# Power Efficient Algorithms for Wireless Sensor Networks

By: Tayler Pino

Department of Mathematics and Computer Science

Submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science

Under the supervision of Dr. Salimur Choudhury

Algoma University

## Abstract

The performance of wireless sensor networks depends greatly on how long the network can run on its limited battery life. There are many ways of increasing the lifetime of a network, with two major techniques being sleep/wake scheduling and topology control algorithms. These two approaches work well, but can be improved by taking into account the variable energy levels in each node in a network. Using this idea, existing sleep/wake scheduling algorithms (the minimum dominating set problem) are improved using three local search techniques: traditional local search, fixed depth and variable depth. As well, an existing local topology control algorithm is studied and improved upon.

## Acknowledgements

I would like to thank my thesis supervisor, Dr. Salimur Choudhury for his help and guidance throughout the year and for sharing his knowledge on this subject. I would also like to thank my parents and sister as well as the rest of my family and friends for their continued support. A special thanks to Juliet Sufady.

## Table of Contents

## List of Figures

## List of Tables

## 1. Introduction

A wireless sensor network is a network that consists of low-cost and low-power nodes that can communicate untethered short distances [1]. These networks do not require existing infrastructure to function since each node functions as a router and host. The nodes that make up the network also have sensing, data processing and communication capabilities. Since these networks are usually deployed in difficult to access locations, a radio is used to transfer collected data from the network back to a base station [2]. Deployment of a sensor network uses a large amount of nodes with the nodes being deployed randomly into the environment. Since this type of deployment is used, sensor networks must be able to self-organize. Self-organization is also important because node failure is a common occurrence due to running out of energy, environmental destruction or node malfunction. The network topology can change after deployment if the nodes deployed are mobile, so the network must be able to re-organize. This also allows redeployment of additional nodes to the network, to replace node failure or change in requirements of the network.

An important constraint in sensor networks is power consumption as most traditional sensor networks have nodes with their own limited power source (<0.5 Ah, 1.2 V) [1]. Power sources for wireless sensors networks can include: macro-scale batteries, micro-scale batteries, micro-fuel cells, micro-heat engines, and radioactive power sources. Macro-scale batteries are the most versatile power source with the three primary chemistry types being zinc-air, lithium and alkaline. Macro-scale secondary (rechargeable) batteries such as the batteries used in consumer products such as cell phones and laptop computers are also common, but require a primary power source in addition to itself. Micro-scale batteries are smaller with the maximum current output of the battery depending on the surface area of the electrodes. Since this surface area is usually very small, this causes the current output to be small as well [3]. Some networks may consist of nodes with a secondary rechargeable power supply such as a solar panel depending on the location of deployment. Other renewable energy sources include solar cells, vibration, fuel cells, acoustic noise, or a mobile supplier [2]. Sleep/wake scheduling improves the lifetime of a sensor network by determining which nodes should have full functionality (awake) and which should be conserving power (sleep). Since each node has a dual importance of a data originator and a data router, inactive nodes cause significant changes in the network topology and could require the network to re-organize. In a wireless sensor network, the power consumption and therefore the lifetime of a network is an importance performance metric.

## 1.1. Aim and Objective

There are two goals of this thesis, the first being to develop and test multiple centralized scheduling algorithms for wireless sensor networks. The algorithms should increase the running time of the network compared to existing algorithms, since current algorithms for sleep/wake scheduling assume all nodes have the same power levels, meaning they are not as efficient as they could be. Node energy levels could be different from the rest, with more nodes having rechargeable power sources, allowing nodes on a sleep cycle to recharge while other nodes continue running and providing full coverage to the network. This is why the algorithms developed will take into account the energy level of each node and provide a list of disjoint sets that will provide complete coverage of the network. A node is considered covered if it is within an active sensors operational range [4]. When a set is active, all nodes in the set are put in the “awake” state, while the remainder are put into a low power, or “sleep” state. Once the set cannot provide complete coverage (a node or multiple nodes in the set have depleted their power), the next set available is activated.

The second goal is to improve upon an existing local topology control algorithm to improve the efficiency of the network topology. The output from this algorithm should provide a connected subgraph that offers an improved lifetime. The topology should allow nodes with the lowest energy to have less neighbors or connections, allowing them to use less energy and therefore increase the running time of the network. As well, nodes with higher amounts of energy should have more connections since the increased performance requirement can be handled with the greater amount of power available.

## 1.2. Rationale

Since the wireless sensor networks have many applications, the knowledge gained from this topic will have many applications improving current wireless sensor networks. Military applications include battlefield surveillance, battle damage assessment and targeting. Nodes can be attached to troops, vehicles or equipment for constant status reports to upper levels of command. Wireless sensor networks also have environmental applications for sensing changes in an area. For example forest fire detection can be accomplished with a large network deployed with a high density of nodes. Improving the performance of these networks by increasing their lifetime benefits all applications of wireless sensor networks.

There are also many home applications for wireless sensor networks such as home automation and smart environments. Nodes and actuators placed in appliances have the ability to interact with other nodes and can be managed remotely or locally by end users. For commercial applications wireless sensor networks can monitor the air conditioning and heating of office buildings. Since the heating and cooling systems are centrally controlled, air flow may not be evenly distributed. Using a wireless sensor network, air flow can be controlled in different parts of the room. This technology can save companies money and reduce carbon emissions. Other uses include vehicle tracking and detection and managing inventory control. Improving the performance of these networks by increasing their lifetime benefits all applications of wireless sensor networks listed above.

## 1.3. Scope

This thesis covers the minimum dominating set algorithm and a variation to produce a series of disjoint sets. Three improvement algorithms using different local search techniques will try to improve these sets. These techniques will include a traditional local search, a fixed depth search and a variable depth search. As well, an existing local topology control algorithm will be studied and improved to allow better performance on networks with variable energy values. Due to lack of hardware, testing on an actual sensor network was not possible, therefore all testing was done on a simulation of a sensor network.

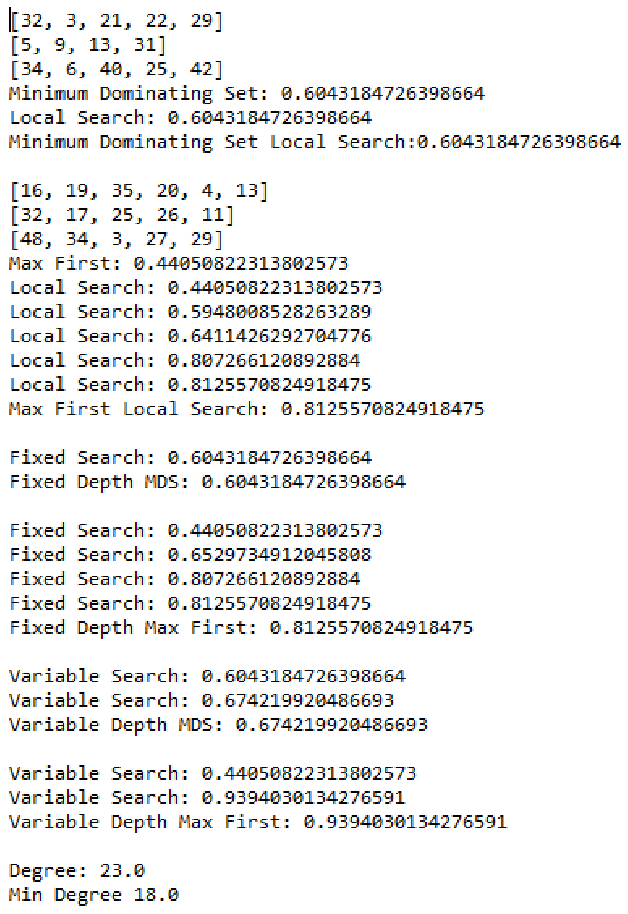
## 1.4. Materials

To write and test the algorithms needed for this thesis, Java version 8 was used. To help with development, Eclipse was used as an IDE.  All algorithms were run on the same computer, running Windows 10 Pro (64 bit) as an operating system.  The system also used an AMD Phenom II X4-955 processor (3.4GHz) and 6.00 GB of RAM. All testing was in the program that will simulate a wireless sensor network.

## 1.5. Procedure

To start, an existing dominating set algorithm (minimum dominating set) was written and tested. This is the baseline for all future results. Using the sets from the output of the minimum dominating set algorithm, other algorithms used this as an input to improve the overall lifetime. The algorithms developed used traditional local search optimization, fixed and variable depth to improve the efficiency of these sets. Once all algorithms were written, each one was run on the same network generated by the program. The program generated a random wireless sensor network at each run, with the size of the network and average degree of each node being variables that could be changed by the user.  Each node is given random neighbors and a random energy value between 0 and 1. The total lifetime of network was calculated by summing the minimum energy values in each of the sets. This is done because once one node dies, the set cannot be used since it cannot provide complete coverage. The local topology control algorithm used a different program that allowed each node to run its own algorithm, knowing only its information and the information of its neighbors. This program used the same rules when generating the network as the previous program with each node being given random neighbors and a random energy value between 0 and 1.

