OPTIMIZATION OF MOBILE SENSOR MOVEMENT IN SELF-HEALING SENSOR NETWORKS

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

By

Samidip Basu

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

Major Department:
Computer Science

November 2010

Fargo, North Dakota

North Dakota State University
Graduate School

Title

OPTIMIZATION OF MOBILE SENSOR MOVEMENT

IN SELF-HEALING SENSOR NETWORKS

By

SAMIDIP BASU

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

ABSTRACT

Basu, Samidip, M.S., Department of Computer Science, College of Science and Mathematics, North Dakota State University, November 2010. Optimization of Mobile Sensor Movement in Self-Healing Sensor Networks. Major Professor: Dr. Kendall Nygard.

This paper moves forward the key idea as proposed in past research works – a self-healing deployment approach for sensor networks, where a small percentage of mobile sensors are deployed along with the static sensors into a field of concern. Mobile sensors can move to make-up for a coverage holes or sensor failure and significantly boost network performance. However, since there are energy constraints on each individual mobile sensor, potentially receiving multiple requests from network holes, the decision to move a mobile sensor has to be optimum, one that maximizes network benefit. In this paper, I propose a hybrid distributed & central decision making algorithm to facilitate optimal moves by each mobile sensor. The algorithm uses several layered techniques like Rough Set analysis, sorting & multi-level auction to provide the best possible decision, given the network scenario and the approach is robust to incompleteness of information. The proposed solution also safeguards against network deadlocks and extensive simulations & statistical analysis have demonstrated superior performance of the algorithm when compared to its peers. Some traits of the algorithm proposed derive inspiration for decision support from Ants' swarm intelligence.

ACKNOWLEDGMENTS

This research work would not have been possible without the guidance of my major advisor, Dr. Kendall Nygard throughout the duration of my graduate program. The thesis is dedicated to my parents for their unwavering support all my life. My wife has played a pivotal role in keeping me sane throughout as I tried to balance work & studies, and she fills my everyday life with love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

Wireless sensor networks have gained prominence in a variety of fields in the recent past. Any such network consists of a number of small sensor nodes that perform sensing tasks of various natures and transmit appropriate information back to a central Base Station [1]. The use of such sensor networks has been increasing in the fields of environmental monitoring, military surveillance & target tracking [1]. With increased usage, the effectiveness of such networks is under increased scrutiny, especially sensor coverage on the network field [3]. Self-healing sensor networks would be the primary focus area of this research thesis. While the solution proposed applies to the field of sensor network coverage, the same principles could be generalized to be applied to broader optimization problems, as discussed henceforth.

In a typical sensor network, as sensors are being deployed on the field, the location of each sensor node cannot be precisely controlled due to their large number, hostile environment, method of distribution etc. As such, geographical terrain might have a serious effect on network coverage and coverage holes will degrade network performance. [1] & [2] proposed a means of increasing network effectiveness by deploying a small proportion of mobile sensors, along with the static sensors in a sensor network. This approach aims at keeping costs low, while trying to maximize self-healing of the sensor network. The core idea is that the small proportion of mobile sensors can greatly increase network effectiveness by moving to coverage holes on detection. The decision to move by each mobile sensor is based on several factors like distance from coverage hole, connectivity gain, coverage gain & overall network benefit. Since the mobile sensors could be limited in energy, several central & distributed approaches have been researched in deciding optimal movement by each mobile sensor. This research work would be aimed at better solving the optimization problem of limited moves by the mobile sensors in a self-healing sensor network.

The following research proposes a new algorithm for a given mobile sensor to make a smart decision on which coverage hole it wants to move to cover, with the Base Station keeping track of movement by other mobile sensors so as to maximize network benefit. The problem under consideration essentially turns out to be an optimization decision, given some network constraints. The proposed solution may easily be used to effectively aid in decision making for most other optimization problems where the decision-making is based on several factors. Given such a scenario, the solution proposed could adapt by simply changing the factors under consideration and attempt to provide an optimal solution. One possible application that readily lends itself to be solved by the proposed solution is to effectively decide how cop patrol cars should visit various troubled neighborhoods to maximize police coverage.

INTRODUCTION

The broad research area in the field of self-healing sensor networks was chosen to be the primary focus for the research thesis. To provide self-optimization and self-healing capabilities in a sensor network, the proposal is to deploy a small proportion of mobile sensors along with the large number of the static sensors [1], [2]. The static sensors are stationary once deployed; but they can talk to their neighbors. Another assumption is that the sensors are Location aware [1] and can sense holes in network coverage. Upon detection of such holes, they would broadcast requests for the neighboring mobile sensors to move in to cover the coverage holes and repair the network.

Each mobile sensor may get multiple requests to cover network holes; but due to energy constraints, we will assume that each mobile sensor can make only one move and only one mobile sensor moves to a hole. Hence, it becomes an optimization problem as to which mobile sensor should move to which hole to provide the maximum benefit to the network. Several entirely centralized schemes have been discussed as in [1] that would help mobile sensors take a decision; but these depend largely on the base station and may not scale very well. I propose a largely distributed algorithm here, with base station involvement kept at a minimum and only when required.

Several ideas were to be investigated as in:

- ✓ Iterative use of Rough Sets in distributed decision making
- ✓ Sorting of best-fit requests
- ✓ Smart multi-level auction protocol
- ✓ Gain inspiration for decision support from Ants' swarm intelligence
- ✓ Optimization through ILP for central decision making

Of these, some appeared to be more promising than others. The proposed solution is for the mobile sensors to be distributed in their decision making as far as feasible. On receiving requests from static sensors to cover network holes, the mobile sensors would apply rough set analysis iteratively to filter out requests that are not at all potentially beneficial. Of the rest, the mobile sensors would apply greedy algorithms to figure out which is the best request for them to serve and communicate with the base station at a minimum, unless conflicts arise between two neighboring mobile sensors about the best request they have chosen to serve. The base station would employ a two-level auction to make choices that would result in maximum network benefit. In all of this, I derive inspiration and mimic the behavior of ants as in ant colony optimization.

ROUGH SETS ANALYSIS

"The main goal of rough set analysis is induction of approximation of concepts" [4, p.4]. Its application can be beneficial in the case of distributed decision-making and rule generation by mobile sensors when they are to filter out requests to move in order to cover holes in the sensor network coverage.

Part I: Consistent Data

Information System/Table:

Let us consider an information table (as in Table 1) for our domain, which may serve as a basis for rough set analysis. The cases represent the requests one mobile sensor may get from neighboring static sensors/sensor groups indicating network holes. These requests would include GPS position of the hole and other necessary details for a mobile sensor to make an educated decision. As attributes, I consider several characteristics of each request [1] as they relate to the particular mobile sensor. These would be calculated on the fly based on the position of the mobile sensor in the network. I realize that not all of this information may be available and would consider incomplete data in subsequent sections. The following table shows 10 possible random requests that one mobile sensor might receive from neighboring static sensors & corresponding attribute values.

Table 1: Request attribute values from sensor network holes

Attributes				
Cases	*Distance*	*Coverage Gain*	*Connectivity Gain*	*Size of Hole*
1	S	M	S	L
2	S	M	L	M
3	M	S	S	M
4	S	L	S	S
5	L	L	M	L
6	M	S	M	S
7	L	M	L	M
8	S	S	S	L
9	L	S	M	S
10	M	L	M	S

Legend explaining Table 1:

- S = Small
- M = Medium
- L = Large
- Distance = Distance between present location of mobile sensor and hole.
- Coverage Gain = Net area covered by the mobile sensor after the move to the corresponding sensor hole.
- Connectivity Gain = Number of new links if and after the given Mobile sensor moved to the corresponding network hole.
- Size of Hole = Number of static sensors missing in hole.

The values of these attributes would be calculated based on the request and the present position of the mobile sensor with respect to the hole in the sensor network's coverage. There would be well-defined formulas to calculate each of the attribute values, some of which are described in [1]. Whether an attribute value would be designated as Small, Medium or Large may be controlled by some threshold values as set by the network administrator. Also, there could be a possibility of increasing the granularity of these attributes values (For e.g.: Small^{+}, Small^{-} etc.), as demonstrated in the test field used for data sampling in measuring the efficacy of the algorithm proposed. This can be achieved without changing any of the rough set analysis to follow; however, the rules generated would be different based on the attribute values. Also, I assume all attributes to have equal weight; future research may be targeted at tweaking the attribute weights based on impact on sensor network to further optimize the decision making for mobile sensors.

Let:

U = set of all cases, also called the Universe

A = set of all attributes

B = a non-empty subset of A

V = set of all attribute values

$U \times A \rightarrow V$: an information function

For each $a \in A$, $U \text{->} V_a$, the Value set of a. [4]

I now define some samples of rough set constructs as they relate to our information system:

Blocks of Attribute-value pairs:

- [(Distance, S)] = {1,2,4,8}
- [(Connectivity Gain, M)] = {5,6,9,10}

Elementary sets of singletons:

$[1]_{\{Distance\}} = [2]_{\{Distance\}} = [4]_{\{Distance\}} = [8]_{\{Distance\}}$ = [(Distance, S)] = {1,2,4,8}

Elementary sets of subsets of A:

$[1]_{\{Distance, Coverage Gain\}} = [2]_{\{Distance, Coverage Gain\}}$ =

[(Distance, S)] ∩ [(Coverage Gain, M)] = {1,2}

Indiscernibility Relation:

Let B be a non-empty subset of A. The Indiscernibility relation IND(B) is a relation on U defined for x,y Є U such that

(x,y) Є IND(B) iff $V_{(x,a)} = V_{(y,a)}$ for all a Є B [7].

The corresponding partition on U will be denoted by B*. For example:

- {Distance}* = { {1,2,4,8}, {3,6,10}, {5,7,9}}
- {Distance, Coverage Gain}* = {{1,2,5,8}, {3}, {4}, {6}, {7}, {9}, {10}}
- IND({Distance, Coverage Gain}) = {{1,2}, {5}, {8}}

The indiscernibility relation IND(*B*) is an equivalence relation. Equivalence classes of IND(*B*) are called *elementary sets* and are denoted by $[x]_B$.

Reducts:

A subset B of the attribute set A is a reduct iff B*=A* and B is minimal with this property [5]. For example:

{Distance, Coverage Gain}* = {{1,2,5,8}, {3}, {4}, {6}, {7}, {9}, {10}}

≠ A* = {{1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}}

However,

{Distance, Coverage Gain, Connectivity Gain}* =

{{1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}} =

A* = {{1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}}, and hence, a reduct.

