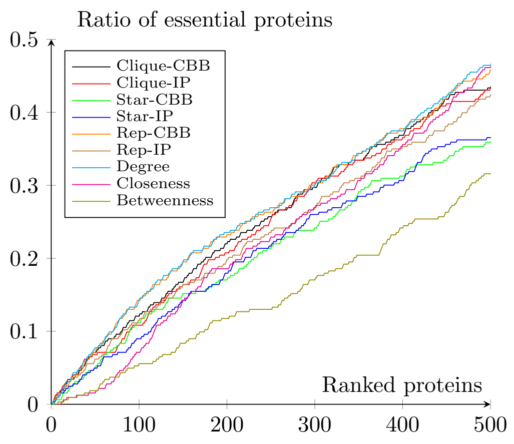


Figure 13.23: The ratio of essential proteins found in the top k ranked proteins using membership measure with a threshold score of 600. The group centrality metric used is based on closeness.

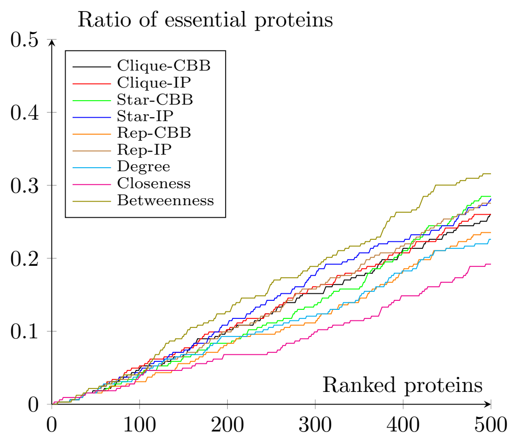


Figure 13.24: The ratio of essential proteins found in the bottom k ranked proteins using membership measure with a threshold score of 600. The group centrality metric used is based on closeness.

13.1.5. Betweenness centrality

13.1.5.1. Comparing the CBB algorithm to the mathematical formulations

Here our analysis of the results is slightly different in the sense that we focus on the CBB. The most striking observation is that Gurobi fails to find a non-trivial feasible solution in all of the problems. On the contrary, our CBB approach solves all of the problems. This is seen in Table 13.16, where we present the number of tree nodes (denoted by B&B nodes) that were fully processed, the percentages of the B&B tree nodes that were fathomed by our bounding condition and fathomed by feasibility, as well as the computational times for solving different versions of the HP graph. The computation time limit imposed here was 60 seconds for each protein. The computational time for calculating the γ_{ij}, γ_{ij}^{k} , γ_{ij}^{kl} and γ_{ij}^{klm} for all $i,j \in V(i \neq j), k,l,m \in V, k \neq l \neq m$, are deducted from the computational time because they are used in both the formulations and the CBB algorithm. Cells containing N/A indicate that the particular instance failed to solve due to memory capacity limits when using Gurobi and the symbol "|" is used to indicate that the time limit was exceeded and no feasible solution was found for that particular instance.

Table 13.16: Results of the experiments for betweenness centrality on the Helicobacter pylori. IP and CBB are our proposed linear IP and algorithm, respectively. Number of B&B tree nodes processed, fathomed by bounds and fathomed by feasibility and running time are presented.

Org	Thr	Str	IP				CBB			
			B&B nodes	Bound Fathoms (%)	Feasibility Fathoms (%)	Time(s)	B&B nodes	Bound Fathoms (%)	Feasibility Fathoms (%)	Time(s)
HP	600	Clique	0	0.00	0.00	N/A	23369	29.53	70.47	80037.20
HP	700	Clique	0	0.00	0.00	N/A	22333	32.06	67.94	73603.57
HP	800	Clique	0	0.00	0.00	|	21580	35.05	64.95	49371.22
HP	600	Star	0	0.00	0.00	N/A	16134	0.49	99.51	92951.78
HP	700	Star	0	0.00	0.00	N/A	16031	0.49	99.51	76953.68
HP	800	Star	0	0.00	0.00	|	17209	0.35	99.65	61195.46
HP	600	Rep	0	0.00	0.00	N/A	14419	2.39	97.61	112312.61
HP	700	Rep	0	0.00	0.00	N/A	13908	1.60	98.40	89497.12
HP	800	Rep	0	0.00	0.00	|	14288	1.27	98.73	82061.02

In order to further evaluate the effectiveness of the proposed CBB algorithm, Table 13.10 lists the percentages of the B&B tree nodes that were fathomed by bounding and fathomed by feasibility condition. Table 13.10 demonstrates that for clique-based problems between 29.53% and 35.05% of the explored nodes are fathomed via the bounding condition, and 64.95% to 70.48% are fathomed by feasibility. While, for star-based problems between 0.35% and 0.49% and 99.51% and 99.65% of the processed nodes are fathomed via the bounding and feasibility condition, respectively and for representative-based problems between 1.27% and 2.39% and 97.61% and 98.73% of the processed nodes are fathomed via the bounding and feasibility condition, respectively. The difference among these percentages for different structures is due to 2 reasons: (1) the upper bound of clique is tighter than those of star and representative because we use graph coloring algorithm to find the maximum cliques in each node of search tree, and (2) Theorem 5 effectively reduces the size of the search tree by fathoming search nodes.

13.1.5.2. Analysis of top ranked proteins per metric

The results for threshold of 600 are shown in Figures 13.29 and 13.30. Degree achieves the highest rate here (detecting 46.44% within the top 500 proteins), as opposed to 46.13% for closeness, with our group extensions falling to third place and below with 41.80% for Rep-CBB, 41.18% for Star-CBB, 40.88% for Clique-CBB. A similar performance, but with closeness outperforming everyone else appears in the bottom *k* proteins, too.

Table 13.17: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using group betweenness centrality and a threshold score of 800.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	12.69	20.12	29.41	36.22	43.03	3.10	8.05	11.46	17.34	22.91
Star-CBB	11.15	19.20	27.86	34.37	39.63	3.10	8.05	11.76	17.34	24.15
Rep-CBB	12.38	20.12	27.55	32.82	40.56	3.10	8.05	11.76	17.03	23.53
Degree	11.46	18.58	26.32	36.53	40.56	2.48	7.12	12.07	16.10	21.05
Closeness	3.41	6.50	14.86	24.77	33.75	5.26	10.22	14.24	19.50	24.15
Betweenness	8.98	15.48	22.29	29.72	35.60	3.10	7.74	13.00	18.27	25.39

Table 13.18: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using group betweenness centrality and a threshold score of 700.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	10.53	20.43	29.10	36.84	43.34	4.95	8.05	11.15	16.72	23.22
Star-CBB	13.00	21.05	28.17	36.22	42.72	4.95	8.05	12.07	17.65	24.15
Rep-CBB	9.91	21.05	29.10	37.46	44.27	4.95	8.05	11.76	16.72	22.91
Degree	14.24	21.98	30.03	39.32	44.58	3.10	8.98	12.07	15.79	21.05
Closeness	3.41	14.55	23.53	33.75	43.96	2.17	6.50	9.29	12.07	16.72
Betweenness	10.22	15.79	25.39	32.51	38.70	5.26	8.98	16.41	19.81	27.24

Table 13.19: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using group betweenness centrality and a threshold score of 600.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	10.84	20.43	26.93	34.67	40.87	4.95	8.05	12.07	17.03	24.15
Star-CBB	9.60	20.12	27.55	35.29	41.18	4.95	8.36	13.00	18.58	25.08
Rep-CBB	10.53	20.12	27.55	33.13	41.80	4.95	8.05	12.07	17.03	23.53
Degree	14.24	23.53	30.03	37.46	46.44	4.02	9.29	12.07	18.27	22.60
Closeness	7.12	18.58	26.93	35.60	46.13	3.72	6.81	9.91	14.86	19.20
Betweenness	5.57	11.76	17.03	24.15	31.58	4.33	12.69	18.89	26.32	31.58

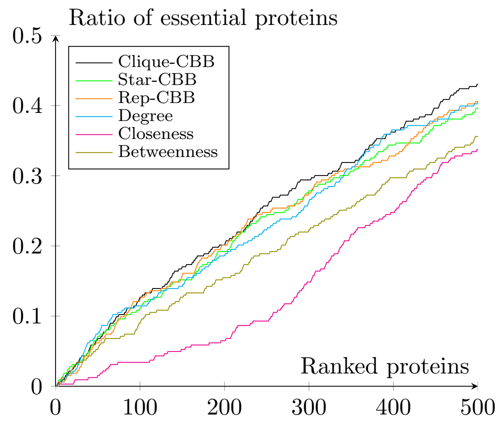


Figure 13.25: The ratio of essential proteins found in the top *k* ranked proteins with a threshold score of 800. The group centrality metric used is based on betweenness.

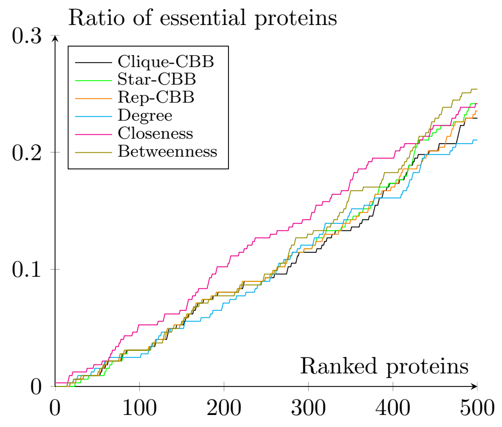


Figure 13.26: The ratio of essential proteins found in the bottom *k* ranked proteins with a threshold score of 800. The group centrality metric used is based on betweenness.

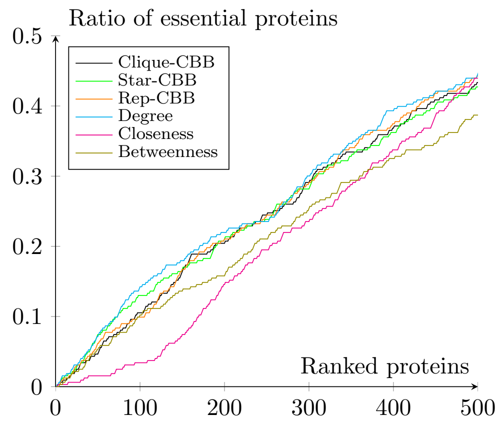


Figure 13.27: The ratio of essential proteins found in the top *k* ranked proteins with a threshold score of 700. The group centrality metric used is based on betweenness.

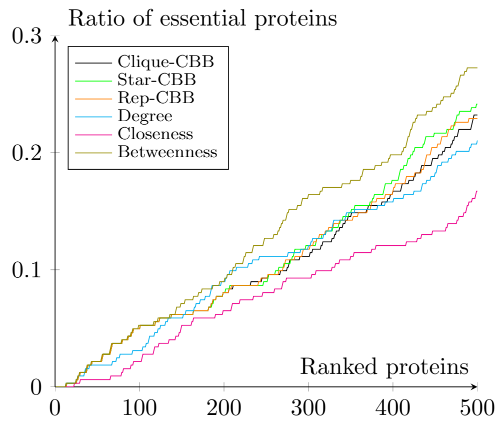


Figure 13.28: The ratio of essential proteins found in the bottom *k* ranked proteins with a threshold score of 700. The group centrality metric used is based on betweenness.

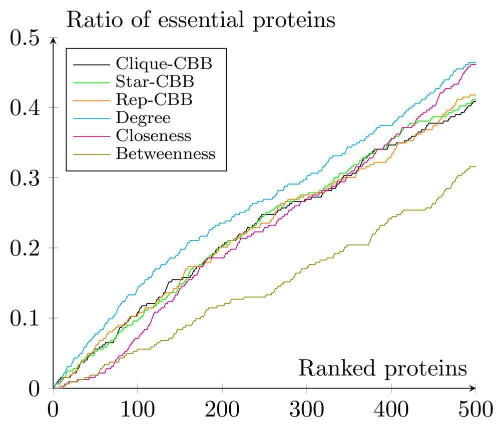


Figure 13.29: The ratio of essential proteins found in the top k ranked proteins with a threshold score of 600. The group centrality metric used is based on betweenness.

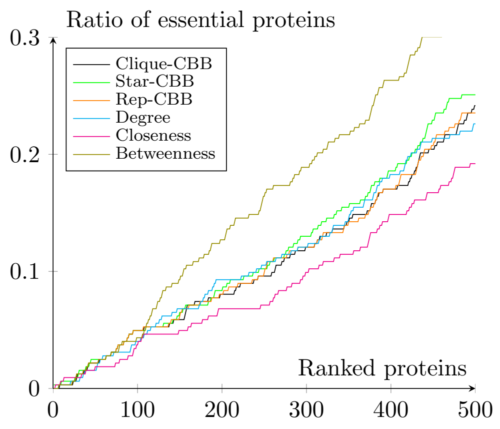


Figure 13.30: The ratio of essential proteins found in the bottom k ranked proteins with a threshold score of 600. The group centrality metric used is based on betweenness.

13.1.5.3. Analysis of top ranked proteins per metric using membership measure

The results for threshold of 600 are shown in Figures 13.35 and 13.36. Degree achieves the highest rate here (detecting 46.44% within the top 500 proteins), as opposed to 46.13% for closeness, with our group extensions falling to third place and below with 37.77% for Clique-CBB, 36.53% for Rep-CBB, 34.67% for Star-CBB. A similar performance, but with closeness outperforming everyone else appears in the bottom k proteins, too.

Table 13.20: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using group betweenness centrality and a threshold score of 800.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	8.98	16.10	24.77	31.89	38.70	2.79	7.43	17.03	21.67	27.55
Star-CBB	8.05	15.79	21.67	29.10	34.06	2.48	8.67	16.41	19.50	24.77
Rep-CBB	8.98	15.17	22.29	30.03	35.91	2.79	8.05	15.79	20.43	25.39
Degree	11.46	18.58	26.32	36.53	40.56	2.48	7.12	12.07	16.10	21.05
Closeness	3.41	6.50	14.86	24.77	33.75	5.26	10.22	14.24	19.50	24.15
Betweenness	8.98	15.48	22.29	29.72	35.60	3.10	7.74	13.00	18.27	25.39

Table 13.21: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using group betweenness centrality and a threshold score of 700.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	8.36	17.96	27.55	36.53	44.27	3.41	10.53	16.10	23.84	27.86
Star-CBB	8.05	17.34	24.15	31.58	37.46	3.41	9.60	15.48	21.98	28.79
Rep-CBB	9.91	18.27	25.08	31.89	39.32	3.41	10.22	14.86	21.05	27.24
Degree	14.24	21.98	30.03	39.32	44.58	3.10	8.98	12.07	15.79	21.05
Closeness	3.41	14.55	23.53	33.75	43.96	2.17	6.50	9.29	12.07	16.72
Betweenness	10.22	15.79	25.39	32.51	38.70	5.26	8.98	16.41	19.81	27.24

Table 13.22: The ratio of essential proteins in top and bottom 100, 200, 300, 400, and 500 proteins using group betweenness centrality and a threshold score of 600.

Approach	**Top**					**Bottom**				
	100	200	300	400	500	100	200	300	400	500
Clique-CBB	9.29	16.41	23.53	30.65	37.77	5.26	11.76	18.89	25.08	30.96
Star-CBB	8.05	15.17	20.43	29.41	34.67	5.88	12.38	18.89	25.70	33.75
Rep-CBB	8.05	16.10	22.91	30.34	36.53	5.57	13.00	19.20	25.08	32.20
Degree	14.24	23.53	30.03	37.46	46.44	4.02	9.29	12.07	18.27	22.60
Closeness	7.12	18.58	26.93	35.60	46.13	3.72	6.81	9.91	14.86	19.20
Betweenness	5.57	11.76	17.03	24.15	31.58	4.33	12.69	18.89	26.32	31.58

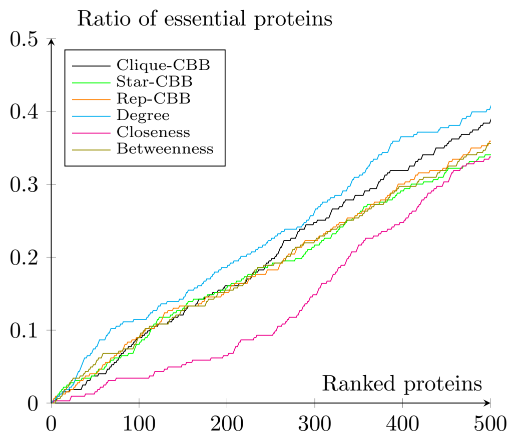


Figure 13.31: The ratio of essential proteins found in the top *k* ranked proteins with a threshold score of 800. The group centrality metric used is based on betweenness.

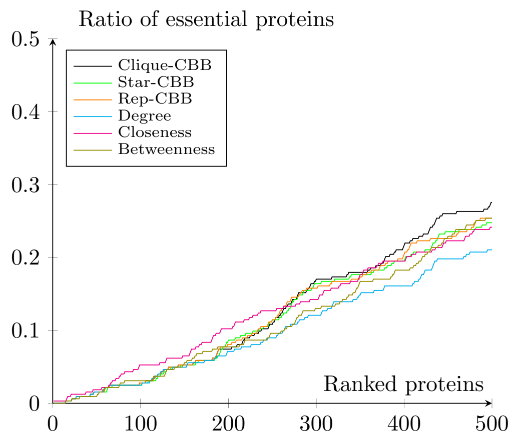


Figure 13.32: The ratio of essential proteins found in the bottom *k* ranked proteins with a threshold score of 800. The group centrality metric used is based on betweenness.

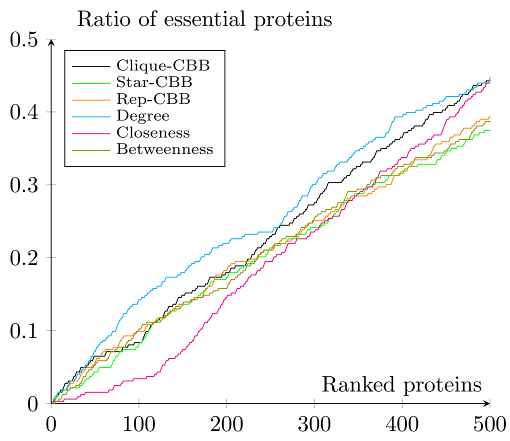


Figure 13.33: The ratio of essential proteins found in the top k ranked proteins with a threshold score of 700. The group centrality metric used is based on betweenness.

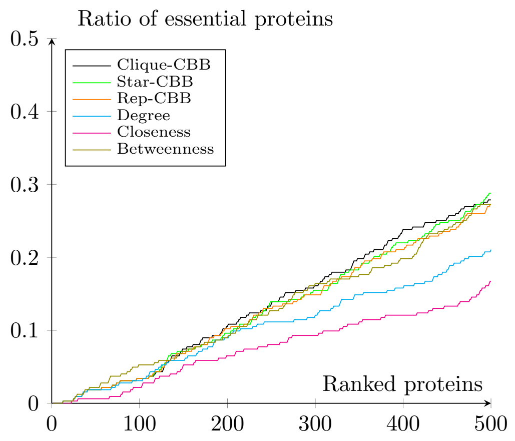


Figure 13.34: The ratio of essential proteins found in the bottom k ranked proteins with a threshold score of 700. The group centrality metric used is based on betweenness.

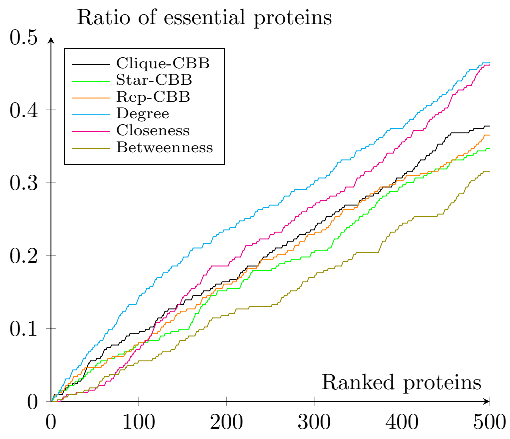


Figure 13.35: The ratio of essential proteins found in the top k ranked proteins with a threshold score of 600. The group centrality metric used is based on betweenness.

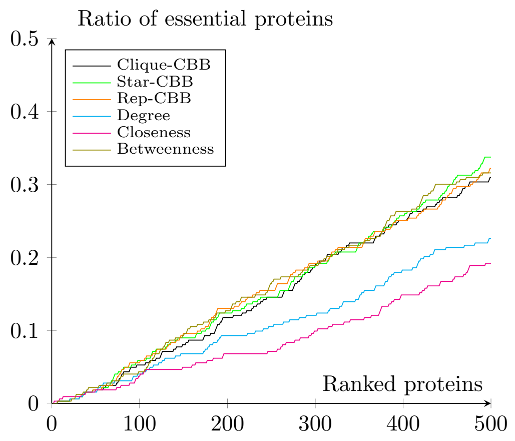


Figure 13.36: The ratio of essential proteins found in the bottom k ranked proteins with a threshold score of 600. The group centrality metric used is based on betweenness.

13.2. Asymmetric probabilistic minimum-cost Hamiltonian cycle problem considering arc and vertex failures

This section presents the computational results of the proposed algorithms and MIP formulations. All computations are performed on a standard PC with a quad-core Intel i7 at 2.8 GHz with 16 GB of RAM. The algorithms are coded in Python and compiled with Xcode 11.5. As a benchmark, we also use Gurobi Optimizer 8.0 to solve the APMCHCP formulated via IP1 – IP4 proposed in Section 11.3. The time limit of computing each instance is 3600 seconds.

The algorithms and formulations are tested on two sets of TSP instances. For each dataset, probability parameters p_i, p_{ij} and $\beta_i (\forall i \in V, \forall (i,j) \in A)$ are generated as follows. The p_i and p_{ij} are generated randomly in intervals $[0.9, 1.0]$ and $[0.1, 1.0]$, respectively. In the same manner with Arigliano et al., 2019, we generate the probability intervals as follows. First, based on a randomly generated Hamiltonian cycle $C = (v_1, v_2, \ldots, v_n, v_1)$, we compute $P^i (\forall i \in C \setminus \{v_1\})$ by (11.1) and P^{v_1} by (11.2), where vertex v_1 is the origin of C. Then, the probability interval for vertex $i (\forall i \in V \setminus v_1)$ is set to $[(1-\alpha)P^i, 1]$, where the probability interval width (PIW) factor $\alpha \in [0,1]$ is used to control the width of the probability intervals.