For the global scheduling algorithm simulation program, the output from figure 1.1 shows all the information from the test on that randomly generated network.



```
[32, 3, 21, 22, 29]
[5, 9, 13, 31]
[34, 6, 40, 25, 42]
Minimum Dominating Set: 0.6043184726398664
Local Search: 0.6043184726398664
Minimum Dominating Set Local Search:0.6043184726398664

[16, 19, 35, 20, 4, 13]
[32, 17, 25, 26, 11]
[48, 34, 3, 27, 29]
Max First: 0.44050822313802573
Local Search: 0.44050822313802573
Local Search: 0.5948008528263289
Local Search: 0.6411426292704776
Local Search: 0.807266120892884
Local Search: 0.8125570824918475
Max First Local Search: 0.8125570824918475

Fixed Search: 0.6043184726398664
Fixed Depth MDS: 0.6043184726398664

Fixed Search: 0.44050822313802573
Fixed Search: 0.6529734912045808
Fixed Search: 0.807266120892884
Fixed Search: 0.8125570824918475
Fixed Depth Max First: 0.8125570824918475

Variable Search: 0.6043184726398664
Variable Search: 0.674219920486693
Variable Depth MDS: 0.674219920486693

Variable Search: 0.44050822313802573
Variable Search: 0.9394030134276591
Variable Depth Max First: 0.9394030134276591

Degree: 23.0
Min Degree 18.0
```

*Figure 1.1*

First the sets found by the minimum dominating set algorithm are displayed, with each node having an integer number as an ID. On a network of n nodes, the nodes are named 1 to n. After that, the network lifetime for the minimum dominating set output is displayed. Then the local search is run on the sets with each iteration showing the new lifetime using the “Local Search:” prefix. At the end of the algorithm, the lifetime found by the local search is shown with the prefix “Minimum Dominating Set Local Search:” followed by the lifetime. The next section is the same as the previous, but using the max first algorithm instead of the minimum dominating set. Once again the sets are displayed along with the original lifetime found by the algorithm and the local search steps with the final result. Below that is the fixed depth search for both the minimum dominating set and max first algorithm. First is the algorithm using the minimum dominating set output, with each iteration being shown as “Fixed Search:” and the final lifetime as “Fixed Depth MDS:”. The same output is displayed below that with the algorithm using the max first output.  The last algorithm is variable search and it is displayed the same way as the fixed depth, with each iteration being displayed along with the final lifetime. Finally, the average degree of the nodes in the network is displayed along with the degree of the node with the least neighbors.

The program used to simulate the local topology control algorithm outputs much less information since only two algorithms were tested on it. The same information is displayed for both algorithms. For every node, its energy and the amount of connections or neighbors is output and the results are used to create a line graph that shows the neighbor count compared to the energy level of the node. The amount of neighbors that the node with the least energy has is also used to determine the success of this algorithm.

## 2. Related Work

When a node is in sleep mode, it loses all major functionality, but it consumes minimal amounts of energy and therefore extends the life of the network. This method of power conservation is called sleep/wake scheduling. There are many algorithms for traditional sleep/wake scheduling, with the most popular being the minimum dominating set algorithm. If two nodes are within transmission range, this implies that their area of coverage overlaps. The minimum dominating set algorithm leverages this to allow only one of these nodes to be active at a time, therefore preserving energy. The algorithm is defined as: "When modeling the network as a graph G=(V,E), a dominating set of G is a subset S ⊆ V such that, each node v ∈ V is either in S or has a neighbor in S. A minimum dominating set produces a dominating set of minimal cardinality [5]". All nodes included in the set can be awake, with the remaining nodes in sleep mode while the network coverage remains complete. This algorithm eliminates redundant nodes covering the same area, and allows power conservation with no performance trade off.

Another algorithm used in sensor networks is the connected dominating set. It is defined as such : "Given a graph G=(V,E) where V and E are sets of nodes and edges respectively, a subset D of V is a dominating set if each node u ∈ V is in D or is adjacent to some node v ∈ D. A subset K of V is a connected dominating set if K is a dominating set and induces a connected subgraph [6]. This produces a set which can act as a virtual backbone in a sensor network, as the connectivity of the set allows for message transmission throughout the nodes that make up the network. Since there is a lack of infrastructure in the formation of a wireless sensor network, the network hierarchy is flat. To make the network scalable and increase efficiency, a virtual network base station must be created. This virtual backbone of the network (VBN) organizes the network into a hierarchical structure and is also used in routing and broadcasting [7].

Recently, improvements to these algorithms have been made. For example the domatic partition problem which uses the energy available in each node to improve the lifetime of the sensor network. The algorithm works as follows: working in rounds, starting from round 1 and in each round *i* a dominating set *Di* is computed such that $Di \cap Dj = \emptyset, i \neq j, i, j \geq 1$. The algorithm runs for $\delta +1$ rounds. However, it stops in round *j* if we cannot form a new disjoint set dominating set *Dj* or we have $j = \delta +1$. If $j = \delta + 1$, then we have reached the optimal (maximum) number of disjoint dominating sets. Otherwise, $j - 1$ is the maximum number of disjoint dominating sets $D_1$, $D_2$, … , $D_{j-l}$ returned by the algorithm [8].

To solve the power issues with existing sleep/wake scheduling algorithms, local search techniques were applied. Local search optimization algorithms are used to improve an initial solution using repeated small changes. At each iteration, the solution is modified and the result is compared against the previous solution. If the new solution improves upon the initial solution, the change is kept and the algorithm restarts. If not, the change is reverted and another change is tested. This repeats until there are no improvements and all changes are exhausted [9].

The variable depth search method was introduced by Lin and Kernighan as an improvement over traditional local search techniques. The technique was applied to the travelling salesman problem and the graph partitioning problem. On a problem of size 30, the probability of global optimization for this method is about 0.5, compared to 0.1 using ordinary local search [10].This algorithm will always find a single favorable swap, if one exists making it locally optimal to the swap neighborhood. The main focus of the algorithm is to adaptively change the size of the swap neighborhood, so the algorithm can traverse a large computational space with minimal time tradeoffs [11].

Another technique used to improve the power conservation of wireless sensor networks is topology control. This results in longer network lifetimes and less radio interference in the network. Topology control organizes the network, changing the communication links between nodes while retaining global graph properties. The goal of this is to reduce energy consumption by controlling the topology of the network. Since wireless sensor networks do not require existing infrastructure, meaning the topology is not fixed, topology control algorithms can leverage this to freely modify the network, for example by modifying the nodes' transmission range [12]. Power consumption is based on the amount of work a node needs to do, such as relaying messages for multiple connections compared to only communicating with one node. The transmission range of a node greatly effects the amount of power used when running, as well effecting as the amount of radio interference in the network. Also, when the transmission range of a node is short, many nodes can communicate simultaneously without interference. One notation of energy efficiency is the energy stretch factor and is defined as: the power required, and hence the energy consumed, for a transmission from *u* to *v* is a polynomial function of the distance between *u* and *v*. Define the energy used for delivering a packet along a path to be the sum of the energy used along the edges of the path. For two nodes *u* and *v*, let $E_G(u, v)$ (resp., $E_T(u, v)$) denote the energy of the minimum energy path between *u* and *v* in G (resp., T). We now define the energy stretch factor of T to be the maximum, over all *u* and *v*, of $E_G(u, v)/E_T(u, v)$. A notion of quality similar to the energy stretch factor is the hop stretch factor which measures the ratios of the hop-counts rather than that of the energy [13].

## 3. Centralized Algorithms

Since the minimum dominating set algorithm does not use battery level as a variable, this was introduced to improve the lifetime of the network by applying it to other algorithms. All improvements of this algorithm used the output sets from the minimum dominating set excluding the max first algorithms which modified the original minimum dominating set algorithm. This excludes the local topology algorithm which runs locally in each node, and does not rely on any of the minimum dominating set algorithms. The local topology control algorithm is based on another existing local algorithm, and is modified to satisfy the variable energy problem.

### 3.1. Minimum Dominating Set

This existing algorithm is an already existing version of the dominating set algorithm. The algorithm works as follows:

i. For a graph G=(E,V), a list of nodes is taken. The node with the most neighbors is picked from the list and put into a set.

ii. All neighbors of this node and itself are removed from the node list and a new node is picked added to the set. This process is repeated until the node list is empty.

iii. The set should provide full coverage of the network, if it does not then the set is dropped and the algorithm terminates. If full coverage is provided, then the algorithm restarts at step 1, excluding the nodes already in a set. All sets provided must be disjoint. A node is defined as covered if it or one of its neighbors is in a set.