Decision Table:

The attribute values in the Information System can be looked upon as independent attributes as the value of one can be calculated irrespective of the value of the other attributes. However, rough set analysis supports some dependent attributes whose values would be calculated based on the values of the independent attributes [6]. This is the basis of a decision support system where decisions could be taken based on the values of few input parameters.

Here, I consider a decision attribute, which is the dependent attribute to be calculated, based on the values of the other attributes. This decision attribute is to mean whether a mobile sensor considers the request to move to cover a network hole somewhat beneficial. The idea here is not to have the rough set analysis help make a final decision as to whether a mobile sensor should move or not; but rather help in ranking all the requests a mobile sensor has got from neighboring holes.

Initially, the rough set analysis is going to look at all the request attributes and flag a request as 'Yes' or 'No' for the decision attribute. This would indicate that based on the rules established by the rough set analysis, a mobile sensor is able to throw away a few requests immediately as the decision attribute to move or not is a 'No'. The requests of interest are the ones that have a decision value of 'Yes'. This would suggest that based on initial evaluation; there is recommendation for the mobile sensor to cater to these requests by moving itself to the corresponding hole. Now, there may be multiple requests that yield a 'Yes' for the decision attribute. To decide which one of these requests is the best one for the mobile sensor to serve, I would apply separate algorithms to sort these requests for their desirability. This would be discussed in subsequent sections.

Based on the sample requests to a Mobile Sensor from Table 1, following are the corresponding computed decision values as shown in Table 2. This, as discussed, is not the final decision; rather an initial screening of requests. Network administrators may optimize decision-making by tweaking the thresholds which guide the Decision Value calculations, based on the attribute values for all the concerned requests to a given mobile sensor.

Table 2: Decision values for given request attributes values

Attributes					*Decision*
Cases	*Distance*	*Coverage Gain*	*Connectivity Gain*	*Size of Hole*	*Move to Hole*
1	S	M	S	L	Y
2	S	M	L	M	Y
3	M	S	S	M	N
4	S	L	S	S	Y
5	L	L	M	L	Y
6	M	S	M	S	N
7	L	M	L	M	Y
8	S	S	S	L	N
9	L	S	M	S	N
10	M	L	M	S	Y

Table 2 shows what the computed Decision Values might look like for the given set of attribute values for requests to a certain Mobile sensor.

Concepts:

Any elementary set of {Decision} is called a concept, which induces partitioning of the Universe. In our decision table, concepts are:

[(Move to Hole, Y)] = {1,2,4,5,7,10}

[(Move to Hole, N)] = {3,6,8,9}

Global Covering:

Let d be a decision. A subset B of attribute set A is a global covering if and only if B* ≤ {d}* and B is minimal with this property. For example:

{Distance} is not a global covering because

{Distance}* = { {1,2,4,8}, {3,6,10}, {5,7,9}}
{Move to Hole}* = {{1,2,4,5,7,10}, {3,6,8,9}}

However, {Distance, Coverage Gain, Connectivity Gain} is a global covering because

{Distance, Coverage Gain, Connectivity Gain}* =
{{1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}, {10}} ≤
{Move to Hole}* = {{1,2,4,5,7,10}, {3,6,8,9}}

Minimal Complex:

Let X is a subset of U. Let T be a set of attribute-value pairs t=(a,v). Set **T** is a minimal complex of X iff X depends on **T** and no proper subset *S* of *T* exists such that *X* depends on *S*. For example:

A minimal complex for [(Move to Hole, Y)] = {1,2,4,5,7,10} is

{(Distance, S), (Coverage Gain, M)}

Local Covering:

Let Γ be a non-empty collection of sets of attribute-value pairs. Γ is a local covering of X iff:

Each member T of Γ is a minimal complex of B, U {T | TЄΓ} = X and Γ is minimal. For example:

Local covering for [(Move to Hole, Y)] = {1,2,4,5,7,10} is

Γ = {{(Distance, S), (Coverage Gain, M)}, {Coverage Gain, L}

{Distance, L}, {Distance, M}}

Some induced rules are:

{(Distance, S), (Coverage Gain, M)} → [(Move to Hole, Y)]

{Coverage Gain, L} → [(Move to Hole, Y)]

{Coverage Gain, S} → [(Move to Hole, N)]

These rules could be used in assigning weights to the Attribute values in arriving at certain Decision values.

Part II: Inconsistent Data

Now, based on how decision values are derived, we might end up having inconsistencies in the Decision values, as demonstrated in Table 3. While this may be a possible outcome in other systems, I will not be considering such a possibility, as explained below.

{Distance, Coverage Gain, Connectivity Gain, Size of Hole}* = {1, 2}

{Move to Hole}* = {{1}, {2}}

Hence, the inconsistency is that

{Distance, Coverage Gain, Connectivity Gain, Size of Hole}* ≠ {Move to Hole}*.

Table 3: Inconsistent decision values for given attributes

Attributes					*Decision*
Cases	*Distance*	*Coverage Gain*	*Connectivity Gain*	*Size of Hole*	*Move to Hole*
1	S	M	S	L	Y
2	S	M	S	L	N

I do not consider the rough set analysis on inconsistent data as this is not a possibility for the case of deriving the decision attribute value for mobile sensors. There would be well-defined rules that would be used to generate the dependent attribute value from a given set of independent attribute values. For two requests that have the exact same values for all of the independent attributes, the decision attribute value would be identical in the given context.

Part III: Incomplete Data

In an incomplete decision table, some assumptions are:

- All decision values are specified
- Lost values are denoted by '?'

- 'Do Not Care' conditions are denoted by '*'
- For each case, at least two attribute values are specified

Blocks of attribute-value pairs:

- If for an attribute a, there exists a case x such that ρ(x,a) = ?, then the case x should not be included in any block [(a,v)] for all values v of attribute a.
- If for an attribute a, there exists a case x such that ρ(x,a) = *, then the case x should always be included in any block [(a,v)] for all values v of attribute a.

For example:

[(Distance, S)] = {1,2,4,8,10}

[(Size of Hole, M)] = {2,3,7,10}

Incomplete Decision Table:

Decision-making systems may not always have complete information on all independent attributes that make up the data set, as demonstrated in Table 4 below. Intelligent systems may have fault tolerance built-in that allows for computing the Decision values, even in the absence of complete information.

Table 4 shows the same information system table with attribute values; but with incomplete information. The "?" indicates unknown value and the "*" indicates don't-care conditions.

Table 4: Decision values with incomplete attribute information

Attributes					*Decision*
Cases	*Distance*	*Coverage Gain*	*Connectivity Gain*	*Size of Hole*	*Move to Hole*
1	S	?	S	L	Y
2	S	M	L	M	Y
3	?	S	S	M	N
4	S	L	S	S	Y
5	L	L	?	L	Y
6	M	S	M	S	N
7	L	M	L	M	Y
8	S	S	S	L	N
9	L	S	M	S	N
10	*	L	M	*	Y

Characteristic Sets:

Incomplete decision tables are described by characteristic relations instead of indiscernibility relations. Also, elementary sets are replaced by characteristic sets. Characteristic set $K_B(x)$ is the intersection of blocks of attribute-value pairs (a,v) for all attribute a from B for which $\rho(x,a)$ is specified and $\rho(x,a)=v$ [7]. For example:

$K_A(1) = \{1,2,4,8,10\} \cap \{1,3,4,8\} \cap \{1,5,8,10\} = \{1,8\}$

$K_A(2) = \{1,2,4,8,10\} \cap \{2.7\} \cap \{2,7\} \cap \{2,3,7,10\} = \{2\}$

$K_A(3) = \{3,6,8,9\} \cap \{1,3,4,8\} \cap \{2,3,7,10\} = \{3\}$

$K_A(4) = \{1,2,4,8,10\} \cap \{4,5,10\} \cap \{1,3,4,8\} \cap \{4,6,9,10\}= \{4\}$

$K_A(5) = \{5,7,9,10\} \cap \{4,5,10\} \cap \{1,5,8,10\} = \{5,10\}$

$K_A(6) = \{6,10\} \cap \{3,6,8,9\} \cap \{6,9,10\} \cap \{4,6,9,10\} = \{6\}$

$K_A(7) = \{5,7,9,10\} \cap \{2,7\} \cap \{2,7\} \cap \{2,3,7,10\} = \{7\}$

$K_A(8) = \{1,2,4,8,10\} \cap \{3,6,8,9\} \cap \{1,3,4,8\} \cap \{1,5,8,10\} = \{8\}$

$K_A(9) = \{5,7,9,10\} \cap \{3,6,8,9\} \cap \{6,9,10\} \cap \{4,6,9,10\} = \{9\}$

$K_A(10) = \{4,5,10\} \cap \{6,9,10\} = \{10\}$

Characteristic Relation R(B):

A relation on U defined for $x,y \in U$ as follows:

$(x,y) \in R(B)$ iff $y \in K_B(x)$. R(B) is reflexive.

Singleton Approximations:

Lower Approximation: $\underline{B}X = \{x \in U \mid K_B(x) \subseteq X\}$ [5]

Upper Approximation: $\overline{B}X = \{x \in U \mid K_B(x) \cap X \neq \emptyset\}$ [5]

Boundary Region: $BN_B(X) = \overline{B}X - \underline{B}X$ [5]

A set is said to be *rough* if its boundary region is non-empty, otherwise the set is crisp [6].

[(Move to Hole, Y)] = {1,2,4,5,7,10}

[(Move to Hole, N)] = {3,6,8,9}

$\underline{A}\{1,2,4,5,7,10\} = \{2,4,5,7,10\}$

$\underline{A}\{3,6,8,9\} = \{3,6,8,9\}$

$\overline{A}\{1,2,4,5,7,10\} = \{1,2,4,5,7,8,10\}$

$\overline{A}\{3,6,8,9\} = \{3,6,8,9\}$

The biggest advantage of using Rough Set analysis in our case of optimization of Mobile sensor movement is the induction of rules (using algorithms like LEM2, a component of LERS based on attribute-value pair blocks [7][8]) in case of incomplete/missing Attribute values for a variety of reasons. Use of rough set approximations allows us to induce Decision values in case the Mobile sensor is missing Attributes for any given request.

SORTING OF RECOMMENDED REQUESTS

Once rough set analysis has been applied, the mobile sensor has a list of requests from other static sensors in the network where a move to cover the corresponding hole could prove beneficial. However under our assumption, the mobile sensor can only move once for energy constraints. Hence, it has to know a way of choosing which request is the best for it. Here, I apply a greedy algorithm to help sort the requests in order of their desirability. Because I do not suspect the number of requests to a mobile sensor to be substantially huge, application of a modified Bubble sort algorithm should work well in the given case, as shown in Table 5. I consider the request list to be a multidimensional array.