The first dataset consists of 135 symmetric Euclidean instances described in Dumas et al. (1995), with number of vertices (or equivalently, instance size) between 20 and 200. We apply the same grouping with Dumas et al. (1995). Instances are grouped into 30 classes, each with five instances (class size=5). The instances in each class have equal number of vertices (n) and PIW factor. The name of each class indicates the number of vertices and the PIW factor. For example, Class $n20\alpha0$ indicates each instance in this class has 20 vertices with $\alpha = 0$. In this dataset, the values of α ranges from 0.0 to 0.1 as showed in Table 13.23. The Dumas benchmark set is available at `http://myweb.uiowa.edu/bthoa/TSPTWBenchmarkDataSets.htm`.

The second set is adapted from the real-world asymmetric traveling salesman (ATSP) instances proposed in Ascheuer (1996), which are part of the TSPLIB (Reinelt, 1991) and is available at `http://elib.zib.de/pub/mp-testdata/tsp/`. These instances are coming from a joint project with industry that aims to minimize the unloaded travel time of a stacker crane within an automatic storage system (see Ascheuer (1996) for details). This set of test problems consists of 314 asymmetric TSP instances with up to 231 vertices, and the number of instances in classes are different (see Table 13.24).

Notations of Tables 13.23 and 13.24:

Class	: Instance class
Size	: Number of instances in the class
$\lvert A\rvert$	: Number of arcs in the original input digraph
$\#PI$	: Number of probability interval adjustments
$\#PR$	: Number of precedence relationships among the vertices
$\lvert A\rvert^r$	: Number of eliminated arcs

13.2.1. Data preprocessing

Tables 13.23 and 13.24 show the impact of applying the three data preprocessing steps (proposed in Section 12.1) on the first and the second data sets, respectively. In both tables, if the size of a class is greater than one, then the values included in the columns are the average of that class.

From columns $\#PI$ of Tables 13.23 and 13.24, one can see that 100% of the probability intervals are tightened. In terms of the construction of precedence relationships, given an instance with size n, the maximum possible number of precedence relationship (MPNPR) is $\frac{n(n-1)}{2}$. The average $\#PR/MPNPR$ for all instances of Datasets 1 and 2 are 4.2% and 4.13%, respectively. Moreover, it can be observed that for different classes with the same instance size, $\#PR$ is negatively correlated with the value of α. In other words, on average, the greater β_i's are, the more such precedence relationship can be found. This is mainly because larger values of β_i's strengthen the significance of chance constraints and hence impose more precedence relationships among vertices.

Furthermore, from Table 13.23, after simple calculations, one can get that for instance sizes $n = 20, 40, 60, 80, 100, 150$, and 200, on average, 45.05%, 45.94%, 51.79%, 47.38%, 42.33%, 38.41%, and 32.30% of the original arcs are eliminated, respectively. In Table 13.24, the rates of the eliminated arcs range from 21.52% ($rbg172\alpha0$) to 66.67% ($rbg19\alpha0.075$) of the original arcs. The average rates of eliminated arcs for all instances in the first and second datasets are 42.37% and 44.74%, respectively. In addition, for a fixed instance size, it can be seen that the number of eliminated arcs increases as α decreases. This is due to the fact that larger β_i's impose more

Table 13.23: Preprocessing results for Dataset 1.

Class	Size	$\|A\|$	#PI	#PR	$\|A\|^r$
n20α0	5	380	20	6	176
n20α0.025	5	380	20	13	187
n20α0.050	5	380	20	1	178
n20α0.075	5	380	20	2	168
n20α0.1	5	380	20	0	147
n40α0	5	1560	40	43	754
n40α0.025	5	1560	40	66	777
n40α0.050	5	1560	40	22	723
n40α0.075	5	1560	40	47	725
n40α0.1	5	1560	40	7	605
n60α0	5	3540	60	213	2205
n60α0.025	5	3540	60	148	2016
n60α0.050	5	3540	60	90	1698
n60α0.075	5	3540	60	141	1729
n60α0.1	5	3540	60	11	1519
n80α0	5	6320	80	202	2984
n80α0.025	5	6320	80	137	3260
n80α0.050	5	6320	80	163	2408
n80α0.075	5	6320	80	234	3389
n80α0.1	5	6320	80	6	2929
n100α0	5	9900	100	235	4370
n100α0.025	5	9900	100	279	4799
n100α0.050	5	9900	100	140	3879
n100α0.075	5	9900	100	149	4354
n100α0.1	5	9900	100	9	3551
n150α0	5	22350	150	501	9118
n150α0.025	5	22350	150	382	7722
n150α0.050	5	22350	150	602	8916
n200α0	5	39800	200	424	13149
n200α0.025	5	39800	200	298	12561

Table 13.24: Preprocessing results for Dataset 2.

Class	Size	$\|A\|$	#PI	#PR	$\|A\|^r$
rbg10α0	1	90	10	4	48
rbg15α0	1	210	15	30	137
rbg15α0.025	1	210	15	17	94
rbg16α0	1	240	16	15	118
rbg16α0.025	1	240	16	12	104
rbg17α0	15	272	17	8	103
rbg17α0.025	17	272	17	9	111
rbg17α0.050	17	272	17	10	102
rbg17α0.075	17	272	17	12	103
rbg17α0.1	17	272	17	3	94
rbg17α0.125	17	272	17	2	86
rbg17α0.15	1	272	17	0	95
rbg19α0	3	342	19	15	185
rbg19α0.025	2	342	19	10	200
rbg19α0.050	2	342	19	30	181
rbg19α0.075	2	342	19	24	228
rbg19α0.1	1	342	19	0	135
rbg19α0.125	1	342	19	7	197
rbg19α0.15	1	342	19	9	162
rbg19α0.175	1	342	19	3	182
rbg20α0.125	1	380	20	0	184
rbg27α0	28	702	27	15	323
rbg27α0.025	27	702	27	11	293
rbg27α0.050	27	702	27	17	308
rbg27α0.075	27	702	27	14	291
rbg27α0.1	27	702	27	2	296
rbg27α0.125	27	702	27	2	275
rbg31α0	1	930	31	29	438
rbg33α0	1	1056	33	0	474
rbg34α0	1	1122	34	59	722
rbg35α0	1	1190	35	38	436
rbg35α0.025	1	1190	35	10	360
rbg38α0	1	1406	38	37	770
rbg40α0	1	1560	40	43	599
rbg41α0	1	1640	41	37	546
rbg42α0	1	1722	42	18	876
rbg48α0	1	2256	48	36	974
rbg49α0	1	2352	49	27	1032
rbg50α0	1	2450	50	24	1379
rbg50α0.025	1	2450	50	78	1210
rbg50α0.050	1	2450	50	0	1145
rbg55α0	1	2970	55	9	1217
rbg67α0	1	4422	67	75	1839
rbg86α0	1	7310	86	215	3085
rbg88α0	1	7656	88	220	2528
rbg92α0	1	8372	92	118	3370
rbg125α0	1	15500	125	200	5208
rbg130α0	1	16770	130	107	7957
rbg130α0.025	1	16770	130	81	4277
rbg150α0	1	22350	150	202	12750
rbg150α0.025	1	22350	150	247	6108
rbg150α0.05	1	22350	150	11	6190
rbg172α0	1	29412	172	383	6330
rbg191α0	1	36290	191	313	10265
rbg191α0.025	1	36290	191	269	11643
rbg201α0	1	40200	201	698	13776
rbg231α0	1	53130	231	766	14635
rbg231α0.025	1	53130	231	228	15243

precedence relationships among vertices and thus enhance the impact of the three arc-elimination steps described in Section 12.1.3.

13.2.2. Comparison of the CBB algorithm and MIP formulations

Computational results of solving both datasets using MIP formulations and the proposed CBB algorithm are summarized in Tables 13.25 and 13.26. For solving the upper bound, the scaling parameter a defined in (12.18a) is set to be $a := \frac{1}{10 \max\limits_{(i,j)\in A} \{c_{ij}\}}$.

Notations of Tables 13.25 and 13.26:

OPT	: Number of instances for which the optimal solution is found within the time limit
CPU (s)	: Mean time in seconds
FS (%)	: Number of instances for which the optimal solution is not found but a feasible solution is obtained within the time limit. Inside the parenthesis is the mean gap for these instances in percent if the optimal solution is available.
IP1	: The first MIP formulation solved by Gurobi
IP2	: The second MIP formulation solved by Gurobi
IP3	: The third MIP formulation solved by Gurobi
IP4	: The forth MIP formulation solved by Gurobi
CBB	: Combinatorial B&B algorithm
|	: Indicating that the time limit is exceeded for all class instances
NA	: Indicating that all instances in the class are failed to be solved by Gurobi due to memory capacity limits

According to Table 13.25, for the first dataset, the CBB solves 143 (95.33%) instances with up to 200 vertices, while IP1, IP2, IP3 and IP4 solve 0 (0.00%), 1 (0.67%), and 19 (12.67%) instances with up to 20 vertices, respectively. Regarding the 19 instances for which both CBB and IP4 find the optimal solutions, the mean computing time of CBB is significantly less than that of IP4 (4.11 versus 1076.70 seconds). Also, regarding the 4 instances for which both CBB and IP2 find the optimal solutions, the mean computing time of CBB and IP2 are 1.24 and 918.98 seconds. Finally, for the only instance for which both CBB and IP3 obtain the optimal solution, the computing time of CBB and IP3 are 0.24 and 2145.9 seconds. In addition, as expected, the CPU time increases with the increase of probability interval width and problem size. We remark that although the CBB algorithm cannot solve 7 problems to optimality in the one-hour time limit, it can find feasible solutions for 6 of them, as we can see from the FS(%) column of CBB in Table 13.25. The only problem that CBB can not provide a feasible solution is in the class $n200\alpha0.025$.

Table 13.25: Results of the experiments for Dataset 1.

Class	**Size**	**IP1**			**IP2**			**IP3**			**IP4**			**CBB**		
		OPT	CPU(s)	FS(%)	OPT	CPU(s)	FS(%)	OPT	CPU(s)	FS(%)	OPT	CPU(s)	FS(%)	OPT	CPU(s)	FS(%)
n20α0	5	0	|	0	1	1699.50	0	0	|	0	5	1363.42	0	5	1.67	0
n20α0.025	5	0	|	0	2	692.42	0	1	2145.92	0	4	879.65	0	5	0.51	0
n20α0.050	5	0	|	0	1	591.57	0	0	|	0	4	729.21	0	5	1.53	0
n20α0.075	5	0	|	0	0	|	0	0	|	0	4	1137.67	0	5	1.34	0
n20α0.1	5	0	|	0	0	|	0	0	|	0	2	1327.05	0	5	15.48	0
n40α0	5	0	|	0	0	|	0	0	|	0	0	|	0	5	14.84	0
n40α0.025	5	0	|	0	0	|	0	0	|	0	0	|	0	5	21.51	0
n40α0.050	5	0	|	0	0	|	0	0	|	0	0	|	0	5	13.79	0
n40α0.075	5	0	|	0	0	|	0	0	|	0	0	|	0	5	15.68	0
n40α0.1	5	0	|	0	0	|	0	0	|	0	0	|	0	5	336.71	0
n60α0	5	0	|	0	0	|	0	0	|	0	0	|	0	5	23.62	0
n60α0.025	5	0	|	0	0	|	0	0	|	0	0	|	0	5	30.32	0
n60α0.050	5	0	|	0	0	|	0	0	|	0	0	|	0	5	18.24	0
n60α0.075	5	0	|	0	0	|	0	0	|	0	0	|	0	5	54.94	0
n60α0.1	5	0	|	0	0	|	0	0	|	0	0	|	0	5	504.86	0
n80α0	5	0	|	0	0	|	0	0	|	0	0	|	0	5	31.96	0
n80α0.025	5	0	|	0	0	|	0	0	|	0	0	|	0	5	162.87	0
n80α0.050	5	0	|	0	0	|	0	0	|	0	0	|	0	5	52.42	0
n80α0.075	5	0	|	0	0	|	0	0	|	0	0	|	0	5	62.25	0
n80α0.1	5	0	|	0	0	|	0	0	|	0	0	|	0	5	1288.64	0
n100α0	5	0	|	0	0	|	0	0	|	0	0	|	0	5	118.19	0
n100α0.025	5	0	|	0	0	|	0	0	|	0	0	|	0	5	250.06	0
n100α0.050	5	0	|	0	0	|	0	0	|	0	0	|	0	5	499.37	0
n100α0.075	5	0	|	0	0	|	0	0	|	0	0	|	0	5	669.81	0
n100α0.1	5	0	|	0	0	|	0	0	|	0	0	|	0	3	638.70	2
n150α0	5	0	NA	0	0	NA	0	0	NA	0	0	|	0	5	697.24	0
n150α0.025	5	0	NA	0	0	NA	0	0	NA	0	0	|	0	5	932.41	0
n150α0.050	5	0	NA	0	0	NA	0	0	NA	0	0	|	0	3	579.21	2
n200α0	5	0	NA	0	0	NA	0	0	NA	0	0	|	0	4	573.85	1
n200α0.025	5	0	NA	0	0	NA	0	0	NA	0	0	|	0	3	840.28	1

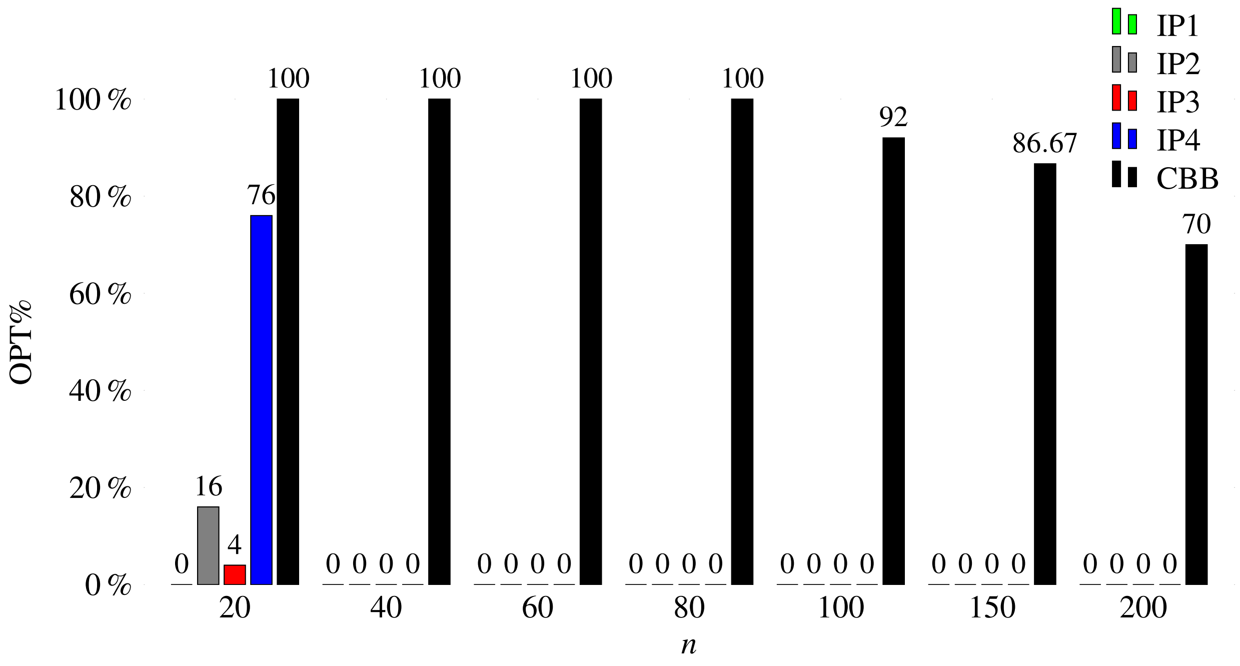


Figure 13.37: Rate of instances solved to optimality by number of customers for Dataset 1.

A graphical comparison among the five computational methods/formulations for Dataset 1 is given in Figure 13.37, where the vertical axis represents the percentage of instances for which optimal solutions are found. Among the MIP formulations, IP4 outperforms others. This is probably because IP4 uses the recursive formulation to represent the arrival probability to all vertices. Although IP3 uses the same idea to find the arrival probabilities, IP4 has much less variables and constraints. However, the biggest problem size that can be solved by IP1 – IP4 is only 20, while by CBB it is 200.

Analogous information is shown in Table 13.26 for each class of asymmetric instances (i.e., Dataset 2). The CBB algorithm solves 311 (99.04%) instances whereas IP1, IP2, IP3, and IP4 solve 13 (74.14%), 50 (15.92%), 49 (15.61%), and 115 (36.62%) instances, respectively. For the 13 instances solved by both CBB and IP1, their mean CPU time are 0.83 and 1123.75 seconds. The CPU time of CBB and IP2 for the 50 instances solved by both methods are 1371.77 and 1.11 seconds. For the 49 problems solved by both CBB and IP3, the mean CPU time are 1.19 and 1431.13 seconds. Finally, regarding the 115 instances solved by both CBB and IP4, their mean CPU time are 3.21 and 907.93 seconds. All these comparisons indicate that the proposed CBB

Table 13.26: Results of the experiments for Dataset 2.

Class	Size	IP1			IP2			IP3			IP4			CBB		
		OPT	CPU(s)	FS(%)	OPT	CPU(s)	FS(%)	OPT	CPU(s)	FS(%)	OPT	CPU(s)	FS(%)	OPT	CPU(s)	FS(%)
rbg10α0	1	1	64.15	0	1	7.76	0	1	1.81	0	1	0.15	0	1	0.28	0
rbg15α0	1	1	182.76	0	1	18.65	0	1	2.61	0	1	0.15	0	1	0.43	0
rbg15α0.025	1	1	1601.94	0	1	377.51	0	1	99.00	0	1	6.02	0	1	0.52	0
rbg16α0	1	1	2238.71	0	1	499.43	0	1	154.78	0	1	8.64	0	1	0.22	0
rbg16α0.025	1	1	2279.79	0	1	3363.81	0	1	99.26	0	1	23.08	0	1	0.24	0
rbg17α0	15	2	2067.52	0	6	2018.09	0	6	1329.81	3(22.38)	14	885.79	0	15	1.01	
rbg17α0.025	17	0		0	9	1676.46	0	8	1496.27	1(76.62)	16	642.37	1(0.72)	17	1.79	0
rbg17α0.050	17	0		0	8	1362.13	0	9	1988.82	2(18.42)	17	750.48	0	17	1.15	0
rbg17α0.075	17	0		0	9	1399.88	0	8	1833.33	1(21.90)	17	1183.90	0	17	1.33	0
rbg17α0.1	17	0		0	4	1496.87	1(6.09)	3	1889.71	2(58.48)	16	974.88	1(17.21)	17	1.81	0
rbg17α0.125	17	0		0	2	2952.30	0	2	2963.77	3(28.79)	15	1567.43	2(14.78)	17	2.33	0
rbg17α0.15	1	0		0	0		0	0		0	0		1(13.51)	1	0.91	0
rbg19α0	3	1	164.78	0	2	415.99	0	2	247.14	0	3	444.33	0	3	2.54	0
rbg19α0.025	2	1	316.32	0	1	176.10	0	1	158.23	0	2	202.90	0	2	1.89	0
rbg19α0.050	2	2	464.55	0	2	50.95	0	2	173.43	0	2	12.71	0	2	0.43	0
rbg19α0.075	2	2	1348.10	0	2	313.36	0	2	1440.99	0	2	31.07	0	2	0.73	0
rbg19α0.1	1	0		0	0		0	0		0	0		0	1	4.58	0
rbg19α0.125	1	0		0	0		0	1	1774.78	0	1	215.88	0	1	2.40	0
rbg19α0.15	1	0		0	0		0	0		0	1	1520.17	0	1	1.67	0
rbg19α0.175	1	0		0	0		0	0		0	1	938.88	0	1	3.31	0
rbg20α0.125	1	0		0	0		0	0		0	1	3600.11	0	1	13.03	0
rbg27α0	28	0		0	0		0	0		0	2	799.83	1(11.73)	28	27.08	0
rbg27α0.025	27	0		0	0		0	0		0	0		0	27	59.86	0
rbg27α0.050	27	0		0	0		0	0		0	0		0	27	97.68	0
rbg27α0.075	27	0		0	0		0	0		0	0		0	27	57.41	0
rbg27α0.1	27	0		0	0		0	0		0	0		0	27	141.93	0
rbg27α0.125	27	0		0	0		0	0		0	0		0	27	128.82	0
rbg31α0	1	0		0	0		0	0		0	0		0	1	13.18	0
rbg33α0	1	0		0	0		0	0		0	0		0	1	28.53	0
rbg34α0	1	0		0	0		0	0		0	0		0	1	1.70	0
rbg35α0	1	0		0	0		0	0		0	0		0	1	6.33	0
rbg35α0.025	1	0		0	0		0	0		0	0		0	1	24.72	0
rbg38α0	1	0		0	0		0	0		0	0		0	1	19.50	0
rbg40α0	1	0		0	0		0	0		0	0		0	1	7.39	0
rbg41α0	1	0		0	0		0	0		0	0		0	1	13.53	0
rbg42α0	1	0		0	0		0	0		0	0		0	1	14.22	0
rbg48α0	1	0		0	0		0	0		0	0		0	1	15.03	0
rbg49α0	1	0		0	0		0	0		0	0		0	1	139.14	0
rbg50α0	1	0		0	0		0	0		0	0		0	1	38.96	0
rbg50α0.025	1	0		0	0		0	0		0	0		0	1	12.60	0
rbg50α0.050	1	0		0	0		0	0		0	0		0	1	98.00	0
rbg55α0	1	0		0	0		0	0		0	0		0	1	48.89	0
rbg67α0	1	0		0	0		0	0		0	0		0	1	66.37	0
rbg86α0	1	0		0	0		0	0		0	0		0	1	71.50	0
rbg88α0	1	0		0	0		0	0		0	0		0	1	30.26	0
rbg92α0	1	0		0	0		0	0		0	0		0	1	69.77	0
rbg125α0	1	0	NA	0	0	NA	0	0	NA	0	0		0	1	514.21	0
rbg130α0	1	0	NA	0	0	NA	0	0	NA	0	0		0	1	764.58	0
rbg130α0.025	1	0	NA	0	0	NA	0	0	NA	0	0		0	1	1224.98	0
rbg150α0	1	0	NA	0	0	NA	0	0	NA	0	0		0	1	798.58	0
rbg150α0.025	1	0	NA	0	0	NA	0	0	NA	0	0		0	1	2975.28	0
rbg150α0.05	1	0	NA	0	0	NA	0	0	NA	0	0		0	0		1
rbg172α0	1	0	NA	0	0	NA	0	0	NA	0	0		0	1	2812.92	0
rbg191α0	1	0	NA	0	0	NA	0	0	NA	0	0		0	1	1726.60	0
rbg191α0.025	1	0	NA	0	0	NA	0	0	NA	0	0		0	0		1
rbg201α0	1	0	NA	0	0	NA	0	0	NA	0	0		0	1	3478.10	0
rbg231α0	1	0	NA	0	0	NA	0	0	NA	0	0		0	1	2551.14	0
rbg231α0.025	1	0	NA	0	0	NA	0	0	NA	0	0		0	0		0

