

NEUTRALIZATION OF CONFLICT AREAS USING AN ANT COLONY
HEURISTIC APPROACH

A Paper
Submitted to the Graduate Faculty
of the
North Dakota State University
of Agriculture and Applied Science

By

Pavan Kumar Bapanpally

In Partial Fulfillment of the Requirements
for the Degree of
MASTER OF SCIENCE

Major Department:
Computer Science

October 2010

Fargo, North Dakota

North Dakota State University
Graduate School

Title

NEUTRALIZATION OF CONFLICT AREAS USING

AN ANT COLONY HEURISTIC APPROACH

By

PAVAN BAPANPALLY

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

MASTER OF SCIENCE

North Dakota State University Libraries Addendum

To protect the privacy of individuals associated with the document, signatures have been removed from the digital version of this document.

ABSTRACT

Bapanpally, Pavan Kumar, M.S., Department of Computer Science, College of Science and Mathematics, North Dakota State University, October 2010. Neutralization of Conflict Areas Using an Ant Colony Heuristic Approach. Major Professor: Dr. Kendall Nygard.

In this paper, we present a unique approach to solve the Neutralization of Conflict Areas problem using an Ant Colony Optimization technique. The Neutralization of Conflict Areas is a known problem, and over the years, considerable research has been conducted to find strategies [7] to solve the problem. To effectively deal with this kind of problem, many search algorithms and techniques have been proposed. The Ant Colony Optimization technique has been very successful in solving problems such as the travelling salesman problem, vehicle routing problem, and routing and scheduling problems. The suggested approach to the problem presented here is to find routes for conflict areas and then neutralize the conflict areas. To find routes to conflict areas, we used the concept of a Dynamic Source Routing protocol. To find the shortest paths, we used the Ant Colony Optimization technique. The goal of this paper is to suggest a swarm-based approach to find conflict areas, neutralize conflict areas, and recruit help from other resource units when needed to neutralize conflict areas.

The proposed solution has been implemented in a simulator. We simulate how two different sets of cooperating ants called explorer ants and worker ants, with different operational abilities work together in finding and neutralizing conflict areas and also in getting help from other resources areas. The solution can be visualized using a graphical user interface. The framework that we implement will allow for experimentation with a wide variety of experimental parameters.

ACKNOWLEDGMENTS

I would like to thank Dr. Kendall E. Nygard for his continued support, encouragement, and invaluable advice, without which this work would not have been completed. Special thanks are due to the committee members, Dr. Vasant Ubhaya, Dr. Wei Jin, and Dr. Karl Altenburg, for their support and suggestions. I would like to thank my family and friends who supported and motivated me to accomplish this task.

TABLE OF CONTENTS

ABSTRACT	iii
ACKNOWLEDGMENTS	iv
LIST OF TABLES	viii
LIST OF FIGURES	ix
1. INTRODUCTION	1
1.1. Overview	1
2. RELATED WORK	6
3. PROBLEM STATEMENT AND APPROACH	9
3.1. Problem Statement	9
3.2. Approach	9
3.2.1. Resource Unit	12
3.2.2. Explorer Ants.....	12
3.2.3. Worker Ants	13
3.2.4. Conflict Area	13
3.2.5. Ant Cycle	13
3.2.6. Dynamic Source Routing Protocol.....	14
3.2.7. The Application User Interface (UI).....	17
4. EXTERNAL VIEW	22

4.1. Block Diagram of Ant Search	22
4.2. Output of Ant Simulator with Test Data	28
5. INTERNAL VIEW	35
5.1. Overview of Class Files.....	35
5.1.1. Program.cs	35
5.1.2. ResourceConflictUI.cs	35
5.1.3. Polygon.cs.....	36
5.1.4. Hexagons.cs.....	36
5.1.5. AreaSize.cs	36
6. EXPERIMENTAL RESULTS	38
6.1. Results by Varying Parameter Resource Unit Range	39
6.1.1. Test Case 1	39
6.1.2. Test Case 2.....	39
6.1.3. Test Case 3.....	40
6.2. Results by Varying Parameter Number of Explorer Ants	42
6.2.1. Test Case 4.....	42
6.2.2. Test Case 5.....	42
6.2.3. Test Case 6.....	43
6.3. Results by Varying Parameter Number of Paths.....	45
6.3.1. Test Case 7.....	45

6.3.2. Test Case 8.....	45
6.3.3. Test Case 9.....	46
6.4. Mean, Standard Deviation and Confidence Interval.....	48
6.4.1. Ten Simulations for Position 1.....	48
6.4.2. Ten Simulations for Position 2.....	49
6.4.3. Ten Simulations for Position 3.....	50
6.4.4. Ten Simulations for Position 4.....	51
6.4.5. Ten Simulations for Position 5.....	52
6.4.6. Ten Simulations for Position 6.....	53
6.4.7. Ten Simulations for Position 7.....	54
6.4.8. Ten Simulations for Position 8.....	55
6.4.9. Ten Simulations for Position 9.....	56
6.4.10. Ten Simulations for Position 10.....	57
7. CONCLUSIONS AND FUTURE WORK.....	59
REFERENCES.....	62

LIST OF TABLES

<u>Table</u>	<u>Page</u>
6.1. Performance Results for Test Case 1	39
6.2. Performance Results for Test Case 2	40
6.3. Performance Results for Test Case 3	41
6.4. Performance Results for Test Case 4	42
6.5. Performance Results for Test Case 5	43
6.6. Performance Results for Test Case 6	44
6.7. Performance Results for Test Case 7	45
6.8. Performance Results for Test Case 8	46
6.9. Performance Results for Test Case 9	47
6.10. Performance Results for Position 1	48
6.11. Performance Results for Position 2	50
6.12. Performance Results for Position 3	51
6.13. Performance Results for Position 4	52
6.14. Performance Results for Position 5	53
6.15. Performance Results for Position 6	54
6.16. Performance Results for Position 7	55
6.17. Performance Results for Position 8	56
6.18. Performance Results for Position 9	57
6.19. Performance Results for Position 10	58

LIST OF FIGURES

<u>Figure</u>	<u>Page</u>
3.1. Route Discovery Example	15
3.2. Route Maintenance Example	16
3.3. Graphical Representation of the Ant Simulator	18
3.4. Form to Enter Resource Unit Input.....	19
3.5. Form to Enter Static Conflict Area Severity	20
3.6. Form to Enter Dynamic Conflict Area Severity	20
3.7. Error Message 1	21
3.8. Error Message 2	21
4.1. Block Diagram of Ant Search Algorithm.....	22
4.2. Ant Search Algorithm.....	23
4.3. Pseudo Code for Explorer Ant Search	24
4.4. Pseudo Code for Worker Ant Search.....	26
4.5. Pseudo Code for Explorer Ant Help Search.....	27
4.6. Test Data for Resource Unit.....	29
4.7. Test Data for Static Conflict Area Severity.....	29
4.8. Test Data for Dynamic Conflict Area.....	29
4.9. Graphical Representation of the Initial Load of the Simulator	30
4.10. Graphical Representation of the Movement of Explorer Ants on Grid.....	31
4.11. Graphical Representation of the Pheromone Lay Down by the Explorer Ants	31
4.12. Graphical Representation of the Movement of Worker Ants	32
4.13. Graphical Representation of the Worker Ants Forming Short Paths	32

4.14. Graphical Representation of Two Resource Units Neutralizing Dynamic Conflict Area 1.....	33
4.15. Graphical Representation of Two Resource Units Neutralizing Dynamic Conflict Area 2.....	33
4.16. Graphical Representation of Two Resource Units Neutralizing Dynamic Conflict Area 3.....	34
4.17. Graphical Representation After Neutralizing All Conflict Areas on the Grid.....	34
6.1. Resource Unit Range vs. Total Simulation Time	41
6.2. Number of Explorer Ants vs. Total Simulation Time	44
6.3. Number of Paths Required to Release Worker Ants vs. Total Simulation Time	47
6.4. Position 1, Mean, Standard Deviation, Confidence Interval	49
6.5. Position 2, Mean, Standard Deviation, Confidence Interval	50
6.6. Position 3, Mean, Standard Deviation, Confidence Interval	51
6.7. Position 4, Mean, Standard Deviation, Confidence Interval	52
6.8. Position 5, Mean, Standard Deviation, Confidence Interval	53
6.9. Position 6, Mean, Standard Deviation, Confidence Interval	54
6.10. Position 7, Mean, Standard Deviation, Confidence Interval	55
6.11. Position 8, Mean, Standard Deviation, Confidence Interval.....	56
6.12. Position 9, Mean, Standard Deviation, Confidence Interval.....	57
6.13. Position 10, Mean, Standard Deviation, Confidence Interval.....	58

1. INTRODUCTION

1.1. Overview

This paper focuses on how the swarming behavior of ants can be applied to solving problems like the Neutralization of Conflict Areas. Generally, we see such scenarios in military applications. In these applications, there is a need to describe how the military assets explore area of interest and recognize the conflict areas; upon finding the conflict areas, how assets mobilize available resources to the conflict areas; and how asset gets additional resources from others when needed. Conflict areas may be terrorist cells, military enemy units, or riots which are spread across the terrain [7]. These conflicts may have different levels of severities. Some conflict areas can be neutralized by one military asset, and some conflict areas require more than one military asset to neutralize them. Military assets should be quick enough to find and neutralize conflict areas before these conflict areas increases in severity.

One solution to the kind of problem just described is the natural behavior of ants. The simple behavior of ants has solved many computational problems, such as the travelling salesman problem, vehicle routing problems, and similar scheduling and routing problems. Ants search randomly for a food source starting from the ant colony. Upon finding a food source, they lay down pheromone trails on their way back to the ant colony. Pheromones are chemical substances that are used to communicate information with other ants regarding the paths of food sources among other things [1]. Other ants, which are randomly moving, come across these pheromones, and then follow these trails to find the food. If they find the food source, the newly recruited ants deposit more pheromone along

the same path. This increased pheromone scent attracts more ants, leading to more pheromone deposits along the path. Over the course of time, the shortest path to a particular food source emerges because less pheromone evaporates on the shorter trail.

Ant pheromone has a distinct feature; as ants lay down pheromone from a food source to the ant colony it decays over time. If the distance between the food source and the ant colony is great, the pheromone laid down by one ant may not be persistent enough to attract other ants from the colony. To make the trail persistent over time, the trail has to be perceived by other ants. The path between the food source and ant colony should be short. The pheromone trail leads ants to the food source. While returning back to the ant colony, the ants reinforce the trail by depositing more pheromone along the same path. As more ants follow that path, more pheromones will be deposited.

The characteristic of pheromone evaporation is helpful in forming the shortest path from the food source to the ant colony. Usually, the number of round trips made by an ant from the ant colony to the food source will be greater if the path is a shorter distance; hence, more pheromone will be deposited along the path. Ants which are moving randomly or which are already in some other less-concentrated pheromone trail will be attracted to this higher concentrated pheromone path. Eventually, all the ants follow the shortest path which has more pheromone. In contrast to shorter paths, longer paths need more round trips to deposit more pheromone; longer paths have less chance to attract other ants due to less pheromone because if the pheromone laid by an ant is not perceived by other ants, the pheromone evaporate over time. This pheromone characteristic helps avoid convergence to a local optimal solution. If pheromone does not evaporate over time, even the pheromone

in longer paths would appear to be the shortest path; in this case pheromone leads to the perception of the longest path as shortest path.

Apart from using Ant Colony Optimization [2], we also adapted the concept of Dynamic Source Routing (DSR) protocol. In our case, artificial ants not only have the behavior of real-world ants, but also run in the context of a DSR protocol. In a traditional DSR protocol first instantiates a *route discovery mechanism* to find routes to a destination node. After finding routes to a destination node sends the actual data packets to the destination nodes. Similarly, instead of releasing all the ants in the search area to look for a food source, we will initially release a group of ants to discover the routes; after finding routes, we will release another group of ants to collect the food, and eventually, they travel in the shortest paths using the pheromone concept. Also as in the DSR protocol for *route discovery*, the distance an ant can move away from the ant colony is also restricted to some predefined value. If an ant does not find any food source within the allowed distance the ant come back to the ant colony and starts exploring along another path. This constraint makes sure that paths within the search limit will be covered; after finding routes to food sources, we mobilize another set of ants to collect the food.

We have developed an ant simulator which mimics the characteristics of an ant colony algorithm and Dynamic Source Routing protocol to solve the Neutralization of Conflict Areas problem. In our simulator, we have resource units which have two types of artificial ants: explorer ants and worker ants. We have two different types of conflict areas: static conflict areas and dynamic conflict areas. Initially, explorer ants examine the search space for conflict areas; if they find any conflict areas within the searchable distance, they lay down pheromone on their way back to the colony. When ants reach the resource unit,

the resource unit checks to see whether it has the minimum required paths, as found by explorer ants, to release worker ants to neutralize the conflict areas. If resource unit has the minimum paths, it releases worker ants that neutralize the conflict area at a pre-defined rate. If it does not have the minimum paths, explorer ants keep moving between the resource unit and the conflict area until the resource unit obtains the minimum paths. If, while neutralizing conflict areas, a worker ant learns about a path which has a high density of pheromone, it chooses a new path to travel between the resource unit and the conflict area. If worker ants are unable to neutralize any conflict area, explorer ants that found the conflict area will search for another resource unit for help, starting with the conflict area. The entire process of finding conflict areas, neutralizing them, and getting help will continue until all conflict areas are neutralized in the search area.

The ant simulator can be represented with the following pseudo code:

```
IF (found conflict area) THEN  
    Release pheromone  
    Move to resource unit  
    IF (min. paths found) THEN  
        Release worker ants  
        IF (can neutralize conflict area) THEN  
            Form shortest path  
            Neutralize conflict area  
        ELSE  
            Get help  
            Neutralize conflict area  
    ELSE  
        Move in between resource and conflict area  
ELSE  
    Search randomly for conflict areas
```

The structure of the paper is organized as follows. In Chapter 2, we present a brief review of the literature. In Chapter 3, we describe the problem statement and proposed solution in detail. Chapter 4 gives an External View of the application, and Chapter 5 gives an overview of the Internal View for the application. Chapter 6 presents Experimental Results, and Chapter 7 presents Conclusions and Future Work.

2. RELATED WORK

The natural behavior of ants forming the shortest paths between ant colonies and food sources leads to many algorithms and techniques. Some of the research work is briefly described below.