### 3.2. Max First

This variation uses the same steps as the minimum dominating set algorithm, with one change. When adding a node to the disjoint set, the original algorithm choses the next available node with the most neighbors. In the max first algorithm this is changed to picking the next available node with the most energy. The rest of the algorithm remains the same, ending when a set does not provide full coverage and outputting a list of disjoint sets.

## 3.3. Local Search

Taking the output from the minimum dominating set algorithm or a variation of it, a local search algorithm is performed on the disjoint sets. This type of algorithm maximizes goal using the candidate solutions provided by the disjoint sets.

i. Every node in these sets is swapped with every node in every set that is not its own. For example consider two sets [A,B,C] and [D,E,F]. The sequence of swaps would be A swapped with every element of all sets that are not its own. For this example there is only one other set, therefore A would swap with nodes D, E and F. The same is done for the remaining nodes.
ii. Each swap has two constraints: if the swap still satisfies the full coverage rule of the sets and if the swap will improve the running time of the network (have a positive energy gain). If both conditions are not satisfied, the pair is swapped back to their original sets and the algorithm tries the next sequential swap.
iii. Once a swap that satisfies both conditions is found, the swap is made and the algorithm starts from the beginning (step 1) on the new sets. If no swaps can satisfy these conditions, the algorithm finishes.

## 3.4. Fixed Depth

This variation of the local search technique algorithm selects the best possible swap available compared to the previous local search algorithm which took the first possible improvement. This algorithm requires a list of disjoint sets as input.

i. For every node in every set, swap with every node that is not in its own set.
ii. After each swap, if all sets still have full coverage, save the swap in a swap list along with the gain in the network lifetime. Swap the nodes back to their original position after each swap.

iii. Once every node is swapped and the sequence of swaps has reached the end, pick the swap from the swap list that has the highest gain value. Make the swap and start the algorithm again at step 1. If all gain values are negative, the algorithm terminates.

## 3.5. Variable Depth

Variable depth is the most complex and resource intensive algorithm tested.  This algorithm also requires a list of disjoint sets as input.

i. Consider every possible pair of nodes (ni,nj) where ni and nj belong to different sets. The pair that gives the maximum gain when swapped is picked and remembered in a swap list. The gain that a pair generates is calculated by getting the difference in energy gained from the new sets and old (new-old).  Same as the other algorithms, all swaps must result in sets that still provide complete coverage.

ii. After picking the max swap, using the new sets from after the swap chosen in step 1, pick the max swap from the new list of available swaps (excluding the nodes that were previously swapped). For example, if nodes 3 and 4 were swapped, this pair cannot be chosen, but 3 and 2 or 4 and 6 are valid swaps. The max gain can be negative. Repeat this sequence until there are no possible swaps left.

iii. Now for each possible pairs (in the worst case it won't be more than n*n), we have gain values and the g list has populated. The g list holds the gains from each swap. To choose the best sequence of swaps, g(k) is calculated as g(k) for each 0<=k<=length of g. For example g(0) is calculated as g(0), g(1) as g(0) + g(1), g(5) as g(0) + g(1) + g(2 ) + g(3) + g(4) + g(5) and so forth for every g value. The greatest g value is picked and the sequence of swaps to get to that value is made, with the new set being the set that the algorithm repeats on. Return to step 1 with the new sets unless the max g value is less than or equal to 0. Then the algorithm will terminate.

## 4. Local Topology Control Algorithm

This algorithm provides a different approach to solving the power conservation issue with wireless sensor networks, by using a local algorithm to efficiently reorganize the topology of the network. Each node runs this algorithm knowing only its own information and the information of nodes within commutation distance, known as neighbors. The information transmitted to its neighbors will be the id of the node. The original algorithm is defined as follows:

i. For each node *u* broadcasts its id to its neighbors *v* ∈ N(*u*).

ii. For each neighbor *v* ∈ N(*u*), *u* receives their ids, id(*v*).

iii. Then, for a node *m* ∈ N(*u*) if there exists some node *k* ∈ N(*u*) such that id(*k*) < id(*m*) and id(*k*) < id(*u*) then m is dropped from the u's list. Otherwise no neighbors are deleted from N(u). The updated neighbor set of u is denoted as N'(*u*) ⊆ N(*u*) [14].

This algorithm takes any connected graph as input and outputs a spanning subgraph. When running on a network with variable power levels in each node, the algorithm does not perform optimally, therefore some modifications were made to improve its performance. When broadcasting information to neighboring nodes, the energy level should be sent along with the id of the node. The improvement of this algorithm will run before the algorithm above and use its output at the input for the original local topology control algorithm.

i. For each node *u* broadcasts its id and energy level to its neighbors *v* ∈ N(*u*).

ii. For each neighbor *v* ∈ N(*u*), *u* receives their ids, id(*v*) and energy energy(*v*).

iii. For each node n in the network, will compare with all its neighbors m. If id(n)< id(m) and energy(n) < energy(m), then swap the id of n and m.

Once complete, the ids of each node should be assigned in a way that the node with the greatest energy will have the most neighbors possible. This improves the performance of the network, since lower energy nodes will last longer, due to less performance required.

# 5. Centralized Algorithm Testing Results

Testing was done by simulating networks with three different sizes, 50, 100 and 200. Each network was generated randomly at the start of each simulation. The energy of each node was a random number between 0 and 1, and each node was assigned a random amount of neighbors. Other variations changed the average degree of the network, with the main three test groups having an average degree of about half the size of the network. Testing for the local topology algorithm was tested separately because the total network lifetime was calculated differently than the other algorithms tested. For all algorithms other than the local topology, the total lifetime was calculated by summing the lowest energy node from each set. For the local topology algorithm, the topology of the network will be examined to determine the efficiency of the new topology.

## 5.1. 50 Node Network

Testing on this network sized showed the ability of the algorithms to perform on networks that would provide less sets with less nodes. As expected the variable depth algorithm performed the best, providing the best lifetime for every test network. To test the consistency of the algorithms on multiple network layouts, 10 random networks comprised of 50 nodes each were generated and each algorithm was used to improve the lifetime using input sets from both the minimum dominating set algorithm and the max first algorithm. The results of the tests are shown in table 5.1.

| Average Degree Size | Minimum Dominating Set | | Max First | | Local Search (MDS) | Local Search (Max First) | Fixed Depth (MDS) | Fixed Depth (Max First) | Variable Depth (MDS) | Variable Depth (Max First) |
|---|---|---|---|---|---|---|---|---|---|---|
| | Sets | Lifetime | Sets | Lifetime | Lifetime | Lifetime | Lifetime | Lifetime | Lifetime | Lifetime |
| 23 | 4 | 0.770 | 5 | 1.364 | 1.439 | 2.417 | 1.552 | 2.442 | 1.552 | 2.657 |
| 25 | 5 | 0.599 | 3 | 0.201 | 1.209 | 0.435 | 1.209 | 0.435 | 1.209 | 0.623 |
| 25 | 3 | 1.253 | 1 | 0.380 | 1.253 | 0.380 | 1.253 | 0.380 | 1.620 | 0.380 |
| 25 | 6 | 1.694 | 5 | 0.699 | 2.728 | 1.746 | 2.860 | 2.153 | 3.23 | 2.153 |
| 24 | 3 | 0.191 | 2 | 0.232 | 0.321 | 0.232 | 0.581 | 0.232 | 0.581 | 0.420 |
| 25 | 4 | 0.784 | 1 | 0.269 | 1.389 | 0.269 | 1.637 | 0.269 | 1.908 | 0.269 |
| 24 | 4 | 0.395 | 3 | 0.703 | 0.754 | 0.868 | 0.941 | 0.703 | 0.941 | 0.703 |
| 23 | 3 | 0.726 | 2 | 0.353 | 0.726 | 0.353 | 0.726 | 0.353 | 1.136 | 0.353 |
| 25 | 4 | 0.666 | 2 | 0.322 | 1.204 | 0.322 | 1.199 | 0.322 | 1.331 | 0.322 |
| 24 | 7 | 1.049 | 2 | 0.101 | 2.584 | 0.101 | 2.741 | 0.101 | 3.047 | 0.101 |

*Table 5.1*

The variable depth algorithm offered the best lifetime for each network compared to other algorithms, including the tests where it found the same result as another algorithm. When this occurred, the lifetime found by both algorithms was the highest on that test network. It never provided a lower network lifetime than another algorithm in this round of testing. The difference in results was the sets provided by the minimum dominating set and max first algorithms. The number of disjoint sets provided directly affected the ability of the algorithms to improve, since there is less options for swapping nodes with less sets. In all of the test networks, the best network lifetime was found using the initial algorithm that provided the most disjoint sets. The minimum dominating set algorithm provided the most disjoint sets consistently except for one test where it found less than the max first algorithm. Due to this, the minimum dominating set algorithm seems to offer the greatest chance at finding the greatest lifetime possible for a network of this size. In cases where this was true and the minimum dominating provided the most disjoint sets, the variable depth algorithm showed the greatest improvement on the lifetime or finding the same solution as the fixed depth algorithm. To better illustrate this, below in figure 4.1 is a clustered bar graph of the results.