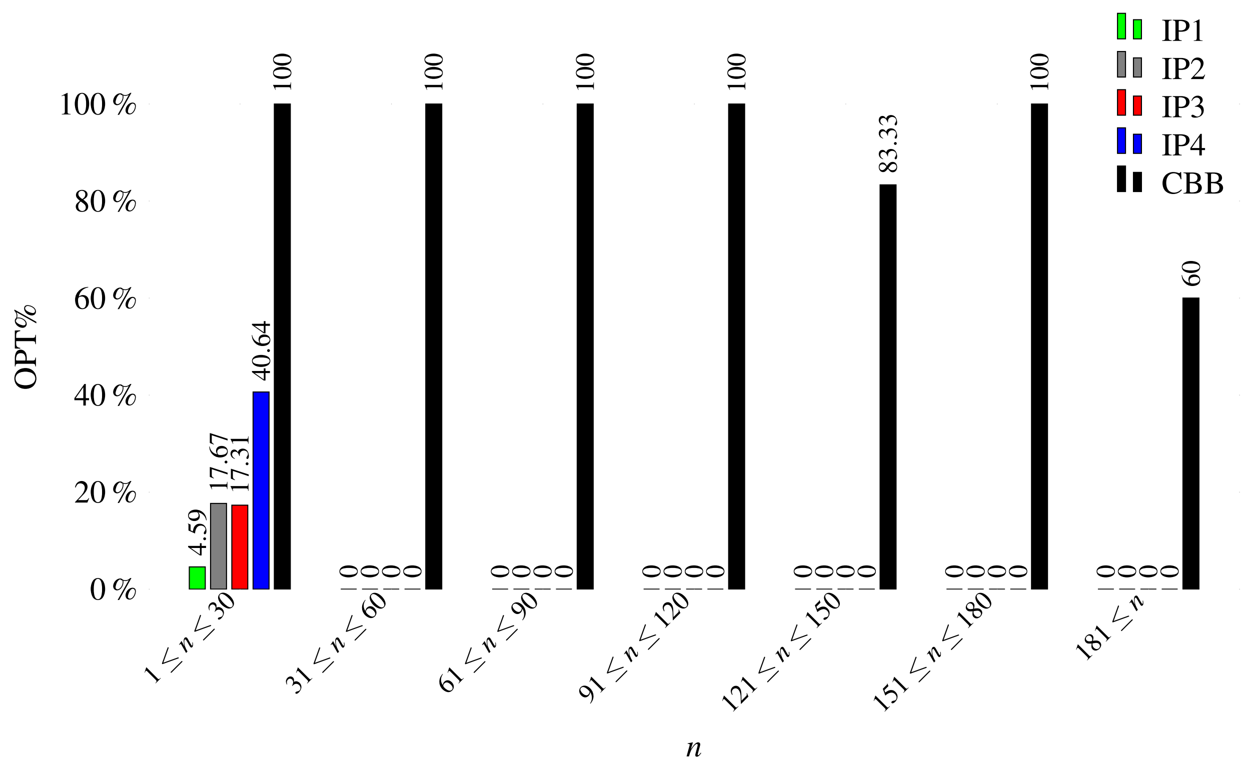


Figure 13.38: Rate of instances solved to optimality by number of customers for Dataset 2.

outperforms the Gurobi solver (for solving IP1 – IP4) not only in terms of the number of problems solved to optimality, but also it is considerably faster than all of them. Analogous to Dataset 1, the CPU time increases with the increase of probability interval width and problem size. According to FS(%) column, among the 3 problems that CBB algorithm cannot solve to optimality, it can find feasible solutions for 2 of them in the one-hour time limit, and the only instance without feasible solution is in the class $rbg231\alpha0.025$.

Figure 13.38 shows the rate of instances solved to optimality by all the methods with respect to the seven problem sizes. The CBB algorithm outperforms IP1 – IP4 in all the problem sizes. As we expect, IP4 and IP1 have the best and the worst performances among the formulations by solving 40.64% and 4.59% of the instances of the first group, respectively. IP2 and IP3 have similar performances by solving 17.67% and 17.31% of the instances, respectively. The CBB algorithm is the only method being able to solve instances with more than 30 vertices. The largest instance solved by the CBB algorithm includes 231 vertices.

Notations of Tables 13.27 and 13.28:

UB	: Number of instances of which an upper bound is found
Gap(%)	: Mean gap for instances with upper bound, gap for an instance in percent is calculated by $\frac{z^*_{ub}-z^*}{z^*}.100\%$
B&B nodes	: Number of generated nodes in the CBB tree
Bound Fathoms(%)	: The percentages of the B&B tree nodes that are fathomed by the bounding conditions
Feasibility Fathoms(%)	: The percentages of the B&B tree nodes that are fathomed by the feasibility conditions

Table 13.27: More information about running the CBB algorithm for Dataset 1.

Class	**Size**	**CBB**				
		UB	Gap(%)	B&B nodes	Feasibility Fathoms(%)	Bound Fathoms(%)
n20α0	5	0		4061	98.77	1.23
n20α0.025	5	0		530	99.50	0.50
n20α0.050	5	1	0	2778	98.40	1.60
n20α0.075	5	0		2168	98.28	1.72
n20α0.1	5	4	7.40	53077	97.82	2.18
n40α0	5	0		23266	99.71	0.29
n40α0.025	5	0		33905	99.98	0.02
n40α0.050	5	0		13951	99.93	0.07
n40α0.075	5	0		18549	99.93	0.07
n40α0.1	5	0		693564	99.44	0.56
n60α0	5	0		13409	99.97	0.03
n60α0.025	5	2	0.00	8688	99.85	0.15
n60α0.050	5	1	0.00	1555	99.03	0.97
n60α0.075	5	0		34044	99.96	0.04
n60α0.1	5	0		382123	99.99	0.01
n80α0	5	0		2187	99.47	0.53
n80α0.025	5	0		66087	99.97	0.03
n80α0.050	5	1	0.00	5619	99.92	0.08
n80α0.075	5	3	0.00	11078	99.82	0.18
n80α0.1	5	1	2.34	507165	99.97	0.03
n100α0	5	0		16151	99.71	0.29
n100α0.025	5	3	0.00	30300	99.88	0.12
n100α0.050	5	1	0.00	136157	99.98	0.02
n100α0.075	5	2	0.00	134105	99.99	0.01
n100α0.1	5	1	2.07	463902	100.00	0.00
n150α0	5	0		17890	99.97	0.03
n150α0.025	5	1	0.00	83953	99.99	0.01
n150α0.050	5	2	0.00	179967	99.98	0.02
n200α0	5	0		70372	99.98	0.02
n200α0.025	5	1	0.00	115741	99.99	0.01

Table 13.28: More information about running the CBB algorithm for Dataset 2.

Class	**Size**	**CBB**				
		UB	Gap(%)	B&B nodes	Feasibility Fathoms(%)	Bound Fathoms(%)
rbg10α0	1	0		31	100.00	0.00
rbg15α0	1	0		23	100.00	0.00
rbg15α0.025	1	0		44	93.33	6.67
rbg16α0	1	0		109	97.83	2.17
rbg16α0.025	1	0		246	97.01	2.99
rbg17α0	15	3	23.07	852	71.58	28.42
rbg17α0.025	17	1	88.31	2710	77.14	22.86
rbg17α0.050	17	1	90.54	1396	78.48	21.52
rbg17α0.075	17	3	63.71	1236	70.00	30.00
rbg17α0.1	17	3	126.97	3067	84.36	15.64
rbg17α0.125	17	0		3440	66.73	33.27
rbg17α0.15	1	0		1438	98.73	1.27
rbg19α0	3	2	0.00	747	90.92	9.08
rbg19α0.025	2	1	0.00	1252	99.38	0.62
rbg19α0.050	2	0		112	98.20	1.80
rbg19α0.075	2	1	0.00	121	95.83	4.17
rbg19α0.1	1	0		3576	99.60	0.40
rbg19α0.125	1	1	1.25	808	95.96	4.04
rbg19α0.15	1	0		890	95.24	4.76
rbg19α0.175	1	1	0.00	1599	96.52	3.48
rbg20α0.125	1	1	7.65	19730	98.63	1.37
rbg27α0	28	0		85463	96.63	3.37
rbg27α0.025	27	0		181063	89.41	10.59
rbg27α0.050	27	2	25.75	310543	80.69	19.31
rbg27α0.075	27	3	20.39	177167	95.72	4.28
rbg27α0.1	27	5	15.72	436447	93.03	6.97
rbg27α0.125	27	5	10.22	392026	96.68	3.32
rbg31α0	1	0		22906	99.86	0.14
rbg33α0	1	0		38858	99.99	0.01
rbg34α0	1	0		1030	99.61	0.39
rbg35α0	1	0		1671	99.56	0.44
rbg35α0.025	1	0		19507	100.00	0.00
rbg38α0	1	0		12596	99.90	0.10
rbg40α0	1	0		3749	99.91	0.09
rbg41α0	1	0		11117	99.90	0.10
rbg42α0	1	0		2457	99.84	0.16
rbg48α0	1	0		2088	99.23	0.77
rbg49α0	1	0		139061	100.00	0.00
rbg50α0	1	0		26597	99.95	0.05
rbg50α0.025	1	0		2182	98.89	1.11
rbg50α0.050	1	0		56691	99.97	0.03
rbg55α0	1	0		30833	99.99	0.01
rbg67α0	1	0		35400	99.96	0.04
rbg86α0	1	0		8017	99.73	0.27
rbg88α0	1	0		384	100.00	0.00
rbg92α0	1	0		7828	99.64	0.36
rbg125α0	1	0		47540	99.89	0.11
rbg130α0	1	0		79993	99.98	0.02
rbg130α0.025	1	0		138526	99.98	0.02
rbg150α0	1	0		45192	99.91	0.09
rbg150α0.025	1	0		342170	100.00	0.00
rbg150α0.05	1	0		291646	100.00	0.00
rbg172α0	1	0		130772	99.98	0.02
rbg191α0	1	0		18163	99.60	0.40
rbg191α0.025	1	0		246753	99.97	0.03
rbg201α0	1	0		182085	99.98	0.02
rbg231α0	1	0		10873	99.63	0.37
rbg231α0.025	1	0		189595	100.00	0.00

13.2.3. Further Information on the CBB algorithm

To further check the efficiency of the proposed CBB algorithm, Tables 13.27 and 13.28 provide more detailed information such as upper bound, upper bound gap, number of generated nodes in the search tree, number of nodes fathomed by bounding, and number of nodes fathomed by feasibility. Table 13.27 demonstrates that 97.82% – 100% of the fathomed nodes in Dataset 1 are fathomed via the feasibility conditions, whereas 0% to 2.18% are fathomed by bounding. For Dataset 2, Table 13.28 shows that 66.73% – 100% of the fathomed nodes are fathomed via the feasibility conditions, while 0% – 33.27% are fathomed by bounding. Overall, the results in Tables 13.27 and 13.28 suggest that the proposed feasibility scheme effectively reduces the size of the search tree.

Tables 13.27 and 13.28 also provide some information about the upper bound which are reported in columns UB and Gap(%). From Table 13.27, the upper bounds of 24 (16.00%) instances from the first dataset can be found by Algorithm 7. Column Gap(%) depicts the quality of the upper bound (by calculating its gap from the corresponding optimal solution, if available) and average the gaps in each class. Among the 24 instances with upper bounds, 18 of them have zero gap, and the average gap of the other 6 instances is 7.15%. According to Table 13.28, we have upper bounds for 33 (10.51%) instances in the asymmetric dataset (i.e., Dataset 2). Six of them have zero gap and the average gap of the remaining instances is 39.68%.

It should be mentioned that one of the main advantages of the proposed CBB algorithm is its low memory requirement, as it only stores one active tree node each time during its execution. This enables the CBB algorithm to avoid memory crash issues which Gurobi encounters. The low memory requirement, however, comes at the cost of not being able to report an optimality gap in the CBB algorithm, as we do not store (and calculate) the lower bound for all active tree nodes.

13.2.4. Impact analysis of the feasibility rules

In this section, we seek to analyze which of the following six configurations has the worst performance

- CBB: CBB is executed with all rules.
- CBB-R1: CBB is executed without feasibility check (Rule 1).
- CBB-R2: CBB is executed without precedence relationship check (Rule 2).
- CBB-R3: CBB is executed without extension check (Rule 3).
- CBB-R4: CBB is executed without PHC check (Rule 4).
- CBB-R5: CBB is executed without returning-to-the-origin check (Rule 5).

The results of Dataset 1 is reported in Table 13.29 and Figures 13.39 and 13.40. From Figure 13.39, one can see that eliminating each of Rule 3 and Rule 5 reduces the number of optimally solved instances, when instance size n is greater than or equal to 100. However, when n is less than or equal to 80, all configurations have the same number of optimally solved instances. This is because the CPU time of the CBB algorithm for $n \leq 80$ is much less than the one-hour time limit, and hence all the configurations can find the optimal solution by spending extra time (see Figure 13.40 for more details). Considering all the 150 instances in this dataset, CBB, CBB-R1, CBB-R2, and CBB-R4 solve 143 (95.33%) instances, whereas CBB-R3 and CBB-R5 solve 137(91.33%) and 140 (93.33%) instances, respectively. Moreover, as one can see in Figure 13.40, the CPU time increases if any of the feasibility rules are eliminated. On average, eliminating Rules 1, 4, 2, 5, and 3 increases the CPU time by 10.69 (4.07%), 13.94 (5.31%), 130.45 (49.65%), 597.47 (227.40%), and 745.93 (283.91%) seconds, respectively. Based on both the number of optimally solved instances and the CPU time, one can conclude that CBB-R3 has the worst performance for Dataset 1.

Analogous information for Dataset 2 is provided in Table 13.30 and Figures 13.41 and 13.42. Table 13.30 indicates the number of instances in each class to which optimal solutions are found. Their corresponding average CPU time are also included. Figure 13.41 and 13.42 give

Table 13.29: Performance of different method configurations for Dataset 1.

Class	**Size**	**CBB**		**CBB-R1**		**CBB-R2**		**CBB-R3**		**CBB-R4**		**CBB-R5**	
		OPT	CPU(s)	OPT	CPU(s)	OPT	CPU(s)	OPT	CPU(s)	OPT	CPU(s)	OPT	CPU(s)
n20α0	5	5	1.67	5	1.61	5	2.13	5	5.21	5	1.19	5	3.28
n20α0.025	5	5	0.51	5	0.52	5	0.89	5	1.05	5	0.27	5	0.69
n20α0.050	5	5	1.53	5	1.47	5	1.59	5	4.28	5	0.98	5	2.95
n20α0.075	5	5	1.34	5	1.55	5	1.60	5	3.55	5	0.72	5	2.23
n20α0.1	5	5	15.48	5	14.54	5	19.01	5	92.98	5	16.11	5	125.53
n40α0	5	5	14.84	5	15.76	5	20.23	5	148.26	5	11.95	5	53.86
n40α0.025	5	5	21.51	5	22.19	5	36.55	5	252.84	5	28.70	5	70.52
n40α0.050	5	5	13.79	5	14.41	5	28.09	5	137.57	5	14.63	5	51.78
n40α0.075	5	5	15.68	5	17.32	5	37.77	5	229.87	5	20.88	5	55.94
n40α0.1	5	5	336.71	5	343.71	5	621.02	5	1445.49	5	550.75	5	1071.24
n60α0	5	5	23.62	5	24.72	5	41.90	5	162.87	5	16.96	5	66.80
n60α0.025	5	5	30.32	5	36.74	5	71.99	5	245.98	5	27.77	5	1164.92
n60α0.050	5	5	18.24	5	19.43	5	58.05	5	59.71	5	6.83	5	37.55
n60α0.075	5	5	54.94	5	55.41	5	122.02	5	696.74	5	66.26	5	406.87
n60α0.1	5	5	504.86	5	500.13	5	904.22	5	2748.86	5	776.20	5	1876.91
n80α0	5	5	31.96	5	29.84	5	101.86	5	189.97	5	14.64	5	157.40
n80α0.025	5	5	162.88	5	163.59	5	332.47	5	1679.21	5	205.89	5	1081.10
n80α0.050	5	5	52.42	5	55.40	5	114.76	5	488.98	5	37.67	5	250.69
n80α0.075	5	5	62.25	5	64.19	5	180.84	5	588.64	5	52.30	5	286.87
n80α0.1	5	5	1288.64	5	1206.93	5	1591.28	5	3600.90	5	1517.31	5	3625.53
n100α0	5	5	118.19	5	137.84	5	322.12	5	1137.09	5	105.93	5	269.91
n100α0.025	5	5	250.06	5	313.25	5	407.25	5	1935.13	5	222.73	5	1402.83
n100α0.050	5	5	499.37	5	562.54	5	725.42	5	915.70	5	592.60	5	927.39
n100α0.075	5	5	669.82	5	638.04	5	834.89	5	1598.12	5	653.98	4	1069.93
n100α0.1	5	3	638.70	3	642.40	3	611.87	2	1949.42	3	469.28	3	2503.50
n150α0	5	5	697.24	5	799.42	5	1229.77	4	2759.90	5	462.08	4	1645.66
n150α0.025	5	5	932.41	5	968.20	5	1308.02	4	2164.15	5	822.04	5	2197.33
n150α0.050	5	3	579.21	3	603.91	3	866.41	3	2782.44	3	339.36	3	1489.18
n200α0	5	4	573.85	4	656.95	4	872.30	3	3600.03	4	839.55	4	3607.09
n200α0.025	5	3	840.28	3	896.36	3	907.91	1	3600.01	3	881.41	2	3605.14

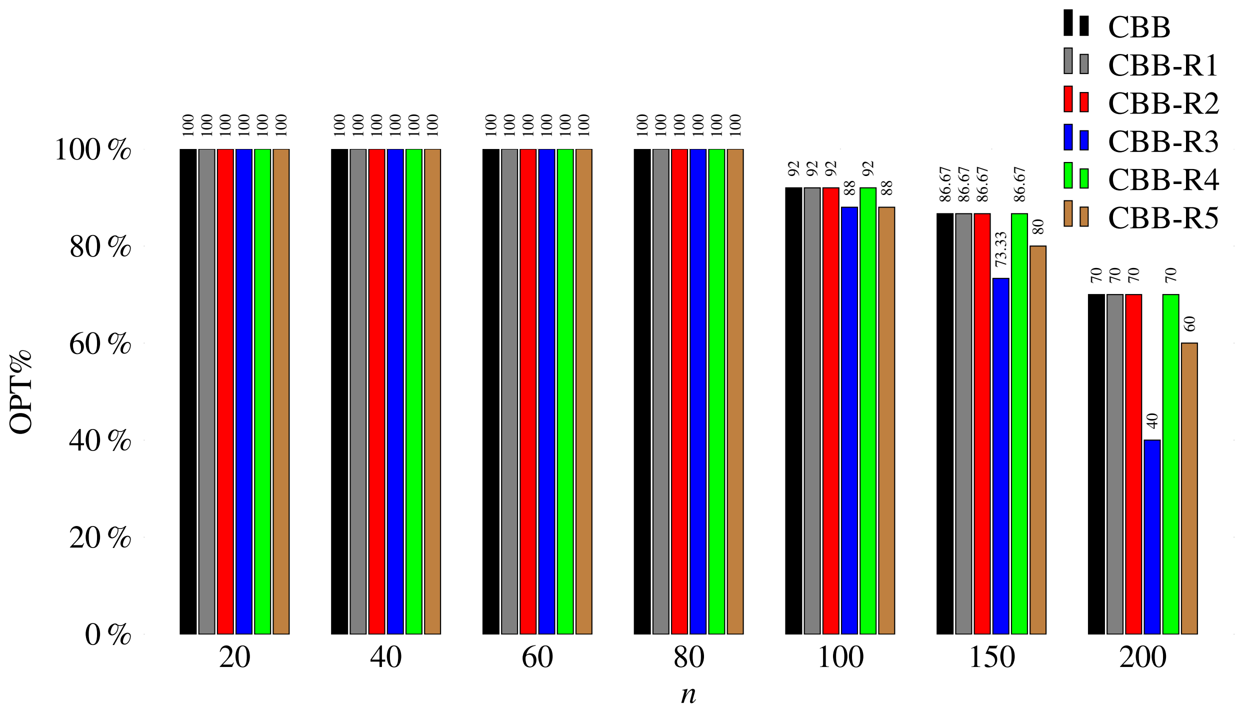


Figure 13.39: Rate of instances solved to optimality by different method configurations for Dataset 1.