Dorigo, Birattari and Stutzle [1] tell how the swarm intelligence behavior is used in finding the shortest paths. Swarm intelligence and ant foraging behavior lead to Ant Colony Optimization (ACO). ACO is applicable to the travelling salesman problem (TSP) in which a set of cities are given and the distance between each of them is known. The goal is to find a Hamiltonian tour of minimal length on a fully connected city. This paper tells about the meta-heuristics for ACO and its usefulness in solving combinatorial optimization problems. The main ACO algorithms are Ant System, MAX-MIN Ant system and Ant Colony System. This paper provides insight about hot topics in this field such as dynamic optimization problems.

Dorigo, Maniezzo, and Colomi [2] present a first heuristic algorithm for Ant Colony Optimization which can be used to solve different combinatorial optimization problems. Three different models have been proposed and among them Ant-cycle gives the best results compared to Ant-density and Ant-quantity.

Stutzle and Hooss [3] present an improved version of the basic ant system algorithm. The best ant is allowed to update the trials in every cycle. Algorithm also uses a local search which helps detect of high-quality solutions and guides a learning mechanism more directly.

Parunak, Brueckner, Matthews, and Sauter [4] present an approach to predict the movements of robots using geospatial data. Instantiates a large population of simple

computer agents that explores possible paths through the landscape which will then estimate the likely behavior of the real-world system. The authors talk about ghosts which will wander through the search area. The ghost that successfully finds the target and comes back establishes a path. Some of the ghosts may not return to the base station, i.e., may not come to the base station because of the threats in the search area. Here, the path between the robot and the target is established by the ghosts on an iterative basis.

Lopes, Molles, and Lima [5] have come up with a method for finding the shortest path for the capacitated-vehicle routing problem. In the capacitated vehicle routing problem there are different customers with various demands. The goal is to serve all the customers with a fixed number of vehicles that have different capacities. No customer should be visited more than once (as with a TSP to find the shortest path). The author has proposed a two level solution. In the first level, method deals with the aggregation of the customers (a type of clustering). An ant searches for a set of tours independently and then, in the second level, permutations of customers to find the shortest path. Once again, each tour is submitted to a new population of ants. Effectively, this is Ant Colony Optimization for the TSP.

Johnson, Maltz, and Brochs [6] propose a routing protocol for multi-hop wireless ad hoc networks of mobile nodes. This protocol has two mechanisms: 1) *route discovery* and 2) *route maintenance*. Before sending the actual data packet in the network, the source nodes find the route by sending route requests, a process called *route discovery*. The route request comes back with source route information to a destination. The source node uses the route path for transmitting data to the destination. While transmitting a packet to the destination, the node which is sending this packet is responsible for confirming that the

packet has been received by the next hop along the source route; the packet is retransmitted until this confirmation of receipt is received.

Banga [8] finds the best possible route with the least travel distance and maximum prize. To solve the prize collecting travelling salesman problem, the author makes assumptions such as there is no need to consider a node which is too far from the set of nodes. In TSP, we need to consider all the nodes. First, proposed solution finds the desired node which has the maximum prize and which is closer to most of the nodes. By using a Euclidean TSP theorem, solution finds the optimal travelling length for a given set of nodes, and it uses a threshold in calculating the optimal travelling length. Then, it applies the local optimization techniques to further reduce the tour length. The author came up with four different techniques and compared those techniques. From the four techniques, 2-opt optimization with probabilistic removal of node gives better results.

3. PROBLEM STATEMENT AND APPROACH

The solution proposed in our model is inspired by Ant Colony Optimization (ACO) and Dynamic Source Routing (DSR) protocol. DSR's *route discovery mechanism* and ACO ant pheromone concept are the key concepts for our model. In this paper, we present a unique approach how these concepts are used in solving problems like Neutralization of Conflict Areas effectively.

3.1. Problem Statement

In a given geographical area, resource units should be able to find conflict areas and neutralize them if it can or else get help from other resource units to neutralize them

The problem is to develop an artificial ant system which effectively walks around the grid to find conflict areas. Upon finding conflict areas, the system should be able to neutralize the conflict areas and, as needed, mobilize more resources from other resource units to neutralize the conflict area. The resource unit should neutralize one conflict area at a time. The solution should be able to make the best use of available resources, to find short paths to conflict areas, to obtain help from other resource units through some mechanism, and to neutralize all the conflict areas in the terrain.

3.2. Approach

In almost all traditional applications which utilize the ant colony optimization technique as a search technique use a set of ants to solve problems such as search the grid, find food, collect food and form short paths between the food source and the ant colony. When it comes to solving problems like the Neutralization of Conflict Areas, our traditional ACO technique needs additional functionalities to find the best and shortest

paths as well as to recruit help. Otherwise, without the additional functionality, ACO technique may end up forming longer paths to conflict areas; also, it may take longer to neutralize all conflict areas in the search area.

By considering the constraints above, we proposed a unique approach by which we were able to find short paths to conflict areas, recruit help from other resources, minimize the time required to neutralize conflict areas, and make the best use of all available resources.

In our proposed solution, we have modified the ACO which adapts the Dynamic Source Routing (DSR) protocol concept for ants move around the grid. In ACO and in applications that use ACO, ants which search the grid get lost in the search area while looking for food sources. Even after finding food, they cannot get back to the food source on their own; they depend on pheromone that is laid down by other ants. If ants are unable to return to the ant colony, then some available resources are lost in the search area. To avoid this situation in our ant system, an ant which is in search of conflict areas can travel only some preset distance from the resource unit as in DSR. With a *route discovery mechanism*, data packets can travel a certain time span in search of a destination node. If the packet is unable to find the destination node within that time span, that packet may be terminated. In our case, instead of terminating an ant, we call it back to the resource unit and have it search for another route to the destination node or conflict area. Calling an ant back ensures that we are making the best use of the available resources. After finding routes, the DSR sends actual data packets to the destination node; if the DSR learns of any shorter path while sending actual data packets, it uses that short path. Here in our solution, we adapt the DSR approach as follows: first, we release a set of ants, the explorer ants, to

find routes to conflict areas; then, we release another set of ants, the worker ants, to neutralize the conflict areas. Worker ants are also responsible for finding the shortest path while neutralizing conflict areas. Explorer ants are also responsible for finding routes to resource units to recruit help when there is a need. By adapting DSR features in ACO, we developed an ant simulator which replicates all the above said features.

Because our proposed solution is heuristic in nature, paths found by ants may not be the shortest path or optimal. In our solution, we use two sets of ants to solve the problem unlike a single set of ants trying to solve the problem. If we use single set of ants to solve the problem, we may come across these issues:

- Each ant is responsible to find a path to the conflict area.
- If an ant does not find the conflict area, it travels away from the colony, and chances of getting lost are high.
- For M paths found by N ants, the time to form a short path will be greater.
- The time to find other resource units to get help will be greater.
- All ants may be not be utilized effectively to find routes and form short paths to conflict areas

To overcome single set of ant issues, we have used two sets of ants such as explorer ants and worker ants:

- Each set of ants will have its own objectives, which allows a separation of concerns.
- The explorer ants' primary objective is to find routes to the conflict areas and resource units.

- Instead of wandering forever to find conflict areas, explorer ants only travel a certain distance away from the resource unit and conflict area.
- If an explorer ant does not find a conflict area within a certain distance from the resource unit, the explorer ant goes back and starts searching in different route. Allowing an ant to search within a certain distance makes sure that ants will not get lost in the search area.
- The worker ants' primary objective is to form short path among the available paths to conflict areas.
- Worker ants make sure the short path is formed in least time.

3.2.1. Resource Unit

In a given territory, the resource unit represents a military base or ant colony. Every resource unit has a predefined value of the range which restricts the distance an ant can move from the resource unit and conflict area. Every resource unit has two different types of ants with a predefined number of explorer ants and worker ants. It also has a predefined value for the minimum number of paths required to release worker ants and a worker ant neutralizing rate.

In reality, an ant colony has different castes of ants like “workers,” “soldiers,” “queens,” or other specialized groups. Our proposed solution needs two different types of ants.

3.2.2. Explorer Ants

Explorer ants mimic the behavior of real world ants such as exploring the terrain to find a conflict area and laying down pheromone on their way back to the resource unit from

the conflict area. One of the constraints is that ants can explore terrain to a certain distance which is defined as resource unit range. The ants cannot explore beyond that range. If they are unable to find a conflict area by one route, they go back to the resource unit and start searching again for a conflict area in another route.

3.2.3. Worker Ants

Worker ants neutralize conflict areas along the paths found by explorer ants.

Worker ants are responsible for finding the shortest paths among the available paths. While finding the shortest path, they use the pheromone concept for finding the shortest paths.

3.2.4. Conflict Area

The conflict area is an area where conflict occurs (a food source in the context of an ant colony) and it has a severity value. Conflict areas are divided into two different types:

Static conflict area: this area can be treated as a low severity area, and over the course of time, severity will not increase or decrease. At least one resource unit could neutralize this area.

Dynamic conflict area: this area can be treated as a high severity area, and over the course of time, its severity increases. It needs at least two conflict areas to neutralize it.

3.2.5. Ant Cycle

The evaporation of pheromone is controlled by an ant cycle. The round trip from the resource unit to the conflict area and from the conflict area to the resource unit is called an ant cycle. An ant will update pheromone density while outbound and uses that same pheromone density to return inbound. In the case of explorer ants finding a conflict area, on their way back to the resource unit, they leave a pheromone trail, and, while coming back

to the conflict area, they use the same pheromone trail to reach the conflict area. For every ant cycle, an ant lay downs the same pheromone density. After completing the first ant cycle and at the start of the second cycle, the pheromone trail from the first cycle evaporates. Pheromone can only persist if more ants follow the same path, meaning more pheromone density. We will see this behavior in worker ants when they try to find the shortest path among the available paths found by the explorer ants.

3.2.6. Dynamic Source Routing Protocol

The DSR protocol is a simple and efficient routing protocol designed specifically for use in multi-hop wireless ad hoc networks of mobile nodes. DSR allows the network to be completely self organizing and self-configuring, without the need for any existing network infrastructure or administration. The protocol is composed of the two mechanisms of *route discovery* and *route maintenance*, which work together to allow nodes to discover and maintain source routes to arbitrary destinations in the ad hoc network [6].

Route discovery: In Figure 3.1, source node A wants to send a packet or data to destination node E. In a network source node tries to obtain a route to node E. To get the route to E, it initiates the *route discovery mechanism* by sending out a *route request* message as a single packet to all the nodes which are in transmission range of node A. Along with the route request packet, it includes a unique ID (in Figure 3.1, ID=2). The node which receives this route request checks to see whether it is the destination for that route request or else it re-transmits the route request to surrounding nodes which are in transmission range. In Figure 3.1, node B checks whether it has route information for the destination node. If not, it includes its path information in that route request and re-

transmits the request to other nodes which are in its transmission range, and it discards all the subsequent route requests with the same ID.

The process continues until the packet reaches destination node E. When the packet reaches destination node E, it sends the *route reply* to the source node by simply reversing the node packet as (D, C, B, A). In the *route discovery* process, a route request received by an intermediate node checks its route against the cache to determine whether it has any route to the destination. If it finds any en-route cache, then it sends a route reply to the source node along with route information.

After sending out a route request for a particular destination, source node for a specified time to determine whether it gets any additional route reply messages. If it does not get any reply, it sends another route-request packet to find the destination node. The rate at which a route request is generated is limited by some other mechanisms beyond the scope of this paper.

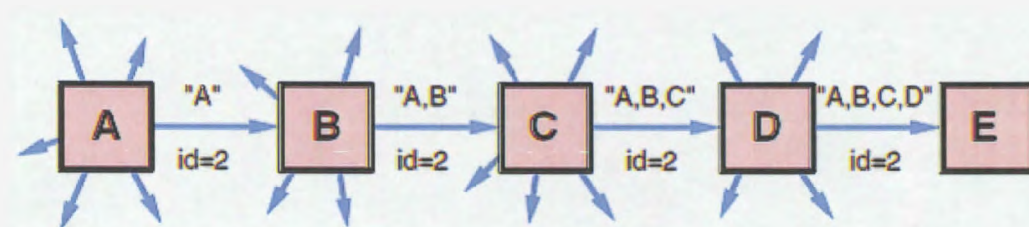


Figure 3.1. Route Discovery Example

Route maintenance: When a node transmits a packet using the source route, the transmitting node is responsible for confirming that the packet has been received by the next node along the source route. The packet is retransmitted until this confirmation of receipt is received.

In figure 3.2, node A sends a data packet for E using the source route through intermediate nodes B, C, and D. In this case, node A is responsible for the receipt of the packet at B, node B is responsible for the receipt of the packet at node C, node C is responsible for the receipt of the packet at node D, and node D is responsible for the receipt finally at the destination E.



Figure 3.2. Route Maintenance Example

In Figure 3.2, node B sends a packet to node C and waits for a receipt, if it did not receive any receipt after some time, it retransmits the packet until it gets the receipt. After the maximum number of attempts, if it is still unable to get a receipt from node C, it generates a *route error message* and sends it to node A, stating that the link from B to C is broken. Node A then removes this broken link from its route cache; if it has any other route to D in its cache, then it uses that other route, or else it, again, starts the *route discovery mechanism*.

We have developed an ant simulator, which incorporates a graphical user interface (GUI) that displays the search space as a hexagonal grid and allows the user to provide required inputs before starting the simulation. While running the simulation, we can study the movements of ants, how they find conflict areas, how they form shortest paths, and also about pheromone features.

Implementation: The simulation was developed using the Microsoft's .NET framework. The .NET framework provides a large library of coded solutions for common programming problems. The Visual Studio 2008 integrated development environment

(IDE) was used. It is a collection of tools to develop a user interface for classic windows desktop or web-based applications in different languages. The language that we used for coding is C#.

3.2.7. The Application User Interface (UI)

The UI of the ant simulator performs motion calculations for all ants. From these ant motions, we can learn how ants neutralizes conflict areas, finds paths, and about their mode of communication using pheromone.

The following assumptions are made in this simulation:

- A resource unit will neutralize only one conflict area at a time.
- Static conflict areas can be neutralized by at least one resource unit.
- Dynamic conflict areas can be neutralized by at least two resource units.
- On a given grid, the number of resource units should be more than the number of dynamic conflict areas. The relation should be as follows: for N number of resource units, there should not be more than N-1 dynamic conflict areas, irrespective of the number of static conflict areas. (This requirement ensures there would not be any dead lock while recruiting help to neutralize dynamic conflict areas.)

$$(\text{Resource units}) N > N-1 (\text{dynamic conflict areas})$$

Figure 3.3 shows the application UI without resource units and conflict areas. The hexagonal grid panel can be changed to fit any size, but for this simulation, we confined it to a 23 by 14 matrix of hexagons that may contain N resource units, M static conflict areas, and N-1 dynamic conflict areas.