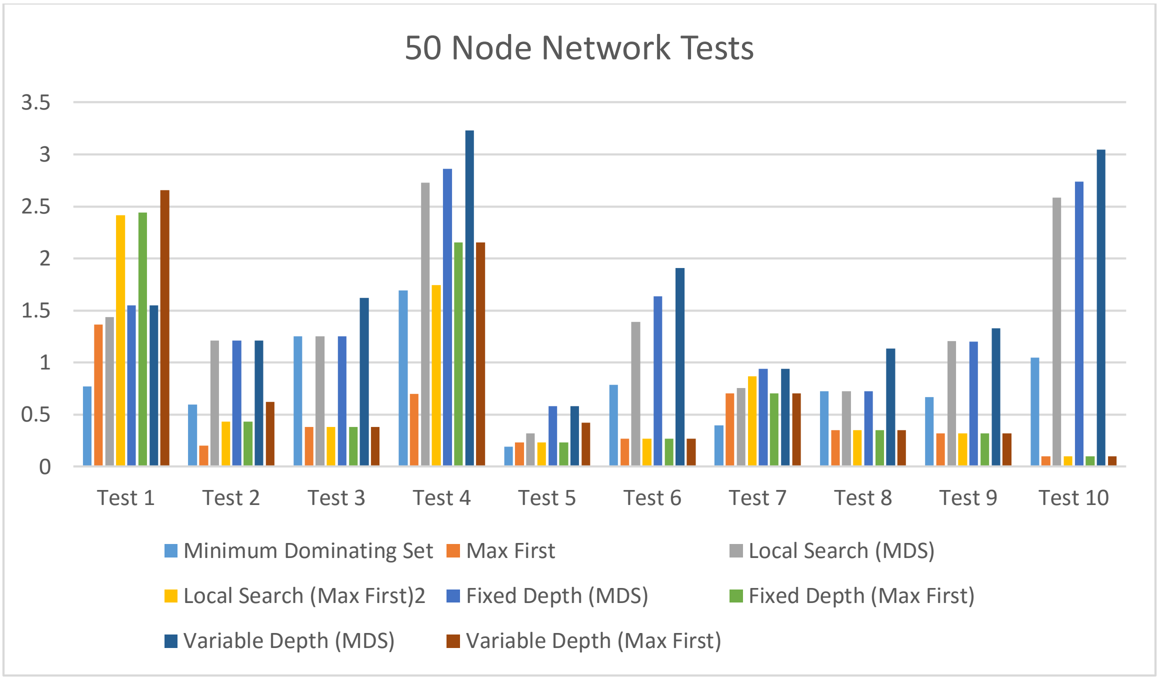


*Figure 5.1*

Since this network size and density provides less opportunities to swap nodes, multiple algorithms found the best network lifetime on more than one test. This was especially common in the results found by the algorithms using sets provided by the max first algorithm, since less sets were provided compared to the minimum dominating set algorithm. However, when improvement were made, the lifetime of the test network increased by the following amounts: using the input from the minimum dominating set algorithm, the local search algorithm had an average improvement of 0.548, the fixed depth algorithm had an average improvement of 0.6572 and the variable depth algorithm had an average improvement of 0.8428. Compared to the same algorithms using the max first algorithm sets as input, the local search algorithm had an average improvement of 0.2499, the fixed depth algorithm had an average improvement of 0.2766 and the variable depth algorithm had an average improvement of 0.3357.

## 5.2. 100 Node Network

On this network size, the results were similar to the previous 50 node test with most algorithms scaling as expected. As usual, the greater number of disjoint sets provided, offered the greatest chance of improving the network's lifetime. Table 5.2 shows the testing results for the algorithms run on 10 randomly generated networks containing 100 nodes each.

| Average Degree Size | Minimum Dominating Set | | Max First | | Local Search (MDS) | Local Search (Max First) | Fixed Depth (MDS) | Fixed Depth (Max First) | Variable Depth (MDS) | Variable Depth (Max First) |
|---|---|---|---|---|---|---|---|---|---|---|
| | Sets | Lifetime | Sets | Lifetime | Lifetime | Lifetime | Lifetime | Lifetime | Lifetime | Lifetime |
| 49 | 5 | 1.236 | 6 | 1.131 | 2.572 | 2.449 | 2.531 | 2.319 | 2.590 | 2.627 |
| 48 | 3 | 0.297 | 4 | 0.331 | 0.764 | 0.857 | 0.764 | 0.768 | 0.973 | 1.466 |
| 49 | 11 | 1.637 | 1 | 0.181 | 1.637 | 0.181 | 4.640 | 0.181 | 4.820 | 0.181 |
| 50 | 3 | 0.548 | 4 | 0.738 | 0.936 | 1.193 | 1.016 | 1.103 | 1.016 | 1.440 |
| 48 | 11 | 1.692 | 3 | 0.424 | 4.495 | 0.734 | 5.020 | 0.808 | 5.471 | 0.829 |
| 48 | 5 | 0.297 | 5 | 0.859 | 1.200 | 1.598 | 1.273 | 1.896 | 1.358 | 1.916 |
| 49 | 8 | 1.698 | 8 | 1.584 | 2.868 | 3.348 | 3.252 | 3.029 | 3.730 | 3.609 |
| 50 | 8 | 1.157 | 5 | 0.652 | 2.954 | 1.760 | 3.037 | 1.605 | 3.338 | 1.952 |
| 49 | 8 | 0.917 | 6 | 0.518 | 2.818 | 2.287 | 2.884 | 2.437 | 3.227 | 2.601 |
| 48 | 8 | 0.765 | 2 | 0.256 | 2.964 | 0.405 | 2.822 | 0.405 | 3.276 | 0.405 |

*Table 5.2*

In this round of testing, the max first algorithm performed better than it did on the previous tests on the 50 node networks. It provided the highest amount of disjoint sets three times and the same amount twice. However the minimum dominating set algorithms still had more consistency in this regard, producing the most disjoint sets five times. This is importance since the highest network lifetime was found was using the output from the algorithm with the most sets in every test. The variable depth search algorithm once again found the best results in every test case, with most disjoint sets allowing it to find more chances at improving the network lifetime. The two test cases where both the minimum dominating set and max first algorithm found the same amount of disjoint sets had two different results. In the first occurrence, five disjoint sets were found by both algorithms. Of these original sets, the max first algorithm found sets with a longer lifetime. This was the same for every algorithm run on the sets, with the max first sets providing a better lifetime compared to the same algorithm run on the minimum dominating set algorithm sets. However on the second occurrence, where both algorithms found eight disjoint sets each, the minimum dominating set algorithm found the initial sets with a greater network lifetime. The local search provided a better result using the max first sets, but for the fixed depth and variable depth algorithms, the better results were found using the minimum dominating set algorithm input. The highest network lifetime was from the variable depth algorithm using the minimum dominating set input. Based on this, there is no conclusive evidence for when both algorithms find the same amount of disjoint sets. Figure 5.2 displays the data from this round of testing.