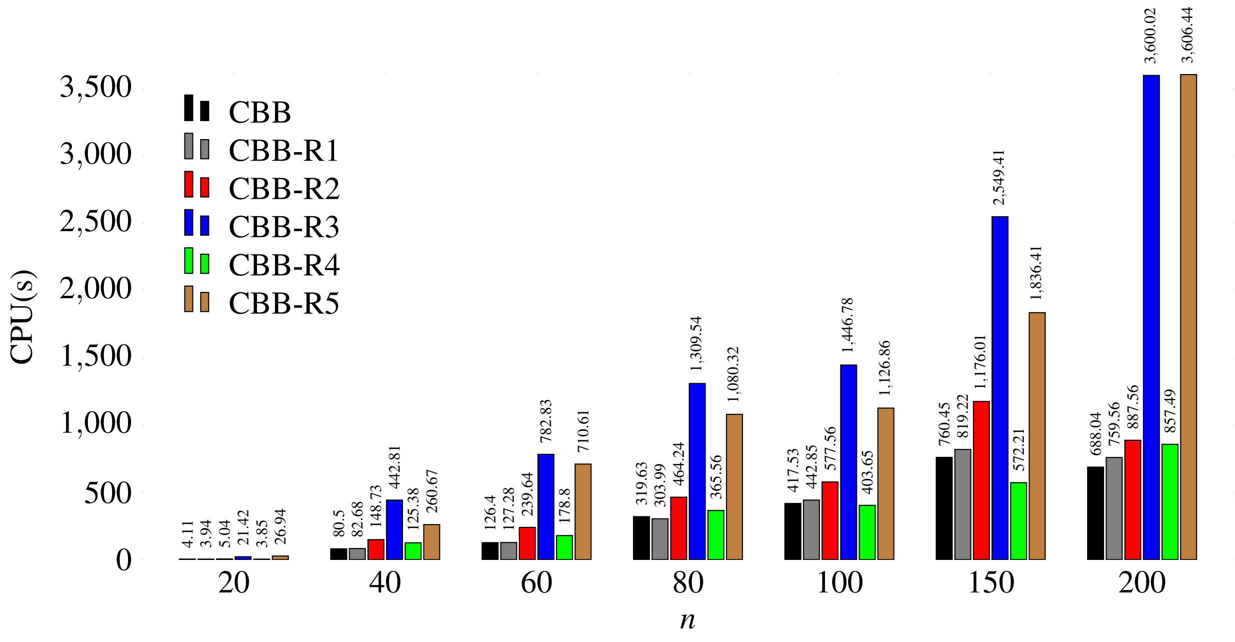


Figure 13.40: CPU time (in seconds) under different method configurations for Dataset 1.

graphical summary of the data included in Table 13.30. According to Figure 13.41, eliminating each of Rule 3 and Rule 5 reduces the number of optimally solved instances, when the instance size is greater than or equal to 150. The CPU time for different method configurations is presented in Figure 13.42, which shows CBB-R3 has the worst computation time and CBB and CBB-R4 have the best. Considering all the 314 instances in Dataset 2, CBB, CBB-R1, CBB-R2, and CBB-R4 solve 311 (99.04%) instances, whereas CBB-R5 and CBB-R3 solve 310 (98.73%) and 308 (98.09%) instances, respectively. In terms of the CPU time, on average, eliminating Rules 4, 1, 2, 5, and 3 increases the CPU time by 23.01 (22.62%), 34.52 (33.94%), 46.11 (45.33%), 164.83 (162.03%), and 223.90 (220.10%) seconds, respectively. Based on both the number of optimally solved instances and the CPU time, again, we can see that CBB-R3 has the worst performance.

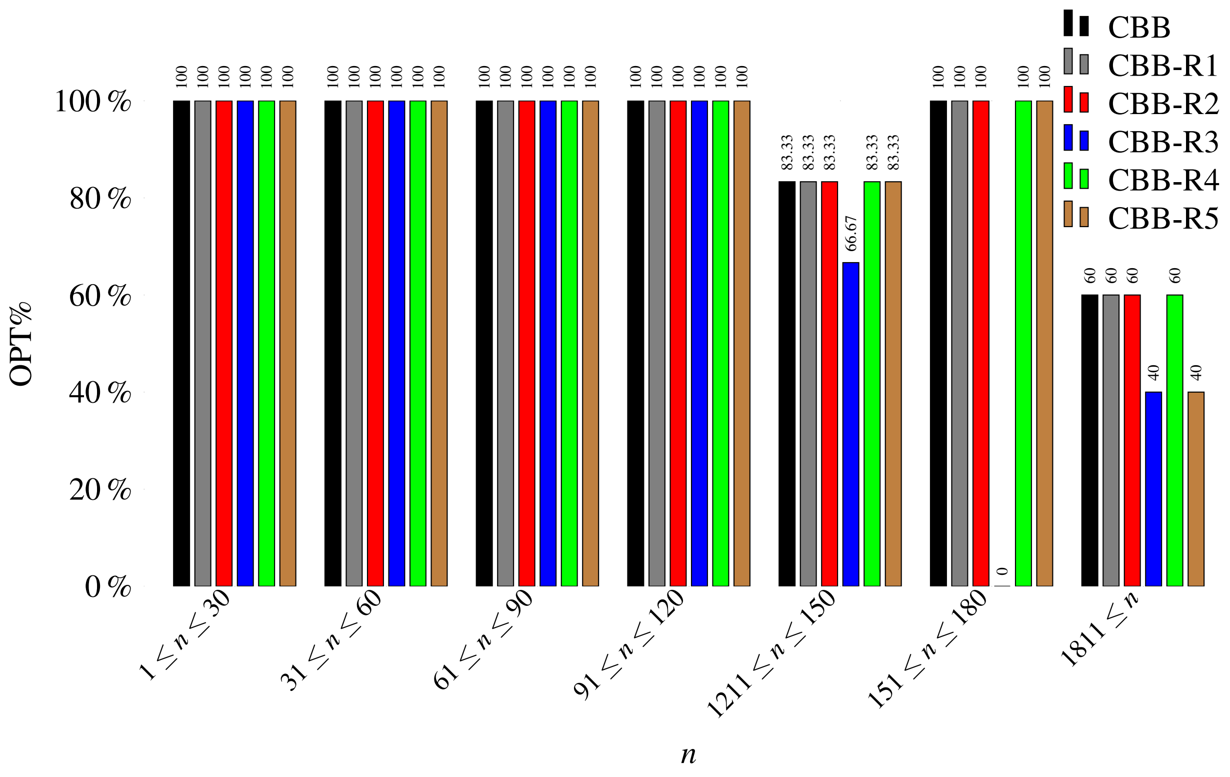


Figure 13.41: Rate of instances solved to optimality by different method configurations for Dataset 2.

Table 13.30: Performance of different method configurations for Dataset 2.

Class	**Size**	**CBB**		**CBB-R1**		**CBB-R2**		**CBB-R3**		**CBB-R4**		**CBB-R5**	
		OPT	CPU(s)	OPT	CPU(s)	OPT	CPU(s)	OPT	CPU(s)	OPT	CPU(s)	OPT	CPU(s)
rbg10α0	1	1	0.28	1	0.10	1	0.08	1	0.09	1	0.04	1	0.09
rbg15α0	1	1	0.43	1	0.13	1	0.32	1	0.15	1	0.11	1	0.13
rbg15α0.025	1	1	0.52	1	0.17	1	0.31	1	0.20	1	0.11	1	0.18
rbg16α0	1	1	0.22	1	0.19	1	0.38	1	0.25	1	0.13	1	0.19
rbg16α0.025	1	1	0.24	1	0.19	1	0.47	1	0.32	1	0.13	1	0.23
rbg17α0	15	15	1.01	15	0.76	15	0.90	15	1.30	15	0.49	15	1.04
rbg17α0.025	17	17	1.79	17	1.14	17	1.36	17	2.55	17	0.93	17	2.51
rbg17α0.050	17	17	1.15	17	0.76	17	1.06	17	1.52	17	0.58	17	1.54
rbg17α0.075	17	17	1.33	17	1.03	17	1.17	17	1.57	17	0.68	17	1.67
rbg17α0.1	17	17	1.81	17	1.43	17	1.57	17	3.17	17	1.09	17	3.31
rbg17α0.125	17	17	2.33	17	1.41	17	1.69	17	3.47	17	1.16	17	4.04
rbg17α0.15	1	1	0.91	1	0.92	1	1.02	1	2.42	1	0.42	1	3.57
rbg19α0	3	3	2.54	3	0.99	3	1.45	3	1.60	3	0.96	3	1.44
rbg19α0.025	2	2	1.89	2	0.86	2	0.99	2	1.66	2	0.42	2	2.19
rbg19α0.050	2	2	0.43	2	0.19	2	0.66	2	0.22	2	0.18	2	0.25
rbg19α0.075	2	2	0.73	2	0.36	2	0.63	2	0.38	2	0.23	2	0.42
rbg19α0.1	1	1	4.58	1	1.96	1	2.09	1	4.87	1	1.53	1	6.30
rbg19α0.125	1	1	2.40	1	0.98	1	1.13	1	1.70	1	0.66	1	1.21
rbg19α0.15	1	1	1.67	1	0.70	1	1.07	1	1.56	1	0.47	1	1.19
rbg19α0.175	1	1	3.31	1	1.31	1	1.58	1	2.40	1	1.04	1	2.62
rbg20α0.125	1	1	13.03	1	5.57	1	6.90	1	20.65	1	6.02	1	26.82
rbg27α0	28	28	27.08	28	36.82	28	50.63	28	259.09	28	47.34	28	238.79
rbg27α0.025	27	27	59.86	27	132.43	27	133.49	27	253.27	27	116.75	27	275.47
rbg27α0.050	27	27	97.68	27	100.46	27	119.57	27	429.14	27	103.15	27	267.81
rbg27α0.075	27	27	57.41	27	58.22	27	61.06	27	412.70	27	58.02	27	222.73
rbg27α0.1	27	27	141.93	27	139.52	27	149.28	27	540.19	27	141.14	27	399.71
rbg27α0.125	27	27	128.82	27	124.30	27	133.73	27	560.81	27	284.55	27	406.64
rbg31α0	1	1	13.18	1	7.72	1	11.24	1	33.96	1	7.31	1	13.14
rbg33α0	1	1	28.53	1	23.09	1	25.01	1	183.95	1	20.64	1	59.84
rbg34α0	1	1	1.70	1	1.33	1	4.93	1	3.95	1	1.16	1	2.18
rbg35α0	1	1	6.33	1	4.98	1	7.35	1	10.84	1	1.80	1	8.59
rbg35α0.025	1	1	24.72	1	13.79	1	14.37	1	105.35	1	10.50	1	19.83
rbg38α0	1	1	19.50	1	11.92	1	13.92	1	67.32	1	7.91	1	17.04
rbg40α0	1	1	7.39	1	4.84	1	11.29	1	32.36	1	3.75	1	6.73
rbg41α0	1	1	13.53	1	12.51	1	16.25	1	88.11	1	8.42	1	15.05
rbg42α0	1	1	14.22	1	7.36	1	12.30	1	20.61	1	2.42	1	9.13
rbg48α0	1	1	15.03	1	10.75	1	17.85	1	29.09	1	3.42	1	16.61
rbg49α0	1	1	139.14	1	148.88	1	175.04	1	2772.53	1	178.09	1	466.93
rbg50α0	1	1	38.96	1	37.88	1	40.59	1	246.30	1	25.85	1	90.57
rbg50α0.025	1	1	12.60	1	13.66	1	24.03	1	43.14	1	4.41	1	25.12
rbg50α0.050	1	1	98.00	1	94.26	1	128.93	1	1879.18	1	99.07	1	2279.20
rbg55α0	1	1	48.89	1	76.84	1	90.97	1	1002.19	1	59.69	1	570.33
rbg67α0	1	1	66.37	1	101.91	1	163.53	1	2368.70	1	110.03	1	343.66
rbg86α0	1	1	71.50	1	98.23	1	209.88	1	1485.60	1	68.95	1	275.42
rbg88α0	1	1	30.26	1	42.96	1	154.59	1	99.73	1	24.16	1	95.33
rbg92α0	1	1	69.77	1	90.01	1	161.34	1	1399.91	1	50.57	1	351.89
rbg125α0	1	1	514.21	1	1145.78	1	1567.30	1	3600.02	1	799.70	1	3290.72
rbg130α0	1	1	764.58	1	1874.45	1	2216.02	1	3600.03	1	1331.39	1	3605.80
rbg130α0.025	1	1	1224.98	1	2463.29	1	2841.38	1	3600.04	1	1726.91	1	3618.57
rbg150α0	1	1	798.58	1	1935.46	1	2283.08	1	3600.00	1	1210.75	1	3602.34
rbg150α0.025	1	1	2975.28	1	3600.83	1	3600.80	0	|	1	3600.67	1	3609.78
rbg150α0.05	1	0	|	0	|	0	|	0	|	0	|	0	|
rbg172α0	1	1	2812.92	1	3600.64	1	3600.55	0	|	1	3600.52	1	3604.13
rbg191α0	1	1	1726.60	1	3601.79	1	3618.50	1	3600.00	1	1065.08	0	|
rbg191α0.025	1	0	|	0	|	0	|	0	|	0	|	0	|
rbg201α0	1	1	3478.10	1	3600.45	1	3600.47	0	|	1	3600.42	1	3601.40
rbg231α0	1	1	2551.14	1	3606.53	1	3673.32	1	3600.01	1	750.71	1	3603.39
rbg231α0.025	1	0	|	0	|	0	|	0	|	0	|	0	|

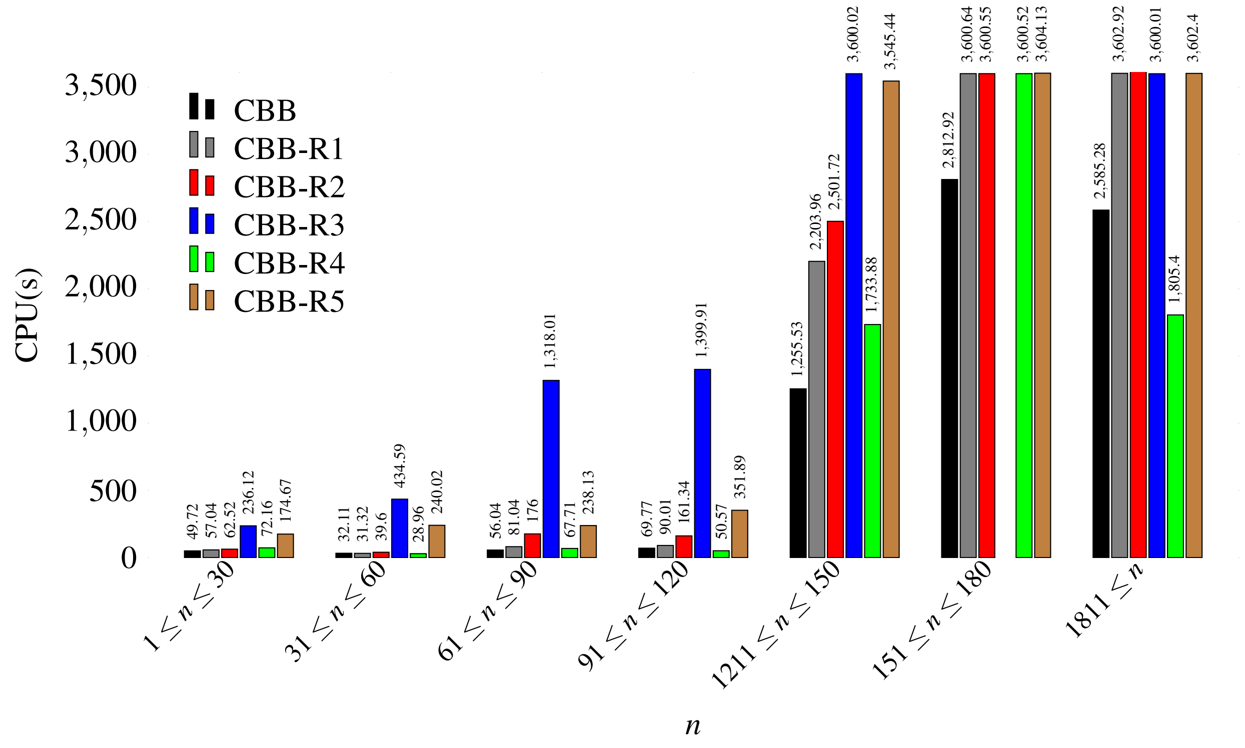


Figure 13.42: CPU time (in seconds) under different method configurations for Dataset 2.

CHAPTER 14. CONCLUSION

14.1. Novel group centrality metrics for studying essentiality in protein-protein interaction networks

In this work, we investigated nine problems based on combining three popular nodal centrality metrics (degree, betweenness, closeness) with three structural requirements (inducing a clique, a star, or having a common node adjacent to every other node in the set). We derived the computational complexity for these problems, and showed that they are indeed NP-hard; we also provided integer programming formulations for solving them exactly. Seeing as our motivation comes from solving problems that arise within protein-protein interaction networks, which tend to be large-scale, we also devised a combinatorial branch-and-bound method that takes advantage of certain interesting properties to speed up the solution process.

The results of our computational experiments can be seen from two different perspectives: (i) comparing solution methods in terms of execution time and solution quality, and (ii) comparing different methods in identifying essential proteins. In the group degree centrality category, considering their solution quality, IP1 and IP2 have a similar performance. From the computational runtime point of view, IP2 and the CBB algorithm perform better. In the group closeness centrality category, the CBB algorithm is 3 times faster on average across all three structures, compared to the integer programming formulation. An even better story is presented in group betweenness centrality experiments, where the integer programming formulation fails to provide a solution, but CBB does.

In terms of identifying essential proteins, the integer programming formulations and the CBB perform equally well in group degree, closeness, and betweenness. Specifically, in group degree, the clique structure performs better; in closeness, all nodal metrics are worse in predicting essentiality than their group counterparts. In group betweenness, an interesting observation is that the performance of group metrics worsens with a decrease in the threshold values used. This observation might be due to the fact that by decreasing the threshold, the network becomes denser and the number of branches in each node of the search tree grows drastically, which in turn affects the quality of the solution in the given time limit. Hence, we obtain a result that shows clique betweenness toe be the best method when the threshold is as high as 800, with simple degree becoming better at a threshold of 600. Overall, considering all three group centrality measures we can say the clique degree centrality (which detects about 50% of the essential proteins in the 500 top ranked proteins) is best at identifying essential proteins.

According to Rasti et al., 2019, one of the discrepancies for PPIN databases is due to high noise rates present in the outputs of the high-throughput techniques used to obtain the interactions. This issue could be addressed by assigning probabilities to the edges which correspond to the probability that these interactions may or may not be present. This change creates a stochastic version of the problem that should be considered as a future research direction for this topic.

14.2. Asymmetric Probabilistic Minimum-cost Hamiltonian Cycle Problem Considering Arc and Vertex Failures

In this research, we propose the APMCHCP considering the possibility of vertex and arc failures in the graph. For this problem, we modify the strengthen MTZ constraints and use direct and recursive methods for modeling the exact chance constraints for the APMCHCP, namely the intuitive formulations and the efficient formulations.

To solve the APMCHCP, we develop data preprocessing procedures, feasibility rules, upper and lower bounds, and utilize them to propose a CBB algorithm. In the computational experiments, it is showed that the proposed CBB algorithm has a superior performance compared to the Gurobi solver (for solving the four formulations), in terms of the size of optimally solved instances and the computing time. Summarizing the presented computational results, the proposed CBB algorithm approach can solve 454 out of 464 ($\approx 97.8\%$) instances with up to 231 nodes from the standard benchmark sets (Dumas, and Ascheuer) to proven optimality.

REFERENCES

Acencio, Marcio L and Ney Lemke (2009). “Towards the prediction of essential genes by integration of network topology, cellular localization and biological process information”. In: *BMC bioinformatics* 10.1, p. 290.

Achterberg, Tobias (2007). “Constraint integer programming”. In:

Achterberg, Tobias and Roland Wunderling (2013). “Mixed integer programming: Analyzing 12 years of progress”. In: *Facets of combinatorial optimization*. Springer, pp. 449–481.

Adamcsek, Balázs et al. (2006). “CFinder: locating cliques and overlapping modules in biological networks”. In: *Bioinformatics* 22.8, pp. 1021–1023.

Adulyasak, Yossiri and Patrick Jaillet (2015). “Models and algorithms for stochastic and robust vehicle routing with deadlines”. In: *Transportation Science* 50.2, pp. 608–626.

Ahuja, Ravindra K, Thomas L Magnanti, and James B Orlin (1988). “Network flows”. In:

Aittokallio, Tero and Benno Schwikowski (2006). “Graph-based methods for analysing networks in cell biology”. In: *Briefings in bioinformatics* 7.3, pp. 243–255.

Akker, Erik van den et al. (2011). “Integrating protein-protein interaction networks with gene-gene co-expression networks improves gene signatures for classifying breast cancer metastasis”. In: *Journal of Integrative Bioinformatics (JIB)* 8.2, pp. 222–238.

Alonso-López, Diego et al. (2016). “APID interactomes: providing proteome-based interactomes with controlled quality for multiple species and derived networks”. In: *Nucleic acids research* 44.W1, W529–W535.

Aloy, Patrick and Robert B Russell (2003). “InterPreTS: Protein Interaction Prediction through Tertiary Structure”. In: *Bioinformatics* 19.1, pp. 161–162.

Altaf-Ul-Amin, Md et al. (2006). “Development and implementation of an algorithm for detection of protein complexes in large interaction networks”. In: *BMC bioinformatics* 7.1, p. 207.

Andersen, Reid, Fan Chung, and Kevin Lang (2006). “Local graph partitioning using pagerank vectors”. In: *Foundations of Computer Science, 2006. FOCS’06. 47th Annual IEEE Symposium on*. IEEE, pp. 475–486.

Antonov, Alexey V (2011). “BioProfiling. de: analytical web portal for high-throughput cell biology”. In: *Nucleic acids research* 39.suppl_2, W323–W327.

Antonov, Alexey V et al. (2009). “PPI spider: a tool for the interpretation of proteomics data in the context of protein–protein interaction networks”. In: *Proteomics* 9.10, pp. 2740–2749.

Aragao, M Poggi de and Eduardo Uchoa (2003). “Integer program reformulation for robust branch-and-cut-and-price algorithms”. In: *Mathematical program in rio: a conference in honour of nelson maculan*. Citeseer, pp. 56–61.

Arigliano, Anna et al. (2019). “Time-dependent asymmetric traveling salesman problem with time windows: Properties and an exact algorithm”. In: *Discrete Applied Mathematics* 261, pp. 28–39.

Arnau, Vicente, Sergio Mars, and Ignacio Marín (2004). “Iterative cluster analysis of protein interaction data”. In: *Bioinformatics* 21.3, pp. 364–378.

Arora, Sanjeev, Béla Bollobás, and László Lovász (2002). “Proving integrality gaps without knowing the linear program”. In: *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.* IEEE, pp. 313–322.

Ascheuer, Norbert (1996). “Hamiltonian path problems in the on-line optimization of flexible manufacturing systems”. In:

Ascheuer, Norbert, Matteo Fischetti, and Martin Grötschel (2001). “Solving the asymmetric travelling salesman problem with time windows by branch-and-cut”. In: *Mathematical Programming* 90.3, pp. 475–506.

Ashburner, Michael et al. (2000). “Gene Ontology: tool for the unification of biology”. In: *Nature genetics* 25.1, p. 25.