Initially, when we launch the simulator, it loads the UI with the default grid, enabling the *Start*, *Reset*, and *Populate with Test Data* buttons, and *Pause* or *Resume* button is grayed because the application is not running. When the application is running, it displays the dynamic results of the simulation in the bottom table.

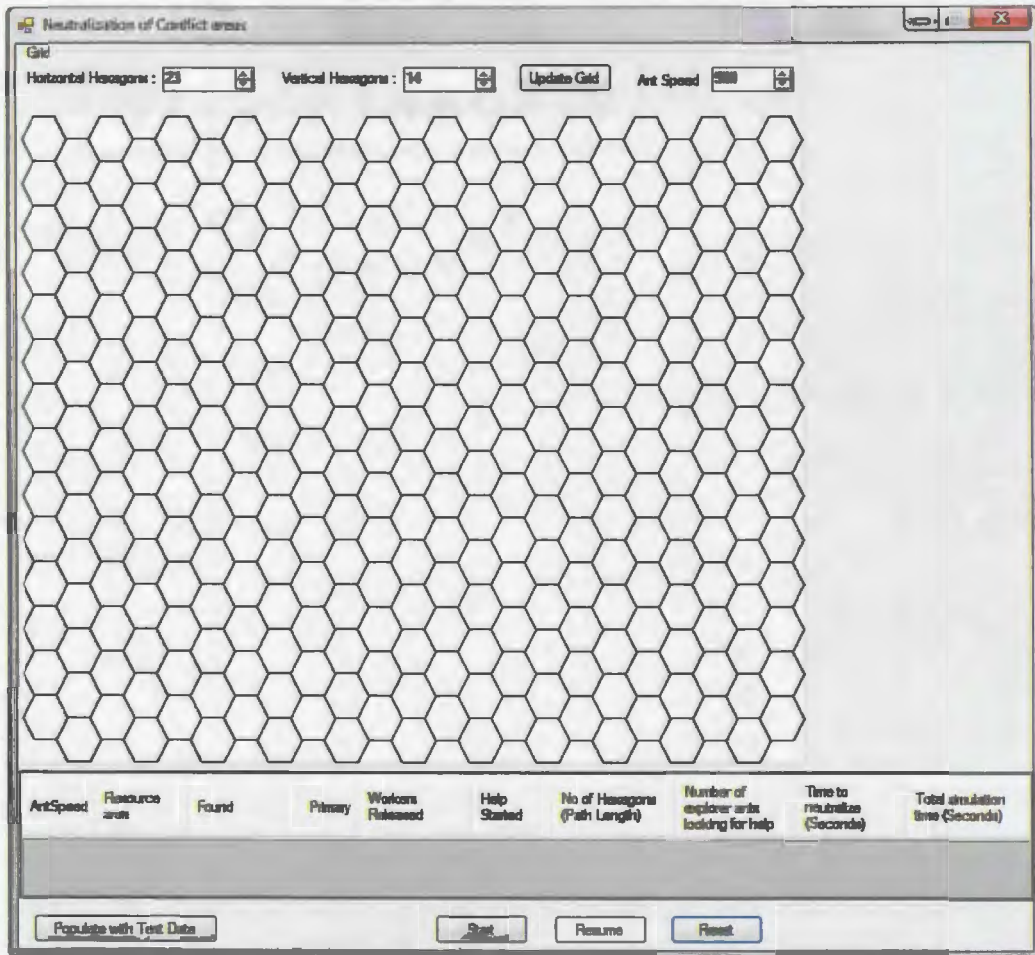


Figure 3.3. Graphical Representation of the Ant Simulator

The *Ant Speed* textbox can be used to vary the ant speed. The numeric dropdowns for *Horizontal Hexagons* and *Vertical Hexagons* are used to vary the size of the grid. After the UI is loaded, we need to place the minimum required resource units and conflict areas, or else we click the *Populate with Test Data* button to populate the grid with resource units

and conflict areas. The *Start* button is used to begin the simulation, and *Reset* button is used to reset the resource units and conflict areas on the grid. The *Pause* button is used to pause the simulation, and the *Stop* button is used to stop the simulation, making it possible to input necessary changes followed by a restart of the application.

To place resource units on the grid, we use the context menu which can be viewed by right clicking on the grid and selecting “resource unit.” A context menu looks like

Figure 3.4.

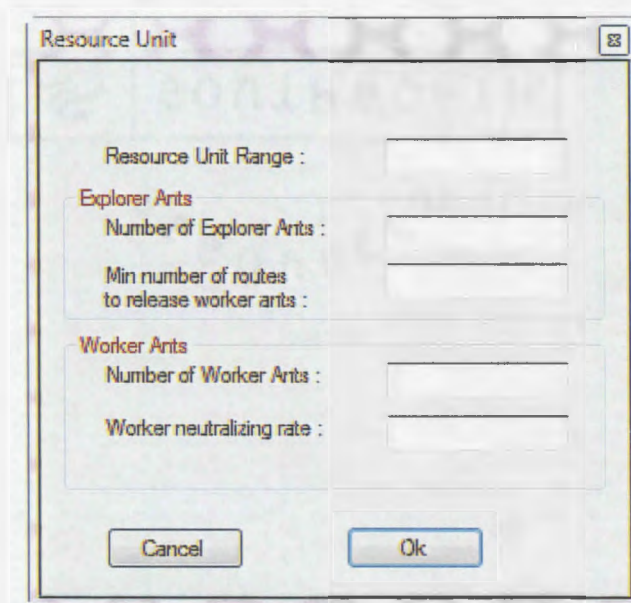


Figure 3.4. Form to Enter Resource Unit Input

Every resource unit has following required input:

- Resource unit range, which is used for how far an explorer ant can move away from the resource unit.
- Number of explorer ants, which represents the number of explorer ants a resource unit, can release to search conflict areas on a grid.

- Minimum number of routes to release worker ants, used for when to release worker ants to neutralize the conflict area.
- Number of worker ants, which represents how many worker ants a resource unit can release to neutralize conflict area.
- Worker ant neutralizing rate, which represents at what rate worker ants can neutralize or kill a conflict area.

To place a conflict area on the grid, use the context menu which can be viewed by right clicking on the grid and select the “conflict area.” Under it, select either “static area” or “dynamic area,” which typically looks like Figures 3.5 and 3.6, respectively. Every conflict area has required Severity, which shows numerical value representation of severity of conflict area

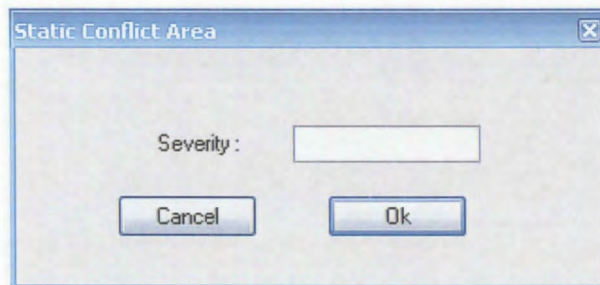


Figure 3.5. Form to Enter Static Conflict Area Severity

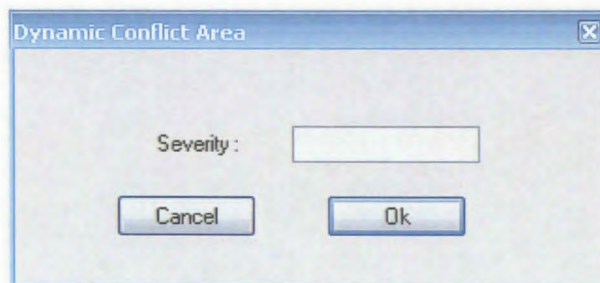


Figure 3.6. Form to Enter Dynamic Conflict Area Severity

To start the simulator, UI should require input; otherwise, it shows the error messages as seen in Figures 3.7. and 3.8.

The error message in Figure 3.7 is shown when there are no resource units and conflict areas on the grid. Error message in Figure 3.8 is shown when the number of resource units is equal to number of dynamic conflict areas on the grid.

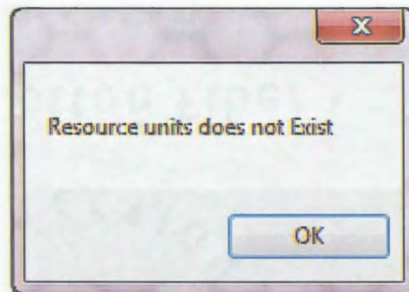


Figure 3.7. Error Message 1

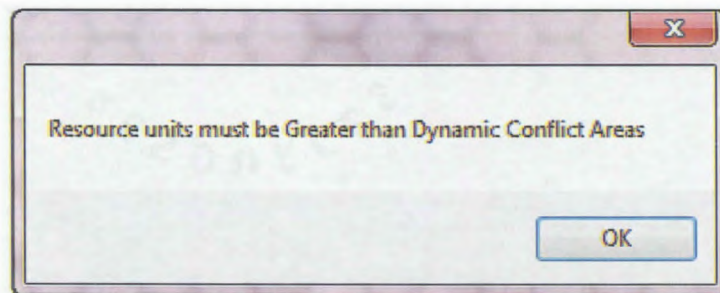


Figure 3.8. Error Message 2

4. EXTERNAL VIEW

4.1. Block Diagram of Ant Search

Figure 4.1 illustrates the block diagram of the ant search. We assume that ants have enough knowledge about how far they can move away from the resource unit and the conflict area. An ant also has knowledge to judge pheromone density in forming shortest paths and also to differentiate between conflict area and other resource units. With this basic knowledge throughout the entire simulation, an ant neutralizes conflict areas in the context of Dynamic Source Routing protocol.

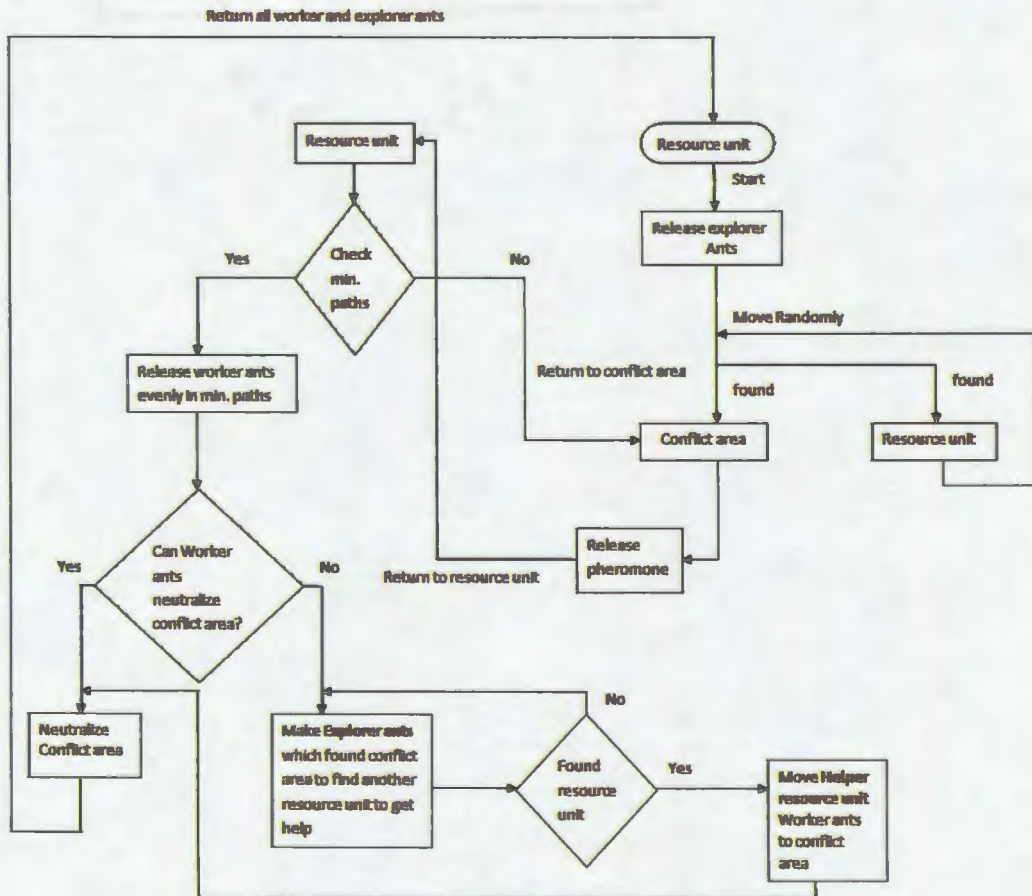


Figure 4.1. Block Diagram of Ant Search Algorithm

Figure 4.2 illustrates the pseudo code of the Ant Search Algorithm. It shows the entire process in three search algorithms: Explorer Ant Search, Worker Ant Search and Explorer Ant Help Search. To begin, the algorithm initially runs the Explorer Ant Search Algorithm. The pseudo code is shown in Figure 4.3. Explorer ants examine the search map randomly to find paths to conflict areas. In each instance of exploring, every explorer ant has six directions to move to find conflict areas, so it chooses one of the hexagons among the six hexagons to move. If the neighbor hexagon is in a Static Conflict Area (SCA) or Dynamic Conflict Area (DCA), it lays down pheromone on its way back to the conflict area, and the same pheromone is used to return to this conflict area; this process is an ant cycle. Ant Cycle helps the explorer ants to form persistent path for worker ants to reach the conflict area and helps workers ants to form short paths between Resource Unit (RU) and conflict area. If the neighbor hexagon has pheromone from some other ant, usually ant chooses that neighbor but it ignores the pheromone because its primary goal is to find a path to the resource unit which is like DSR's *route discovery mechanism*. Later, among the available paths using the pheromone concept, we find the shortest path. If a neighbor is empty, it again repeats the Explorer Ant Search process until it finds a RU.

```
Ant Search (Iterative process)  
BEGIN  
  Explorer Ant Search  
  Worker Ant Search  
  Explorer Ant Help Search  
END
```

Figure 4.2. Ant Search Algorithm

```

Explorer Ant Search (Iterative process)
BEGIN
Get list of unvisited neighbor
IF (not MAX distance travelled) THEN
    Randomly choose a neighbor
    IF (neighbor = SCA) THEN
        Return to RU laying down pheromone
        IF (RU reached) THEN
            Do Worker Ant Search process
        END IF
    ELSE IF (neighbor = DCA) THEN
        Return to RU laying down pheromone
        IF (RU reached) THEN
            Do Worker Ant Search process
        END IF
    ELSE IF (neighbor has pheromone) THEN
        Ignore pheromone choose neighbor
    ELSE
        Move to neighbor
        Repeat steps 2 to 17
    END IF
ELSE
    Return to RU
END IF
Repeat Explorer Ant search
END

```

Figure 4.3. Pseudo Code for Explorer Ant Search (RU=Resource Unit, CA=Conflict Area, SCA=Static Conflict Area; DCA= Dynamic Conflict Area, EA=Explorer ant; WA=Worker Ant)

An explorer ant cycle is the process of moving from a conflict area to a resource unit and from a resource unit to a conflict area. In every ant cycle, the ant cycle pheromone is updated only while moving from a conflict area to a resource unit and at the start of second cycle, pheromone from the first cycle is evaporated. In the context of an explorer ant's cycle, pheromone density for each cycle will be the same because there will only be one explorer ant in each path. Pheromone density increases only when there is more than one ant in that particular path. Explorer's ant pheromone density will only be increased by other worker ants following that path.