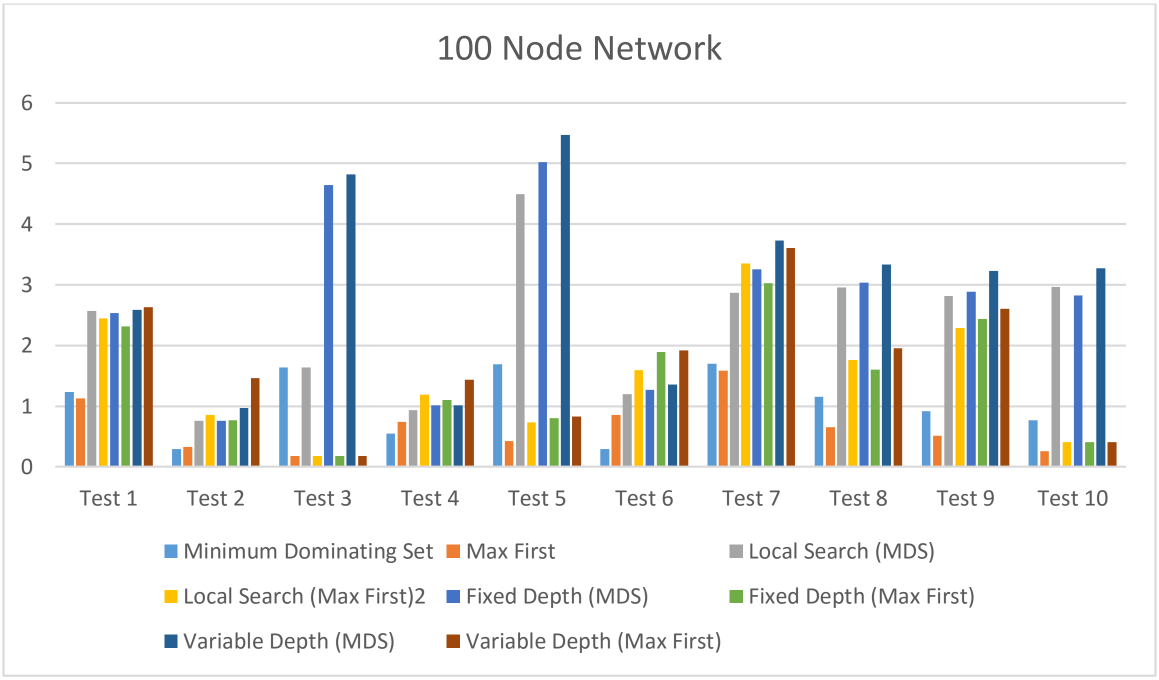


*Figure 5.2*

After this round of testing, the average increase in network lifetime was calculated for each algorithm. Using the input from the minimum dominating set algorithm, the local search algorithm had an average improvement of 1.2964, the fixed depth algorithm had an average improvement of 1.6995 and the variable depth algorithm had an average improvement of 1.9555. Compared to the same algorithms using the max first algorithm sets as input, the local search algorithm had an average improvement of 0.8138, the fixed depth algorithm had an average improvement of 0.7877 and the variable depth algorithm had an average improvement of 1.0352. Testing at this network size, compared to the previous test using 50 node networks provided an increase in average improvement for every algorithm compared to the previous averages.

## 5.3. 200 Node Network

This network size was the largest tested and continued the pattern set by the previous two test groups. The main observation from testing on 200 node networks was that the resource intensive algorithms such as fixed and variable depth, became much slower as more possibilities made themselves available with the increase of nodes. The slowest algorithm was by far the variable depth algorithm since every swap possibility was considered before making the swap or sequence of swaps and with the increase in network size, this slowed the algorithm down. However it still performed the best at increasing the network lifetime compared to other algorithms. This stage of testing best illustrated the time tradeoff compared to results. Table 4.3 shows the testing results on 10 randomly generated networks consisting of 200 nodes.

| Average Degree Size | Minimum Dominating Set | | Max First | | Local Search (MDS) | Local Search (Max First) | Fixed Depth (MDS) | Fixed Depth (Max First) | Variable Depth (MDS) | Variable Depth (Max First) |
|---|---|---|---|---|---|---|---|---|---|---|
| | Sets | Lifetime | Sets | Lifetime | Lifetime | Lifetime | Lifetime | Lifetime | Lifetime | Lifetime |
| 99 | 10 | 1.260 | 6 | 0.876 | 4.253 | 2.472 | 4.313 | 2.575 | 4.715 | 2.670 |
| 99 | 14 | 1.399 | 5 | 0.874 | 5.826 | 2.129 | 6.031 | 1.719 | 6.504 | 2.297 |
| 99 | 4 | 0.486 | 9 | 1.001 | 1.208 | 3.065 | 1.035 | 3.433 | 1.370 | 3.752 |
| 99 | 4 | 0.874 | 8 | 0.696 | 1.456 | 2.781 | 1.420 | 3.116 | 1.880 | 3.203 |
| 99 | 4 | 0.545 | 2 | 0.225 | 1.1969 | 0.357 | 1.389 | 0.357 | 1.731 | 0.357 |
| 99 | 10 | 1.556 | 2 | 0.0548 | 3.858 | 0.242 | 3.853 | 0.242 | 4.416 | 0.261 |
| 99 | 11 | 0.962 | 3 | 0.440 | 3.687 | 0.440 | 3.873 | 0.440 | 4.386 | 0.901 |
| 97 | 6 | 1.046 | 5 | 0.912 | 2.023 | 1.273 | 2.034 | 1.406 | 2.623 | 1.768 |
| 99 | 6 | 0.689 | 5 | 0.610 | 2.261 | 1.353 | 2.029 | 1.677 | 2.385 | 1.739 |
| 98 | 8 | 0.654 | 1 | 0.369 | 2.578 | 0.369 | 2.777 | 0.369 | 3.112 | 0.369 |

*Table 5.3*

As with the previous tests, the variable depth algorithm found the highest total network lifetime for each test. The original algorithm providing the disjoint sets (the minimum dominating set and max first algorithms) had a direct impact on the best network lifetime with the algorithm providing the most disjoint sets giving the highest lifetime when having the variable depth algorithm run on them. The fixed depth and local search both provided improvements over the original lifetime of the original sets, but not as much as the variable depth algorithm. However they did have a much faster running time, reaching the improvement much quicker. Something that was noticed during testing at this network size was that the variable depth algorithm found network lifetimes that were significantly greater than other algorithms on more tests at this size. Compared to testing on the 50 node networks, the difference between the variable depth algorithm and the others was smaller. This can be attributed to the larger network size providing more sets, with more nodes allowing the algorithm to find more possibilities of improvement on the sets provided. The max first algorithm provided the most sets twice on this round of testing, whereas the minimum dominating set algorithm proved to be more consistent at finding the most disjoint sets. Usually when the minimum dominating set algorithm found more sets than the max first algorithm at this stage of testing, the difference was much greater compared to when the max first algorithm found more sets. This can be seen in figure 5.3.

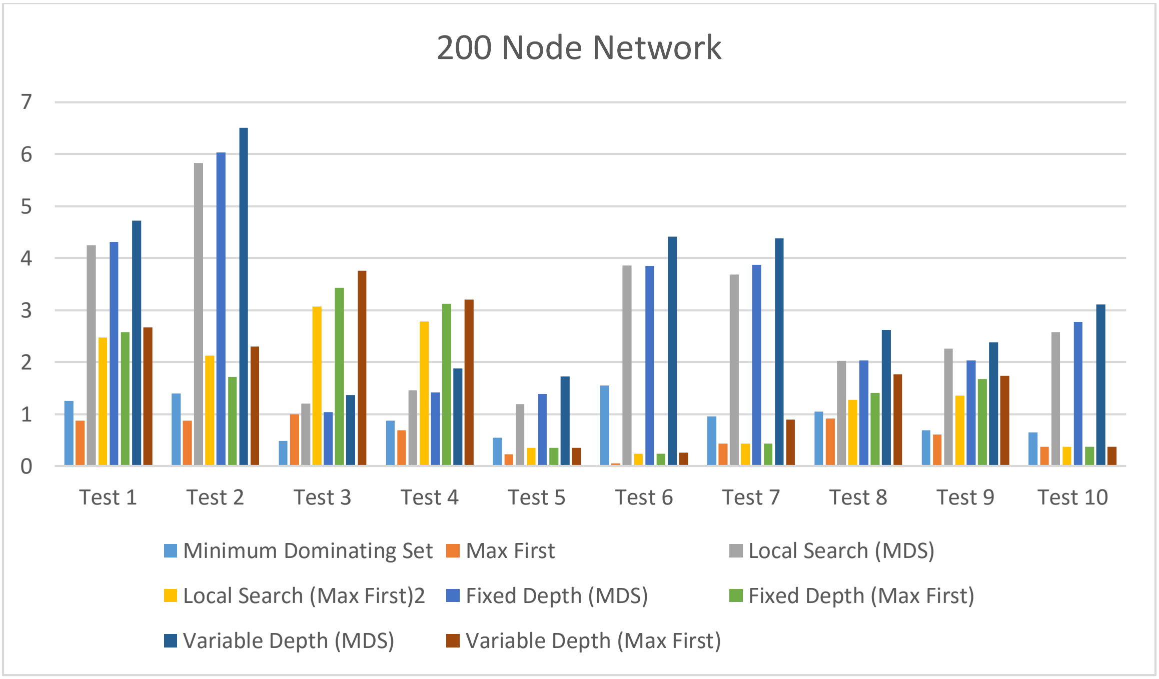


*Figure 5.3*

The average increase that each algorithm provided to the network lifetime was once again greater than the previous test (100 node network). However, the difference was not as big when compared to the difference found between the 100 node network test and the 50 node network test. Using the input from the minimum dominating set algorithm, the local search algorithm had an average improvement of 1.88759, the fixed depth algorithm had an average improvement of 1.9283 and the variable depth algorithm had an average improvement of 2.3651. Compared to the same algorithms using the max first algorithm sets as input, the local search algorithm had an average improvement of 0.84232, the fixed depth algorithm had an average improvement of 0.92762 and the variable depth algorithm had an average improvement of 1.12592.