Asur, Sitaram, Duygu Ucar, and Srinivasan Parthasarathy (2007). “An ensemble framework for clustering protein–protein interaction networks”. In: *Bioinformatics* 23.13, pp. i29–i40.

Atamtürk, Alper (2005). “Cover and pack inequalities for (mixed) integer programming”. In: *Annals of Operations Research* 139.1, pp. 21–38.

Atamtürk, Alper and Martin WP Savelsbergh (2005). “Integer-programming software systems”. In: *Annals of operations research* 140.1, pp. 67–124.

Aytuna, A Selim, Attila Gursoy, and Ozlem Keskin (2005). “Prediction of protein–protein interactions by combining structure and sequence conservation in protein interfaces”. In: *Bioinformatics* 21.12, pp. 2850–2855.

Babel, Luitpold (1994). “A fast algorithm for the maximum weight clique problem”. In: *Computing* 52.1, pp. 31–38.

Bader, Gary D et al. (2001). “BIND: the biomolecular interaction network database”. In: *Nucleic acids research* 29.1, pp. 242–245.

Bader, Gary D and Christopher WV Hogue (2003a). “An automated method for finding molecular complexes in large protein interaction networks”. In: *BMC bioinformatics* 4.1, p. 2.

Bader, Gary D, Doron Betel, and Christopher WV Hogue (2003b). “BIND: the biomolecular interaction network database”. In: *Nucleic acids research* 31.1, pp. 248–250.

Bader, Joel S et al. (2004). “Gaining confidence in high-throughput protein interaction networks”. In: *Nature biotechnology* 22.1, p. 78.

Balas, Egon (1971). “Intersection cuts–a new type of cutting planes for integer programming”. In: *Operations Research* 19.1, pp. 19–39.

Balas, Egon, Sebastián Ceria, and Gérard Cornuéjols (1993). “A lift-and-project cutting plane algorithm for mixed 0–1 programs”. In: *Mathematical programming* 58.1-3, pp. 295–324.

Balas, Egon et al. (1996a). “Gomory cuts revisited”. In: *Operations Research Letters* 19.1, pp. 1–9.

Balas, Egon, Sebastián Ceria, and Gérard Cornuéjols (1996b). “Mixed 0-1 programming by lift-and-project in a branch-and-cut framework”. In: *Management Science* 42.9, pp. 1229–1246.

Balas, Egon and Michael Perregaard (2003). “A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming”. In: *Mathematical Programming* 94.2-3, pp. 221–245.

Balas, Egon and Anureet Saxena (2008). "Optimizing over the split closure". In: *Mathematical Programming* 113.2, pp. 219–240.

Barabási, Albert-László and Réka Albert (1999). "Emergence of scaling in random networks". In: *science* 286.5439, pp. 509–512.

Barabasi, Albert-Laszlo and Zoltan N Oltvai (2004). "Network biology: understanding the cell's functional organization". In: *Nature reviews. Genetics* 5.2, p. 101.

Barnhart, Cynthia, Christopher A Hane, and Pamela H Vance (2000). "Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems". In: *Operations Research* 48.2, pp. 318–326.

Batada, Nizar N, Laurence D Hurst, and Mike Tyers (2006). "Evolutionary and physiological importance of hub proteins". In: *PLoS computational biology* 2.7, e88.

Baumeister, Wolfgang, Rudo Grimm, and Jochen Walz (1999). "Electron tomography of molecules and cells". In: *Trends in cell biology* 9.2, pp. 81–85.

Beale, EML and John JH Forrest (1976). "Global optimization using special ordered sets". In: *Mathematical Programming* 10.1, pp. 52–69.

Beale, Evelyn Martin Lansdowne and John A Tomlin (1970). "Special facilities in a general mathematical programming system for non-convex problems using ordered sets of variables". In: *OR* 69.447-454, p. 99.

Belov, Gleb and Guntram Scheithauer (2006). "A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting". In: *European journal of operational research* 171.1, pp. 85–106.

Bender, ALAN and JOHN R Pringle (1991). "Use of a screen for synthetic lethal and multi-copy suppressee mutants to identify two new genes involved in morphogenesis in Saccharomyces cerevisiae." In: *Molecular and cellular biology* 11.3, pp. 1295–1305.

Benders, JF (1962). "Partitioning procedures for solving mixed-variables programming problems, Num Math". In: *Num. Math. Benders41962*.

Bensimon, Ariel, Albert JR Heck, and Ruedi Aebersold (2012). “Mass spectrometry–based proteomics and network biology”. In: *Annual review of biochemistry* 81, pp. 379–405.

Berggård, Tord, Sara Linse, and Peter James (2007). “Methods for the detection and analysis of protein–protein interactions”. In: *Proteomics* 7.16, pp. 2833–2842.

Berman, Oded and David Simchi-Levi (1988). “Finding the optimal a priori tour and location of a traveling salesman with nonhomogeneous customers”. In: *Transportation Science* 22.2, pp. 148–154.

Bertsimas, Dimitris (1988). “Probabilistic combinatorial optimization problems”. PhD thesis. Massachusetts Institute of Technology.

Bertsimas, Dimitris and Louis H Howell (1993). “Further results on the probabilistic traveling salesman problem”. In: *European Journal of Operational Research* 65.1, pp. 68–95.

Bertsimas, Dimitris and John N Tsitsiklis (1997). *Introduction to linear optimization*. Vol. 6. Athena Scientific Belmont, MA.

Bertsimas, Dimitris J (1992). “A vehicle routing problem with stochastic demand”. In: *Operations Research* 40.3, pp. 574–585.

Bertsimas, Dimitris J, Patrick Jaillet, and Amedeo R Odoni (1990). “A priori optimization”. In: *Operations Research* 38.6, pp. 1019–1033.

Bhowmick, Sourav S and Boon Siew Seah (2016). “Clustering and summarizing protein-protein interaction networks: a survey”. In: *IEEE Transactions on Knowledge and Data Engineering* 28.3, pp. 638–658.

Birattari, Mauro et al. (2008). “Estimation-based local search for stochastic combinatorial optimization using delta evaluations: a case study on the probabilistic traveling salesman problem”. In: *INFORMS Journal on Computing* 20.4, pp. 644–658.

Birlutiu, Adriana and Tom Heskes (2014). “Using topology information for protein-protein interaction prediction”. In: *IAPR International Conference on Pattern Recognition in Bioinformatics*. Springer, pp. 10–22.

Blatt, Marcelo, Shai Wiseman, and Eytan Domany (1996). “Superparamagnetic Clustering of Data”. In: *Phys. Rev. Lett.* 76 (18), pp. 3251–3254. DOI: `10.1103/PhysRevLett.76.3251`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.76.3251`.

BnnoBRs, JF (1962). “Partitioning procedures for solving mixed-variables programming problems”. In:

Borch, Jonas, Peter Roepstorff, and Jakob Møller-Jensen (2011). “Nanodisc-based co-immunoprecipitation for mass spectrometric identification of membrane-interacting proteins”. In: *Molecular & Cellular Proteomics* 10.7, O110–006775.

Borgatti, Stephen P (2006). “Identifying sets of key players in a social network”. In: *Computational & Mathematical Organization Theory* 12.1, pp. 21–34.

Boulon, Séverine et al. (2010). “Establishment of a protein frequency library and its application in the reliable identification of specific protein interaction partners”. In: *Molecular & Cellular Proteomics* 9.5, pp. 861–879.

Bowers, Peter M et al. (2004). “Prolinks: a database of protein functional linkages derived from coevolution”. In: *Genome biology* 5.5, R35.

Bray, Dennis (1995). “Protein molecules as computational elements in living cells”. In: *Nature* 376.6538, pp. 307–312.

Brohee, Sylvain and Jacques Van Helden (2006). “Evaluation of clustering algorithms for protein-protein interaction networks”. In: *BMC bioinformatics* 7.1, p. 488.

Brohée, Sylvain et al. (2008). “Network Analysis Tools: from biological networks to clusters and pathways”. In: *Nature protocols* 3.10, p. 1616.

Bu, Dongbo et al. (2003). “Topological structure analysis of the protein–protein interaction network in budding yeast”. In: *Nucleic acids research* 31.9, pp. 2443–2450.

Campbell, Ann M and Barrett W Thomas (2008). “Probabilistic traveling salesman problem with deadlines”. In: *Transportation Science* 42.1, pp. 1–21.

Campbell, Ann Melissa and Philip C Jones (2011). “Prepositioning supplies in preparation for disasters”. In: *European Journal of Operational Research* 209.2, pp. 156–165.

Carpaneto, Giorgio and Paolo Toth (1980). “Some new branching and bounding criteria for the asymmetric travelling salesman problem”. In: *Management Science* 26.7, pp. 736–743.

Chang, Tsung-Sheng, Yat-wah Wan, and Wei Tsang Ooi (2009). “A stochastic dynamic traveling salesman problem with hard time windows”. In: *European Journal of Operational Research* 198.3, pp. 748–759.

Chatr-aryamontri, Andrew et al. (2008). “Protein interactions: integration leads to belief”. In: *Trends in biochemical sciences* 33.6, pp. 241–242.

Chatr-aryamontri, Andrew et al. (2017). “The BioGRID interaction database: 2017 update”. In: *Nucleic acids research* 45.D1, pp. D369–D379.

Chaurasia, Gautam et al. (2006). “UniHI: an entry gate to the human protein interactome”. In: *Nucleic acids research* 35.suppl_1, pp. D590–D594.

Chen, Bolin et al. (2011). “An improved graph entropy-based method for identifying protein complexes”. In: *Bioinformatics and Biomedicine (BIBM), 2011 IEEE International Conference on*. IEEE, pp. 123–126.

Chen, Bolin, Jinhong Shi, and Fang-Xiang Wu (2012). “Not AU protein complexes exhibit dense structures in S. cerevisiae PPI network”. In: *Bioinformatics and Biomedicine (BIBM), 2012 IEEE International Conference on*. IEEE, pp. 1–4.

Chen, Bolin et al. (2013). “Identifying protein complexes and functional modules—from static PPI networks to dynamic PPI networks”. In: *Briefings in bioinformatics* 15.2, pp. 177–194.

Chen, Feng et al. (2006a). “OrthoMCL-DB: querying a comprehensive multi-species collection of ortholog groups”. In: *Nucleic acids research* 34.suppl_1, pp. D363–D368.

Chen, Jake Y, Ragini Pandey, and Thanh M Nguyen (2017). “HAPPI-2: a Comprehensive and High-quality Map of Human Annotated and Predicted Protein Interactions”. In: *BMC genomics* 18.1, p. 182.

Chen, Jake Yue, SudhaRani Mamidipalli, and Tianxiao Huan (2009). “HAPPI: an online database of comprehensive human annotated and predicted protein interactions”. In: *BMC genomics* 10.1, S16.

Chen, Jin et al. (2005). “Discovering reliable protein interactions from high-throughput experimental data using network topology”. In: *Artificial intelligence in medicine* 35.1, pp. 37–47.

Chen, Jin et al. (2006b). “Increasing confidence of protein-protein interactomes”. In: *Genome Informatics* 17.2, pp. 284–297.

Chen, Jingchun and Bo Yuan (2006c). “Detecting functional modules in the yeast protein–protein interaction network”. In: *Bioinformatics* 22.18, pp. 2283–2290.

Chen, Yu and Dong Xu (2004). “Understanding protein dispensability through machine-learning analysis of high-throughput data”. In: *Bioinformatics* 21.5, pp. 575–581.

Cheng, Jian et al. (2013). “A new computational strategy for predicting essential genes”. In: *BMC genomics* 14.1, p. 910.

Cheng, Jian et al. (2014). “Training set selection for the prediction of essential genes”. In: *PloS one* 9.1, e86805.

Chiang, Tony et al. (2007). “Coverage and error models of protein-protein interaction data by directed graph analysis”. In: *Genome biology* 8.9, R186.

Chinneck, John W (2006). “Practical optimization: a gentle introduction”. In: *Systems and Computer Engineering), Carleton University, Ottawa. http://www. sce. carleton. ca/faculty/chinneck/po. html*, p. 11.

Cho, Young-Rae et al. (2007). “Semantic integration to identify overlapping functional modules in protein interaction networks”. In: *BMC bioinformatics* 8.1, p. 265.

Cho, Young-Rae, Lei Shi, and Aidong Zhang (2008). “Functional module detection by functional flow pattern mining in protein interaction networks”. In: *BMC Bioinformatics* 9.10, O1.

Chua, Hon Nian, Wing-Kin Sung, and Limsoon Wong (2006). “Exploiting indirect neighbours and topological weight to predict protein function from protein–protein interactions”. In: *Bioinformatics* 22.13, pp. 1623–1630.

Chua, Hon Nian and Limsoon Wong (2008). “Increasing the reliability of protein interactomes”. In: *Drug discovery today* 13.15, pp. 652–658.

Chvatal, Vasek (1973). “Edmonds polytopes and a hierarchy of combinatorial problems”. In: *Discrete mathematics* 4.4, pp. 305–337.

Clatworthy, Anne E, Emily Pierson, and Deborah T Hung (2007). “Targeting virulence: a new paradigm for antimicrobial therapy”. In: *Nature chemical biology* 3.9, pp. 541–548.

Clausen, Jens (1999). “Branch and bound algorithms-principles and examples”. In: *Department of Computer Science, University of Copenhagen*, pp. 1–30.

Cook, William, Ravindran Kannan, and Alexander Schrijver (1990). “Chvátal closures for mixed integer programming problems”. In: *Mathematical Programming* 47.1-3, pp. 155–174.

Cooper, Matthew A (2003). “Label-free screening of bio-molecular interactions”. In: *Analytical and bioanalytical chemistry* 377.5, pp. 834–842.

Cormen, Thomas H et al. (2009). *Introduction to algorithms*. MIT press.

Cornuéjols, Gérard (2008). “Valid inequalities for mixed integer linear programs”. In: *Mathematical Programming* 112.1, pp. 3–44.

Coulomb, Stéphane et al. (2005). “Gene essentiality and the topology of protein interaction networks”. In: *Proceedings of the Royal Society of London B: Biological Sciences* 272.1573, pp. 1721–1725.

Cowley, Mark J et al. (2011). “PINA v2. 0: mining interactome modules”. In: *Nucleic acids research* 40.D1, pp. D862–D865.

Craig, Roger A and Li Liao (2007). “Phylogenetic tree information aids supervised learning for predicting protein-protein interaction based on distance matrices”. In: *Bmc Bioinformatics* 8.1, p. 6.

Crowder, Harlan, Ellis L Johnson, and Manfred Padberg (1983). “Solving large-scale zero-one linear programming problems”. In: *Operations Research* 31.5, pp. 803–834.

Cuatrecasas, Pedro (1970). “Protein purification by affinity chromatography derivatizations of agarose and polyacrylamide beads”. In: *Journal of Biological Chemistry* 245.12, pp. 3059–3065.

Cui, Guangyu et al. (2008). “An algorithm for finding functional modules and protein complexes in protein-protein interaction networks”. In: *BioMed Research International* 2008.

Danon, Leon et al. (2005). “Comparing community structure identification”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2005.09, P09008.

Dantzig, George B and Philip Wolfe (1960). “Decomposition principle for linear programs”. In: *Operations research* 8.1, pp. 101–111.

Das, Jishnu and Haiyuan Yu (2012). “HINT: High-quality protein interactomes and their applications in understanding human disease”. In: *BMC systems biology* 6.1, p. 92.

Daud, Mimi Suriani Mat et al. (2016). “Humanitarian logistics and its challenges: The literature review”. In: *International Journal of Supply Chain Management* 5.3, pp. 107–110.

Deane, Charlotte M et al. (2002). “Protein interactions two methods for assessment of the reliability of high throughput observations”. In: *Molecular & Cellular Proteomics* 1.5, pp. 349–356.

Dechter, Rina and Judea Pearl (1985). “Generalized best-first search strategies and the optimality of A”. In: *Journal of the ACM (JACM)* 32.3, pp. 505–536.

Deng, Jingyuan et al. (2010). “Investigating the predictability of essential genes across distantly related organisms using an integrative approach”. In: *Nucleic acids research* 39.3, pp. 795–807.

Deng, Minghua, Fengzhu Sun, and Ting Chen (2002). “Assessment of the reliability of protein-protein interactions and protein function prediction”. In: *Pac. Symp. Biocomputing (PSB 2003). Singapore: World Scientific*, pp. 140–51.

Deng, Minghua et al. (2004). “Mapping gene ontology to proteins based on protein–protein interaction data”. In: *Bioinformatics* 20.6, pp. 895–902.

Derényi, Imre, Gergely Palla, and Tamás Vicsek (2005). “Clique percolation in random networks”. In: *Physical review letters* 94.16, p. 160202.

Desaulniers, Guy, Jacques Desrosiers, and Marius M Solomon (2006). *Column generation*. Vol. 5. Springer Science & Business Media.

Desrochers, Martin and Gilbert Laporte (1991). “Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints”. In: *Operations Research Letters* 10.1, pp. 27–36.

Desrosiers, Jacques et al. (1995). “Time constrained routing and scheduling”. In: *Handbooks in operations research and management science* 8, pp. 35–139.

Desrosiers, Jacques, Raf Jans, and Yossiri Adulyasak (2013). *Improved column generation algorithms for the job grouping problem*. Groupe d’études et de recherche en analyse des décisions.

Ding, Yijun et al. (2012). “atBioNet–an integrated network analysis tool for genomics and biomarker discovery”. In: *BMC genomics* 13.1, p. 325.

Dittrich, Marcus T et al. (2008). “Identifying functional modules in protein–protein interaction networks: an integrated exact approach”. In: *Bioinformatics* 24.13, pp. i223–i231.

Dotan-Cohen, Dikla, Avraham A Melkman, and Simon Kasif (2007). “Hierarchical tree snipping: clustering guided by prior knowledge”. In: *Bioinformatics* 23.24, pp. 3335–3342.

Dror, Moshe (1994). “Note on the complexity of the shortest path models for column generation in VRPTW”. In: *Operations Research* 42.5, pp. 977–978.

Dror, Moshe, Gilbert Laporte, and Pierre Trudeau (1989). “Vehicle routing with stochastic demands: Properties and solution frameworks”. In: *Transportation science* 23.3, pp. 166–176.

Dumas, Yvan et al. (1995). “An optimal algorithm for the traveling salesman problem with time windows”. In: *Operations research* 43.2, pp. 367–371.

Duran, Serhan, Marco A Gutierrez, and Pinar Keskinocak (2011). “Pre-positioning of emergency items for CARE international”. In: *Interfaces* 41.3, pp. 223–237.

Edwards, Aled M et al. (2002). “Bridging structural biology and genomics: assessing protein interaction data with known complexes”. In: *TRENDS in Genetics* 18.10, pp. 529–536.

Ekman, Diana et al. (2006). “What properties characterize the hub proteins of the protein-protein interaction network of Saccharomyces cerevisiae?” In: *Genome biology* 7.6, R45.

Enright, Anton J et al. (1999). “Protein interaction maps for complete genomes based on gene fusion events”. In: *Nature* 402.6757, p. 86.

Enright, Anton J, Stijn Van Dongen, and Christos A Ouzounis (2002). “An efficient algorithm for large-scale detection of protein families”. In: *Nucleic acids research* 30.7, pp. 1575–1584.

Eremin, Andrew (2004). “Using dual values to integrate row and column generation into constant logic”. PhD thesis. Imperial College London (University of London).

Eremin, Andrew and Mark Wallace (2001). “Hybrid Benders decomposition algorithms in constraint logic programming”. In: *International Conference on Principles and Practice of Constraint Programming*. Springer, pp. 1–15.

Errico, Fausto et al. (2018). “The vehicle routing problem with hard time windows and stochastic service times”. In: *EURO Journal on Transportation and Logistics* 7.3, pp. 223–251.

Estrada, Ernesto (2006). “Virtual identification of essential proteins within the protein interaction network of yeast”. In: *Proteomics* 6.1, pp. 35–40.

Estrada, Ernesto and Juan A Rodriguez-Velazquez (2005). “Subgraph centrality in complex networks”. In: *Physical Review E* 71.5, p. 056103.

Everett, Martin G and Stephen P Borgatti (1999). “The centrality of groups and classes”. In: *The Journal of mathematical sociology* 23.3, pp. 181–201.

Everett, Martin G and Stephen P Borgatti (2005). “Extending centrality”. In: *Models and methods in social network analysis* 35.1, pp. 57–76.

Fang, Yi et al. (2011). “Global geometric affinity for revealing high fidelity protein interaction network”. In: *PloS one* 6.5, e19349.

Farkas, Illés J, Ádám Szántó-Várnagy, and Tamás Korcsmáros (2012). “Linking proteins to signaling pathways for experiment design and evaluation”. In: *PloS one* 7.4, e36202.

Fields, Stanley and Ok-kyu Song (1989). “A novel genetic system to detect protein–protein interactions”. In: *Nature* 340.6230, pp. 245–246.

Fischetti, Matteo and Domenico Salvagnin (2010). “Pruning moves”. In: *INFORMS Journal on Computing* 22.1, pp. 108–119.

Fischetti, Matteo and Domenico Salvagnin (2013). “Approximating the split closure”. In: *INFORMS Journal on Computing* 25.4, pp. 808–819.

Flippo, Olaf E and Alexander HG Rinnooy Kan (1993). “Decomposition in general mathematical programming”. In: *Mathematical Programming* 60.1-3, pp. 361–382.

Fortunato, Santo (2010). “Community detection in graphs”. In: *Physics Reports* 486.3, pp. 75–174.

Fortunato, Santo and Marc Barthélemy (2007). “Resolution limit in community detection”. In: *Proceedings of the National Academy of Sciences* 104.1, pp. 36–41.

Fraser, Hunter B et al. (2002). “Evolutionary rate in the protein interaction network”. In: *Science* 296.5568, pp. 750–752.

Friedel, Caroline, Jan Krumsiek, and Ralf Zimmer (2008). “Bootstrapping the interactome: unsupervised identification of protein complexes in yeast”. In: *Research in Computational Molecular Biology*. Springer, pp. 3–16.

Fryxell, Karl J (1996). “The coevolution of gene family trees”. In: *Trends in Genetics* 12.9, pp. 364–369.

Fujimori, Shigeo et al. (2012). “IRView: a database and viewer for protein interacting regions”. In: *Bioinformatics* 28.14, pp. 1949–1950.

Fukasawa, Ricardo et al. (2006). “Robust branch-and-cut-and-price for the capacitated vehicle routing problem”. In: *Mathematical programming* 106.3, pp. 491–511.