Figure 4.4, shows the second step in the Ant Search Algorithm. When a resource unit sees the minimum required paths to a conflict area, it may be SCA or DCA; then, it releases worker ants evenly on all the paths. While moving to the conflict area, the ant lays down pheromone along its path; if it is SCA in its round trips, it forms the shortest path using the ant cycle mechanism and neutralizes SCA; otherwise, if it is DCA, it does the same steps and also, to neutralize DCA, instantiates the Explorer Ant Help Search process which is the third step in Ant Search Algorithm

Figure 4.5 shows pseudo code for an Explorer Ant Help Search process. To neutralize a DCA, resource unit sends out those explorer ants that found the DCA to search for the RU. While searching, if the explorer ant comes across an SCA, another DCA, or its own RU, the ant ignores that neighbor and continues searching for other RUs. If it finds a RU, the explorer ant checks whether that RU is engaged in neutralizing other conflict areas; if it is, explorer ant ignores that RU and continues to search for another RU or else ant mobilizes RU workers ants to DCA using pheromone on its way. Throughout the entire process of neutralization, in a given time, the RU should work only on one conflict area.

Worker Ant Search (*Iterative process*)

BEGIN

IF (paths found by EA = Min. required paths) THEN

 Release worker ants evenly in all paths to conflict area

 Lay down pheromone along the path to CA

IF (SCA) THEN

WHILE (move in between SCA and RU)

 Form shortest path using pheromone

IF (another new path found to SCA by other EA) THEN

 Make EA attract to higher pheromone density of shortest path

 Evaporate pheromone in new path

END IF

 Neutralize SCA

END WHILE

 Return worker ants to RU from SCA

 Repeat Explorer Ant Search

END IF

IF (DCA) THEN

WHILE (move in between DCA and RU)

 Form shortest path using pheromone

IF (another new path found to DCA by other EA) THEN

 Make EA attract to higher pheromone density of shortest path

 Evaporate pheromone in new path

END IF

IF (more than two RU engaged in neutralizing DCA) THEN

 Neutralize DCA

ELSE IF (only current RU is working on DCA) THEN

 Do Explorer Ant Help Search

 Neutralize DCA

ELSE

 Repeat steps 18 to 28

END IF

Figure 4.4. Pseudo Code for Worker Ant Search

```

END WHILE
  Return worker ants to RU from DCA
  Repeat Explorer Ant Search
END IF
END

```

Figure 4.4. (Continued)

```

Explorer Ant Help Search
BEGIN
  Get list of explorer ants found DCA
  Initiate search for Helper RU
  WHILE (searching)
    IF (not MAX distance travelled) THEN
      Randomly choose neighbor
      IF (neighbor = SCA or DCA) THEN ignore
        Move randomly
      ELSE IF (neighbor = RU) THEN
        Return to DCA laying pheromone from Helper RU
        IF (Helper RU not engaged) THEN
          Move worker Ants of Helper RU to DCA
        ELSE
          Move in between DCA and Helper RU until Helper RU is free
        END IF
      ELSE
        Move randomly
      END IF
    ELSE
      Return to DCA
      Go to step 3
    END IF
  END WHILE
END

```

Figure 4.5. Pseudo Code for Explorer Ant Help Search

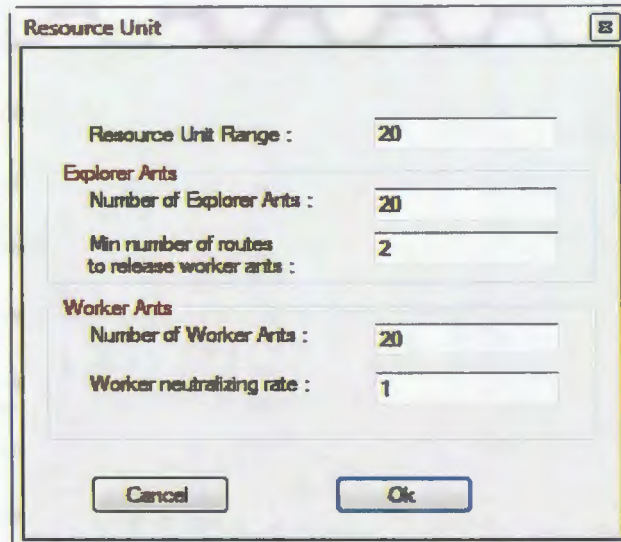
4.2. Output of Ant Simulator with Test Data

In this section, we will see the actual simulation. Before we actually run the simulator, we need to provide the required input which always has N resource units, not more than N-1 dynamic conflict areas, and M static conflict areas; otherwise, there may be a chance of forming deadlock in getting help to neutralize DCA. For example if there are 2 resource areas and 2 dynamic conflict areas on the grid. One resource unit found dynamic conflict area and another resource unit found another dynamic conflict area; at this point both resource units looking for help which is a dead lock. To avoid dead lock we need follow above constraint. On the User Interface (UI), there are two ways we can populate the grid: either by with manually right clicking on the grid to place resource units and conflict areas or by clicking on the *Populate with Test Data* button to populate the grid with test data. In this section, we will test the simulation with test data to observe and derive the results.

Our test data have four resource units (RU), three dynamic conflict areas (DCA), and two static conflict areas (SCA). Each resource unit will have the same input values. We may have different values, but to be consistent for these test data, we have used the same input for all resource units. The input form is depicted in Figure 4.6.

For all static conflict areas, we have used same input values, and for all dynamic conflict areas, we have the same input value. Here, input value represents the severity of the conflict area which is shown in Figure 4.7 and 4.8. A point to remember is that SCA represents a conflict area where the severity is static in nature which means its severity will not increase over time and can be neutralized by at least one resource unit. For DCA, we

need at least two resource units to neutralize the DCA because the conflict area's severity may increase over time.

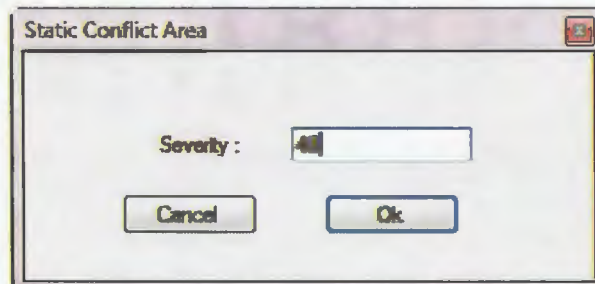


The 'Resource Unit' dialog box contains the following settings:

Parameter	Value
Resource Unit Range	20
Explorer Ants	
Number of Explorer Ants	20
Min number of routes to release worker ants	2
Worker Ants	
Number of Worker Ants	20
Worker neutralizing rate	1

Buttons: Cancel, Ok

Figure 4.6. Test Data for Resource Unit

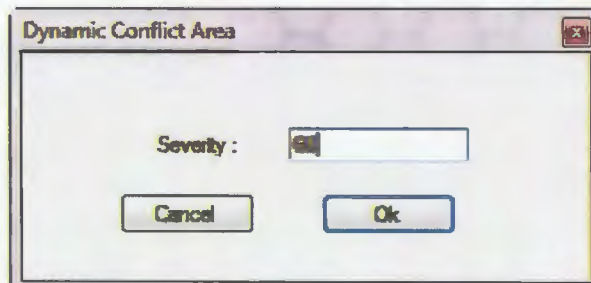


The 'Static Conflict Area' dialog box contains the following setting:

Parameter	Value
Severity	40

Buttons: Cancel, Ok

Figure 4.7. Test Data for Static Conflict Area Severity



The 'Dynamic Conflict Area' dialog box contains the following setting:

Parameter	Value
Severity	60

Buttons: Cancel, Ok

Figure 4.8. Test Data for Dynamic Conflict Area

After entering all the required input values. The ant simulator would appear as shown in Figure 4.9. Here hexagons in maroon color represent resource units, hexagons in light green color represent static conflict area, and dark green hexagons represent dynamic conflict areas. Group boxes on the right side of the Figure 4.9 provides information of each resource unit about location, range, explorer ants, minimum number of explorer ants, and worker neutralizing rate; and also about conflict area's type and their severity. The table below the hexagon grid shows the performance results of the simulation such as total simulation time etc.

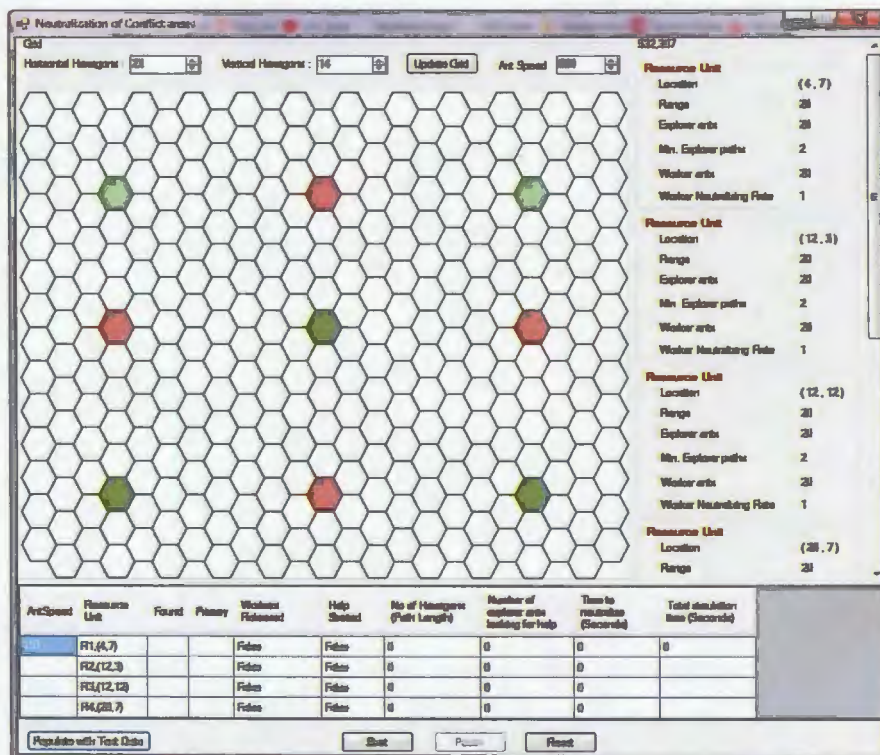


Figure 4.9. Graphical Representation of the Initial Load of the Simulator

Figures 4.10 through 4.17 show one scenario of solving the problem with artificial ants. In figures red marks represent explorer ants, green marks represent worker ants, and hexagons in light blue color represent pheromone.

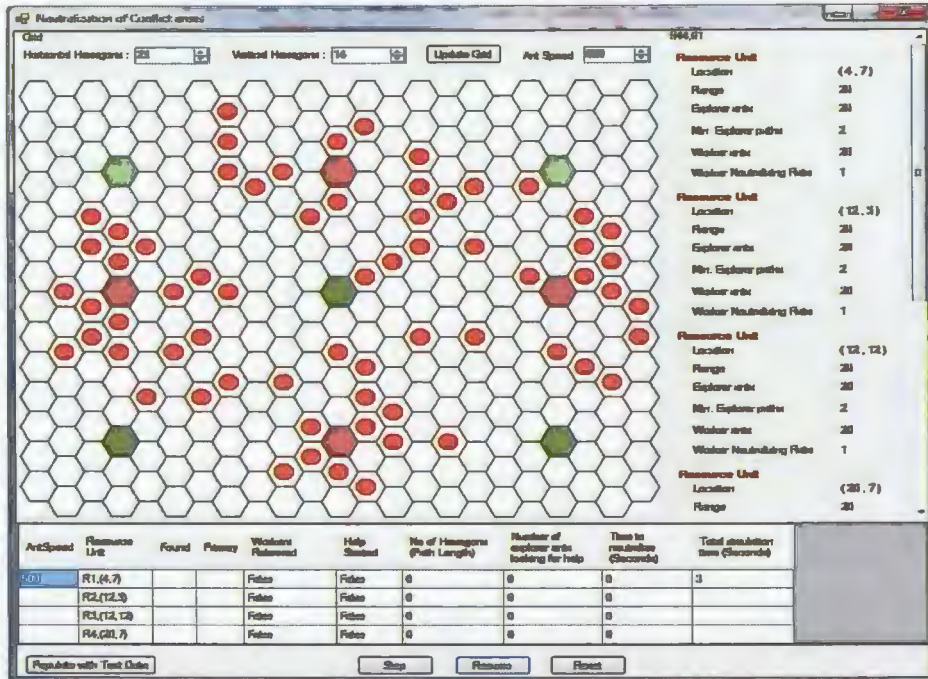


Figure 4.10. Graphical Representation of the Movement of Explorer Ants on Grid

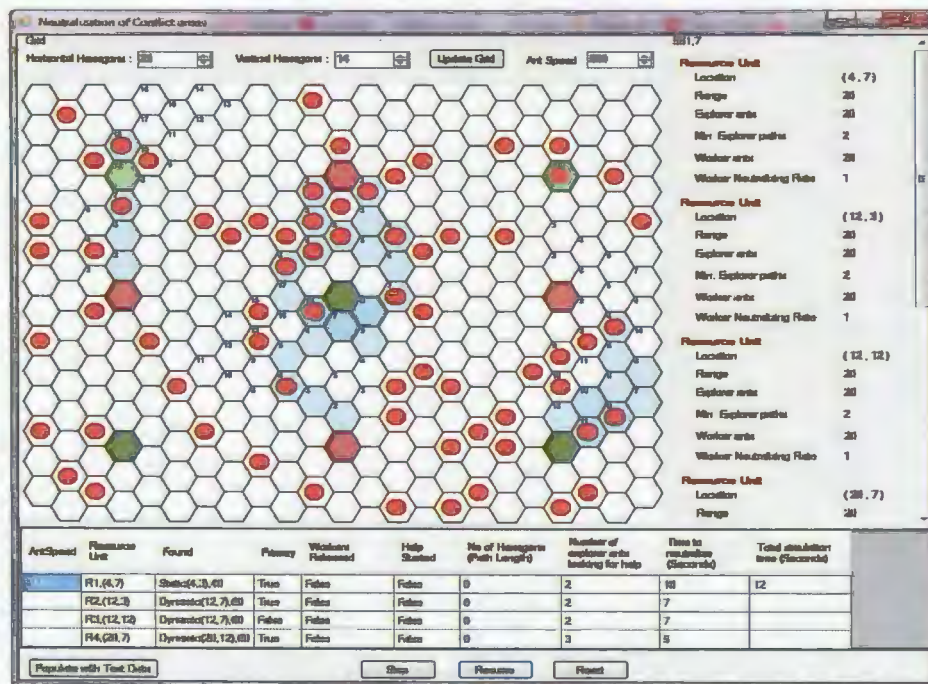


Figure 4.11. Graphical Representation of the Pheromone Lay Down by the Explorer Ants