Let us assume that a given mobile sensor has short-listed the following requests to be favorable in the initial screening:

Table 5: Short-listed requests for given mobile sensor

Attributes					*Decision*
Cases	*Distance*	*Coverage Gain*	*Connectivity Gain*	*Size of Hole*	*Move to Hole*
1	S	M	S	L	Y
2	S	M	M	S	Y
3	M	L	L	L	Y
4	M	L	M	L	Y
5	S	L	L	L	Y

In order to calculate the relative strength of each request, I assign some numbers to the values of each of the attributes in the request. The sum of these numbers would be used in the evaluation algorithm below with the intention of higher numbers indicating greater desirability. Based on decided granularity of attribute values, the numbers to be used would be:

Least Desirable = 1

Middle Value = 2

Most Desirable = 3

Don't Care condition (*) = maximum = 3

Incomplete value (?) = minimum = 1

Algorithm 1:

```
Procedure bubbleSortRequestList (A[i][j]: request list to be sorted)
Do
    Bool Swapped = false
    For each i in 0 to [length (A) – 2] do:
        Int Current = 0
        Int Next    = 0
        Current += Procedure GetDesirabilityValue (A[i][Distance],
        SmallerBetter)
        Current += Procedure GetDesirabilityValue (A[i][Coverage Gain],
        BiggerBetter)
        Current += Procedure GetDesirabilityValue (A[i][Connectivity
        Gain],BiggerBetter)
        Current += Procedure GetDesirabilityValue (A[i][Size of Hole],
        BiggerBetter)
        Next += Procedure GetDesirabilityValue (A[i+1][Distance],
        SmallerBetter)
        Next += Procedure GetDesirabilityValue (A[i+1][Coverage Gain],
        BiggerBetter)
        Next += Procedure GetDesirabilityValue (A[i+1][Connectivity
        Gain], BiggerBetter)
        Next += Procedure GetDesirabilityValue (A[i+1][Size of Hole],
        BiggerBetter)
        If Next> Current then
            Swap (A[i], A[i+1])
            Swapped = true
        End if
    End for
While Swapped
End Procedure
```

Algorithm 2:

```
Procedure GetDesirabilityValue (x: incoming attribute, y: desirability order)
    If x = 'S' then
        If y = SmallerBetter then
            Return 3
        Else
            Return 1
        End If
    End If
    If x = 'M' then
```

```
        Return 2
    End If
    If x = 'L' then
        If y = SmallerBetter then
            Return 1
        Else
            Return 3
        End If
    End If
    If x = '*' then
        Return 3
    End If
    If x = '?' then
        Return 1
    End If
End Procedure
```

Hence, at the end of applying the above two algorithms, the concerned mobile sensor is able to sort and rank its requests as shown in Table 6. Higher rank indicates favorable requests for the mobile sensor to serve. The algorithms used may be tweaked to serve special conditions and the individuals weights tied to each attribute may also be adjusted to suit various network priorities. If after the sorting algorithm, two requests are ranked the same with matching totals, the mobile sensor randomly picks one.

Table 6: Sorted & ranked (weighted) requests for given mobile sensor

Attributes					*Decision*	*Rank*
Cases	*Distance*	*Coverage Gain*	*Connectivity Gain*	*Size of Hole*	*Move to Hole*	*With Totals*
1	S	M	S	L	Y	4(9)
2	S	M	M	S	Y	5(8)
3	M	L	L	L	Y	2(11)
4	M	L	M	L	Y	3(10)
5	S	L	L	L	Y	1(12)

COMMUNICATION WITH BASE STATION

Each mobile sensor in the sensor network may receive requests from surrounding static sensors indicating there are holes in coverage and requesting the mobile sensor to move to cover that hole & increase network coverage. On reception of each request, the concerned mobile sensor waits *T* seconds (a predefined threshold set by network administrator) for any other requests. If not, it treats this request as the only request it has received and lets the base station know about this. If it receives other requests, then for each new request, the mobile sensor resets its timer clock till it has received all possible requests. Once it has all the requests, the mobile sensor uses Rough Set analysis to throw out the requests that have no/very little potential benefit. Among the existing ones, the mobile sensor could potentially serve any request as the decision attribute value on the decision table is a 'Yes'. However, since we are operating under the assumption that due to energy constraints, each mobile sensor can move only once, it becomes that much more important to take the best educated decision. Hence, each mobile sensor lists all the possible favorable requests and applies the algorithms described above to sort/rank them. At this stage, the mobile sensor has picked the request that is the best for it to serve. If multiple requests have rank a of "1", the mobile sensor would randomly choose one of them as its best request to move to.

However, here there is a possibility of conflicts. The static sensors that detect sensor coverage holes will broadcast their requests and every mobile sensor in the vicinity will pick those up. What if two mobile sensors get the same request and ultimately rank that as their individual best? Although attribute values for the same request would be different based on the relative position of each mobile sensor, this is quite possible an outcome. Now, the two mobile sensors who have the ranked the same request as their best would want to move to the same corresponding hole, resulting in wasted network resources. Hence, there is some need for conflict resolution.

The proposed conflict resolution would be centralized and will be handled by the base station monitoring the sensor network. One reason to propose such a model would be energy considerations of the mobile sensors [2]. If they have to resolve such conflicts, it is going to demand multiple communications between the neighboring mobile sensors and further processing needs. Instead, if the base station handles such conflicts, communication is limited to only one request and response from the concerned mobile sensors to and from the base station.

BASE STATION AUCTION

Once each mobile sensor has ranked some request as its best, it will send a notification to the base station and wait for a response. This request is informing the base station that the mobile sensor considers it most prudent to move to a certain coverage hole and is asking for clearance. On receipt, the base station would check to see if any other mobile sensor also treats the same request (identified by the coverage hole's GPS position or some other unique parameter on the request) as its best. If not, there is no conflict and the base station immediately clears the mobile sensor to move to the corresponding hole by serving that request. This clearance would also be given if the base station does not hear about any conflicts from any other mobile sensor within a certain predefined threshold time. The base station would have some memory to remember which mobile sensors are moving to which hole so that no other mobile sensor in the near future is allowed to move to the same hole.

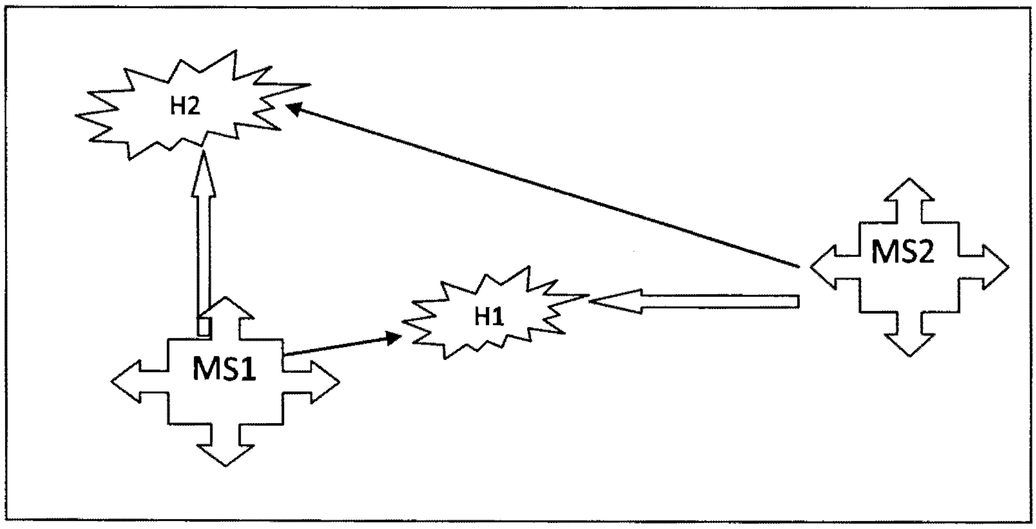
However, if the base station is informed by more than one mobile sensor that they have ranked the same request as their best, then there is a conflict. The base station has to come up with a decision as to which mobile sensor offers the 'best price' for the given request. This basically means an auction among the competing mobile sensors to determine which one is best suited to serve the given request. I utilize a contract net auction protocol that is extended to two levels. This partially addresses the problem of inherent non-linearity of best price auctions [10].

The auctioning is done by a comparative analysis of evaluating benefits as to which mobile sensor moving to which coverage hole has the best impact on network coverage. The simplest form of auction would be the case where each mobile sensor, after sorting its best requests, sends out notification to the base station. This notification is the mobile sensor's bid in case there are conflicts where some other mobile sensor also picks the same hole as its best. The base station applies a greedy algorithm to decide on the best bid and come up with the 'best price' for the request.

On hearing the base station's decision, the winning mobile senor immediately moves to cover the corresponding hole as pertained to its best request. The other mobile sensors cannot move to the same hole and have to 'forget' about their present best request and re-evaluate what's best for them. Hence, they remove the conflicting request from their list, wait for a threshold time for fresh requests and then re-apply rough set analysis and the sorting algorithms to come up with a new best request. This is followed by communication to the base station and the whole cycle repeats itself.

However, following such a greedy algorithm at the base station to determine the best bid in case of conflicts, may not always serve the network best. There could be possibilities where picking the 'best price' choice for a given coverage hole may not lead to an overall optimal solution. Such an example situation is demonstrated as below in Figure 1:

Figure 1: Actual & desired movement of mobile sensors



Legends in Field of concern:

Mobile Sensors = ✥

Network Coverage Holes = ✴

Actual Movement = ⟶

Ideal movement = ⇨

In the given setup, I assume that there are two coverage holes – H1 and H2. Two mobile sensors that are close by are called MS1 and MS2. I also assume that the field of concern does not change between iterations of rough set/sorting analysis, that is, no new holes or mobile sensors show up in the vicinity. All other parameters behaving as they are, our focus is on the 'Distance' parameter between the network holes and their nearest mobile sensors.

After rough set analysis, both MS1 and MS2 rank H1 to be their best request in the area of concern, with H2 being second in rank. Under the simple auction scheme, both MS1 and MS2 will submit a bid to move to H1 and transmit the information to the base station. The base station would apply a greedy algorithm to rank the bids and pick the best price for H1. In this case, all other parameters remaining constant, MS1 would offer a better price because of shorter distance and hence the fitter bid to move to H1. The base station would thus pick MS1 to move to H1 and broadcast the decision. However, MS2 now has to forget about H1 and reiterate through its rough set analysis and sorting of best request. Given the assumption that no holes show up in our field of concern, in the second iteration, MS2 is going to rank H2 as its best request. Being the only mobile sensor in the vicinity, MS2 this time wins the auction and moves to H2. Hence, the final assignment is MS1 to H1 and MS2 to H2.