Futschik, Matthias E, Gautam Chaurasia, and Hanspeter Herzel (2007). “Comparison of human protein–protein interaction maps”. In: *Bioinformatics* 23.5, pp. 605–611.

Gao, Guanghua, Jason G Williams, and Sharon L Campbell (2004). “Protein-protein interaction analysis by nuclear magnetic resonance spectroscopy”. In: *Protein-Protein Interactions: Methods and Applications*, pp. 79–91.

Gao, Jing et al. (2008). “Integrating and annotating the interactome using the MiMI plugin for cytoscape”. In: *Bioinformatics* 25.1, pp. 137–138.

Gavin, Anne-Claude et al. (2002). “Functional organization of the yeast proteome by systematic analysis of protein complexes”. In: *Nature* 415.6868, pp. 141–147.

Gavin, Anne-Claude et al. (2006). “Proteome survey reveals modularity of the yeast cell machinery”. In: *Nature* 440.7084, p. 631.

Ge, Hui et al. (2001). “Correlation between transcriptome and interactome mapping data from Saccharomyces cerevisiae”. In: *Nature genetics* 29.4, p. 482.

Gendron, Bernard, Paul-Virak Khuong, and Frédéric Semet (2016). “A Lagrangian-based branch-and-bound algorithm for the two-level uncapacitated facility location problem with single-assignment constraints”. In: *Transportation Science* 50.4, pp. 1286–1299.

Gene Ontology Consortium (2001). “Creating the gene ontology resource: design and implementation”. In: *Genome research* 11.8, pp. 1425–1433.

Gene Ontology Consortium (2004). “The Gene Ontology (GO) database and informatics resource”. In: *Nucleic acids research* 32.suppl 1, pp. D258–D261.

Geoffrion, Arthur M (1969). “An improved implicit enumeration approach for integer programming”. In: *Operations Research* 17.3, pp. 437–454.

Geoffrion, Arthur M (1972). “Generalized benders decomposition”. In: *Journal of optimization theory and applications* 10.4, pp. 237–260.

Georgii, Elisabeth et al. (2009). “Enumeration of condition-dependent dense modules in protein interaction networks”. In: *Bioinformatics* 25.7, pp. 933–940.

Giaever, Guri et al. (2002). “Functional profiling of the Saccharomyces cerevisiae genome”. In: *nature* 418.6896, p. 387.

Gingras, Anne-Claude et al. (2007). “Analysis of protein complexes using mass spectrometry”. In: *Nature reviews. Molecular cell biology* 8.8, p. 645.

Girvan, Michelle and Mark EJ Newman (2002). “Community structure in social and biological networks”. In: *Proceedings of the national academy of sciences* 99.12, pp. 7821–7826.

Glass, John I et al. (2006). “Essential genes of a minimal bacterium”. In: *Proceedings of the National Academy of Sciences of the United States of America* 103.2, pp. 425–430.

Glass, John I et al. (2009). “A systems biology tour de force for a near-minimal bacterium”. In: *Molecular systems biology* 5.1, p. 330.

Goel, Renu et al. (2012). "Human Protein Reference Database and Human Proteinpedia as resources for phosphoproteome analysis". In: *Molecular bioSystems* 8.2, pp. 453–463.

Goh, Chern-Sing et al. (2000). "Co-evolution of proteins with their interaction partners". In: *Journal of molecular biology* 299.2, pp. 283–293.

Goh, K-I et al. (2003). "Betweenness centrality correlation in social networks". In: *Physical Review E* 67.1, p. 017101.

Goldberg, Debra S and Frederick P Roth (2003). "Assessing experimentally derived interactions in a small world". In: *Proceedings of the National Academy of Sciences* 100.8, pp. 4372–4376.

Goll, Johannes and Peter Uetz (2006). "The elusive yeast interactome". In: *Genome biology* 7.6, p. 223.

Golomb, Solomon W and Leonard D Baumert (1965). "Backtrack programming". In: *Journal of the ACM (JACM)* 12.4, pp. 516–524.

Gomory, Ralph E (2010). "Outline of an algorithm for integer solutions to linear programs and an algorithm for the mixed integer problem". In: *50 Years of Integer Programming 1958-2008*. Springer, pp. 77–103.

Greene, Derek et al. (2008). "Ensemble non-negative matrix factorization methods for clustering protein–protein interactions". In: *Bioinformatics* 24.15, pp. 1722–1728.

Grigoriev, Andrei (2001). "A relationship between gene expression and protein interactions on the proteome scale: analysis of the bacteriophage T7 and the yeast Saccharomyces cerevisiae". In: *Nucleic acids research* 29.17, pp. 3513–3519.

Guo, Yanzhi et al. (2008). "Using support vector machine combined with auto covariance to predict protein–protein interactions from protein sequences". In: *Nucleic acids research* 36.9, pp. 3025–3030.

Gustafson, Adam M et al. (2006). "Towards the identification of essential genes using targeted genome sequencing and comparative analysis". In: *Bmc Genomics* 7.1, p. 265.

Gutin, G, A Yeo, and A Zverovich (2002). *Exponential Neighborhoods and Domination Analysis for the TSP. Chapter 6 in: The Traveling Salesman Problem and Its Variations. G. Gutin, AP Punnen*.

Gutin, Gregory and Abraham P Punnen (2006). *The traveling salesman problem and its variations*. Vol. 12. Springer Science & Business Media.

Guzelsoy, Menal, George Nemhauser, and Martin Savelsbergh (2013). "Restrict-and-relax search for 0-1 mixed-integer programs". In: *EURO Journal on Computational Optimization* 1.1-2, pp. 201–218.

Gygi, Steven P et al. (2002). "Proteome analysis of low-abundance proteins using multidimensional chromatography and isotope-coded affinity tags". In: *Journal of proteome research* 1.1, pp. 47–54.

Hagberg, Aric A., Daniel A. Schult, and Pieter J. Swart (2008). "Exploring network structure, dynamics, and function using NetworkX". In: *Proceedings of the 7th Python in Science Conference (SciPy2008)*. Pasadena, CA USA, pp. 11–15.

Hahn, Matthew W and Andrew D Kern (2004a). "Comparative genomics of centrality and essentiality in three eukaryotic protein-interaction networks". In: *Molecular biology and evolution* 22.4, pp. 803–806.

Hahn, Matthew W, Gavin C Conant, and Andreas Wagner (2004b). "Molecular evolution in large genetic networks: does connectivity equal constraint?" In: *Journal of molecular evolution* 58.2, pp. 203–211.

Hahn, Matthew W and Andrew D Kern (2005). "Comparative genomics of centrality and essentiality in three eukaryotic protein-interaction networks". In: *Molecular biology and evolution* 22.4, pp. 803–806.

Hakes, Luke et al. (2006). "Protein interactions from complexes: a structural perspective". In: *Comparative and functional genomics* 2007.

Hakes, Luke et al. (2008). "Protein-protein interaction networks and biology—what's the connection?" In: *Nature biotechnology* 26.1, pp. 69–72.

Hall, David A, Jason Ptacek, and Michael Snyder (2007). “Protein microarray technology”. In: *Mechanisms of ageing and development* 128.1, pp. 161–167.

Han, Dong-Soo et al. (2004a). “PreSPI: a domain combination based prediction system for protein–protein interaction”. In: *Nucleic Acids Research* 32.21, pp. 6312–6320.

Han, Jing-Dong J et al. (2004b). “Evidence for dynamically organized modularity in the yeast protein-protein interaction network”. In: *Nature* 430.6995, p. 88.

Hao, Tong et al. (2016). “Reconstruction and application of protein–protein interaction network”. In: *International journal of molecular sciences* 17.6, p. 907.

Hart, Darren J et al. (1999). “The salt dependence of DNA recognition by N-κB p50: A detailed kinetic analysis of the effects on affinity and specificity”. In: *Nucleic acids research* 27.4, pp. 1063–1069.

Hart, G Traver, Arun K Ramani, and Edward M Marcotte (2006). “How complete are current yeast and human protein-interaction networks?” In: *Genome biology* 7.11, p. 120.

Hart, G Traver, Insuk Lee, and Edward M Marcotte (2007). “A high-accuracy consensus map of yeast protein complexes reveals modular nature of gene essentiality”. In: *BMC bioinformatics* 8.1, p. 236.

Hartmanis, Juris (1982). “Computers and intractability: a guide to the theory of NP-completeness (michael r. garey and david s. johnson)”. In: *Siam Review* 24.1, p. 90.

He, Xionglei and Jianzhi Zhang (2006). “Why do hubs tend to be essential in protein networks?” In: *PLoS genetics* 2.6, e88.

Hegde, Shubhada R, Palanisamy Manimaran, and Shekhar C Mande (2008). “Dynamic changes in protein functional linkage networks revealed by integration with gene expression data”. In: *PLoS computational biology* 4.11, e1000237.

Held, Stephan, William Cook, and Edward C Sewell (2012). “Maximum-weight stable sets and safe lower bounds for graph coloring”. In: *Mathematical Programming Computation* 4.4, pp. 363–381.

Henchiri, Abir, Monia Bellalouna, and Walid Khaznaji (2014). “A probabilistic traveling salesman problem: a survey.” In: *FedCSIS Position Papers* 3, pp. 55–60.

Hernández-Pérez, Hipólito and Juan-José Salazar-González (2004). “A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery”. In: *Discrete Applied Mathematics* 145.1, pp. 126–139.

Hoffmann, Robert and Alfonso Valencia (2002). “A gene network for navigating the literature”. In: *Phys. Rev. E Stat. Nonlin. Soft Matter Phys* 65, p. 065102.

Hooker, John N and Greger Ottosson (2003). “Logic-based Benders decomposition”. In: *Mathematical Programming* 96.1, pp. 33–60.

Hosur, Raghavendra et al. (2011). “iWRAP: an interface threading approach with application to prediction of cancer-related protein–protein interactions”. In: *Journal of molecular biology* 405.5, pp. 1295–1310.

Hsing, Michael, Kendall Grant Byler, and Artem Cherkasov (2008). “The use of Gene Ontology terms for predicting highly-connected’hub’nodes in protein-protein interaction networks”. In: *BMC systems biology* 2.1, p. 80.

Huang, Sui et al. (2005). “Cell fates as high-dimensional attractor states of a complex gene regulatory network”. In: *Physical review letters* 94.12, p. 128701.

Hubner, Nina C et al. (2010). “Quantitative proteomics combined with BAC TransgeneOmics reveals in vivo protein interactions”. In: *The Journal of cell biology* 189.4, pp. 739–754.

Huynen, Martijn et al. (2000). “Predicting protein function by genomic context: quantitative evaluation and qualitative inferences”. In: *Genome research* 10.8, pp. 1204–1210.

Hwang, Hee-Su, Siriwat Visoldilokpun, and Jay M Rosenberger (2008). “A branch-and-price-and-cut method for ship scheduling with limited risk”. In: *Transportation science* 42.3, pp. 336–351.

Hwang, Yih-Chii et al. (2009). “Predicting essential genes based on network and sequence analysis”. In: *Molecular BioSystems* 5.12, pp. 1672–1678.

Ibaraki, Toshihide (1976). “Theoretical comparisons of search strategies in branch-and-bound algorithms”. In: *International Journal of Computer & Information Sciences* 5.4, pp. 315–344.

Ibaraki, Toshihide (1977). “The power of dominance relations in branch-and-bound algorithms”. In: *Journal of the ACM (JACM)* 24.2, pp. 264–279.

Ideker, Trey et al. (2001). “Integrated genomic and proteomic analyses of a systematically perturbed metabolic network”. In: *Science* 292.5518, pp. 929–934.

Ideker, Trey et al. (2002). “Discovering regulatory and signalling circuits in molecular interaction networks”. In: *Bioinformatics* 18.suppl_1, S233–S240.

Irnich, Stefan and Guy Desaulniers (2005). “Shortest path problems with resource constraints”. In: *Column generation*. Springer, pp. 33–65.

Ishitsuka, Masayuki, Tatsuya Akutsu, and Jose C Nacher (2016). “Critical controllability in proteome-wide protein interaction network integrating transcriptome”. In: *Scientific reports* 6.

Isserlin, Ruth, Rashad A El-Badrawi, and Gary D Bader (2011). “The biomolecular interaction network database in PSI-MI 2.5”. In: *Database* 2011.

Ito, Takashi et al. (2001). “A comprehensive two-hybrid analysis to explore the yeast protein interactome”. In: *Proceedings of the National Academy of Sciences* 98.8, pp. 4569–4574.

Jaillet, Patrick (1985). “Probabilistic traveling salesman problems”. PhD thesis. Massachusetts Institute of Technology.

Jaillet, Patrick and Michael R Wagner (2008). “Online vehicle routing problems: A survey”. In: *The Vehicle Routing Problem: Latest Advances and New Challenges*. Springer, pp. 221–237.

Jain, Vipul and Ignacio E Grossmann (2001). “Algorithms for hybrid MILP/CP models for a class of optimization problems”. In: *INFORMS Journal on computing* 13.4, pp. 258–276.

James, Philip, John Halladay, and Elizabeth A Craig (1996). "Genomic libraries and a host strain designed for highly efficient two-hybrid selection in yeast". In: *Genetics* 144.4, pp. 1425–1436.

Jansen, Ronald, Dov Greenbaum, and Mark Gerstein (2002). "Relating whole-genome expression data with protein-protein interactions". In: *Genome research* 12.1, pp. 37–46.

Jansen, Ronald et al. (2003). "A Bayesian networks approach for predicting protein-protein interactions from genomic data". In: *science* 302.5644, pp. 449–453.

Jayapandian, Magesh et al. (2006). "Michigan Molecular Interactions (MiMI): putting the jigsaw puzzle together". In: *Nucleic acids research* 35.suppl_1, pp. D566–D571.

Jeong, H et al. (2001). "Lethality and centrality in protein networks". In: *Nature* 411.6833, p. 41.

Jeong, Hawoong et al. (2000). "The large-scale organization of metabolic networks". In: *arXiv preprint cond-mat/0010278*.

Jiang, Peng and Mona Singh (2010). "SPICi: a fast clustering algorithm for large biological networks". In: *Bioinformatics* 26.8, pp. 1105–1111.

Jiinger, Michael, Gerhard Reinelt, and Stefan Thienel (1995). "Practical problem solving with cutting plane algorithms in combinatorial optimization". In: *Combinatorial Optimization, Dimacs* 20, pp. 111–152.

Jin, Ruoming et al. (2007). *Identify dynamic network modules with temporal and spatial constraints*. Tech. rep. Lawrence Livermore National Laboratory (LLNL), Livermore, CA.

Jordan, I King et al. (2002). "Essential genes are more evolutionarily conserved than are nonessential genes in bacteria". In: *Genome research* 12.6, pp. 962–968.

Joy, Maliackal Poulo et al. (2005). "High-betweenness proteins in the yeast protein interaction network". In: *BioMed Research International* 2005.2, pp. 96–103.

Jula, Hossein, Maged Dessouky, and Petros A Ioannou (2006). "Truck route planning in nonstationary stochastic networks with time windows at customer locations". In: *IEEE Transactions on Intelligent Transportation Systems* 7.1, pp. 51–62.

Junker, Björn H and Falk Schreiber (2011). *Analysis of biological networks*. Vol. 2. John Wiley & Sons.

Kalathur, Ravi Kiran Reddy et al. (2013). "UniHI 7: an enhanced database for retrieval and interactive analysis of human molecular interaction networks". In: *Nucleic acids research* 42.D1, pp. D408–D414.

Kamath, Ravi S et al. (2003). "Systematic functional analysis of the Caenorhabditis elegans genome using RNAi". In: *Nature* 421.6920, p. 231.

Kamburov, Atanas et al. (2012a). "Cluster-based assessment of protein-protein interaction confidence". In: *BMC bioinformatics* 13.1, p. 262.

Kamburov, Atanas, Ulrich Stelzl, and Ralf Herwig (2012b). "IntScore: a web tool for confidence scoring of biological interactions". In: *Nucleic acids research* 40.W1, W140–W146.

Kao, Gio K, Edward C Sewell, and Sheldon H Jacobson (2009). "A Branch, Bound, and remember algorithm for the 1| r i*lsum* t i scheduling problem". In: *Journal of Scheduling* 12.2, p. 163.

Karaoz, Ulas et al. (2004). "Whole-genome annotation by using evidence integration in functional-linkage networks". In: *Proceedings of the National Academy of Sciences of the United States of America* 101.9, pp. 2888–2893.

Karp, Richard M (1972). "Reducibility among combinatorial problems". In: *Complexity of computer computations*. Springer, pp. 85–103.

Kenley, Edward Casey and Young-Rae Cho (2011). "Detecting protein complexes and functional modules from protein interaction networks: A graph entropy approach". In: *Proteomics* 11.19, pp. 3835–3844.

Kenyon, Astrid S and David P Morton (2003). "Stochastic vehicle routing with random travel times". In: *Transportation Science* 37.1, pp. 69–82.

Kerrien, Samuel et al. (2011). "The IntAct molecular interaction database in 2012". In: *Nucleic acids research* 40.D1, pp. D841–D846.

Keshava Prasad, TS et al. (2008). "Human protein reference database—2009 update". In: *Nucleic acids research* 37.suppl_1, pp. D767–D772.

Keskin, Ozlem, Nurcan Tuncbag, and Attila Gursoy (2016). “Predicting protein–protein interactions from the molecular to the proteome level”. In: *Chemical reviews* 116.8, pp. 4884–4909.

Kim, Jongkwang and Kai Tan (2010). “Discover protein complexes in protein-protein interaction networks using parametric local modularity”. In: *BMC bioinformatics* 11.1, p. 521.

King, Andrew D, N Pržulj, and Igor Jurisica (2004). “Protein complex prediction via cost-based clustering”. In: *Bioinformatics* 20.17, pp. 3013–3020.

Kolesar, Peter J (1967). “A branch and bound algorithm for the knapsack problem”. In: *Management science* 13.9, pp. 723–735.

Komurov, Kakajan and Michael White (2007). “Revealing static and dynamic modular architecture of the eukaryotic protein interaction network”. In: *Molecular systems biology* 3.1, p. 110.

Kondrashov, Fyodor A, Aleksey Y Ogurtsov, and Alexey S Kondrashov (2004). “Bioinformatical assay of human gene morbidity”. In: *Nucleic acids research* 32.5, pp. 1731–1737.

Korf, Richard E (1985). “Depth-first iterative-deepening: An optimal admissible tree search”. In: *Artificial intelligence* 27.1, pp. 97–109.

Kritikos, George D et al. (2011). “Noise reduction in protein-protein interaction graphs by the implementation of a novel weighting scheme”. In: *BMC bioinformatics* 12.1, p. 239.

Krogan, Nevan J et al. (2006). “Global landscape of protein complexes in the yeast Saccharomyces cerevisiae”. In: *Nature* 440.7084, p. 637.

Kumar, Anuj and Michael Snyder (2002). “Proteomics: Protein complexes take the bait”. In: *Nature* 415.6868, pp. 123–124.

Kumar, Vipin (1992). “Algorithms for constraint-satisfaction problems: A survey”. In: *AI magazine* 13.1, pp. 32–32.

Land, AH and AG Doig (1960). “An automatic method of solving discrete programming problems. Econometrica. v28”. In:

Laporte, Gilbert (1992). “The vehicle routing problem: An overview of exact and approximate algorithms”. In: *European journal of operational research* 59.3, pp. 345–358.

Laporte, Gilbert, Francois V Louveaux, and Hélene Mercure (1994). “A priori optimization of the probabilistic traveling salesman problem”. In: *Operations research* 42.3, pp. 543–549.

Laporte, Gilbert, François V Louveaux, and Luc Van Hamme (2002). “An integer L-shaped algorithm for the capacitated vehicle routing problem with stochastic demands”. In: *Operations Research* 50.3, pp. 415–423.

Laporte, Gilbert and Inmaculada Rodríguez Martín (2007). “Locating a cycle in a transportation or a telecommunications network”. In: *Networks: An International Journal* 50.1, pp. 92–108.

Larsen, Allan, Oli BG Madsen, and Marius M Solomon (2008). “Recent developments in dynamic vehicle routing systems”. In: *The vehicle routing problem: Latest advances and new challenges*. Springer, pp. 199–218.

Lasdon, Leon S (2002). *Optimization theory for large systems*. Courier Corporation.

Lawler, Eugene L and David E Wood (1966). “Branch-and-bound methods: A survey”. In: *Operations research* 14.4, pp. 699–719.

Lee, Sheng-An et al. (2008). “Ortholog-based protein-protein interaction prediction and its application to inter-species interactions”. In: *BMC bioinformatics* 9.12, S11.

Lei, Hongtao, Gilbert Laporte, and Bo Guo (2012). “A generalized variable neighborhood search heuristic for the capacitated vehicle routing problem with stochastic service times”. In: *Top* 20.1, pp. 99–118.

Leung, Henry CM et al. (2009). “Predicting protein complexes from PPI data: a core-attachment approach”. In: *Journal of Computational Biology* 16.2, pp. 133–144.

Levy, Emmanuel D and Jose B Pereira-Leal (2008). “Evolution and dynamics of protein interactions and networks”. In: *Current opinion in structural biology* 18.3, pp. 349–357.

Li, Ai and Steve Horvath (2006a). “Network neighborhood analysis with the multi-node topological overlap measure”. In: *Bioinformatics* 23.2, pp. 222–231.

Li, Gaoshi et al. (2016a). “Predicting essential proteins based on subcellular localization, orthology and PPI networks”. In: *BMC bioinformatics* 17.8, p. 279.

Li, Haiquan, Jinyan Li, and Limsoon Wong (2006b). “Discovering motif pairs at interaction sites from protein sequences on a proteome-wide scale”. In: *Bioinformatics* 22.8, pp. 989–996.

Li, Min, Jianxin Wang, and Jian’er Chen (2008a). “A fast agglomerate algorithm for mining functional modules in protein interaction networks”. In: *BioMedical Engineering and Informatics, 2008. BMEI 2008. International Conference on*. Vol. 1. IEEE, pp. 3–7.

Li, Min et al. (2008b). “Modifying the DPClus algorithm for identifying protein complexes based on new topological structures”. In: *BMC bioinformatics* 9.1, p. 398.

Li, Min et al. (2009). “Hierarchical organization of functional modules in weighted protein interaction networks using clustering coefficient”. In: *International Symposium on Bioinformatics Research and Applications*. Springer, pp. 75–86.