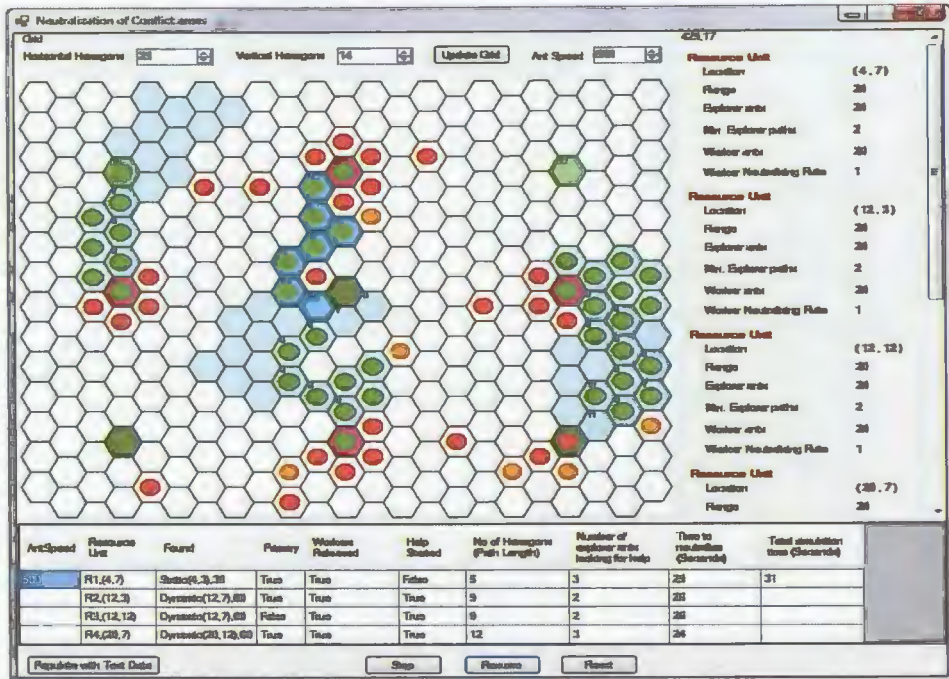


Figure 4.12. Graphical Representation of the Movement of Worker Ants

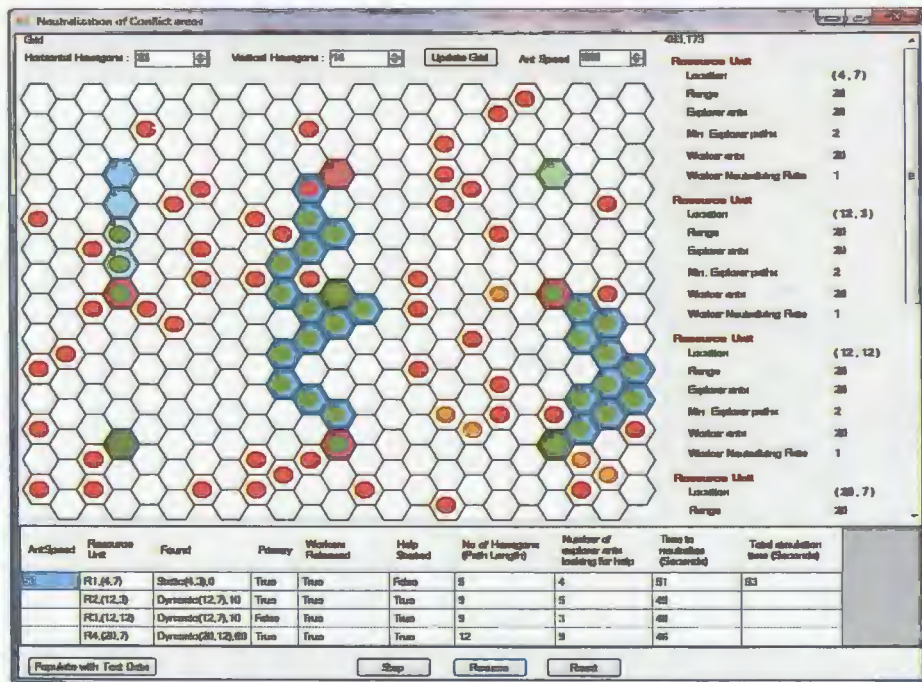


Figure 4.13. Graphical Representation of the Worker Ants Forming Short Paths

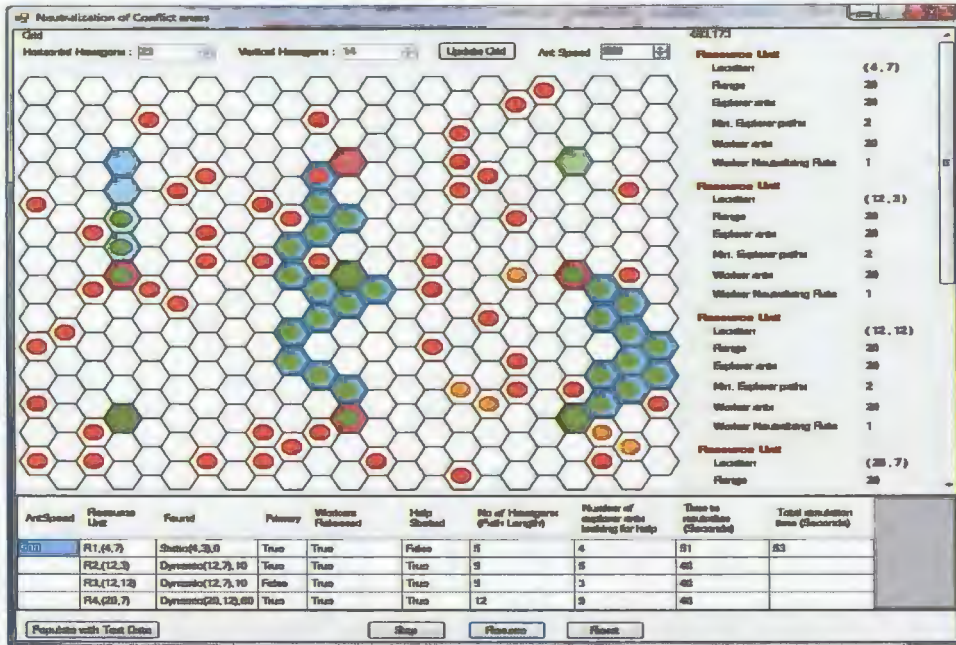


Figure 4.14. Graphical Representation of Two Resource Units Neutralizing Dynamic Conflict Area 1

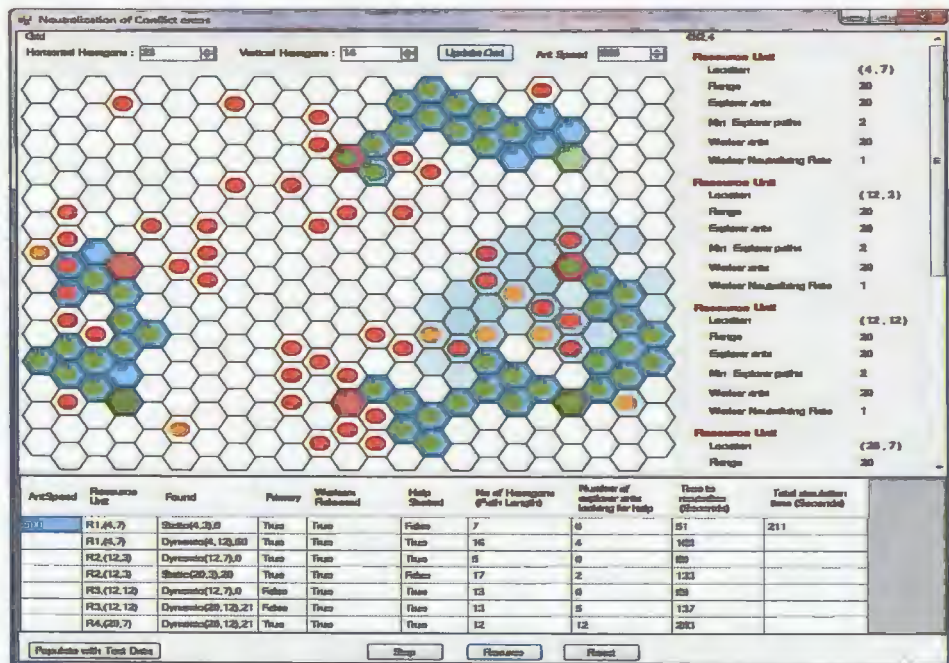


Figure 4.15. Graphical Representation of Two Resource Units Neutralizing Dynamic Conflict Area 2

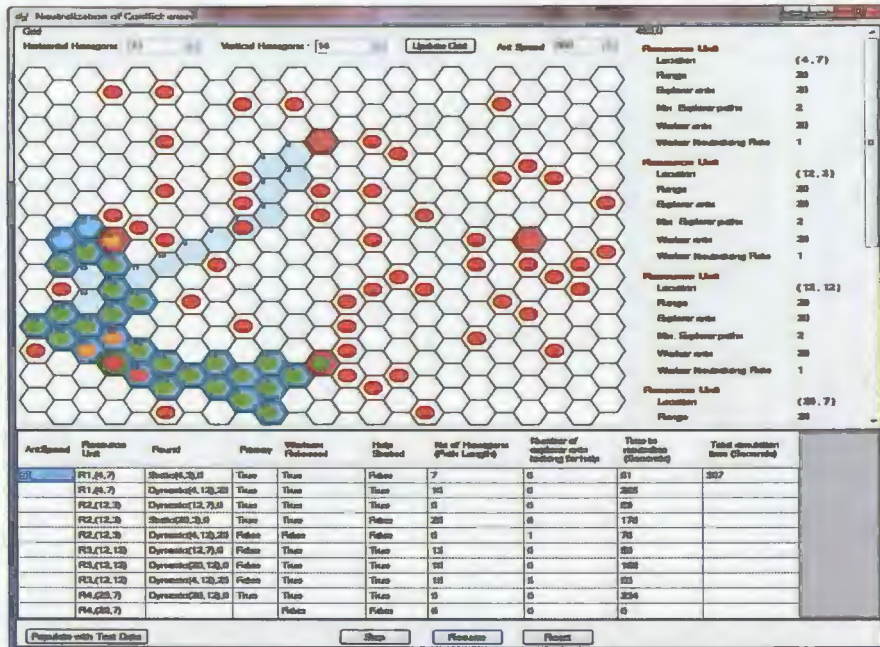


Figure 4.16. Graphical Representation of Two Resource Units Neutralizing Dynamic Conflict Area 3

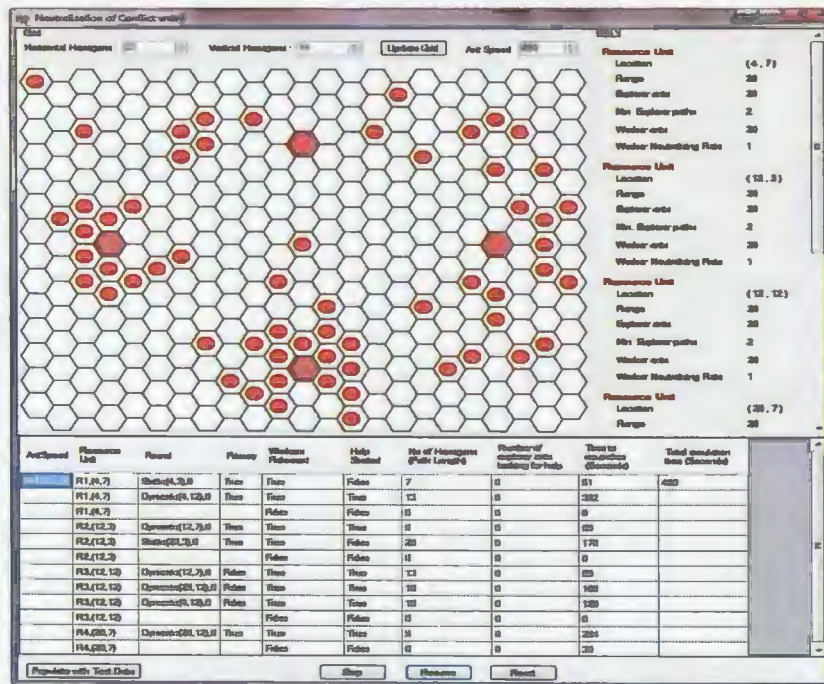


Figure 4.17. Graphical Representation After Neutralizing All Conflict Areas on the Grid

5. INTERNAL VIEW

In this chapter, we will go through internal structure of our application, including some important class files that we created to successfully develop this simulator. The ant simulator was developed using DOTNET Framework 3.5, Visual Studio 2008, Win Forms, built-in controls, and C# language. It was built on Windows OS 7 Professional, with 2.1 GHz and 3GB RAM.

5.1. Overview of Class Files

In our application we have developed following classes,

- Program.cs
- ResourceConflictUI.cs
- Polygon.cs
- Hexagon.cs
- AreaSize.cs

5.1.1. Program.cs

This class file is responsible for actually building the application and bringing up the user interface (UI) to run the simulator. To bring up the UI program uses ResourceConflictUI.cs and its associated interrelated class files to compile and execute the .EXE file.

5.1.2. ResourceConflictUI.cs

This class file is a Win Form which is a UI provided by Visual Studio 2008 (VS 2008) to develop user interfaces. To develop the UI, we have used built-in controls which

are again classes that come with the .NET Framework VS 2008 integrated development environment (IDE) like panel, numeric dropdowns, textboxes, groupboxes, gridview, and buttons.

When the application is run, this class file is responsible for all the pre calculations and initializes all the controls on the UI. This class is also responsible for drawing the hexagon grid, resource units, conflict areas, results table, and summary group boxes while running the simulation.