However, this is clearly not ideal. MS2, winning the auction in its second iteration, has to travel a much longer distance to cover H2 and the sum of the distances travelled by MS1 & MS2 is much higher. The optimal assignment would have been MS1 moving to H2 and MS2 moving to H1. This would have been much more beneficial for network coverage in terms of distance covered. However, it would mean that to make such a decision, the base station cannot be too greedy and get caught up in local optimums. The auction would have to be smart enough to make decisions where the most favorable choice is at times neglected for the overall benefit of the network.

I propose that the base station does a smart auction by being less greedy and evaluate consequences of a decision before informing the mobile sensors. The auction would be extended to two levels and the base station evaluates choices that are little beyond the one with the best price [10]. However, to support this, the mobile sensors would have to do more so that the base station can take smarter decisions. At each iteration, the mobile sensor applies rough set analysis followed by sorting to figure out which request is its best. It then communicates its decision to the base station by sending out information about its best request. For the base station to be smarter in its decision-making, I propose that each mobile sensor send out *two* of its best requests (if available) in each iteration, as shown in Table 7 below. This would allow the base station to extend the auction process to two stages. Given that the base station knows about each mobile sensor's best and second best requests, it can afford to look beyond the best price for each request. This has the potential of providing more optimized decisions for improving network coverage.

In case of the given example, both MS1 and MS2 would send two of their best requests to the base station, those being H1 and H2. To the base station, the information could look somewhat like as follows:

Table 7: Best two mobile sensor decisions in auction

MS1	Best Request (H1)	Total = 12
	Second Best (H2)	Total = 10
MS2	Best Request (H1)	Total = 8
	Second Best (H2)	Total = 2

Here, the totals indicate desirability values as utilized by the mobile sensors to sort their requests such that higher numbers mean better. Here, it is clear to the base station that the best request of MS1 is fitter than the best request of MS2, as both pertain to the same coverage hole. MS1, thus presents, a better price for serving that request. However, considering the assumption that nothing changes in the field of concern, the second best choice for MS2 is a rather bad one. To avoid these situations, the base station extends the auction to two levels and considers second best options.

In this case, even though MS1 wins the bid at start, the base station considers the second best requests of both MS1 and MS2. Here, the network administrator can set a threshold value that the base station utilizes to evaluate the relative strengths of the second best requests. Let us consider that threshold value be 7 in terms of desirability numbers. I propose that the base station compare the desirability of the second best requests of each conflicting mobile sensor and reverse its decision if the difference is more than the threshold value. In this case, the second best requests of the conflicting sensors MS1 and MS2 differ by more than 7, indicating that the second choice for MS2 is much worse as compared to that of MS1. Given such circumstances, the base station neglects the initial winning bid of MS1 and awards coverage hole H1 to MS2 for the overall benefit of the network. Hence, the final assignment becomes MS1 to H2 and MS2 to H1.

ITERATIVE APPLICATION OF ROUGH SETS

Once a mobile sensor picks a best request for it to serve, but gets turned down in auctioning from the base station, the mobile sensor must 'forget' about its decision. It must then re-evaluate existing requests after waiting a threshold amount of time for newer requests. This would mean re-application of Rough Set analysis to filter out favorable requests and applying the sorting algorithms to evaluate potential benefit of each request against others. Here, Rough Set analysis is being revisited and acts as sort of a controller for the lower level search mechanism as in Tabu Searches.

With a single pass of the Rough Set analysis, there could have been a possibility that requests that are deemed unfit at the start through the filtering, never actually have a chance to get served. This would be because the concerned mobile sensor, based on the request's attribute values has perpetually blocked the request. To get around this potential deadlock situation, I suggest re-application of Rough Set analysis by the mobile sensor once it loses out on its best price auction bid with the base station. To avoid having the same results again, the Rough Set analysis is applied with refined thresholds after each iteration. The idea is to be less aggressive in throwing out requests on each pass.

Rough Set analysis is based on evaluating requests for attribute values as they pertain to the concerned mobile sensor. On successive iterations, the network administrator sets up the Rough Set mechanism to be less aggressive and more optimistic than the last application. This is where the granularity of attribute values comes on handy. As mentioned before, I consider nine possible attribute values as in: [Small$^+$, Small, Small$^-$], [Medium$^+$, Medium, Medium$^-$] and [Large$^+$, Large, Large$^-$]. On each successive iteration, the Rough Set analysis assigns attribute values to be one better than the last run, if feasible. For example, if the Distance attribute for a request was assigned a 'Medium$^+$' on first try, the second Rough Set pass is going to assign it to be 'Medium', as shorter distances are better. Another possible solution could be being more optimistic about a request in successive passes by auto-incrementing the ranking score for the given request to the concerned Mobile sensor. With multiple Rough Set passes, the hope is that the request would not be deadlocked and would eventually be considered fit enough for the Decision attribute value to be a 'Yes'. Beyond this point, each mobile sensor begins its own sorting and things roll as usual.

INTEGER LINEAR PROGRAMMING (ILP) SOLUTION

The optimal mobile sensor movement problem in the entire network can also be formulated as an Integer Linear Programming (ILP) problem [1[. The formulation seeks the global optimal solution of the mobile sensor movement problem, given a test field with static sensors broadcasting coverage hole details. The ILP formulation is presented as follows.

Mathematical Model:

Given:

M = *Set of Mobile Sensors*

H = *Set of Coverage Holes*

C_{ij} = *Connectivity Gain if* M_i *moves to* H_j;

for each $i \in M$ *and* $j \in H$

S_{ij} = *Size of Hole as* M_i *moves to* H_j;

for each $i \in M$ *and* $j \in H$

D_{ij} = *Distance to be covered by Mobile Sensor if* M_i *moves to* H_j;

for each $i \in M$ *and* $j \in H$ [1]

Define Variables:

X_{ij} = *1, if* M_i *moves to* H_j ; *for each* $i \in M$ *and* $j \in H$

= *0, otherwise*

Maximize Network Gain:

$$\sum_{i \in M, j \in H} X_{ij} * (C_{ij} + S_{ij} - D_{ij})$$

Subject To:

$$0 <= X_{ij} <= 1$$

$$1 <= C_{ij} <= 9$$

$$1 <= S_{ij} <= 9$$

$$1 <= D_{ij} <= 9$$

AMPL Formulation:

Based on the above mathematical model, the ILP problem can be formulated & solved through regular ILP solvers. AMPL is a great language for specifying such optimization problems. It provides an algebraic notation that is very close to the way the problem is described mathematically. The separation of model and data is the key to describing complex linear programs in a concise & understandable fashion.

Hence, I set about rewriting the above mobile sensor movement optimization problem as an AMPL Model & the simulation tests provide the data. The NetworkGain Model is described as below:

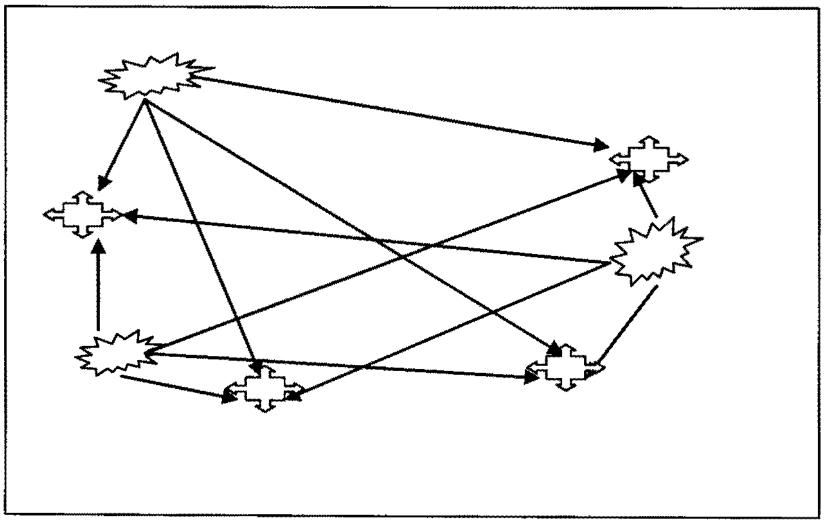
```
set M;
set H;
param C {i in M, j in H};
param S {i in M, j in H};
param D {i in M, j in H};
var X {i in M, j in H};

maximizeNetwork_Gain: sum {i in M,j in H} X [i,j] * (C [i,j] + S [i,j] - D [i,j]);
subject to Limit1 { i in M, j in H }: 0 <= X [i,j] <= 1;
subject to Limit3 { i in M, j in H }: 1 <= C [i,j] <= 9;
subject to Limit4 { i in M, j in H }: 1 <= S [i,j] <= 9;
subject to Limit5 { i in M, j in H }: 1 <= D [i,j] <= 9;
subject to TotalMoves: sum { i in M, j in H } X [i,j] = 4;
subject to MS1Move: sum {j in H} X[1,j] = 1;
subject to MS2Move: sum {j in H} X[2,j] = 1;
subject to MS3Move: sum {j in H} X[3,j] = 1;
subject to MS4Move: sum {j in H} X[4,j] = 1;
```

Standard AMPL commands & Minos Version 5.5 solver are then used to solve the above ILP solution for every simulation of the field test, which provides the data for the AMPL model. The results & its comparison with the proposed algorithm are discussed in the Performance Evaluation section.

INSPIRATION FROM ANT COLONY OPTIMIZATION

“The behavior of ants has long fascinated scientists” [9, p.1]. In recent years, computer science has tried to gain inspiration from the way a colony of ants can solve complex problems; in particular, finding the shortest path from nest to food source [9]. Our model of self-healing sensor network also mimics ant colony optimization techniques, as we start demonstrating in Figure 2 below:

Figure 2: Requests/responses in an ant colony



Legends in Field of concern:

Nest of ants/Mobile Sensors = ✣

Food Source/Network Coverage Holes = ✷

Broadcast Requests = ⟶

"Each ant in a colony operates on its own agenda, and yet the group as a whole appears to be highly organized" [9, p.1]. "Ants form and maintain a line to their food source by laying a trail of pheromone, i.e. a chemical to which other members of the same species are very sensitive. They deposit a certain amount of pheromone while walking, and each ant prefers to follow a direction rich in pheromone. This enables the ant colony to quickly find the shortest route. The first ants to return should normally be those on the shortest route, so this will be the first to be doubly marked by pheromone (once in each direction). Thus other ants will be more attracted to this route than to longer ones not yet doubly marked, which means it will become even more strongly marked with pheromone" [9, p.1].

Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over faster, and thus the pheromone density remains high as it is laid on the path as fast as it can evaporate. Pheromone evaporation has also the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained around a local optimum or greedy solution.

The decision making behaviors of the mobile sensors in our case seems to have a direct analogy to the swarming intelligence demonstrated by the ants in ant colony optimization. Each mobile sensor can be thought of as a nest of ants and the holes in network coverage as food sources. The idea is to establish the shortest path to a food source, in essence picking a request to move to a hole that will prove most beneficial for the network. Each mobile sensor gets multiple requests from neighboring holes, thought of as food sources; and ants belonging to the given nest as in the mobile sensor must find the shortest route to the food. The mobile sensor applying rough set analysis on the requests it has received can be thought of being analogous to ants from a nest looking around randomly at first and trying to determine the shortest path to a food source. Once favorable requests have been chosen, each mobile sensor applies a greedy algorithm to pick the best request; something that could be thought of as ants leaving pheromone trails so that the shortest path is selected eventually.

Once a mobile sensor has picked a request to be its best, it would inform the base station of its decision. If no conflicts occur, then this is the best choice for the mobile sensor and it would move to cover the corresponding network hole. The base station would remember this move and would disallow any other mobile sensors from moving to the same hole in future even if they treat the request as their best. This could be thought of as a family of ants belonging to a given nest releasing pheromones once they have found a shortest path to a food source;

chemicals that would actually repulse ants from other neighboring nests from finding the same food source. If there is a conflict between the mobile sensors and if a neighboring mobile sensor wins, the concerned mobile sensor has to 'forget' about its best request as the other mobile sensor offered a better price for the request. The concerned mobile sensor then drops its best request from its list, waits for a predefined time for fresh requests and starts the process of evaluating requests from the scratch. This could be thought of being analogous to pheromone evaporation on a certain trail of food in the ant world. It is this 'evaporation' that would prevent a local optima; in this case, from two mobile sensors moving to the same hole.

PERFORMANCE EVALUATION

Simulation experiments were designed to evaluate the performance of the proposed rough set analysis and sorting-based distributed decision-making algorithm. Each simulation considered a field test bed where static sensors were deployed over a desired coverage area. To provide self-optimization and self-healing capabilities in a sensor network, the proposal is to deploy a small proportion of mobile sensors along with the large number of the static sensors. Each request to cover a network hole includes details about the coverage hole that is to be covered, which the concerned mobile sensor would use to evaluate the needed attribute values on the fly. These attribute values (as described in sections above) allow the mobile sensor to inspect a request to move to a coverage hole and evaluate the benefit derived from doing so. Based on network administrator-set thresholds, I consider that attribute values of the four factors could range anywhere in between 10 values (Small-, Small, Small+, Medium-, Medium, Medium+, Large-, Large, Large+ & Unknown). In our experiments (sample in Figure 3), these attribute values are generated completely randomly for each request.

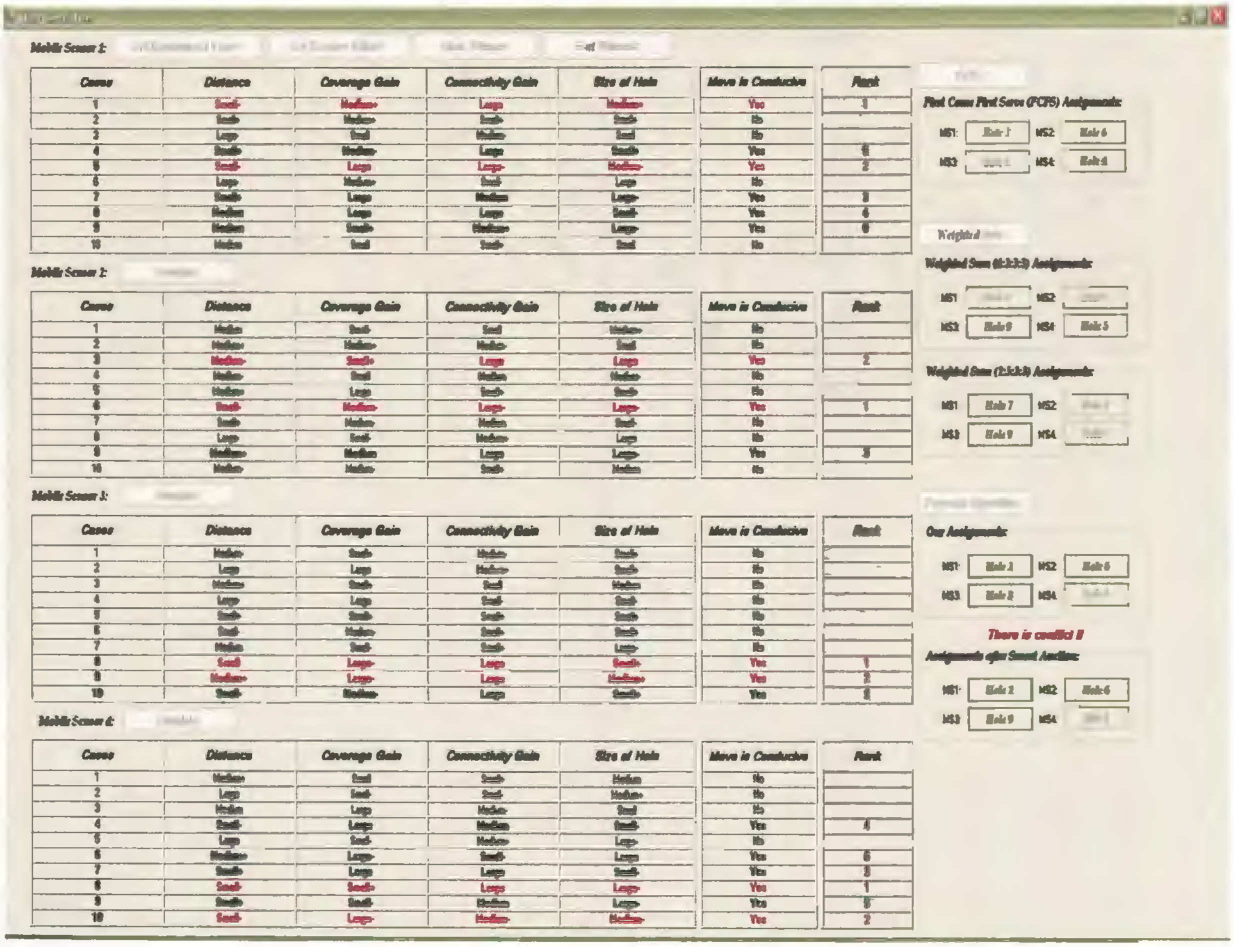
For experimental purposes, the field test comprised of a maximum of 4 mobile sensors in range and up to 10 coverage holes as detected by the static sensors. This test is repeated in 20 completely random simulations to get an accurate average of how the different algorithms fare against each other. Each algorithm comes up what it thinks is the best optimized solution as how the mobile sensors should move, given the coverage holes and their positions.

The performance of the proposed algorithm is compared with two other heuristic algorithms - First Call First Served (FCFS) algorithm and Weighted Sum (WS) algorithm (as in Figure 6), as well as with that of Integer Linear Programming (ILP) solution, as demonstrated in Figure 4 & Figure 5. The two heuristic algorithms are presented below:

FCFS algorithm: "Once a mobile sensor receives a request from a hole, it moves to the hole immediately" [1, p.6]. This would mean that the concerned mobile sensor always serves its first request to cover a network hole. It should also be noted that the assumption is that the request to serve the closest coverage hole gets to a mobile sensor before others.

WS algorithm: A mobile sensor waits for requests for a pre-defined time and then moves to the hole that has the maximum weighted sum of four factors: Distance, Coverage gain, Connectivity gain and Size of Hole [1]. These are the same attributes used in our algorithm. The weight of each factor is given (obtained from extensive simulation experiments). The sets of weights used for the following experiments are: Distance: Coverage: Connectivity: Size = 4:3:3:3 and 2:5:3:3 [1].

Figure 3: A screenshot of the C#.NET simulation program used to evaluate the field test bed and optimize movement for the mobile sensors. The same simulation values are also used to evaluate the solutions from other algorithms for a fair comparison



Mobile Sensor 1:

Cases	Distance	Coverage Gain	Connectivity Gain	Size of Hole	Move is Conducive	Rank
1	Small	Medium	Large	Medium	Yes	1
2	Small	Medium	Small	Small	No	
3	Large	Small	Medium	Small	No	
4	Small	Medium	Large	Small	Yes	6
5	Small	Large	Large	Medium	Yes	2
6	Large	Medium	Small	Large	No	
7	Small	Large	Medium	Large	Yes	3
8	Medium	Large	Large	Small	Yes	4
9	Medium	Small	Medium	Large	Yes	5
10	Medium	Small	Small	Small	No	

Mobile Sensor 2:

Cases	Distance	Coverage Gain	Connectivity Gain	Size of Hole	Move is Conducive	Rank
1	Medium	Small	Small	Medium	No	
2	Medium	Medium	Medium	Small	No	
3	Medium	Small	Large	Large	Yes	2
4	Medium	Small	Medium	Medium	No	
5	Medium	Large	Small	Small	No	
6	Small	Medium	Large	Large	Yes	1
7	Small	Medium	Medium	Small	No	
8	Large	Small	Medium	Large	No	
9	Medium	Medium	Large	Large	Yes	3
10	Medium	Medium	Small	Medium	No	

Mobile Sensor 3:

Cases	Distance	Coverage Gain	Connectivity Gain	Size of Hole	Move is Conducive	Rank
1	Medium	Small	Medium	Small	No	
2	Large	Large	Medium	Small	No	
3	Medium	Small	Small	Medium	No	
4	Large	Large	Small	Small	No	
5	Small	Small	Small	Small	No	
6	Small	Medium	Small	Small	No	
7	Medium	Small	Small	Large	No	
8	Small	Large	Large	Small	Yes	1
9	Medium	Large	Large	Medium	Yes	2
10	Small	Medium	Large	Small	Yes	3

Mobile Sensor 4:

Cases	Distance	Coverage Gain	Connectivity Gain	Size of Hole	Move is Conducive	Rank
1	Medium	Small	Small	Medium	No	
2	Large	Small	Small	Medium	No	
3	Medium	Large	Medium	Small	No	
4	Small	Large	Medium	Small	Yes	4
5	Large	Small	Medium	Large	No	
6	Medium	Large	Small	Large	Yes	6
7	Small	Large	Large	Small	Yes	3
8	Small	Small	Large	Large	Yes	1
9	Small	Small	Medium	Large	Yes	5
10	Small	Large	Medium	Medium	Yes	2

First Come First Serve (FCFS) Assignments:

MS1:	Hole 1	MS2:	Hole 6
MS3:	[illegible]	MS4:	Hole 4

Weighted Sum (1:3:3:3) Assignments:

MS1	[illegible]	MS2	[illegible]
MS3	Hole 9	MS4	Hole 5

Weighted Sum (2:3:3:3) Assignments:

MS1	Hole 7	MS2	[illegible]
MS3	Hole 9	MS4	[illegible]

Our Assignments:

MS1:	Hole 1	MS2	Hole 6
MS3	Hole 8	MS4	[illegible]

There is conflict !!