## 5.4. Variable Degree Test Networks

In addition to the previous test groups, networks with a higher average node degree were used in testing. Previously, when generating the random networks each node had a 50% chance of having each node as its neighbor. To increase the degree of each node, this was changed to 70% which in turn increased the number of disjoint sets provided by the minimum dominating set and max first algorithms. This was tested on 10 randomly generated networks containing 50 nodes each. The results for this round of testing are displayed in table 5.4

| Average Degree Size | Minimum Dominating Set | | Max First | | Local Search (MDS) | Local Search (Max First) | Fixed Depth (MDS) | Fixed Depth (Max First) | Variable Depth (MDS) | Variable Depth (Max First) |
|---|---|---|---|---|---|---|---|---|---|---|
| | Sets | Lifetime | Sets | Lifetime | Lifetime | Lifetime | Lifetime | Lifetime | Lifetime | Lifetime |
| 34 | 4 | 0.437 | 5 | 0.682 | 1.078 | 0.894 | 1.016 | 1.147 | 1.338 | 1.433 |
| 33 | 12 | 4.113 | 4 | 0.714 | 5.723 | 1.003 | 5.867 | 0.959 | 6.314 | 1.631 |
| 34 | 7 | 1.351 | 4 | 0.559 | 3.196 | 1.213 | 2.705 | 0.893 | 3.198 | 1.213 |
| 35 | 10 | 2.704 | 6 | 0.715 | 5.003 | 2.354 | 5.020 | 2.459 | 5.157 | 2.611 |
| 33 | 7 | 1.600 | 8 | 2.349 | 2.933 | 3.814 | 2.670 | 3.394 | 3.059 | 3.948 |
| 33 | 6 | 1.162 | 5 | 0.923 | 2.461 | 2.214 | 2.413 | 2.238 | 2.632 | 2.238 |
| 35 | 9 | 1.474 | 6 | 0.751 | 3.342 | 1.824 | 3.265 | 1.441 | 3.547 | 1.954 |
| 35 | 4 | 1.152 | 1 | 0.266 | 1.772 | 0.266 | 1.772 | 0.266 | 1.772 | 0.266 |
| 33 | 7 | 1.166 | 3 | 0.489 | 1.821 | 0.502 | 1.945 | 0.502 | 2.185 | 0.502 |
| 33 | 13 | 3.967 | 5 | 0.852 | 5.787 | 0.852 | 5.839 | 1.765 | 6.343 | 1.796 |

*Table 5.4*

The increased density of the networks in this testing group is reflected in the average degree size of each network generated. Compared to the average degree on the previous 50 node networks which had around an average degree of 25, this testing group had an average degree closer to 35 neighbors per node. This density increase also allowed for an increase in sets provided by the minimum dominating set and max first algorithms. In the testing for the previous 50 node network, the most sets found by an algorithm was 7, found by the minimum dominating set algorithm. In this test, the most was 13 which was also found by the minimum dominating set algorithm. This allowed the local search, fixed depth and variable depth algorithms more opportunities to increase the network lifetime using the sets. In the 200 node test, the algorithms found greater increases compared to the increases found in the 50 node network tests. This test with higher density found improvements comparable to the 100 node tests, with only 50 nodes. These results show that the local search, fixed and variable depth algorithm perform better on higher density networks. Figure 5.4 illustrates the increases found by each algorithm.

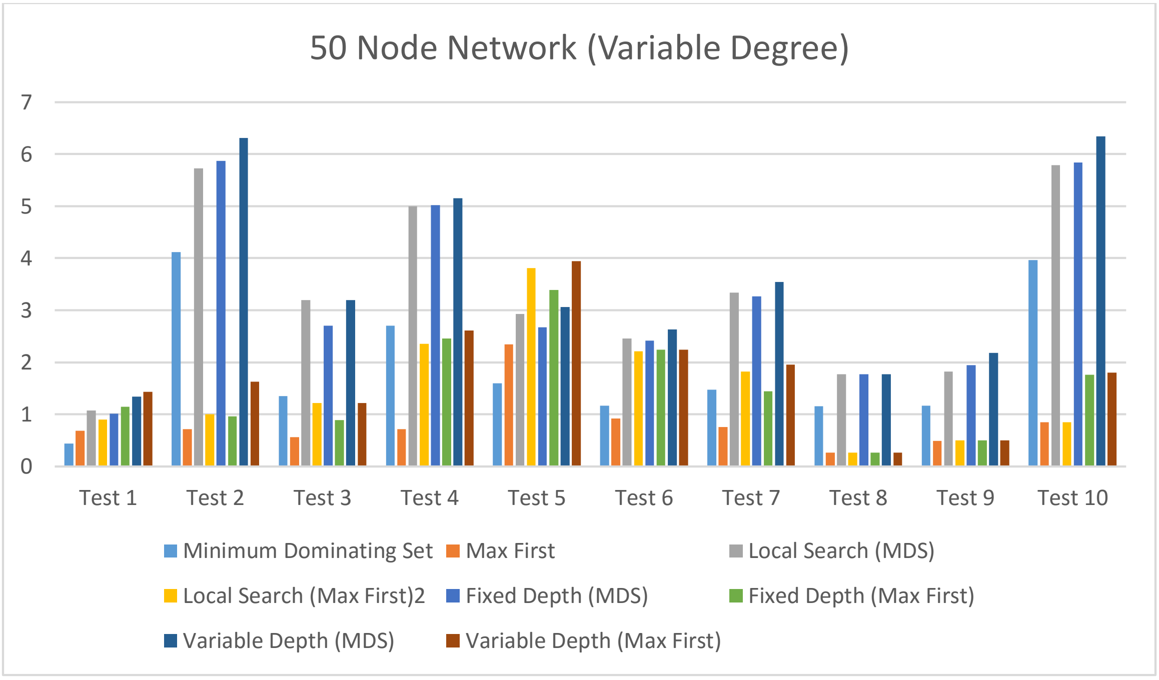


*Figure 5.4*

The pattern of the minimum dominating set algorithm providing more disjoint sets continued in this test group with the max first algorithm proving more sets only twice. As with previous tests, the algorithm that provided the most disjoint sets allowed the variable depth algorithm to find the greatest lifetime for the network. As well, when the minimum dominating set algorithm found more sets compared to the max first algorithm, the difference was much greater when compared to the two times where the max first algorithm found more sets.

The average increases provided by each algorithm was higher than the 50 node network with the standard testing density. This testing group provided results that were closer (slightly lower) to the 100 node network test group, with none of the average increases being higher than any result from the 200 node network test. The results are as follows: using the input from the minimum dominating set algorithm, the local search algorithm had an average improvement of 1.399, the fixed depth algorithm had an average improvement of 1.3386 and the variable depth algorithm had an average improvement of 1.6419. Compared to the same algorithms using the max first algorithm sets as input, the local search algorithm had an average improvement of 0.6636, the fixed depth algorithm had an average improvement of 0.6764 and the variable depth algorithm had an average improvement of 0.9292. This is a good increase for a small network, showing that the density of the sensor network directly effects the ability of the algorithms to provide a better lifetime.

## 5.5. Average Algorithm Improvement

At the end of testing the minimum dominating set, max first, local search, fixed depth and variable depth algorithms, the improvement that each algorithm provided on each test network was calculated. The improvements were separated into two groups: using the minimum dominating sets as input and using the max first sets as input. The averages were calculated by getting the difference from the total lifetime of the network using the new sets and the original lifetime. An average was taken from the differences for each algorithm with each input algorithm type. The results can be found in table 5.5.

| Input | Local Search | Fixed Depth | Variable Depth |
|---|---|---|---|
| Minimum Dominating Set | 1.282748 | 1.4059 | 1.701325 |
| Max First | 0.642405 | 0.66708 | 0.856505 |

*Table 5.5*

As the results show, the average increase for the algorithms using the minimum dominating set was significantly higher than the increases from the max first algorithm. Out of all the results, the variable depth algorithm gave the greatest average increase in all test cases with the fixed depth coming in second. The local search algorithm came last, but was not far behind the fixed depth algorithm with the max first versions being very close. Although slower than the other algorithms, the variable depth algorithm appears to be the best option at increasing the lifetime of a wireless sensor network. This works best using sets provided by the minimum dominating set algorithm. Max first does not perform as well, or as consistent compared to the algorithm it is based off.

## 5.6. Local Search Technique

To show how a local search improves a network, an example is shown on a small test graph of 7 nodes. The topology of this network is shown in figure 5.5. Each vertex in the graph represents a node in the network. Each node has a ID and below it, the energy level. Each edge represents that the nodes connected are within communication range and are therefore neighbors.