Li, Min et al. (2012a). “A new essential protein discovery method based on the integration of protein-protein interaction and gene expression data”. In: *BMC systems biology* 6.1, p. 15.

Li, Min et al. (2012b). “Towards the identification of protein complexes and functional modules by integrating PPI network and gene expression data”. In: *BMC bioinformatics* 13.1, p. 109.

Li, Min et al. (2013). “Identification of essential proteins from weighted protein–protein interaction networks”. In: *Journal of bioinformatics and computational biology* 11.03, p. 1341002.

Li, Min et al. (2014). “Effective identification of essential proteins based on priori knowledge, network topology and gene expressions”. In: *Methods* 67.3, pp. 325–333.

Li, Min et al. (2015). “A topology potential-based method for identifying essential proteins from PPI networks”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 12.2, pp. 372–383.

Li, Min et al. (2016b). “A reliable neighbor-based method for identifying essential proteins by integrating gene expressions, orthology, and subcellular localization information”. In: *Tsinghua Science and Technology* 21.6, pp. 668–677.

Li, Min et al. (2017). “United complex centrality for identification of essential proteins from PPI networks”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 14.2, pp. 370–380.

Li, Xiao-Li et al. (2005). “Interaction graph mining for protein complexes using local clique merging”. In: *Genome Informatics* 16.2, pp. 260–269.

Li, Xiao-Li, Chuan-Sheng Foo, and See-Kiong Ng (2007). “Discovering protein complexes in dense reliable neighborhoods of protein interaction networks”. In: *Comput Syst Bioinformatics Conf.* Vol. 6, pp. 157–168.

Li, Xiaoli et al. (2010). “Computational approaches for detecting protein complexes from protein interaction networks: a survey”. In: *BMC genomics* 11.1, S3.

Li, Zhenping et al. (2008c). “Quantitative function for community detection”. In: *Physical review E* 77.3, p. 036109.

Lian, Hao, Chengsen Song, and Young-Rae Cho (2010). “Decomposing protein interactome networks by graph entropy”. In: *Bioinformatics and Biomedicine (BIBM), 2010 IEEE International Conference on*. IEEE, pp. 585–589.

Liang, Han and Wen-Hsiung Li (2007). “Gene essentiality, gene duplicability and protein connectivity in human and mouse”. In: *Trends in Genetics* 23.8, pp. 375–378.

Licata, Luana et al. (2011). “MINT, the molecular interaction database: 2012 update”. In: *Nucleic acids research* 40.D1, pp. D857–D861.

Lichtenberg, Ulrik de et al. (2005). “Dynamic complex formation during the yeast cell cycle”. In: *science* 307.5710, pp. 724–727.

Lin, Chen-Ching et al. (2009). “Essential core of protein- protein interaction network in Escherichia coli”. In: *Journal of proteome research* 8.4, pp. 1925–1931.

Lin, Chen-Ching et al. (2010). “Dynamic functional modules in co-expressed protein interaction networks of dilated cardiomyopathy”. In: *BMC systems biology* 4.1, p. 138.

Lin, Chi et al. (2016). “MREA: a minimum resource expenditure node capture attack in wireless sensor networks”. In: *Security and Communication Networks* 9.18, pp. 5502–5517.

Lin, Tzu-Wen, Jian-Wei Wu, and Darby Tien-Hao Chang (2013). “Combining phylogenetic profiling-based and machine learning-based techniques to predict functional related proteins”. In: *PloS one* 8.9, e75940.

Linderoth, Jeff T and Martin WP Savelsbergh (1999). “A computational study of search strategies for mixed integer programming”. In: *INFORMS Journal on Computing* 11.2, pp. 173–187.

Liu, Guimei et al. (2004). “Efficient mining of frequent patterns using ascending frequency ordered prefix-tree”. In: *Data Mining and Knowledge Discovery* 9.3, pp. 249–274.

Liu, Guimei, Jinyan Li, and Limsoon Wong (2008). “Assessing and predicting protein interactions using both local and global network topological metrics”. In: *Genome Informatics* 21, pp. 138–149.

Liu, Guimei, Limsoon Wong, and Hon Nian Chua (2009). “Complex discovery from weighted PPI networks”. In: *Bioinformatics* 25.15, pp. 1891–1897.

Liu, Yin, Nianjun Liu, and Hongyu Zhao (2005). “Inferring protein–protein interactions through high-throughput interaction data from diverse organisms”. In: *Bioinformatics* 21.15, pp. 3279–3285.

Lodree Jr, Emmett J, Kandace N Ballard, and Chang H Song (2012). “Pre-positioning hurricane supplies in a commercial supply chain”. In: *Socio-Economic Planning Sciences* 46.4, pp. 291–305.

Lord, Phillip W. et al. (2003). “Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation”. In: *Bioinformatics* 19.10, pp. 1275–1283.

Lovász, László and Alexander Schrijver (1991). “Cones of matrices and set-functions and 0–1 optimization”. In: *SIAM journal on optimization* 1.2, pp. 166–190.

Lu, Hongchao et al. (2006). “Integrated analysis of multiple data sources reveals modular structure of biological networks”. In: *Biochemical and biophysical research communications* 345.1, pp. 302–309.

Lu, Long, Hui Lu, and Jeffrey Skolnick (2002). “MULTIPROSPECTOR: an algorithm for the prediction of protein–protein interactions by multimeric threading”. In: *Proteins: Structure, Function, and Bioinformatics* 49.3, pp. 350–364.

Lu, Xin et al. (2007). “Hubs in biological interaction networks exhibit low changes in expression in experimental asthma”. In: *Molecular systems biology* 3.1, p. 98.

Lu, Yao et al. (2014). “Predicting essential genes for identifying potential drug targets in Aspergillus fumigatus”. In: *Computational biology and chemistry* 50, pp. 29–40.

Lübbecke, Marco E and Jacques Desrosiers (2005). “Selected topics in column generation”. In: *Operations research* 53.6, pp. 1007–1023.

Lubovac, Zelmina, Jonas Gamalielsson, and Björn Olsson (2006). “Combining functional and topological properties to identify core modules in protein interaction networks”. In: *Proteins: Structure, Function, and Bioinformatics* 64.4, pp. 948–959.

Luo, Fei, Juan Liu, and Jinyan Li (2010a). “Discovering conditional co-regulated protein complexes by integrating diverse data sources”. In: *BMC systems biology* 4.2, S4.

Luo, Feng et al. (2006). “Modular organization of protein interaction networks”. In: *Bioinformatics* 23.2, pp. 207–214.

Luo, Hao et al. (2013). “DEG 10, an update of the database of essential genes that includes both protein-coding genes and noncoding genomic elements”. In: *Nucleic acids research* 42.D1, pp. D574–D580.

Luo, Qibin et al. (2010b). “DIMA 3.0: domain interaction map”. In: *Nucleic acids research* 39.suppl_1, pp. D724–D729.

MacBeath, Gavin and Stuart L Schreiber (2000). “Printing proteins as microarrays for high-throughput function determination”. In: *Science* 289.5485, pp. 1760–1763.

Malaguti, Enrico, Michele Monaci, and Paolo Toth (2011). “An exact approach for the vertex coloring problem”. In: *Discrete Optimization* 8.2, pp. 174–190.

Maraziotis, Ioannis A, Konstantina Dimitrakopoulou, and Anastasios Bezerianos (2007). “Growing functional modules from a seed protein via integration of protein interaction and gene expression data”. In: *Bmc Bioinformatics* 8.1, p. 408.

Marchand, Hugues (1998). “A polyhedral study of the mixed knapsack set and its use to solve mixed integer programs”. PhD thesis. UCL-Université Catholique de Louvain.

Marchand, Hugues and Laurence A Wolsey (2001). “Aggregation and mixed integer rounding to solve MIPs”. In: *Operations research* 49.3, pp. 363–371.

Marchand, Hugues et al. (2002). “Cutting planes in integer and mixed integer programming”. In: *Discrete Applied Mathematics* 123.1-3, pp. 397–446.

Marcotte, Edward M et al. (1999a). “A combined algorithm for genome-wide prediction of protein function”. In: *Nature* 402.6757, p. 83.

Marcotte, Edward M et al. (1999b). “Detecting protein function and protein-protein interactions from genome sequences”. In: *Science* 285.5428, pp. 751–753.

Margot, François (2002). “Pruning by isomorphism in branch-and-cut”. In: *Mathematical Programming* 94.1, pp. 71–90.

Margot, François (2003). “Exploiting orbits in symmetric ILP”. In: *Mathematical Programming* 98.1-3, pp. 3–21.

Mariano, Rachelle and Stefan Wuchty (2017). “Structure-based prediction of host–pathogen protein interactions”. In: *Current Opinion in Structural Biology* 44, pp. 119–124.

Martin, Alexander and Robert Weismantel (1997). “Contributions to general mixed integer knapsack problems”. In:

McDowall, Mark D, Michelle S Scott, and Geoffrey J Barton (2008). “PIPs: human protein–protein interaction prediction database”. In: *Nucleic acids research* 37.suppl_1, pp. D651–D656.

Mehrotra, Anuj and Michael A Trick (1996). “A column generation approach for graph coloring”. In: *informs Journal on Computing* 8.4, pp. 344–354.

Meseguer, Pedro (1997). “Interleaved depth-first search”. In: *IJCAI*. Vol. 97. Citeseer, pp. 1382–1387.

Mete, Mutlu et al. (2008). “A structural approach for finding functional modules from large biological networks”. In: *Bmc Bioinformatics* 9.9, S19.

Michnick, Stephen W et al. (2011). “Protein-fragment complementation assays for large-scale analysis, functional dissection and dynamic studies of protein–protein interactions in living cells”. In: *Signal Transduction Protocols*, pp. 395–425.

Michnick, Stephen W et al. (2016). “Protein-Fragment Complementation Assays for Large-Scale Analysis, Functional Dissection, and Spatiotemporal Dynamic Studies of Protein–Protein Interactions in Living Cells”. In: *Cold Spring Harbor Protocols* 2016.11, pdb–top083543.

Miller, Clair E, Albert W Tucker, and Richard A Zemlin (1960). “Integer programming formulation of traveling salesman problems”. In: *Journal of the ACM (JACM)* 7.4, pp. 326–329.

Mishra, Gopa R et al. (2006). “Human protein reference database—2006 update”. In: *Nucleic acids research* 34.suppl_1, pp. D411–D414.

Mitchell, John E (2002). “Branch-and-cut algorithms for combinatorial optimization problems”. In: *Handbook of applied optimization* 1, pp. 65–77.

Moresco, James J, Paulo C Carvalho, and John R Yates (2010). “Identifying components of protein complexes in C. elegans using co-immunoprecipitation and mass spectrometry”. In: *Journal of proteomics* 73.11, pp. 2198–2204.

Morrison, David R et al. (2014a). “A wide branching strategy for the graph coloring problem”. In: *INFORMS Journal on Computing* 26.4, pp. 704–717.

Morrison, David R, Edward C Sewell, and Sheldon H Jacobson (2014b). “An application of the branch, bound, and remember algorithm to a new simple assembly line balancing dataset”. In: *European Journal of Operational Research* 236.2, pp. 403–409.

Morrison, David R et al. (2016). “Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning”. In: *Discrete Optimization* 19, pp. 79–102.

Morrison, David R et al. (2017). “Cyclic best first search: Using contours to guide branch-and-bound algorithms”. In: *Naval Research Logistics (NRL)* 64.1, pp. 64–82.

Mosca, Roberto et al. (2013). “3did: a catalog of domain-based interactions of known three-dimensional structure”. In: *Nucleic acids research* 42.D1, pp. D374–D379.

Mrowka, Ralf, Andreas Patzak, and Hanspeter Herzel (2001). “Is there a bias in proteome research?” In: *Genome research* 11.12, pp. 1971–1973.

Mucha, Peter J et al. (2010). “Community structure in time-dependent, multiscale, and multiplex networks”. In: *science* 328.5980, pp. 876–878.

Muff, Stefanie, Francesco Rao, and Amedeo Caflisch (2005). “Local modularity measure for network clusterizations”. In: *Physical Review E* 72.5, p. 056107.

Myers, Chad L et al. (2005). “Discovery of biological networks from diverse functional genomic data”. In: *Genome biology* 6.13, R114.

Narayanan, Tejaswini et al. (2011). “Modularity detection in protein-protein interaction networks”. In: *BMC research notes* 4.1, p. 569.

Nasirian, Farzaneh, Foad Mahdavi Pajouh, and Balabhaskar Balasundaram (2020). “Detecting a most closeness-central clique in complex networks”. In: *European Journal of Operational Research* 283.2, pp. 461–475.

Navlakha, Saket and Carl Kingsford (2010a). “Exploring biological network dynamics with ensembles of graph partitions.” In: *Pacific Symposium on Biocomputing*. Vol. 15, pp. 166–177.

Navlakha, Saket et al. (2010b). “Finding biologically accurate clusterings in hierarchical tree decompositions using the variation of information”. In: *Journal of Computational Biology* 17.3, pp. 503–516.

Nemhauser, George L and Laurence A Wolsey (1990). “A recursive procedure to generate all cuts for 0–1 mixed integer programs”. In: *Mathematical Programming* 46.1-3, pp. 379–390.

Nepusz, Tamás, Haiyuan Yu, and Alberto Paccanaro (2012). “Detecting overlapping protein complexes in protein-protein interaction networks”. In: *Nature methods* 9.5, pp. 471–472.

Newman, Mark EJ and Michelle Girvan (2004). “Finding and evaluating community structure in networks”. In: *Physical review E* 69.2, p. 026113.

Newman, MEJ (2016). “Community detection in networks: Modularity optimization and maximum likelihood are equivalent”. In: *arXiv preprint arXiv:1606.02319*.

Ning, Kang et al. (2010). “Examination of the relationship between essential genes in PPI network and hub proteins in reverse nearest neighbor topology”. In: *BMC bioinformatics* 11.1, p. 505.

Öncan, Temel, İ Kuban Altınel, and Gilbert Laporte (2009). “A comparative analysis of several asymmetric traveling salesman problem formulations”. In: *Computers & Operations Research* 36.3, pp. 637–654.

Orchard, Sandra et al. (2013). “The MIntAct project—IntAct as a common curation platform for 11 molecular interaction databases”. In: *Nucleic acids research* 42.D1, pp. D358–D363.

Ostrowski, James et al. (2011). “Orbital branching”. In: *Mathematical Programming* 126.1, pp. 147–178.

Oughtred, Rose et al. (2016). “BioGRID: a resource for studying biological interactions in yeast”. In: *Cold Spring Harbor Protocols* 2016.1, pdb–top080754.

Padberg, Manfred and Giovanni Rinaldi (1991). “A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems”. In: *SIAM review* 33.1, pp. 60–100.

Padberg, Manfred W, Tony J Van Roy, and Laurence A Wolsey (1985). “Valid linear inequalities for fixed charge problems”. In: *Operations Research* 33.4, pp. 842–861.

Pagel, Philipp, Philip Wong, and Dmitrij Frishman (2004a). “A domain interaction map based on phylogenetic profiling”. In: *Journal of molecular biology* 344.5, pp. 1331–1346.

Pagel, Philipp et al. (2004b). “The MIPS mammalian protein–protein interaction database”. In: *Bioinformatics* 21.6, pp. 832–834.

Pagel, Philipp et al. (2007). “DIMA 2.0—predicted and known domain interactions”. In: *Nucleic acids research* 36.suppl_1, pp. D651–D655.

Papadimitriou, Christos H and Kenneth Steiglitz (1998). *Combinatorial optimization: algorithms and complexity*. Courier Corporation.

Pattillo, Jeffrey, Nataly Youssef, and Sergiy Butenko (2013). “On clique relaxation models in network analysis”. In: *European Journal of Operational Research* 226.1, pp. 9–18.

Pazos, Florencio et al. (1997). “Correlated mutations contain information about protein-protein interaction”. In: *Journal of molecular biology* 271.4, pp. 511–523.

Pazos, Florencio and Alfonso Valencia (2001). “Similarity of phylogenetic trees as indicator of protein–protein interaction”. In: *Protein engineering* 14.9, pp. 609–614.

Pazos, Florencio and Alfonso Valencia (2002). “In silico two-hybrid system for the selection of physically interacting protein pairs”. In: *Proteins: Structure, Function, and Bioinformatics* 47.2, pp. 219–227.

Pazos, Florencio et al. (2005). “Assessing protein co-evolution in the context of the tree of life assists in the prediction of the interactome”. In: *Journal of molecular biology* 352.4, pp. 1002–1015.

Pazos, Florencio and Alfonso Valencia (2008). “Protein co-evolution, co-adaptation and interactions”. In: *The EMBO journal* 27.20, pp. 2648–2655.

Pei, Pengjun and Aidong Zhang (2007). “A “seed-refine” algorithm for detecting protein complexes from protein interaction data”. In: *IEEE Transactions on Nanobioscience* 6.1, pp. 43–50.

Peng, Junmin et al. (2003). “Evaluation of multidimensional chromatography coupled with tandem mass spectrometry (LC/LC- MS/MS) for large-scale protein analysis: the yeast proteome”. In: *Journal of proteome research* 2.1, pp. 43–50.

Peng, Wei et al. (2012). “Iteration method for predicting essential proteins based on orthology and protein-protein interaction networks”. In: *BMC systems biology* 6.1, p. 87.

Peng, Wei et al. (2015). “UDoNC: an algorithm for identifying essential proteins based on protein domains and protein-protein interaction networks”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 12.2, pp. 276–288.

Peng, Xiaoqing et al. (2016). “Protein–protein interactions: detection, reliability assessment and applications”. In: *Briefings in bioinformatics*.

Pereira-Leal, José B et al. (2004a). “An exponential core in the heart of the yeast protein interaction network”. In: *Molecular biology and evolution* 22.3, pp. 421–425.

Pereira-Leal, Jose B, Anton J Enright, and Christos A Ouzounis (2004b). “Detection of functional modules from protein interaction networks”. In: *PROTEINS: Structure, Function, and Bioinformatics* 54.1, pp. 49–57.

Pereira-Leal, Jose B, Emmanuel D Levy, and Sarah A Teichmann (2006). “The origins and evolution of functional modules: lessons from protein complexes”. In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 361.1467, pp. 507–517.

Peri, Suraj et al. (2003). “Development of human protein reference database as an initial platform for approaching systems biology in humans”. In: *Genome research* 13.10, pp. 2363–2371.

Phan, Dzung T (2012). “Lagrangian duality and branch-and-bound algorithms for optimal power flow”. In: *Operations Research* 60.2, pp. 275–285.

Plaimas, Kitiporn, Roland Eils, and Rainer König (2010). “Identifying essential genes in bacterial metabolic networks with machine learning methods”. In: *BMC systems biology* 4.1, p. 56.

Prieto, Carlos and Javier De Las Rivas (2006). “APID: agile protein interaction DataAnalyzer”. In: *Nucleic acids research* 34.suppl_2, W298–W302.

Przytycka, Teresa M, Mona Singh, and Donna K Slonim (2010). “Toward the dynamic interactome: it’s about time”. In: *Briefings in bioinformatics* 11.1, pp. 15–29.

Puzis, Rami, Yuval Elovici, and Shlomi Dolev (2007a). “Fast algorithm for successive computation of group betweenness centrality”. In: *Phys. Rev. E* 76 (5), p. 056709. DOI: 10.1103/PhysRevE.76.056709. URL: https://link.aps.org/doi/10.1103/PhysRevE.76.056709.

Puzis, Rami, Yuval Elovici, and Shlomi Dolev (2007b). “Finding the most prominent group in complex networks”. In: *AI communications* 20.4, pp. 287–296.

Qi, Yanjun, Judith Klein-Seetharaman, and Ziv Bar-Joseph (2005). “Random forest similarity for protein-protein interaction prediction from multiple sources.” In: *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pp. 531–542.

Qi, Yanjun, Ziv Bar-Joseph, and Judith Klein-Seetharaman (2006). “Evaluation of different biological data and computational classification methods for use in protein interaction prediction”. In: *Proteins: Structure, Function, and Bioinformatics* 63.3, pp. 490–500.

Qin, Chao, Yongqi Sun, and Yadong Dong (2017). “A new computational strategy for identifying essential proteins based on network topological properties and biological information”. In: *PloS one* 12.7, e0182031.

Radicchi, Filippo et al. (2004). “Defining and identifying communities in networks”. In: *Proceedings of the National Academy of Sciences of the United States of America* 101.9, pp. 2658–2663.

Raghavachari, Balaji et al. (2007). “DOMINE: a database of protein domain interactions”. In: *Nucleic acids research* 36.suppl_1, pp. D656–D661.

Ramadan, Emad, Arijit Tarafdar, and Alex Pothen (2004). “A hypergraph model for the yeast protein complex network”. In: *18th International Parallel and Distributed Processing Symposium, 2004. Proceedings.* IEEE, p. 189.

Ramani, Arun K et al. (2008). “A map of human protein interactions derived from co-expression of human mRNAs and their orthologs”. In: *Molecular systems biology* 4.1, p. 180.

Rasti, Saeid and Chrysafis Vogiatzis (2019). “A survey of computational methods in protein–protein interaction networks”. In: *Annals of Operations Research* 276.1-2, pp. 35–87.

Reinelt, Gerhard (1991). “TSPLIB traveling salesman problem library”. In: *ORSA journal on computing* 3.4, pp. 376–384.

Remy, Ingrid and Stephen W Michnick (2015). “Mapping biochemical networks with protein fragment complementation assays”. In: *Protein-Protein Interactions: Methods and Applications*, pp. 467–481.

Ren, Jun et al. (2015). “Discovering essential proteins based on PPI network and protein complex”. In: *International journal of data mining and bioinformatics* 12.1, pp. 24–43.

Rigaut, Guillaume et al. (1999). “A generic protein purification method for protein complex characterization and proteome exploration”. In: *Nature biotechnology* 17.10, pp. 1030–1032.

Ritzinger, Ulrike, Jakob Puchinger, and Richard F Hartl (2016). “A survey on dynamic and stochastic vehicle routing problems”. In: *International Journal of Production Research* 54.1, pp. 215–231.

Rivera, Corban G, Rachit Vakil, and Joel S Bader (2010). “NeMo: network module identification in Cytoscape”. In: *BMC bioinformatics* 11.1, S61.

Rives, Alexander W and Timothy Galitski (2003). “Modular organization of cellular networks”. In: *Proceedings of the National Academy of Sciences* 100.3, pp. 1128–1133.