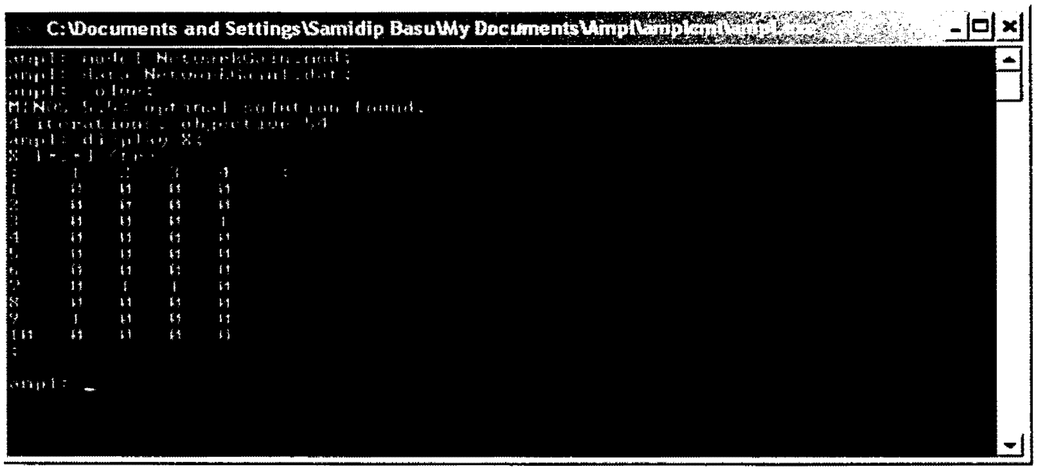
Assignments after Smart Auction:

MS1:	Hole 1	MS2	Hole 6
MS3	Hole 9	MS4	[illegible]

Legend explaining Figure 3:

- Cases = Requests from sensor holes
- Columns = Request Attributes/Decisions/Ranks
- Attribute Values = Random from Uniform Distribution
- MS = Mobile Sensors
- Red/Bold = Favorable Decisions in Proposed Algorithm
- Right of the tables = Decisions by competing algorithms

Figure 4: Screenshot of the same simulation test bed used through AMPL



```
C:\Documents and Settings\Samidip Basu\My Documents\Ampl\amplcml\ampl
ampl: model NetworkGain.mod;
ampl: data NetworkSmall.dat;
ampl: solve;
MINOS 5.5: optimal solution found.
4 iterations, objective 54
ampl: display X;
X [*,*] (tr)
:     1    2    3    4     :=
1     0    0    0    0
2     0    0    0    0
3     0    0    0    1
4     0    0    0    0
5     0    0    0    0
6     0    0    0    0
7     0    1    1    0
8     0    0    0    0
9     1    0    0    0
10    0    0    0    0
;

ampl: _
```

Results of the simulations and comparisons between the Proposed Algorithm & ILP are shown below in Table 8:

Table 8: Comparison of proposed solution to ILP

	Simulations	Proposed	ILP		Legend:	
Rounded Coverage Gain:	1	6	4			
	2	5	4		*Small-*	*1*
	3	6	0		*Small*	*2*
	4	6	6		*Small+*	*3*
	5	6	3		*Medium-*	*4*
	6	5	4		*Medium*	*5*
	7	6	6		*Medium+*	*6*
	8	6	3		*Large-*	*7*
	9	6	4		*Large*	*8*
	10	6	2		*Large+*	*9*
	11	7	4			
	12	6	4			
	13	6	4			
	14	8	4			
	15	6	5			
	16	6	4			
	17	5	6			
	18	4	5			
	19	7	4			
	20	7	5			
Mean		6	4.05			
Variance		0.736842105	1.944736842			
Standard Deviation		0.858395075	1.394538218			

Figure 5: Graphical comparison of performance of proposed algorithm versus that of ILP

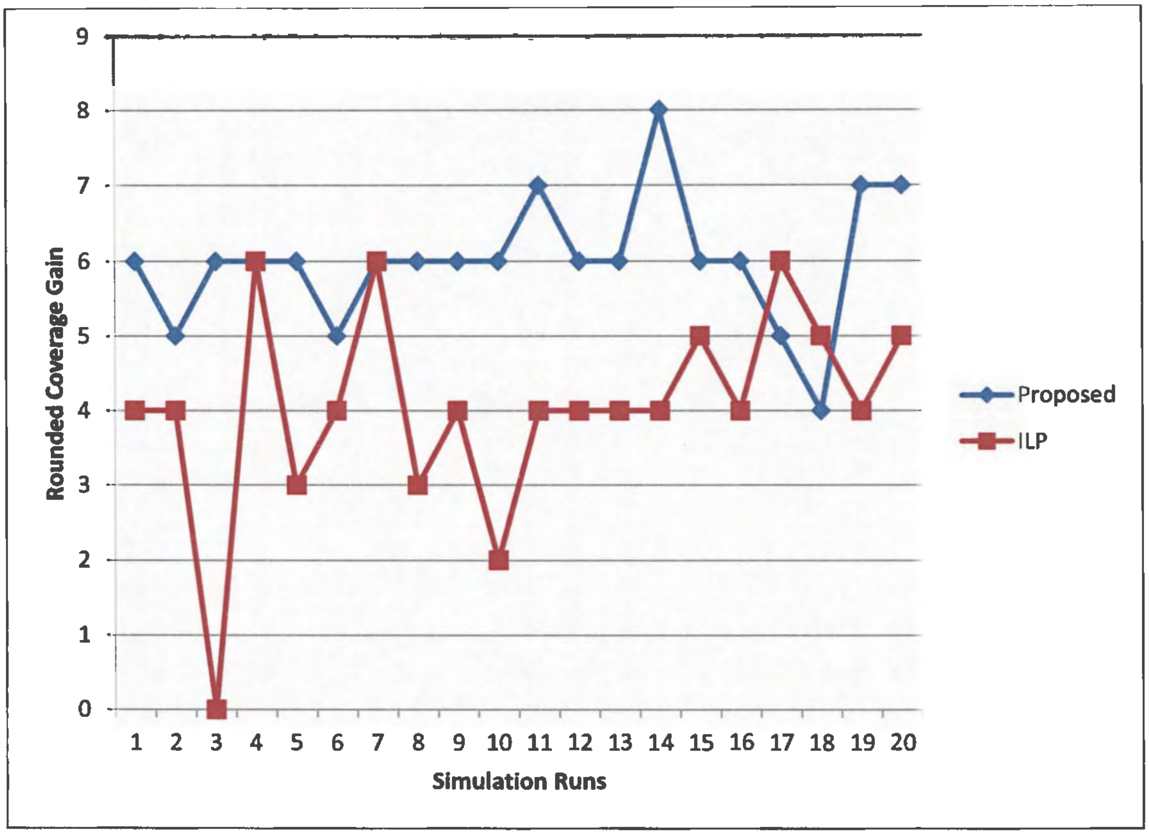


Figure 5 shows a graph of the Average rounded Coverage Gains gained by our sample network when using the Proposed Algorithm versus that of ILP. Also, to note is the fact that computational times were identical & minimal.

A breakdown of the performance comparison, in terms of Rounded Coverage Gain, between the Proposed algorithm and the closest competition is shown as below in Table 9.

Table 9: Comparison of proposed solution to other methods

	Simulations						
Rounded Coverage Gain		Proposed	FCFS	WS(4:3:3:3)	WS(2:5:3:3)	*Legend:*	
	1	6	4	6	6		
	2	5	4	3	3	*Small-*	*1*
	3	6	4	8	8	*Small*	*2*
	4	6	5	7	7	*Small+*	*3*
	5	6	4	5	5	*Medium-*	*4*
	6	5	1	5	6	*Medium*	*5*
	7	6	3	7	7	*Medium+*	*6*
	8	6	2	7	4	*Large-*	*7*
	9	6	2	6	6	*Large*	*8*
	10	6	2	4	4	*Large+*	*9*
	11	7	6	6	7		
	12	6	3	5	6		
	13	6	4	6	5		
	14	8	7	5	7		
	15	6	6	6	6		
	16	6	5	3	5		
	17	5	6	3	5		
	18	4	5	3	3		
	19	7	7	6	6		
	20	7	6	5	5		
Mean		6	4.3	5.3	5.55		
Variance		0.736842105	3.063157895	2.221052632	1.839473684		
Standard Deviation		0.858395075	1.75018796	1.490319641	1.35627198		

Figure 6: Graphical comparison of performance of proposed algorithm versus that of other competing methods