5.1.3. Polygon.cs

This class represents an object which can hold an array of seven 2-dimensional points. These points are used to draw a single hexagon on hexagon grid and also locations of resource units, explorer ants, and worker ants.

5.1.4. Hexagons.cs

This class is responsible for representing the hexagon grid and its associated resource unit and conflict area properties. The class also does the calculations about how to move an ant to another hexagon randomly and also maintains a list of hexagons about the ant path, and also responsible for drawing pheromone colors in those hexagons.

5.1.5. AreaSize.cs

This UI user control utilized to display a UI box that collects user-entered input values for resource units and conflict areas. It also validates the input values and interacts with the context menu to display resource units and conflict areas on the grid.

The class files are responsible for the movement of explorer ants, worker ants, and explorer ants looking for help. The ant movements are shown on the GUI in each frame. A

Frame is a particular instance of the application where the program calculates the next location to which each ant moves. Each new frame is triggered by a timer control which is built-in control of .NET Framework. The timer is responsible to start the release of explorer ants and worker ants on the grid for each resource unit. As soon as calculations are done for each ant of resource units, in every frame GUI draws the hexagonal grid, explorer ants, and worker ants at their calculated locations. The application simulates the movements of ants as per the pseudo code presented in Figures 4.3, 4.4, and 4.5.

The parallelism of ants can be represented by following pseudo code:

BEGIN

Get the delay interval for each frame

Start the timer

FOR each delay interval of timer

 Get the list of all RUs

FOR each EA in RU

 Calculate next position to move

CALL Explorer Ant Search process

IF (EA found minimum paths to CA) **THEN**

CALL Worker Ant Search process

IF (WA unable to neutralize CA) **THEN**

CALL Explorer Ant Help Search process

END IF

ELSE

CALL Explorer Ant Search process

END IF

END FOR

END FOR

END

(RU = Resource unit, EA = Explorer ant, WA = Worker ant, CA = Conflict area)

6. EXPERIMENTAL RESULTS

Parameters were varied during the simulation run:

- Resource unit range
- Number of explorer ants
- Number of paths required to release worker ants

Our simulation was run three times. Each time one of the above variable parameters is changed, and the remaining parameters, such as worker ant neutralizing rate and conflict areas severity are not changed. In each simulation, we observe the total simulation time and shortest path to conflict areas as well as some of the columns added to/deleted from the actual results table as per the requirement.

Also, we presented tables and graphs for 10 different positions of resource units and conflict areas. Here, each position was run 10 times to calculate the mean, standard deviation, and confidence interval of the run.

Each simulation was run with 4 resource units, 3 dynamic conflict areas, and 2 static conflict areas on 23 by 14 hexagonal grid. First we will see the performance of simulator by varying resource unit range. We believe, by varying resource unit range, explorer ants can find the conflict locations in the long range and also it helps in finding resource units in the long range when there is a need for help. By varying the number of explorer ants, we believe that the probability of finding conflict locations in the search area will be high and also will have a high probability of finding resource units for help. By varying the number of paths required to release works, we believe that there is a high probability of finding shortest paths among available paths which will have impact on total simulation time.

6.1. Results by Varying Parameter Resource Unit Range

In this section performance results are observed for test cases 1, 2, and 3 by varying parameter resource unit range and keeping the remaining parameters the same.

6.1.1. Test Case 1

In this test, the resource unit range is 15. The static conflict area severity is 40. The dynamic conflict area severity is 60. The number of explorer ants is 20 and the number of worker ants is 20. For this test case results are presented in Table 6.1.

Table 6.1. Performance Results for Test Case 1

Resource unit range	Resource unit	Primary	Found	No. of hexagons (Path length)	Time to neutralize (Seconds)	Total simulation time (Seconds)
15	(4,7)	True	Static (4,3)	15	33	215
	(4,7)	True	Dynamic (4,12)	13	178	
	(12,12)	False	Dynamic (4,12)	10	96	
	(12,3)	True	Dynamic (12,7)	5	117	
	(20,7)	False	Dynamic (12,7)	9	75	
	(12,12)	True	Dynamic (20,12)	14	81	
	(20,7)	False	Dynamic (20,12)	6	60	
	(20,7)	True	Static (20,3)	15	34	

6.1.2. Test Case 2

In this test, the resource unit range is 20. The static conflict area severity is 40. The dynamic conflict area severity is 60. The number of explorer ants is 20, and the number of worker ants is 20. For this test case results are presented in Table 6.2.

Table 6.2. Performance Results for Test Case 2

Resource unit range	Resource unit	Primary	Found	No. of hexagons (Path length)	Time to neutralize (Seconds)	Total simulation time (Seconds)
20	(4,7)	True	Static (4,3)	12	32	110
	(12,12)	True	Dynamic (4,12)	10	73	
	(4,7)	False	Dynamic (4,12)	6	52	
	(20,7)	True	Dynamic (12,7)	12	50	
	(12,3)	False	Dynamic (12,7)	7	41	
	(12,12)	True	Dynamic (20,12)	11	67	
	(20,7)	False	Dynamic (20,12)	5	63	
	(20,7)	True	Static (20,3)	7	24	

6.1.3. Test Case 3

In this test, the resource unit range is 25. The static conflict area severity is 40. The dynamic conflict area severity is 60. The number of explorer ants is 20, and number of worker ants is 20. For this test case results are presented in Table 6.3.

From Figure 6.1, we infer that

- A higher resource unit range leads to higher simulation time.
 - The time taken to travel from resource unit to maximum range and come back to resource unit will be greater
 - In large search area, long range will have significant impact on total simulation time.
- A high resource range will have a high probability of finding conflict locations in the long range.

Table 6.3. Performance Results for Test Case 3

Resource unit range	Resource unit	Primary	Found	No. of hexagons (Path length)	Time to neutralize (Seconds)	Total simulation time (Seconds)
25	(4,7)	True	Static (4,3)	13	73	440
	(12,12)	True	Dynamic (4,12)	17	219	
	(4,7)	False	Dynamic (4,12)	9	185	
	(4,7)	True	Dynamic (12,7)	25	274	
	(12,3)	False	Dynamic (12,7)	18	230	
	(12,12)	False	Dynamic (12,7)	6	45	
	(20,7)	True	Dynamic (20,12)	24	152	
	(12,12)	False	Dynamic (20,12)	10	86	
	(12,2)	True	Static (20,3)	17	97	

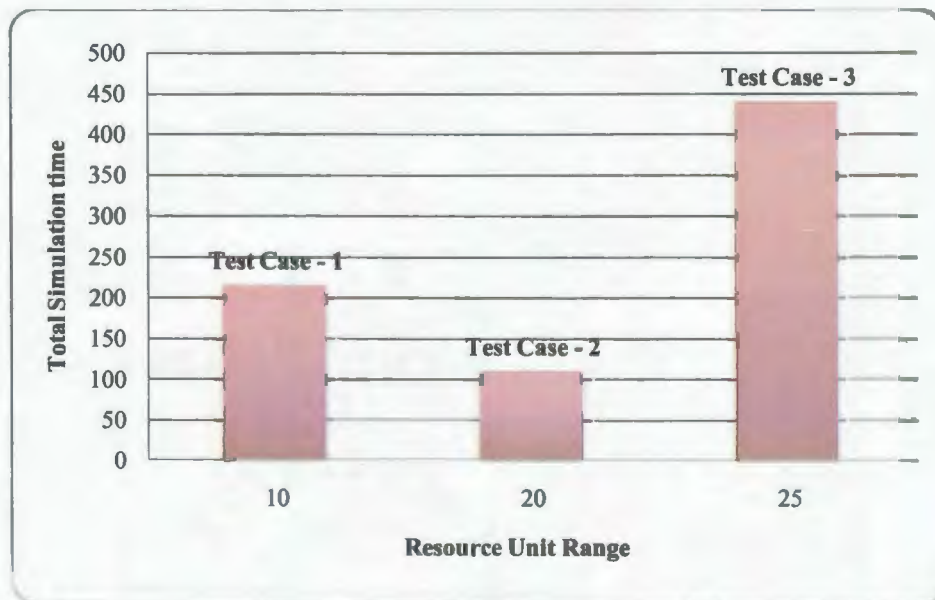


Figure 6.1. Resource Unit Range vs. Total Simulation Time

6.2. Results by Varying Parameter Number of Explorer Ants

In this section performance results are observed for test cases 4, 5, and 6 by varying parameter number of explorer ants and keeping the remaining parameters the same.

6.2.1. Test Case 4

In this test, the resource unit range is 20. The static conflict area severity is 40. The dynamic conflict area severity is 60. The number of explorer ants is 15, and number of worker ants is 20. For this test case results are presented in Table 6.4.

Table 6.4. Performance Results for Test Case 4

Number of Explorer Ants	Resource unit	Primary	Found	No. of hexagons (Path length)	Time to neutralize (Seconds)	Total simulation time (Seconds)
15	(4,7)	True	Static (4,3)	9	49	249
	(4,7)	True	Dynamic (4,12)	10	70	
	(12,12)	False	Dynamic (4,12)	13	55	
	(12,3)	True	Dynamic (12,7)	7	81	
	(12,12)	False	Dynamic (12,7)	6	47	
	(20,7)	True	Dynamic (20,12)	6	203	
	(12,12)	False	Dynamic (20,12)	11	114	
	(12,2)	True	Static (20,3)	12	40	

6.2.2. Test Case 5

In this test, the resource unit range is 20. The static conflict area severity is 40. The dynamic conflict area severity is 60. The number of explorer ants is 20, and number of worker ants is 20. For this test case results are presented in Table 6.5.

Table 6.5. Performance Results for Test Case 5

Number of Explorer Ants	Resource unit	Primary	Found	No. of hexagons (Path length)	Time to neutralize (Seconds)	Total simulation time (Seconds)
20	(4,7)	True	Static (4,3)	10	59	244
	(4,7)	True	Dynamic (4,12)	18	165	
	(12,12)	False	Dynamic (4,12)	15	77	
	(12,3)	True	Dynamic (12,7)	12	157	
	(12,12)	False	Dynamic (12,7)	10	57	
	(20,7)	True	Dynamic (20,12)	8	91	
	(12,12)	False	Dynamic (20,12)	17	92	
	(20,7)	True	Static (20,3)	7	26	

6.2.3. Test Case 6

In this test, the resource unit range is 20. The static conflict area severity is 40, dynamic conflict area severity is 60, number of explorer ants is 25, and number of worker ants is 20. For this test case results are presented in Table 6.6.

From Figure 6.2, we infer that

- More explorer ants lead to a high processing time for the simulator.
 - In small search area like 23 by 14 hexagonal grid. Processing time will have less impact on the performance of the simulator.
 - In large search area like 100 by 100 hexagonal grid. Processing time will hve high impact on the performance of the simulator.
- In large search area, more explorer ants will be helphul in finding conflict locations in a shorter time.

Table 6.6. Performance Results for Test Case 6

Number of Explorer Ants	Resource unit	Primary	Found	No. of hexagons (Path length)	Time to neutralize (Seconds)	Total simulation time (Seconds)
25	(4,7)	True	Static (4,3)	7	45	413
	(4,7)	True	Dynamic (4,12)	9	53	
	(12,12)	False	Dynamic (4,12)	17	37	
	(12,3)	True	Dynamic (12,7)	6	57	
	(12,12)	False	Dynamic (12,7)	12	57	
	(20,7)	True	Dynamic (20,12)	17	381	
	(12,12)	False	Dynamic (20,12)	19	273	
	(20,7)	True	Static (20,3)	7	39	

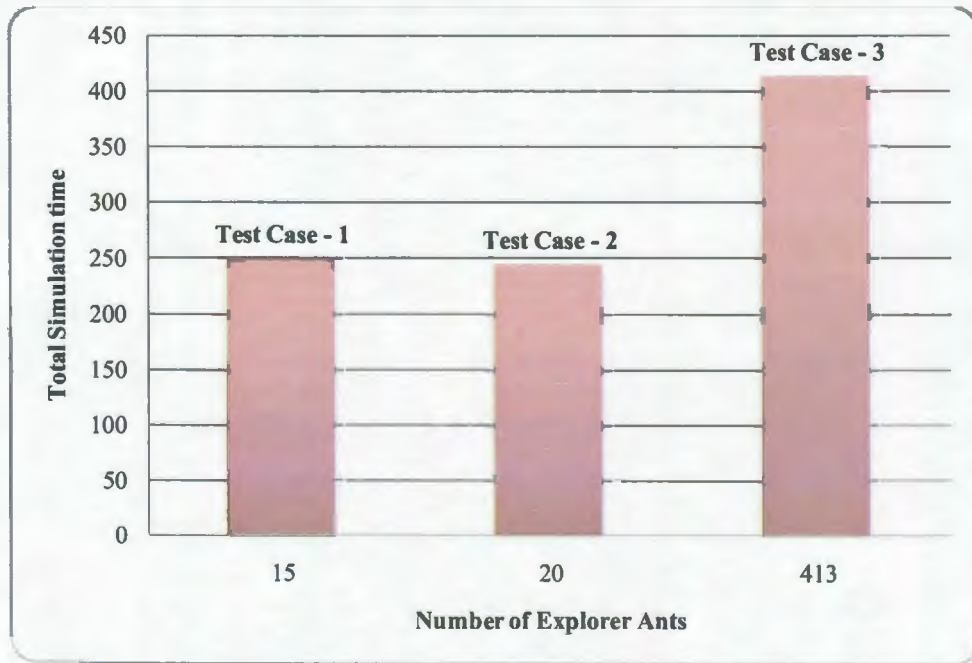


Figure 6.2. Number of Explorer Ants vs. Total Simulation Time

6.3. Results by Varying Parameter Number of Paths

In this section performance results are observed for test cases 7, 8, and 9 by varying parameter number of paths and keeping the remaining parameters the same.

6.3.1. Test Case 7

In this test, the resource unit range is 20. The static conflict area severity is 40. The dynamic conflict area severity is 60. The number of explorer ants is 20. The number of worker ants is 20, and the number of paths required to release worker ants is 1. For this test case results are presented in Table 6.7.

Table 6.7. Performance Results for Test Case 7

Number of paths required to release worker ants	Resource unit	Primary	Found	No. of hexagons (Path length)	Time to neutralize (Seconds)	Total simulation time (Seconds)
1	(12,3)	True	Static (4,3)	18	49	190
	(4,7)	True	Dynamic (4,12)	5	185	
	(12,12)	False	Dynamic (4,12)	7	54	
	(12,3)	True	Dynamic (12,7)	8	55	
	(12,12)	False	Dynamic (12,7)	12	55	
	(20,7)	True	Dynamic (20,12)	7	80	
	(12,12)	False	Dynamic (20,12)	19	45	
	(20,7)	True	Static (20,3)	6	23	

6.3.2. Test Case 8

In this test, the resource unit range is 20. The static conflict area severity is 40. The dynamic conflict area severity is 60. The number of explorer ants is 20. The number of

worker ants is 20, and the number of paths required to release worker ants is 2. For this test case results are presented in Table 6.8.