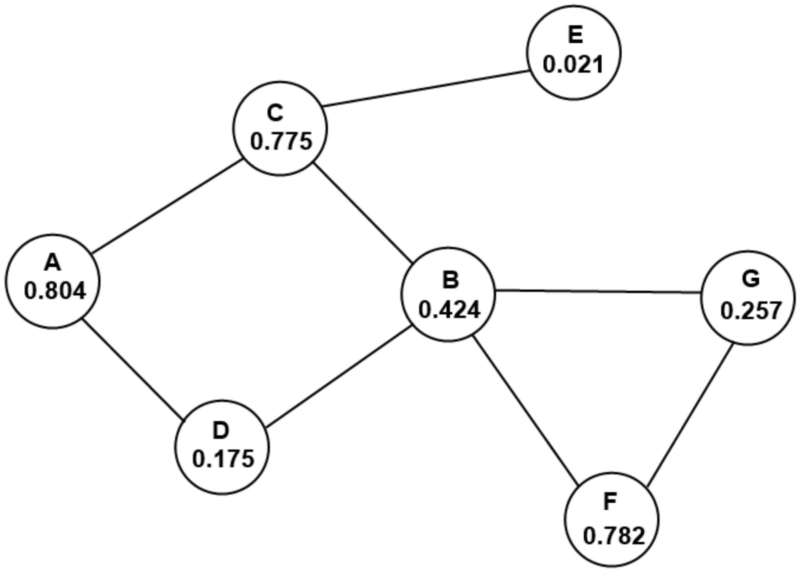


*Figure 5.5*

On this network, after running the minimum dominating set algorithm, two disjoint sets are found: [A,B,E] and [C,D,F]. As per the conditions of the algorithm, both sets satisfy the full coverage rule. The total time of this schedule would be the minimum energy of the first set (0.021) and minimum energy of the second set (0.175). Therefore the total energy of this network with this schedule would be 0.021 + 0.175=0.196. To improve this, the local search is done on the two sets. First nodes A and C are swapped and the two new sets, [C,B,E] and [A,D,F] are checked to see if both provide full coverage. The first set meets the criteria but the second does not provide coverage to node E. Since the rule is broken, the sets are swapped back to the previous state which in this case is [A,B,E] and [C,D,F]. The next pair of nodes in the sequence is swapped, node A and D. As with the step before, two new sets are created: [B,D,E] and [A,C,F]. Again these sets are tested for coverage, but unlike the previous step, both sets pass. Now the total energy of these new sets is compared to the previous sets. The total energy would be 0.021 + 0.775 = 0.796. Since this is greater than the previous sets (0.196), the swap remains and the algorithm starts from the beginning. This continues until no swaps provide an improvement over the previous solution. For this example, swapping nodes A and D is the only swap made and the best solution found by the algorithm giving these two sets: [B,D,E] and [A,C,F].

## 5.7. Fixed Depth

On the network shown in figure 5.6, the minimum dominating set algorithm provides two sets: [A,B,D] and [C,E,F] which give a network lifetime of 0.561. As per the algorithm definition, every swap is made, and if the swap results in all sets retaining their full network coverage, the energy gained is recorded along with the nodes swapped in a swap list.

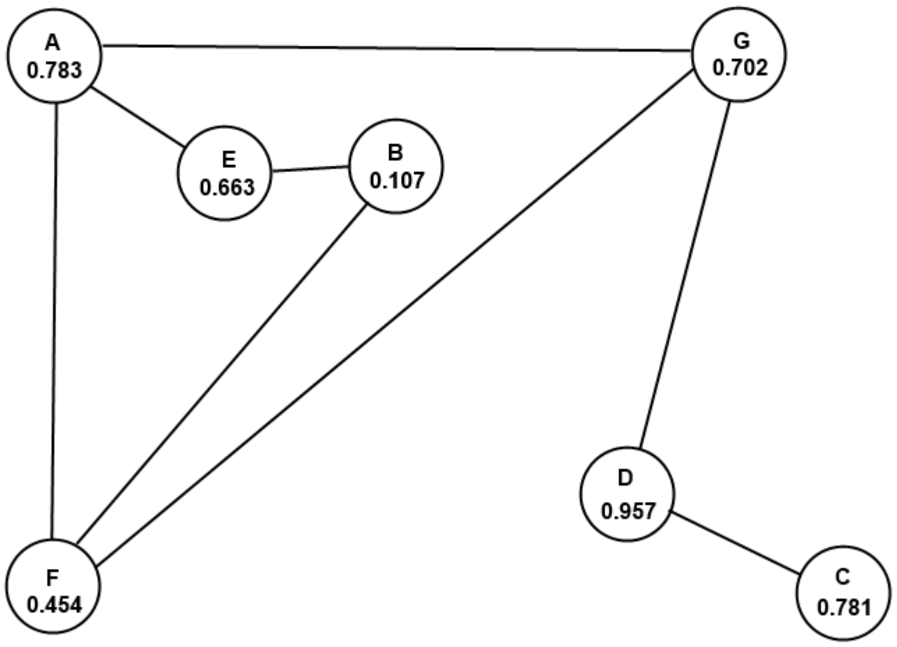


*Figure 5.6*

After the first iteration, the swap list contains two possible swaps: swap nodes D and C which gains 0 energy compared to the original sets, and swap nodes A and F which gains 0.209 energy compared to the original sets. The algorithm choses the swap with the greatest gain (in this case it would be A and F), and swaps the nodes. The algorithm restarts with the new sets acting as the original.

On this iteration of the algorithm, the swap list would contain two possible swaps: swap nodes B and E which gains -0.209 energy compared to the original sets, and swap nodes F and A which gains- 0.209 energy compared to the original sets. Since all gains in the swap list are negative, the algorithm can terminate. The resulting sets found are [B,D,F] and [A,C,E], which give a network lifetime of 0.770.

## 5.8. Variable Depth

To better explain the process of the variable depth algorithm, this section has a step by step of the algorithm being run on a small network of 7 nodes. The topology of the network is shown in figure 5.7. This is a simple example since a larger or more complicated solution would require too many steps because of the nature of the variable depth algorithm.

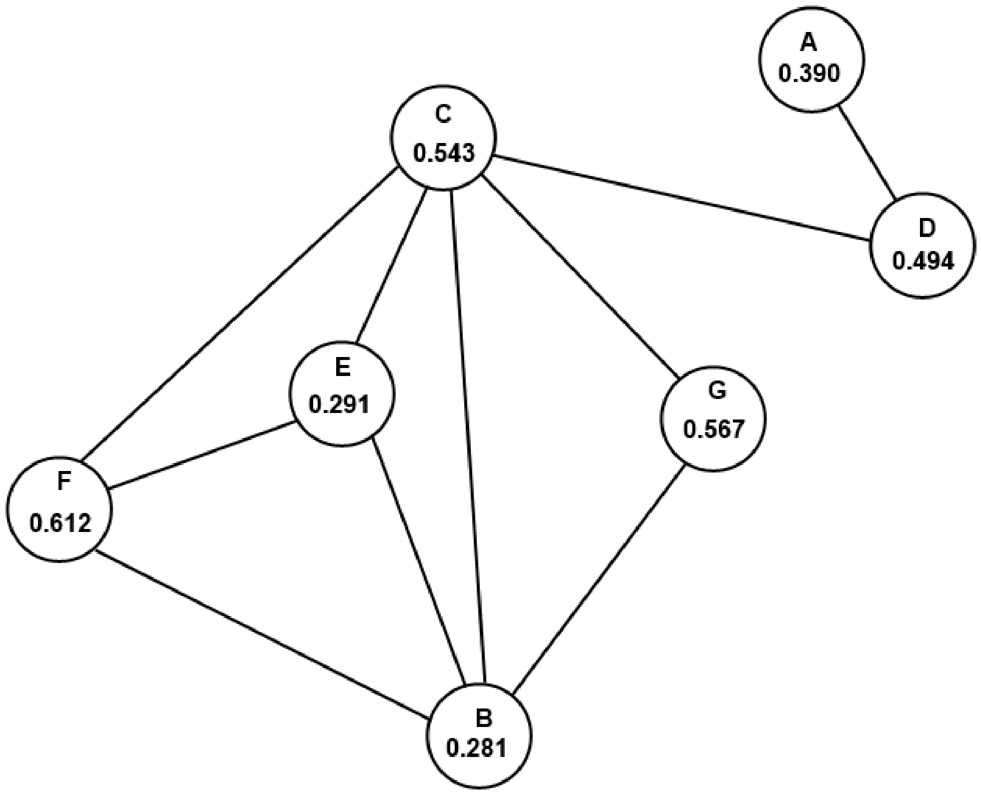


*Figure 5.7*

Using the minimum dominating set algorithm, the following disjoint sets were found: [A,C] and [B,D] with network lifetime of 0.671 (0.390 + 0.281). The variable depth algorithm uses the sets as input and makes a swap list consisting of every possible swap. For these sets, the possible swaps would be (A,B), (A,D), (C,B) and (C,D). Swaps that break the full coverage rule in any set are eliminated from the swap list, meaning swap (A,B) and (C,D) cannot be in the list. Therefore the possible swaps that can be chosen are (A,D) which gives a gain of 0.104 and (C,B) which gives a gain of 0.104 as well. The swap with the highest gain is chosen from this list and since both give the same increase, the first swap, (A,D) is chosen and added to the final list. The new sets are now [C,D] and [A,B] and the list of possible swaps is remade choosing all possible swaps, excluding (A,D) since it was chosen at the last iteration and swaps that break the full coverage provided by the sets. The swap list now contains only one swap (C,B) which gives a gain of -0.104, so this swap is chosen and added to the final list. Remember the highest gain can be negative unlike the other algorithms, since that is not the stopping point. The sets now look like this: [B,D] and [A,C]. Another iteration of the above sets occur on these new sets, and the list of possible swaps are made. At this stage, there are no swaps that keep the full coverage of the sets and were not already made. Remember swaps cannot be repeated at this stage, so swapping the pairs (A,D) and (C,B) cannot be made. There are no possible swaps at this stage, so this stage of the algorithm ends and the final list is examined. The final list of swaps made, along with their gains contains two swaps: [(A,D),(C,B)] and the associated gain list would be [0.104, -0.104]. To choose the sequence of swaps, g(k) is calculated as g(k) for each 0<=k<=length of g. So for this example,