Rohila, Jai S et al. (2004). “Improved tandem affinity purification tag and methods for isolation of protein heterocomplexes from plants”. In: *The Plant Journal* 38.1, pp. 172–181.

Ropke, Stefan and Jean-François Cordeau (2009). “Branch and cut and price for the pickup and delivery problem with time windows”. In: *Transportation Science* 43.3, pp. 267–286.

Ruan, Jianhua and Weixiong Zhang (2008). “Identifying network communities with a high resolution”. In: *Physical Review E* 77.1, p. 016104.

Rutherford, Suzanne L et al. (2000). “From genotype to phenotype: buffering mechanisms and the storage of genetic information”. In: *Bioessays* 22.12, pp. 1095–1105.

Rysz, Maciej, Foad Mahdavi Pajouh, and Eduardo L Pasiliao (2018). “Finding clique clusters with the highest betweenness centrality”. In: *European Journal of Operational Research* 271.1, pp. 155–164.

Saito, Rintaro, Harukazu Suzuki, and Yoshihide Hayashizaki (2002). “Interaction generality, a measurement to assess the reliability of a protein–protein interaction”. In: *Nucleic acids research* 30.5, pp. 1163–1168.

Saito, Rintaro, Harukazu Suzuki, and Yoshihide Hayashizaki (2003). “Construction of reliable protein–protein interaction networks with a new interaction generality measure”. In: *Bioinformatics* 19.6, pp. 756–763.

Satuluri, Venu, Srinivasan Parthasarathy, and Duygu Ucar (2010). “Markov clustering of protein interaction networks with improved balance and scalability”. In: *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*. ACM, pp. 247–256.

Savelsbergh, Martin (1997). “A branch-and-price algorithm for the generalized assignment problem”. In: *Operations research* 45.6, pp. 831–841.

Scott, Michelle S and Geoffrey J Barton (2007). “Probabilistic prediction and ranking of human protein-protein interactions”. In: *BMC bioinformatics* 8.1, p. 239.

Segal, Eran, Haidong Wang, and Daphne Koller (2003). “Discovering molecular pathways from protein interaction and gene expression data”. In: *Bioinformatics* 19.suppl_1, pp. i264–i272.

Seringhaus, Michael et al. (2006). “Predicting essential genes in fungal genomes”. In: *Genome research* 16.9, pp. 1126–1135.

Sewell, Edward C et al. (2012a). “A BB&R algorithm for minimizing total tardiness on a single machine with sequence dependent setup times”. In: *Journal of Global Optimization* 54.4, pp. 791–812.

Sewell, Edward C and Sheldon H Jacobson (2012b). “A branch, bound, and remember algorithm for the simple assembly line balancing problem”. In: *INFORMS Journal on Computing* 24.3, pp. 433–442.

Sharan, Roded et al. (2005). “Conserved patterns of protein interaction in multiple species”. In: *Proceedings of the National Academy of Sciences of the United States of America* 102.6, pp. 1974–1979.

Sherali, Hanif D and Cihan H Tuncbilek (1992). “A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique”. In: *Journal of Global Optimization* 2.1, pp. 101–112.

Shi, Lei and Aidong Zhang (2010). “Semi-supervised learning protein complexes from protein interaction networks”. In: *Bioinformatics and Biomedicine (BIBM), 2010 IEEE International Conference on*. IEEE, pp. 247–252.

Shih, Yu-Keng and Srinivasan Parthasarathy (2012). “Identifying functional modules in interaction networks through overlapping Markov clustering”. In: *Bioinformatics* 28.18, pp. i473–i479.

Shoemaker, Benjamin A and Anna R Panchenko (2007). “Deciphering protein–protein interactions. Part I. Experimental techniques and databases”. In: *PLoS computational biology* 3.3, e42.

Sidhu, Sachdev S, Wayne J Fairbrother, and Kurt Deshayes (2003). “Exploring protein–protein interactions with phage display”. In: *Chembiochem* 4.1, pp. 14–25.

Sidhu, Sachdev S and Shohei Koide (2007). “Phage display for engineering and analyzing protein interaction interfaces”. In: *Current opinion in structural biology* 17.4, pp. 481–487.

Silva, Joao Paulo Müller da et al. (2008). “In silico network topology-based prediction of gene essentiality”. In: *Physica A: Statistical Mechanics and its Applications* 387.4, pp. 1049–1055.

Slate, David J and Lawrence R Atkin (1983). “Chess 4.5–the Northwestern University chess program”. In: *Chess skill in Man and Machine*. Springer, pp. 82–118.

Snel, Berend et al. (2000). “STRING: a web-server to retrieve and display the repeatedly occurring neighbourhood of a gene”. In: *Nucleic acids research* 28.18, pp. 3442–3444.

Snel, Berend, Peer Bork, and Martijn A Huynen (2002). “The identification of functional modules from the genomic association of genes”. In: *Proceedings of the National Academy of Sciences* 99.9, pp. 5890–5895.

Song, Jimin and Mona Singh (2013). “From hub proteins to hub modules: the relationship between essentiality and centrality in the yeast interactome at different scales of organization”. In: *PLoS computational biology* 9.2, e1002910.

Spielman, Daniel A and Shang-Hua Teng (2008). “A local clustering algorithm for massive graphs and its application to nearly-linear time graph partitioning”. In: *arXiv preprint arXiv:0809.3232*.

Spirin, Victor and Leonid A Mirny (2003). “Protein complexes and functional modules in molecular networks”. In: *Proceedings of the National Academy of Sciences* 100.21, pp. 12123–12128.

Sprinzak, Einat and Hanah Margalit (2001). “Correlated sequence-signatures as markers of protein-protein interaction”. In: *Journal of molecular biology* 311.4, pp. 681–692.

Sprinzak, Einat, Shmuel Sattath, and Hanah Margalit (2003). “How reliable are experimental protein–protein interaction data?” In: *Journal of molecular biology* 327.5, pp. 919–923.

Srihari, Sriganesh, Kang Ning, and HonWai Leong (2009). “Refining Markov Clustering for protein complex prediction by incorporating core-attachment structure”. In: *Genome Informatics* 23.1, pp. 159–168.

Srinivas, K et al. (2008). “Methodology for phylogenetic tree construction”. In: *Journal of Proteomics & Bioinformatics* 1, S005–S011.

Stark, Chris et al. (2006). “BioGRID: a general repository for interaction datasets”. In: *Nucleic acids research* 34.suppl_1, pp. D535–D539.

Stein, Amelie, Arnaud Céol, and Patrick Aloy (2010). “3did: identification and classification of domain-based interactions of known three-dimensional structure”. In: *Nucleic acids research* 39.suppl_1, pp. D718–D723.

Stephenson, Karen and Marvin Zelen (1989). “Rethinking centrality: Methods and examples”. In: *Social networks* 11.1, pp. 1–37.

Sun, Siqi et al. (2011). “An iterative network partition algorithm for accurate identification of dense network modules”. In: *Nucleic acids research* 40.3, e18–e18.

Szklarczyk, Damian et al. (2014). “STRING v10: protein–protein interaction networks, integrated over the tree of life”. In: *Nucleic acids research* 43.D1, pp. D447–D452.

Szklarczyk, Damian et al. (2017). “The STRING database in 2017: quality-controlled protein–protein association networks, made broadly accessible”. In: *Nucleic acids research* 45.D1, pp. D362–D368.

Tague, Patrick and Radha Poovendran (2007). “Modeling adaptive node capture attacks in multi-hop wireless networks”. In: *Ad Hoc Networks* 5.6, pp. 801–814.

Tan, Powell Patrick Cheng, Daryanaz Dargahi, and Frederic Pio (2010). “Predicting protein complexes by data integration of different types of interactions”. In: *International journal of computational biology and drug design* 3.1, pp. 19–30.

Tanay, Amos et al. (2004). “Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data”. In: *Proceedings of the National Academy of Sciences of the United States of America* 101.9, pp. 2981–2986.

Tang, Xiwei et al. (2011). “A comparison of the functional modules identified from time course and static PPI network data”. In: *BMC bioinformatics* 12.1, p. 339.

Tang, Xiwei et al. (2014). “Predicting essential proteins based on weighted degree centrality”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)* 11.2, pp. 407–418.

Tarcea, V Glenn et al. (2008). “Michigan molecular interactions r2: from interacting proteins to pathways”. In: *Nucleic acids research* 37.suppl_1, pp. D642–D646.

Tarjan, Robert (1972). “Depth-first search and linear graph algorithms”. In: *SIAM journal on computing* 1.2, pp. 146–160.

Taş, Duygu et al. (2013). “Vehicle routing problem with stochastic travel times including soft time windows and service costs”. In: *Computers & Operations Research* 40.1, pp. 214–224.

Taylor, Ian W et al. (2009). “Dynamic modularity in protein interaction networks predicts breast cancer outcome”. In: *Nature biotechnology* 27.2, pp. 199–204.

Templin, Markus F et al. (2002). “Protein microarray technology”. In: *Drug discovery today* 7.15, pp. 815–822.

Terentiev, AA, NT Moldogazieva, and KV Shaitan (2009). “Dynamic proteomics in modeling of the living cell. Protein-protein interactions”. In: *Biochemistry (Moscow)* 74.13, pp. 1586–1607.

Thompson, Peter M, Moriah R Beck, and Sharon L Campbell (2015). “Protein-protein interaction analysis by nuclear magnetic resonance spectroscopy”. In: *Protein-Protein Interactions: Methods and Applications*, pp. 267–279.

Tomita, Etsuji, Akira Tanaka, and Haruhisa Takahashi (2006). “The worst-case time complexity for generating all maximal cliques and computational experiments”. In: *Theoretical computer science* 363.1, pp. 28–42.

Tong, Amy Hin Yan et al. (2001). “Systematic genetic analysis with ordered arrays of yeast deletion mutants”. In: *Science* 294.5550, pp. 2364–2368.

Tong, Amy Hin Yan et al. (2002). “A combined experimental and computational strategy to define protein interaction networks for peptide recognition modules”. In: *Science* 295.5553, pp. 321–324.

Tong, Amy Hin Yan et al. (2004). “Global mapping of the yeast genetic interaction network”. In: *science* 303.5659, pp. 808–813.

Toth, Paolo and Daniele Vigo (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

Trinkle-Mulcahy, Laura et al. (2008). “Identifying specific protein interaction partners using quantitative mass spectrometry and bead proteomes”. In: *The Journal of cell biology* 183.2, pp. 223–239.

Tsoka, Sophia and Christos A Ouzounis (2000). “Prediction of protein interactions: metabolic enzymes are frequently involved in gene fusion”. In: *Nature Genetics* 26.2, pp. 141–143.

Uchoa, Eduardo et al. (2008). “Robust branch-cut-and-price for the capacitated minimum spanning tree problem over a large extended formulation”. In: *Mathematical Programming* 112.2, pp. 443–472.

Uetz, Peter et al. (2000). “A comprehensive analysis of protein-protein interactions in Saccharomyces cerevisiae”. In: *Nature* 403.6770, p. 623.

Ulitsky, Igor and Ron Shamir (2007). “Identification of functional modules using network topology and high-throughput data”. In: *BMC systems biology* 1.1, p. 8.

Ulitsky, Igor and Ron Shamir (2009). “Identifying functional modules using expression profiles and confidence-scored protein interactions”. In: *Bioinformatics* 25.9, pp. 1158–1164.

Valente, Guilherme T et al. (2013). “The development of a universal in silico predictor of protein-protein interactions”. In: *PLoS One* 8.5, e65587.

Vanderbeck, François (2011). “Branching in branch-and-price: a generic scheme”. In: *Mathematical Programming* 130.2, pp. 249–294.

Venkatesan, Kavitha et al. (2009). “An empirical framework for binary interactome mapping”. In: *Nature methods* 6.1, pp. 83–90.

Veremyev, Alexander, Oleg A Prokopyev, and Eduardo L Pasiliao (2017). “Finding groups with maximum betweenness centrality”. In: *Optimization Methods and Software* 32.2, pp. 369–399.

Vila, Mariona and Jordi Pereira (2014). “A branch-and-bound algorithm for assembly line worker assignment and balancing problems”. In: *Computers & Operations Research* 44, pp. 105–114.

Voevodski, Konstantin, Shang-Hua Teng, and Yu Xia (2009). “Finding local communities in protein networks”. In: *BMC bioinformatics* 10.1, p. 297.

Vogiatzis, Chrysafis et al. (2015). “An integer programming approach for finding the most and the least central cliques”. In: *Optimization Letters* 9.4, pp. 615–633.

Vogiatzis, Chrysafis and Mustafa Can Camur (2019). “Identification of Essential Proteins Using Induced Stars in Protein–Protein Interaction Networks”. In: *INFORMS Journal on Computing* 31.4, pp. 703–718. DOI: 10.1287/ijoc.2018.0872.

Von Mering, Christian et al. (2002). “Comparative assessment of large-scale data sets of protein-protein interactions”. In: *Nature* 417.6887, p. 399.

Von Mering, Christian et al. (2005). “STRING: known and predicted protein–protein associations, integrated and transferred across organisms”. In: *Nucleic acids research* 33.suppl_1, pp. D433–D437.

Wallace, Mark and Joachim Schimpf (2002). “Finding the right hybrid algorithm–A combinatorial meta-problem”. In: *Annals of Mathematics and Artificial Intelligence* 34.4, pp. 259–269.

Wan, Kyu Kim, Jong Park, and Jung Keun Suh (2002). “Large scale statistical prediction of protein-protein interaction by potentially interacting domain (PID) pair”. In: *Genome Informatics* 13, pp. 42–50.

Wang, Haidong et al. (2009). “A complex-based reconstruction of the Saccharomyces cerevisiae interactome”. In: *Molecular & Cellular Proteomics* 8.6, pp. 1361–1381.

Wang, Haixun et al. (2002). “Clustering by pattern similarity in large data sets”. In: *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*. ACM, pp. 394–405.

Wang, James Z et al. (2007). “A new method to measure the semantic similarity of GO terms”. In: *Bioinformatics* 23.10, pp. 1274–1281.

Wang, Jianxin et al. (2011). “A fast hierarchical clustering algorithm for functional modules discovery in protein interaction networks”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8.3, pp. 607–620.

Wang, Jianxin et al. (2012a). “Identification of essential proteins based on edge clustering coefficient”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 9.4, pp. 1070–1080.

Wang, Jianxin et al. (2013). “Identifying essential proteins based on protein domains in protein-protein interaction networks”. In: *Bioinformatics and Biomedicine (BIBM), 2013 IEEE International Conference on*. IEEE, pp. 133–138.

Wang, Pei, Xinghuo Yu, and Jinhu Lu (2014). “Identification and evolution of structurally dominant nodes in protein-protein interaction networks”. In: *IEEE transactions on biomedical circuits and systems* 8.1, pp. 87–97.

Wang, Yi-Ming, Shi-Hao Chen, and Mango C-T Chao (2012b). “An Efficient Hamiltonian-cycle power-switch routing for MTCMOS designs”. In: *17th Asia and South Pacific Design Automation Conference*. IEEE, pp. 59–65.

Wilson, Allan C, Steven S Carlson, and Thomas J White (1977). “Biochemical evolution”. In: *Annual review of biochemistry* 46.1, pp. 573–639.

Winzeler, Elizabeth A et al. (1999). “Functional characterization of the S. cerevisiae genome by gene deletion and parallel analysis”. In: *science* 285.5429, pp. 901–906.

Wojcik, Jérôme and Vincent Schächter (2001). “Protein-protein interaction map inference using interacting domain profile pairs”. In: *Bioinformatics* 17.suppl_1, S296–S305.

Wolsey, Laurence A and George L Nemhauser (1999). *Integer and combinatorial optimization*. Vol. 55. John Wiley & Sons.

Wu, Jianmin et al. (2009a). “Integrated network analysis platform for protein-protein interactions”. In: *Nature methods* 6.1, pp. 75–77.

Wu, Min et al. (2009b). “A core-attachment based method to detect protein complexes in PPI networks”. In: *BMC bioinformatics* 10.1, p. 169.

Wuchty, Stefan (2002). “Interaction and domain networks of yeast”. In: *Proteomics* 2.12, pp. 1715–1723.

Wuchty, Stefan (2014). “Controllability in protein interaction networks”. In: *Proceedings of the National Academy of Sciences* 111.19, pp. 7156–7160.

Wuchty, Stefan and Peter F Stadler (2003). “Centers of complex networks”. In: *Journal of Theoretical Biology* 223.1, pp. 45–53.

Wuchty, Stefan and Eivind Almaas (2005). “Peeling the yeast protein network”. In: *Proteomics* 5.2, pp. 444–449.

Wuchty, Stefan, Toni Boltz, and Hande Küçük-McGinty (2017). “Links between critical proteins drive the controllability of protein interaction networks”. In: *Proteomics* 17.10.

Xenarios, Ioannis et al. (2000). “DIP: the database of interacting proteins”. In: *Nucleic acids research* 28.1, pp. 289–291.

Xenarios, Ioannis et al. (2002). “DIP, the Database of Interacting Proteins: a research tool for studying cellular networks of protein interactions”. In: *Nucleic acids research* 30.1, pp. 303–305.

Xiao, Qianghua et al. (2015). “Identifying essential proteins from active PPI networks constructed with dynamic gene expression”. In: *BMC genomics* 16.3, S1.

Xiong, Hui et al. (2005). “Identification of functional modules in protein complexes via hyperclique pattern discovery.” In: *Pacific symposium on biocomputing*. Vol. 10, pp. 221–232.

Xu, Bo, Hongfei Lin, and Zhihao Yang (2011). “Ontology integration to identify protein complex in protein interaction networks”. In: *Proteome science* 9.1, S7.

Yan, Yuling and Gerard Marriott (2003). “Analysis of protein interactions using fluorescence technologies”. In: *Current opinion in chemical biology* 7.5, pp. 635–640.

Yang, Yong, Hong Wang, and Dorothy A Erie (2003). “Quantitative characterization of biomolecular assemblies and interactions using atomic force microscopy”. In: *Methods* 29.2, pp. 175–187.

Yu, Haiyuan et al. (2004). “Genomic analysis of essentiality within protein networks”. In: *TRENDS in Genetics* 20.6, pp. 227–231.

Yu, Haiyuan et al. (2007). “The importance of bottlenecks in protein networks: correlation with gene essentiality and expression dynamics”. In: *PLoS computational biology* 3.4, e59.

Yu, Haiyuan et al. (2008). “High-quality binary protein interaction map of the yeast interactome network”. In: *Science* 322.5898, pp. 104–110.

Yu, Liang, Lin Gao, and ChuiLiang Kong (2011). “Identification of core–attachment complexes based on maximal frequent patterns in protein–protein interaction networks”. In: *Proteomics* 11.19, pp. 3826–3834.

Zahiri, Javad, Joseph Hannon Bozorgmehr, and Ali Masoudi-Nejad (2013). “Computational prediction of protein–protein interaction networks: algorithms and resources”. In: *Current genomics* 14.6, pp. 397–414.

Zaki, Nazar, Dmitry Efimov, and Jose Berengueres (2013). “Protein complex detection using interaction reliability assessment and weighted clustering coefficient”. In: *BMC bioinformatics* 14.1, p. 163.

Zhang, Bing et al. (2008a). “From pull-down data to protein interaction networks and complexes with biological relevance”. In: *Bioinformatics* 24.7, pp. 979–986.

Zhang, Qiangfeng Cliff et al. (2012). “Structure-based prediction of protein-protein interactions on a genome-wide scale”. In: *Nature* 490.7421, p. 556.

Zhang, Ren, Hong-Yu Ou, and Chun-Ting Zhang (2004). “DEG: a database of essential genes”. In: *Nucleic acids research* 32.suppl_1, pp. D271–D272.

Zhang, Ren and Yan Lin (2008b). “DEG 5.0, a database of essential genes in both prokaryotes and eukaryotes”. In: *Nucleic acids research* 37.suppl_1, pp. D455–D458.

Zhang, Shihua, Xuemei Ning, and Xiang-Sun Zhang (2006a). “Identification of functional modules in a PPI network by clique percolation clustering”. In: *Computational biology and chemistry* 30.6, pp. 445–451.

Zhang, Shihua et al. (2006b). “Prediction of protein complexes based on protein interaction data and functional annotation data using kernel methods”. In: *LECTURE NOTES IN COMPUTER SCIENCE* 4115, p. 514.

Zhang, Shihua et al. (2010). “Determining modular organization of protein interaction networks by maximizing modularity density”. In: *BMC systems biology* 4.2, S10.

Zhang, Wei et al. (2016a). “A New Method for Identifying Essential Proteins by Measuring Co-Expression and Functional Similarity”. In: *IEEE transactions on nanobioscience* 15.8, pp. 939–945.

Zhang, Xue, Jin Xu, and Wang-xin Xiao (2013). “A new method for the discovery of essential proteins”. In: *PloS one* 8.3, e58763.

Zhang, Xue et al. (2016b). “An ensemble framework for identifying essential proteins”. In: *BMC bioinformatics* 17.1, p. 322.

Zhang, Xue, Marcio Luis Acencio, and Ney Lemke (2016c). “Predicting essential genes and proteins based on machine learning and network topological features: a comprehensive review”. In: *Frontiers in physiology* 7.

Zhang, Yijia et al. (2016d). “A method for predicting protein complex in dynamic PPI networks”. In: *BMC bioinformatics* 17.7, p. 229.

Zhao, Bihai et al. (2014). “Prediction of essential proteins based on overlapping essential modules”. In: *IEEE transactions on nanobioscience* 13.4, pp. 415–424.

Zheng, Huiru, Haiying Wang, and David H Glass (2008). “Integration of genomic data for inferring protein complexes from global protein–protein interaction networks”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38.1, pp. 5–16.

Zhong, Haonan, Foad Mahdavi Pajouh, and Oleg A Prokopyev (2020). “Finding influential groups in networked systems: the most degree-central clique problem”. In: *Omega*, p. 102262.

Zhong, Jiancheng et al. (2013). “Prediction of essential proteins based on gene expression programming”. In: *BMC genomics* 14.4, S7.

Zhong, Jiancheng et al. (2015). “A feature selection method for prediction essential protein”. In: *Tsinghua Science and Technology* 20.5, pp. 491–499.

Zotenko, Elena et al. (2008). “Why do hubs in the yeast protein interaction network tend to be essential: reexamining the connection between the network topology and essentiality”. In: *PLoS computational biology* 4.8, e1000140.