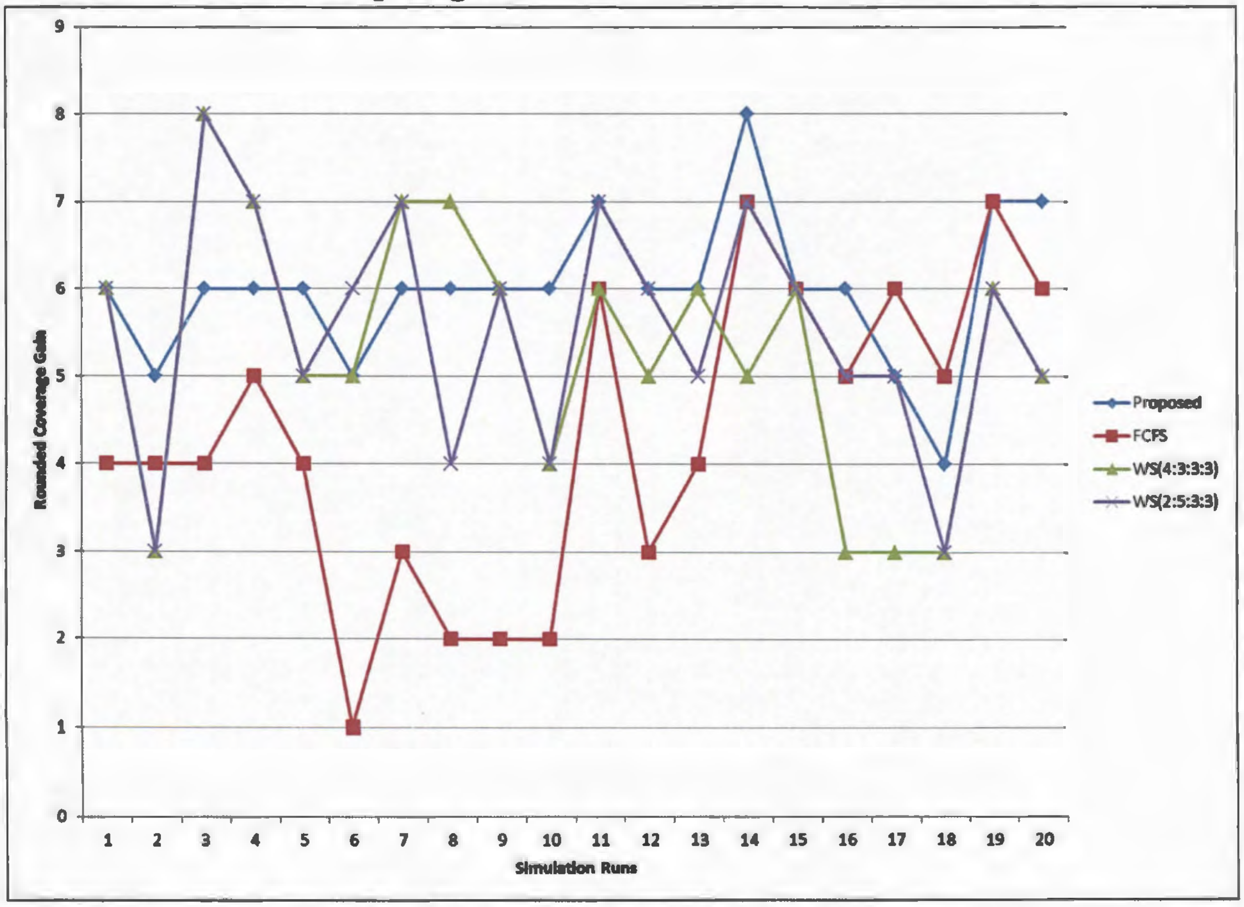


Figure 6 shows a graph of the Average rounded Coverage Gains gained by our sample network when using the Proposed Algorithm versus that of several competing algorithms. Computational times were identical & minimal.

STATISTICAL ANALYSIS OF PERFORMANCE COMPARISONS

Given our simulations that compare performance of the proposed algorithm against that of other algorithms, I would use statistical measures to further increase our confidence about the efficacy of the proposed methodology.

The 10 simulation test runs demonstrate the choices made by the competing algorithms to arrive at their best solutions, measured in terms of one variable – the rounded Coverage Gain of the Network as the concerned Mobile Sensors make their corresponding moves. Since the test bed with the position of the coverage holes is completely random, these 10 simulations can be considered to be a good representative sample of the actual performance of the different algorithms in a much larger distribution. The rounded Coverage Gain of the network in every simulation effectively becomes the statistical variable consistently measuring the performance of different approaches to solve the same optimization problem. Statistics provides various methods in which to evaluate this variable closely and see how well the Coverage Gain represents what happened to each sensor network across simulations.

Also, a large population could be sampled multiple times for better accuracy of results. Each of the samples has an average and a variance associated with it. If we aggregated all of the averages from these samples, we can create a sampling distribution of the Mean for the population. The standard deviation of this distribution is commonly known as the *Standard Error of the Mean* (SEM or, simply, the *Standard Error*, SE). Since our simulation runs for each algorithm are completely random, the samples can be used to compute SEM to have an indicative idea of the dispersion of the overall population.

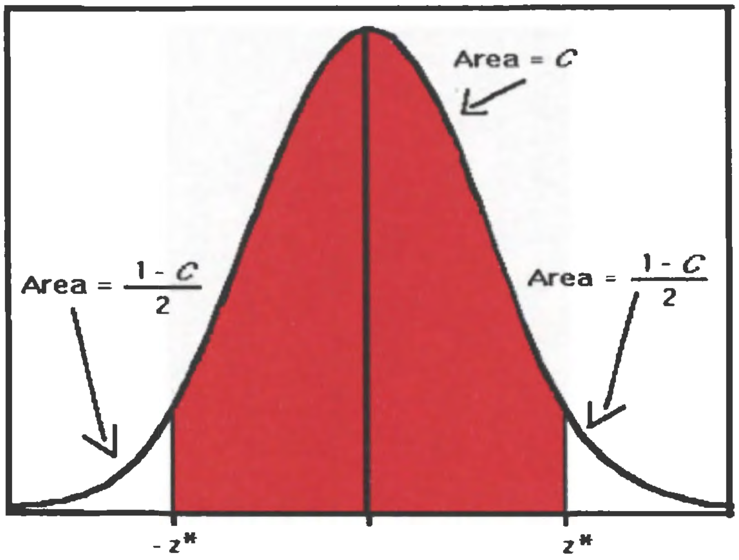
In probability theory and statistics, the normal distribution or Gaussian distribution is a continuous probability distribution that describes data that cluster around a mean or average [12]. The graph of the associated probability density function is bell-shaped (as shown in Figure 7), with a peak at the mean, and is known as the Gaussian function or bell curve. Normal distributions are symmetric around their Mean and are denser in the center, but less dense in the tails of the bell curve [12]. The normal distribution can be used to describe, at least approximately, any variable that tends to cluster around the mean. Closer inspection of the simulation runs in our experiment for every algorithm demonstrates that the rounded Coverage Gain for each run centers around Mean; and hence, the Coverage Gain variable can be described using a normal distribution.

By the central limit theorem, the sum of a large number of independent random variables is distributed approximately normally. For this reason, normal distribution is used throughout statistics, natural science, and social science as a simple model for complex phenomena. A confidence interval gives an estimated range of values, which is likely to include an unknown population parameter, the estimated range being calculated from a given set of sample data.

Per [11], the selection of a confidence level for an interval determines the probability that the confidence interval produced will contain the true parameter value. Common choices for the confidence level [C] are 0.90, 0.95, and 0.99. These levels correspond to percentages of the area of the normal density curve. For example, a 95% confidence interval covers 95% of the normal curve -- the probability of observing a value outside of this area is less than 0.05. Because the normal curve is symmetric, half of the area is in the left tail of the curve, and the other half of the area is in the right tail of the curve. For a confidence interval with level *C*, the area in each tail of the curve is equal to (1-*C*)/2. For a 95% confidence interval, the area in each tail is equal to 0.05/2 = 0.025.

"The value z^* representing the point on the standard normal density curve such that the probability of observing a value greater than z^* is equal to p is known as the upper p critical value of the standard normal distribution. For example, if p = 0.025, the value z^* such that $P(Z > z^*)$ = 0.025, or $P(Z \leq z^*) = 0.975$, is equal to 1.96. For a confidence interval with level *C*, the value p is equal to (1-*C*)/2. A 95% confidence interval for the standard normal distribution, then, is the interval (-1.96, 1.96), since 95% of the area under the curve falls within this interval" [11, p.1].

Figure 7: Bell curve of a normal distribution



(Diagram taken from [11])

Now, what if we have a normal distribution but do not know the standard deviation? This is possible if a random sample of the performance comparisons was chosen out a larger population of simulation runs. We could sample N values and compute the sample mean (M) and estimate the standard error of the mean (s_M) [12]. What is the probability that M will be within 1.96 s_M of the population mean? "This is a difficult problem because there are two ways in which M could be more than 1.96 s_M from mean: (1) M could, by chance, be either very high or very low and (2) s_M could, by chance, be very low. Fortunately, the way to work out this type of problem was solved in the early 20th century by W. S. Gossett who determined the distribution of a mean divided by an estimate of the standard error. This distribution is called the *student's t distribution* or sometimes just the *t distribution*" [12, p.1]. "The t distribution is very similar to the normal distribution when the estimate of variance is based on many degrees of freedom but has relatively more scores in its tails when there are fewer degrees of freedom {DF}" [12, p.1]. We would also consider the t distribution in computing our Confidence Intervals for better comparison of the different algorithms.

With these explanations in place, I set out to explore the results of applying these statistical measures to the results of our simulation runs for different algorithms (as shown in Figure 8). At least two measures are needed to describe the distribution of any statistical variable across a population [14]:

(a) Measure of Central Tendency - i.e., where is the 'center' of the distribution? For our case, *Mean* would be the chosen measure of central tendency.

Mean (Average): M= sum ($\sum$) of the values divided by N

(b) Measure of Dispersion - i.e., how much variation among values exists in the distribution? For our case, *Variance, Standard Deviation* & *Confidence Levels* with *Intervals* would serve as the measures of Dispersion.

Variance: s2 = sum of squared deviations from the Mean divided by N

Standard deviation: s = $\sqrt{}$s2

Standard error of the mean: S.E.M. = s / $\sqrt{}$N

95% Confidence limits (which define the confidence interval): [13][14]

For *Normal Distribution*: M ± (S.E.M. * C95); where C95 = 1.96

(99%, C_{99}= 2.576 ; 90%, C_{90}= 1.645)

For *T-Distribution*: M ± (S.E.M. * $t_{crit(.05)}$); where $t_{crit(.05)} = 2.26$

Statistical Analysis of Proposed Algorithm's Performance:

Population sample from Simulations:

X = {6, 5, 6, 6, 6, 5, 6, 6, 6, 6}

Summary Values:

$$N = 10$$

$$\Sigma X = 58$$

$$\Sigma X^2 = 338$$

$$\text{Mean} = 5.8$$

$$\text{Variance} = 0.1778$$

$$\text{Standard Deviation} = 0.4216$$

$$\text{S.E.M} = 0.1333$$

$$\text{DF} = 9$$

$$t_{crit(.05)} = 2.26$$

$$C_{95} = 1.96$$

95% Confidence Intervals for Estimated Mean of Population:

Normal Distribution: 5.8 ± (.13 * 1.96) = (5.55, 6.05)

T-Distribution: 5.8 ± (.13 * 2.26) = (5.51, 6.09)

Statistical Analysis of ILP Performance:

Population sample from Simulations:

X = {4, 4, 0, 6, 3, 4, 6, 3, 4, 2}

Summary Values:

$$N = 10$$

$$\Sigma X = 36$$

$$\Sigma X^2 = 158$$

$$\text{Mean} = 3.6$$

$$\text{Variance} = 3.1556$$

$$\text{Standard Deviation} = 1.7764$$

$$\text{S.E.M} = 0.5617$$

$$DF = 9$$

$$t_{crit(.05)} = 2.26$$

$$C_{95} = 1.96$$

95% Confidence Intervals for Estimated Mean of Population:

Normal Distribution: 3.6 ± (.56 * 1.96) = (2.51, 4.69)

T-Distribution: 3.6 ± (.56 * 2.26) = (2.34, 4.86)

Statistical Analysis of FCFS Performance:

Population sample from Simulations:

X = {4, 4, 4, 5, 4, 1, 3, 2, 2, 2}

Summary Values:

$N = 10$

$\Sigma X = 31$

$\Sigma X^2 = 111$

Mean = 3.1

Variance = 1.6556

Standard Deviation = 1.2867

S.E..M = 0.4069

DF = 9

$t_{crit(.05)} = 2.26$

$C_{95} = 1.96$

95% Confidence Intervals for Estimated Mean of Population:

Normal Distribution: 3.1 ± (.40 * 1.96) = (2.32, 3.88)

T-Distribution: 3.1 ± (.40 * 2.26) = (2.2, 4.0)

Statistical Analysis of WS (4:3:3:3) Performance:

Population sample from Simulations:

X = {6, 3, 8, 7, 5, 5, 7, 7, 6, 4}

Summary Values:

$N = 10$

$\Sigma X = 58$

$\Sigma X^2 = 358$

Mean = 5.8

Variance = 2.4

Standard Deviation = 1.5492

S.E.M = 0.4899

DF = 9

$t_{crit(.05)} = 2.26$

$C_{95} = 1.96$

95% Confidence Intervals for Estimated Mean of Population:

Normal Distribution: 5.8 ± (.48 * 1.96) = (4.86, 6.74)

T-Distribution: 5.8 ± (.48 * 2.26) = (4.72, 6.88)

Statistical Analysis of WS (2:5:3:3) Performance:

Population sample from Simulations:

X = {6, 3, 8, 7, 5, 6, 7, 4, 6, 4}

Summary Values:

$N = 10$

$\Sigma X = 56$

$\Sigma X^2 = 336$

Mean = 5.6

Variance = 2.48

Standard Deviation = 1.57

S.E.M = 0.4989

DF = 9

$t_{crit(.05)} = 2.26$

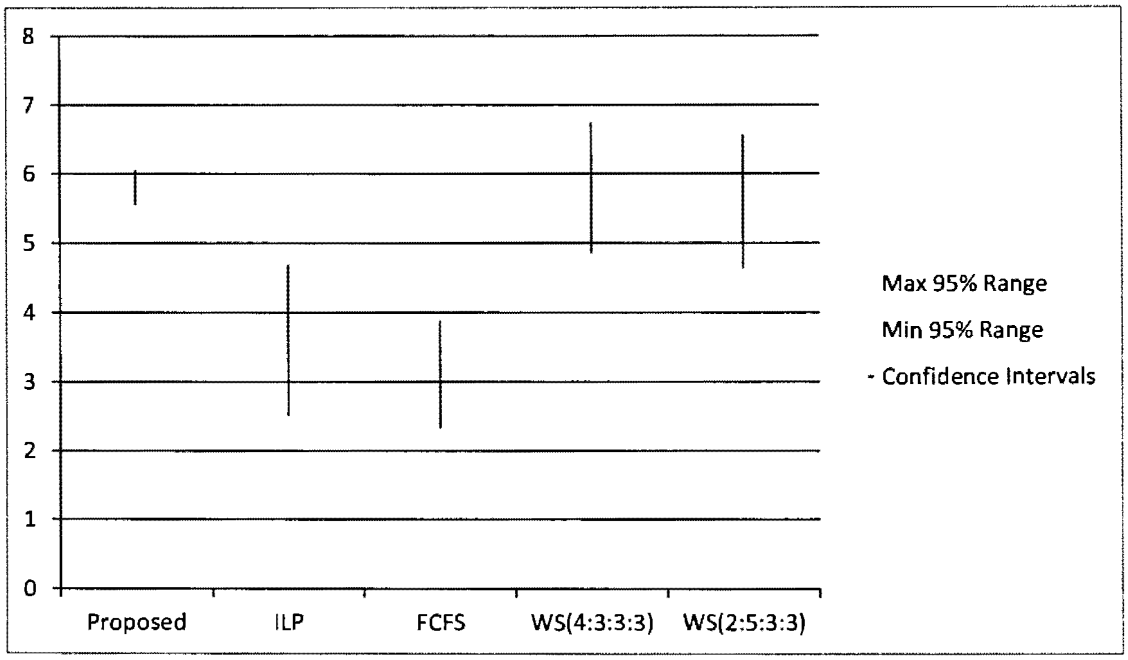
$C_{95} = 1.96$

95% Confidence Intervals for Estimated Mean of Population:

Normal Distribution: 5.6 ± (.49 * 1.96) = (4.64, 6.56)

T-Distribution: 5.6 ± (.49 * 2.26) = (4.5, 6.7)

Figure 8: Graphical comparisons of the 95% confidence intervals as demonstrated by the sample distributions of the proposed solution versus that of other competing algorithms



Findings from Figure 8 explained in next section.

CONCLUSION

To provide self-healing capabilities for sensor networks, I propose to deploy a few mobile sensors in addition to a large number of static sensors in a sensor network. Mobile sensors respond to multiple coverage requests by moving to connectivity holes and improve network performance. In this paper, I proposed a distributed decision-making algorithm, based on iterative Rough Set analysis and sorting, that has low computation requirements on the sensors and is robust to incomplete information. Our simulations demonstrate that the proposed algorithm performed well and mobile sensors could significantly increase sensor network coverage if using such optimization techniques before their move.

The above statistical analysis shows the performance of the four competing algorithms on a single measurable variable across 10 completely random simulations. Detailed breakdown of the measures of Central Tendency & measures of Dispersion for each algorithm demonstrate why the proposed algorithm outshines other ones. The Mean value of Rounded Coverage gain (the higher the better) for the sample obtained through the 10 simulations clearly points out the proposed algorithm to be performing at or above the levels demonstrated by the other algorithms. The measures of Central Tendency & Dispersion indicate that the proposed algorithm not only has a higher Mean result; but also performs with much less deviation compared to others. In other words, the

Rounded Coverage Gain to be obtained for any sample using the proposed algorithm is much closer to the Mean of the distribution as compared to the other algorithms. In fact, a significant win for the proposed algorithm stems from the fact that it has the lowest Standard Deviation for the given population compared to all other algorithms. The 95% Confidence Intervals are computed & represented for each algorithm for the samples drawn through the simulations. The Confidence Interval for the proposed algorithm clearly beats ILP & FCFS methods. There is closer competition with the weighted sum algorithms since there is overlap of the Confidence Interval. However, the Confidence Interval graph clearly demonstrates why the Interval of the proposed algorithm is more reliable; it is a smaller interval. The true mean of any sample population in the test field will be somewhere between these values (or within this confidence interval) in 95/100 samples. This provides confidence in the uniformity of results expected by applying the proposed solution. While with the weighted sum techniques, it is possible by chance to get a better network coverage result; it is also equally possible to get a worst result. Further, the weights for the different network attributes may have to be realized by trial and error, which does not guarantee the same results under varying weights. In comparison, the proposed algorithm outlines a fixed set of steps to be taken by the network stakeholders which guarantees a high quality result in increased network coverage.

REFERENCES

[1] Xiaojiang Du, Ming Zhang, Kendall Nygard et al. (2006). Self-Healing sensor networks with distributed decision making. Int. J. Sensor Networks, Vol. 1, Nos. 1/2/3.

[2] M. Zhang, X. Du, and K. Nygard (2005). Improving coverage performance in sensor networks by using Mobile sensors. IEEE MILCOM 2005.

[3] Tatiana Bokareva, Nirupama Bulusu and Sanjay Jha (n.d). SASHA: Toward a Self-Healing Hybrid Sensor Network Architecture. Web retrieve on May 6, 2010 from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.8588&rep=rep1&type=pdf

[4] Andrzej Skowron and Ning Zhong (2000). Rough Sets in KDD. Pacific-Asia Conference on Knowledge Discovery and Data-mining, Japan. Web on retrieve May 6, 2010 from: http://roughsets.home.pl/IRSS/rs-kdd/index.htm

[5] Andrzej Skowron (n.d). Rough Sets in KDD. Web on retrieve May 6, 2010 from: http://logic.mimuw.edu.pl/Grant2003/prace/GIFIPSkowron1.pdf

[6] Jan Komorowski, Lech Polkowski, Andrzej Skowron (n.d). Rough Sets: a Tutorial. Web on retrieve May 6, 2010 from: http://folli.loria.fr/cds/1999/library/pdf/skowron.pdf

[7] Jerzy W. Grzymala-Busse, Sachin Siddhaye (2004). Rough Set Approaches to Rule Induction from Incomplete Data. 10th International Conference on Information Processing & Management of Uncertainty in Knowledge-based systems, Italy. Vol. 2, 923-930.

[8] Marco Dorigo, Mauro Birattari, Thomas Stutzle (2006). Ant Colony Optimization. Iridia – Technical Report Series: Report # TR/IRIDIA/2006-023. Web retrieve on May 7, 2010 from: http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=7E60A6BC0411C7032AB1E359ACDEE984?doi=10.1.1.64.9532&rep=rep1&type=pdf

[9] Paula Weston (2001). Ants: swarm intelligence. Web retrieve on May 7, 2010 from: http://www.answersingenesis.org/creation/v24/i1/ants.asp

[10] Jared W. Patterson, Kendall E. Nygard (2006). Market Based Adaptive Task Allocations for Autonomous Agents. 6th International Conference on Cooperative Control and Optimization, Feb 1-3, 2006, Hilton University of Florida, Gaineville, FL.

[11] Confidence Intervals (n.d). Web retrieve on May 7, 2010 from: http://www.stat.yale.edu/Courses/1997-98/101/confint.htm

[12] Introduction to Normal and T_Distributions (n.d). Web retrieve on May 7, 2010 from:
http://onlinestatbook.com/chapter8/t_distribution.html
http://onlinestatbook.com/chapter6/intro.html

[13] Richard Lowry (n.d). Confidence Interval for the Estimated Mean of a Population. Web retrieve on May 7, 2010 from: http://faculty.vassar.edu/lowry/conf_mean.html

[14] Edward Waltz (n.d). Understanding Confidence Intervals. Web retrieve on May 7, 2010 from: http://www.albany.edu/sph/data/confidence_intervals.pdf