Table 6.8. Performance Results for Test Case 8

Number of paths required to release worker ants	Resource unit	Primary	Found	No. of hexagons (Path length)	Time to neutralize (Seconds)	Total simulation time (Seconds)
2	(4,7)	True	Static (4,3)	15	42	277
	(4,7)	True	Dynamic (4,12)	7	93	
	(12,12)	False	Dynamic (4,12)	9	87	
	(12,12)	True	Dynamic (12,7)	6	100	
	(12,3)	False	Dynamic (12,7)	5	99	
	(20,7)	True	Dynamic (20,12)	13	243	
	(12,12)	False	Dynamic (20,12)	19	76	
	(20,7)	True	Static (20,3)	12	28	

6.3.3. Test Case 9

In this test, the resource unit range is 20. The static conflict area severity is 40. The dynamic conflict area severity is 60. The number of explorer ants is 20. The number of worker ants is 20, and the number of paths required to release worker ants is 3. For this test case, results are presented in Table 6.9.

From Figure 6.3, we infer that

- The Number of paths required is directly proportional to the total simulation time.
- Useful in large maps to find short path, when the conflict areas are far from resource units.

Table 6.9. Performance Results for Test Case 9

Number of paths required to release worker ants	Resource unit	Primary	Found	No. of hexagons (Path length)	Time to neutralize (Seconds)	Total simulation time (Seconds)
3	(4,7)	True	Static (4,3)	7	27	431
	(4,7)	True	Dynamic (4,12)	14	146	
	(12,12)	False	Dynamic (4,12)	13	134	
	(12,3)	True	Dynamic (12,7)	8	127	
	(12,12)	False	Dynamic (12,7)	14	60	
	(20,7)	True	Dynamic (20,12)	6	424	
	(12,12)	False	Dynamic (20,12)	10	171	
	(12,3)	True	Static (20,3)	14	147	

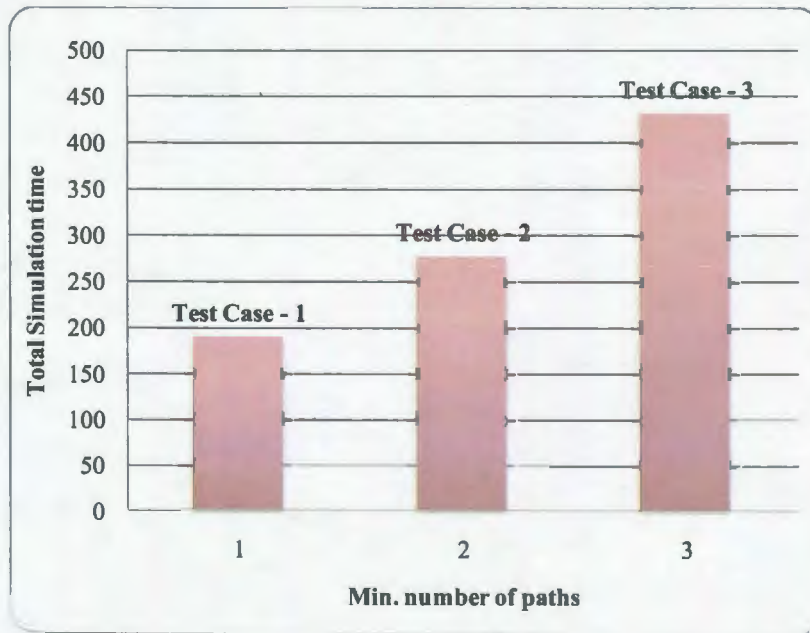


Figure 6.3. Number of Paths Required to Release Worker Ants vs. Total Simulation Time

6.4. Mean, Standard Deviation and Confidence Interval

We have calculated the mean, standard deviation, and confidence interval for 10 different positions. In all 10 positions, each resource unit has a range of 20: the number of explorer ants is 20; the number of worker ants is 20; and each static conflict area has a severity of 40 while each dynamic conflict area has a severity of 60. The confidence interval is calculated at the 95% confidence level, and the calculated confidence interval represents the confidence limits of the simulation for that particular position.

For mean X , standard deviation σ , and sample n . Confidence interval at 95% can be calculated as,

$$0.95 = P(X - 1.96 \sigma/\sqrt{n} \leq X + 1.96 \sigma/\sqrt{n})$$

6.4.1. Ten Simulations for Position 1

Resource units are at locations (4, 7), (12, 3), (12, 12) and (20, 7). Conflict areas are at locations (4, 12), (12, 7), (20, 12), (4, 3) and (20, 3). Results of 10 simulations for position 1 were presented in Table 6.10. Mean, standard deviation, and confidence interval are shown in Figure 6.4.

Table 6.10. Performance Results for Position 1

Number of runs	Total simulation time (Seconds)
1	215
2	354
3	104
4	309
5	116
6	251
7	166
8	319
9	257
10	317

The calculated mean, standard deviation, and confidence interval are,

Mean = 240.8

Standard Deviation = 88.38

Confidence Interval = (186.1, 295.5)

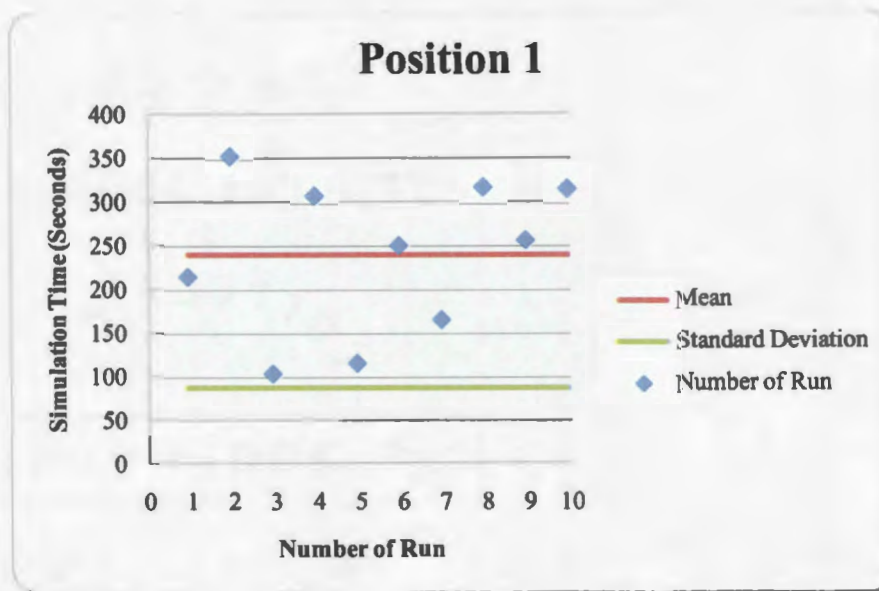


Figure 6.4. Position 1, Mean, Standard Deviation, Confidence Interval

6.4.2. Ten Simulations for Position 2

Resource units are at locations (4, 3), (12, 7), (12, 12) and (20, 3). Conflict areas are at locations (4, 12), (12, 3), (20, 12), (4, 7), and (20, 7). Results of 10 simulations for position 2 were presented in Table 6.11. Mean, standard deviation, and confidence interval are shown in Figure 6.5.

The calculated mean, standard deviation, and confidence interval are,

Mean = 332.8

Standard Deviation = 66.2

Confidence interval = (291.77, 373.83)

Table 6.11. Performance Results for Position 2

Number of runs	Total simulation time (Seconds)
1	354
2	344
3	339
4	426
5	312
6	233
7	372
8	288
9	240
10	420

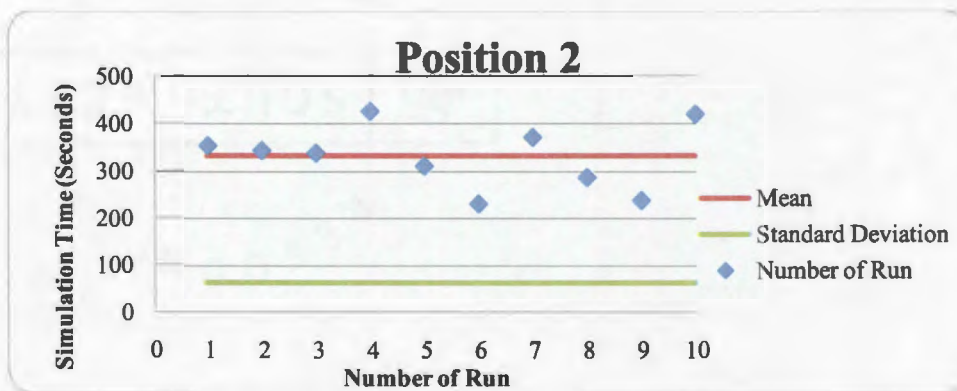


Figure 6.5. Position 2, Mean, Standard Deviation, Confidence Interval

6.4.3. Ten Simulations for Position 3

Resource units are at locations (4, 3), (12, 7), (4, 12) and (2, 7). Conflict areas are at locations (12, 12), (12, 3), (20, 12), (4, 7), and (20, 3). Results of 10 simulations for position 3 were presented in Table 6.12. Mean, standard deviation, and confidence interval are shown in Figure 6.6.

The calculated mean, standard deviation, and confidence interval are,

Mean = 653.7

Standard Deviation = 141.1

Confidence Interval = (566.25, 741.15)

Table 6.12. Performance Results for Position 3

Number of runs	Total simulation time (Seconds)
1	876
2	833
3	725
4	617
5	667
6	589
7	447
8	482
9	565
10	736

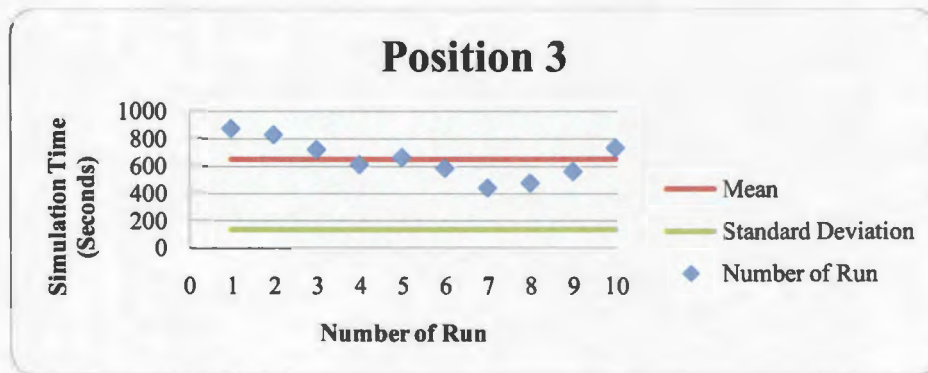


Figure 6.6. Position 3, Mean, Standard Deviation, Confidence Interval

6.4.4. Ten Simulations for Position 4

Resource units are at locations (4, 7), (12, 7), (20, 12) and (20, 3). Conflict areas are at locations (4, 3), (12, 3), (20, 7), (4, 12), and (12, 12). Results of 10 simulations for position 4 were presented in Table 6.13. Mean, standard deviation, and confidence interval are shown in Figure 6.7.

The calculated mean, standard deviation, and confidence interval are,

Mean = 453.3

Standard Deviation = 140.73

Confidence Interval = (366.08, 540.52)

Table 6.13. Performance Results for Position 4

Number of runs	Total simulation time (Seconds)
1	382
2	655
3	299
4	586
5	344
6	294
7	320
8	547
9	340
10	586

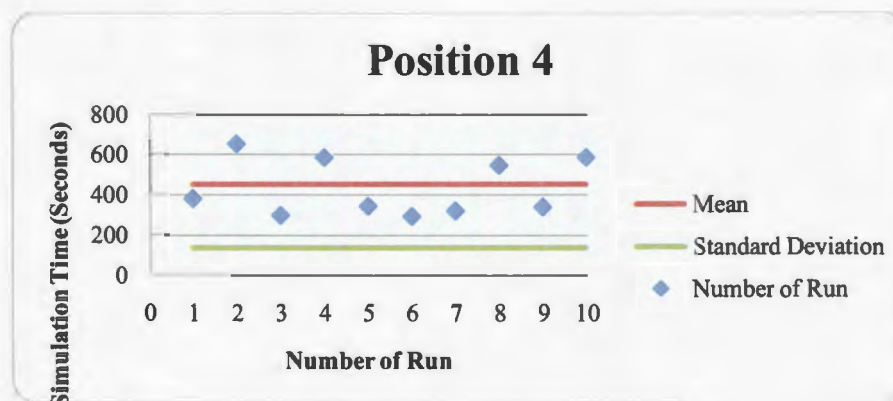


Figure 6.7. Position 4, Mean, Standard Deviation, Confidence Interval

6.4.5. Ten Simulations for Position 5

Resource units are at locations (4, 12), (12, 7), (20, 12), and (12, 3). Conflict areas are at locations (4, 7), (20, 7), (12, 12), (4, 3), and (20, 3). Results of 10 simulations for position 5 were presented in Table 6.14. Mean, standard deviation, and confidence interval are shown in Figure 6.8.

The calculated mean, standard deviation, and confidence interval are,

Mean = 457.2

Standard Deviation = 66.17

Confidence Interval = (416.3, 498.3)

Table 6.14. Performance Results for Position 5

Number of runs	Total simulation time (Seconds)
1	540
2	430
3	517
4	486
5	401
6	351
7	452
8	397
9	445
10	553

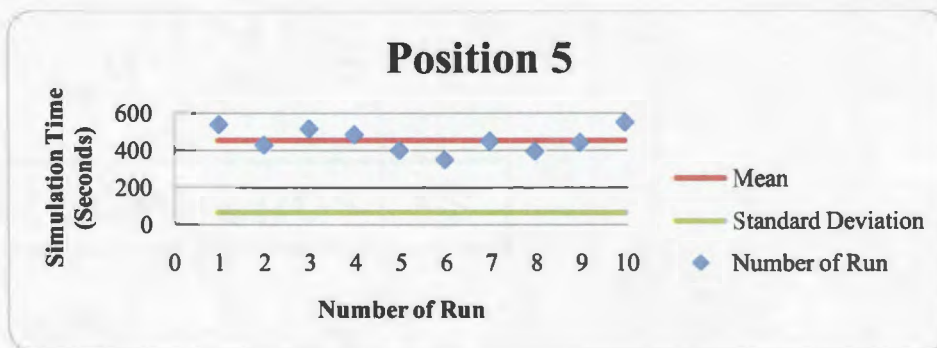


Figure 6.8. Position 5, Mean, Standard Deviation, Confidence Interval

6.4.6. Ten Simulations for Position 6

Resource units are at locations (4, 12), (12, 7), (20, 12) and (4, 3). Conflict areas are at locations (4, 7), (20, 7), (12, 12), (12, 3), and (20, 3). Results of 10 simulations for position 6 were presented in Table 6.15. Mean, standard deviation, and confidence interval are shown in Figure 6.9.

The calculated mean, standard deviation, and confidence interval are,

Mean = 413.5

Standard Deviation = 135.14

Confidence Interval = (329.74, 497.26)

Table 6.15. Performance Results for Position 6

Number of runs	Total simulation time (Seconds)
1	445
2	498
3	312
4	461
5	338
6	640
7	264
8	407
9	208
10	562

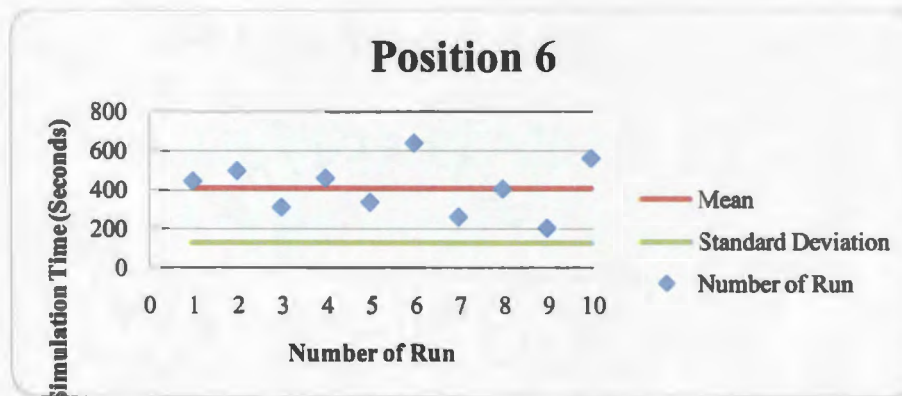


Figure 6.9. Position 6, Mean, Standard Deviation, Confidence Interval

6.4.7. Ten Simulations for Position 7

Resource units are at locations (20, 3), (12, 7), (20, 12) and (4, 3). Conflict areas are at locations (4, 7), (20, 7), (12, 12), (12, 3) and (4, 12). Results of 10 simulations for position 7 were presented in Table 6.16. Mean, standard deviation, and confidence interval are shown in Figure 6.10.

The calculated mean, standard deviation, and confidence interval are,

Mean = 433.6

Standard Deviation = 76.32

Confidence Interval = (386.3, 480.9)

Table 6.16. Performance Results for Position 7

Number of runs	Total simulation time (Seconds)
1	406
2	489
3	285
4	386
5	454
6	445
7	466
8	532
9	356
10	517

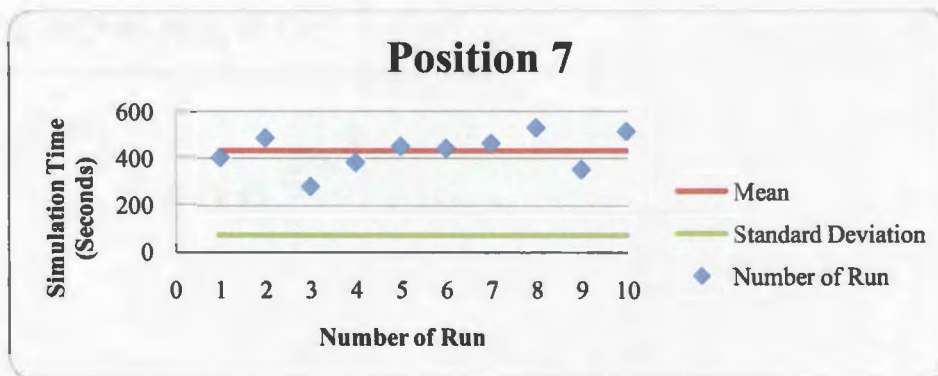


Figure 6.10. Position 7, Mean, Standard Deviation, Confidence Interval

6.4.8. Ten Simulations for Position 8

Resource units are at locations (4, 12), (4, 3), (20, 12) and (20, 3). Conflict areas are at locations (4, 7), (20, 7), (12, 12), (12, 3), and (12, 7). Results of 10 simulations for position 8 were presented in Table 6.17. Mean, standard deviation, and confidence interval are shown in Figure 6.11.

The calculated mean, standard deviation, and confidence interval are,

Mean = 247.8

Standard Deviation = 59.02

Confidence Interval = (211.22, 248.38)

Table 6.17. Performance Results for Position 8

Number of runs	Total simulation time (Seconds)
1	373
2	250
3	284
4	223
5	191
6	207
7	293
8	177
9	269
10	211

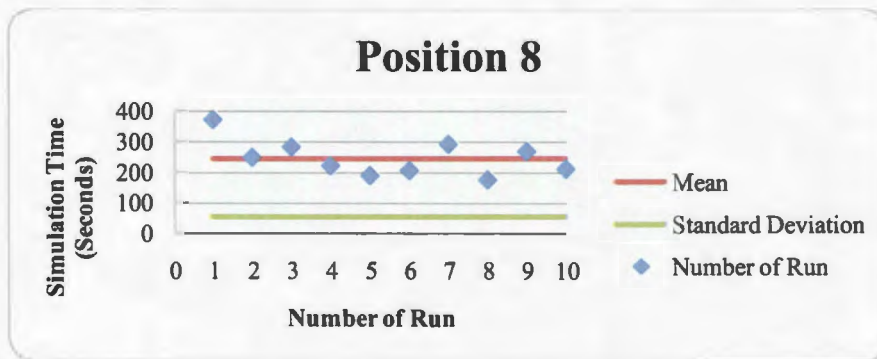


Figure 6.11. Position 8, Mean, Standard Deviation, Confidence Interval

6.4.9. Ten Simulations for Position 9

Resource units are at locations (4, 7), (12, 7), (12, 3) and (20, 7). Conflict areas are at locations (4, 3), (12, 12), (20, 3), (4, 12), and (20, 12). Results of 10 simulations for position 9 were presented in Table 6.18. Mean, standard deviation, and confidence interval are shown in Figure 6.12.

The calculated mean, standard deviation, and confidence interval are,

Mean = 346.3

Standard Deviation = 71.88

Confidence Interval = (301.75, 390.85)

Table 6.18. Performance Results for Position 9

Number of runs	Total simulation time (Seconds)
1	415
2	380
3	374
4	395
5	185
6	337
7	418
8	320
9	274
10	365

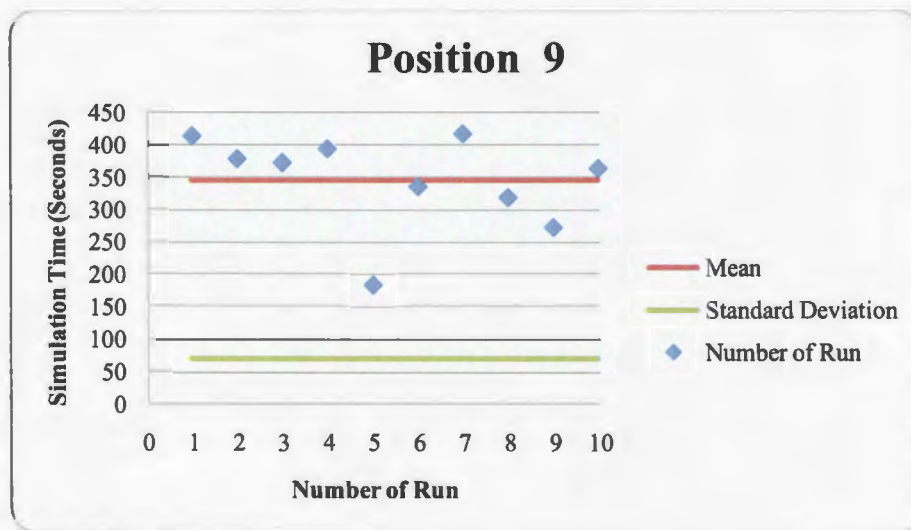


Figure 6.12. Position 9, Mean, Standard Deviation, Confidence Interval

6.4.10. Ten Simulations for Position 10

Resource units are at locations (4, 7), (12, 7), (12, 12) and (20, 7). Conflict areas are at locations (4, 3), (12, 3), (20, 3), (4, 12) and (20, 12). Results of 10 simulations for position 10 were presented in Table 6.19. Mean, standard deviation, and confidence interval are shown in Figure 6.13.

The calculated mean, standard deviation, and confidence interval are,

Mean = 728.4

Standard Deviation =137.06

Confidence Interval = (643.45, 813.35)

Table 6.19. Performance Results for Position 10

Number of runs	Total simulation time (Seconds)
1	577
2	914
3	630
4	759
5	854
6	586
7	853
8	644
9	588
10	879

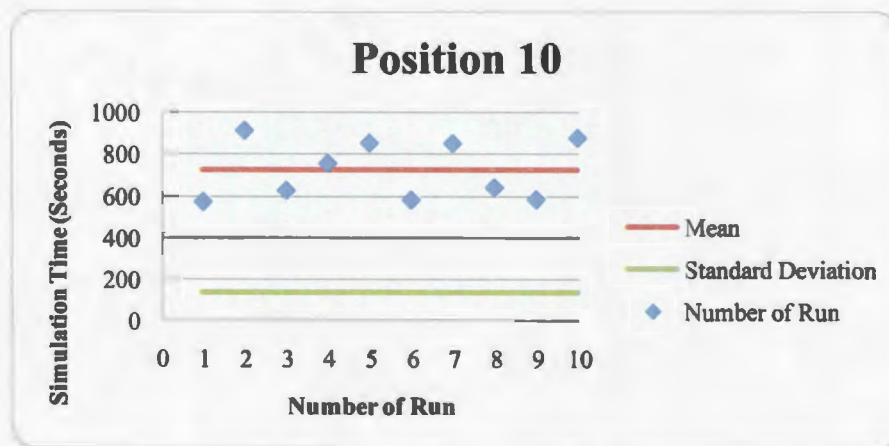


Figure 6.13. Position 10, Mean, Standard Deviation, Confidence Interval

From above results we can infer that, positions 1 and 8 are able to solve the problem in less time.

7. CONCLUSIONS AND FUTURE WORK

We conclude the following points from our research

- The ant heuristic search approach, with adapted features of Dynamic Source Routing protocol, can be applied to solve problems such as the Neutralization of Conflict Areas. This kind of approach is unique, and nobody has previously applied this type of solution to the problem.
- Unlike traditional ant applications, to solve this problem we have used two types of ants: explorer ants and worker ants. Each type of ant has different concerns. The explorer ant's responsibility is to find routes, and the worker ant's responsibility is to form the shortest paths, and neutralize conflict areas.
- Our solution is heuristic in nature. It guarantees that the solution is close to optimal and that the paths found are short paths.
- As the number of resource units and conflict areas increase, the time taken by the simulator to solve the problem will be more.
- Our solution can neutralize any number of conflict areas provided with the required number of resource units in a reasonable time using short paths to conflict areas in order to neutralize and can also get help when there is a need.
- Our procedure mimics real ant behavior finding short paths to conflict areas using the ant pheromone concept.
- The exploration of the search area for conflict zones is very efficient because of adapted features from Dynamic Source Routing protocol's *route discovery mechanism*. This mechanism makes sure that every resource is used very effectively unlike in traditional ant applications where ants wander the search space randomly

and where there are chances of getting lost in the search space. In our solution every ant is under the control the of resource unit range.

- To a certain number of resource units and conflict areas, the problem-solving time can be minimized by varying performance-tuning parameters such as ant speed, resource unit range, explorer ants, etc.
- For a finite grid, results were consistent for a certain number of resource units and conflict areas. In this paper we showed results for a 23 by 12 hexagonal grid with 4 resource units, 3 dynamic conflict areas, and 2 static conflict areas. We ran this test case over 10 times. In all cases, results were consistent in finding the shortest path length to conflict areas, time to neutralize them.
- We established a framework that would allow for experimentation with a wide variety of inputs.

As we said, we can tune the performance of the application by varying the inputs.

We can reduce the complexity of the solver exponentially, and we also documented experimental results by varying those parameters which are critical for application performance.

For enhancements to this application, we can introduce obstacles which are like unmovable paths in the map and then make ants to detect and avoid the obstacles when reaching conflict areas. It would be good if there were some mechanism to dynamically increase the severity of conflict areas, for example, dynamically changing a static conflict area to a dynamic conflict area or vice versa. Another suggested enhancement is to counter attack the resource units by conflict areas.

With these enhancements, this application can be more useful in military applications such as the neutralization of rioters' conflict areas. Another potential field is gaming applications. The application is scalable to similar kind of applications.

REFERENCES

1. Marco Dorigo, Mauro Birattari, and Thomas Stutzle. (2006). Ant Colony Optimization: Artificial Ants as a Computational Intelligence Technique. Universite Libre de Bruxelles, Belgium, IRIDIA.
2. Marco Dorigo, Vittorio Maniezzo and Alberto Coloni. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. Universite Libre de Bruxelles, Belgium, IRIDIA.
3. Thomas Stutzle and Holger Hoos. (2000). Max-Min Ant System. Universite Libre de Bruxelles, Belgium, IRIDIA.
4. H. Van Dyke Parunak, Sven S. Brueckner, Robert Matthews and John Sauter. (2005). Swarming Methods for Geospatial Reasoning. Altarum Institute, 3520 Green Court, Suite 300, Ann Arbor, MI.
5. Heitor S. Lopes, Vilson L. Dalle Molle, and Carlos R. Erig Lima. (2005). An Ant Colony Optimization System for the Capacitated Vehicle Routing Problem. Av. 7 de setembro, 3165, 80230-901, Curitiba (PR) – Brazil.
6. David B. Johnson, David A. Maltz and Josh Broch. (2001). DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks. Carnegie Mellon University Pittsburgh, PA.
7. Joseph Moses, Karl Alterberg and Kendall Nygard. (2009). Strategies for Neutralizing Emergent Conflict Areas. *First International Conference on the Dynamics of Information Systems*. Gainesville, FL.
8. Surjeet Banga. (2008). Solving Prize Collecting Travel Salesman Problem Using Ant colony optimization. (Master's paper, North Dakota State University).