**g(0) = 0.104**
**g(1) = 0.104 + (-0.104)= 0**

In this case, g(0) contained the highest g value, so all swaps before it, and itself are made to the original sets. Since it was the first swap made, there are no swaps proceeding it and only one swap needs to be made. If g(1) was chosen, swaps 0 and 1 would need to be made to the original sets. The new sets are [C,D] and [A,B] with a total network lifetime of 0.775. Now the algorithm restarts using the new sets, in this example we can skip to the end of the next iteration. After the second iteration, the final swap list is [(C,B), (A,D)] and the gain list is [-0.104, 0.104]. The g values are calculated:

**g(0) = -0.104**

**g(1) =-0.104 + 0.104 = 0**

The highest g value is g(1)=0, and since the max is less than or equal to zero, the algorithm can terminate. The sets found in the iteration before become the final sets, in this case being [C,D] and [A,B] which give a total network lifetime of 0.775.

# 6. Local Topology Algorithm Testing Results

## 6.1. Results

Testing for this local algorithm was done on networks with 50, 100 and 200 nodes, examining the topology of the networks after running both the local topology control algorithm and the optimized version.  The goal is to minimize the amount of connections to nodes with low energy, allowing them to run longer and transfer the responsibly to nodes with higher energy levels. After multiple rounds of testing, there appears to be an improvement over the previous algorithm on most tests. Three tests are shown in figure 6.1, 6.2 and 6.3. The graphs show the average neighbor count for each range of node energies. Graphed are nine ranges from 0 to 1, with intervals of 0.1. For each interval, the average amount of neighbors was calculated and graphed.

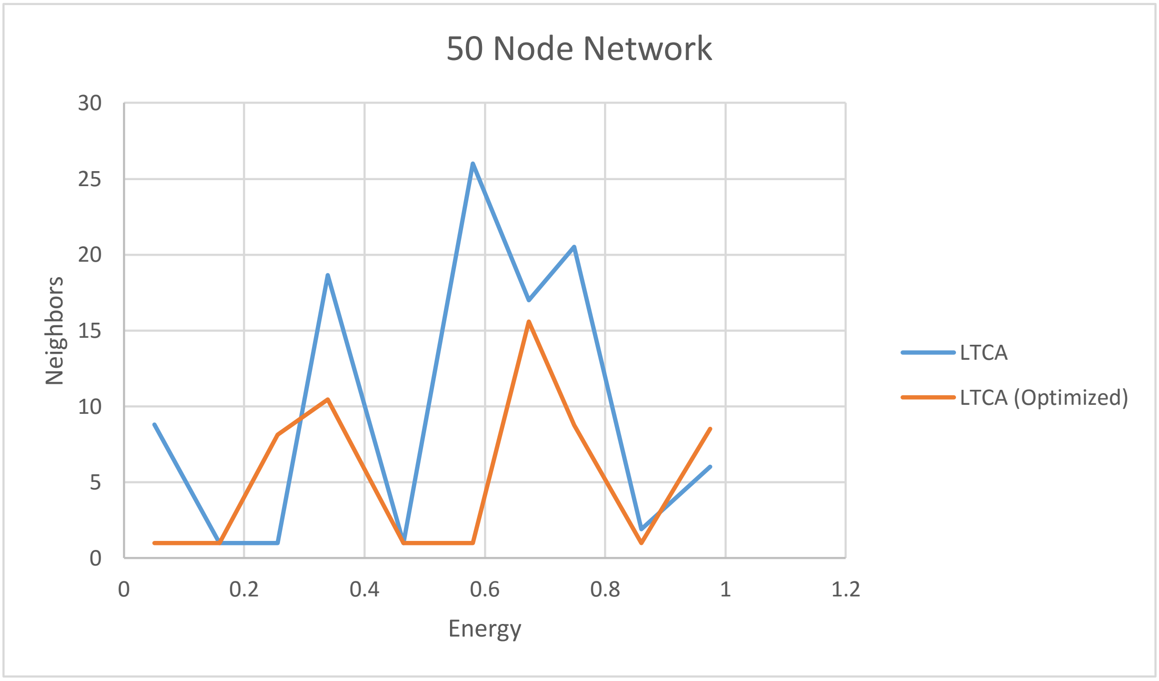


*Figure 6.1*

For this 50 node network, the optimized algorithm performed well, having less neighbors on low energy nodes and more neighbors on higher energy nodes. This is compared to the original local topology control algorithm which had more neighbors in the mid-range (0.5 energy), but comparing the highest energy and lowest energy, the lowest had more neighbors. As well, the optimized version of the algorithm has more nodes with only one neighbor with the graph showing that overall the nodes contained much less neighbors on average compared to the original algorithm.

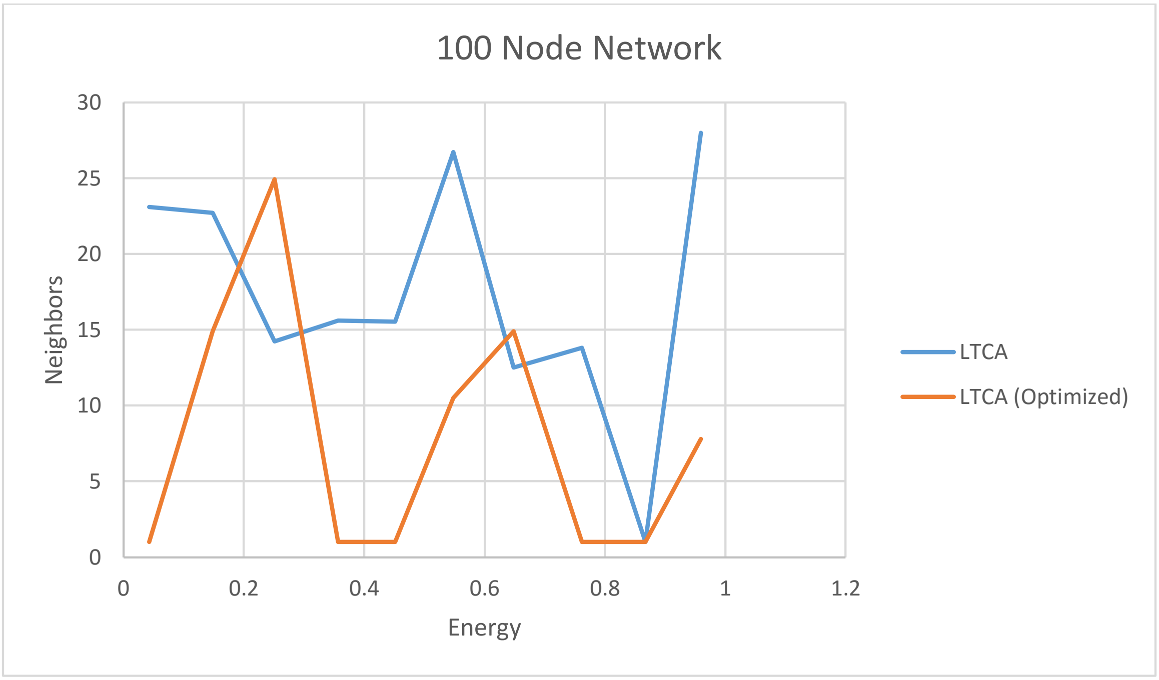


*Figure 6.2*

The 100 node network also showed promising results, with the lower energy nodes have sufficiently less neighbors using the optimized algorithm. Compared to the rest of the nodes in the topology created by the optimized algorithm, the nodes with less than 0.1 energy had the least neighbors. The original local topology control algorithm has the least amount of neighbors on the nodes with energy values between 0.8 and 0.9. This is not optimal since these nodes should have more neighbors to take advantage of the greater amounts on energy available.

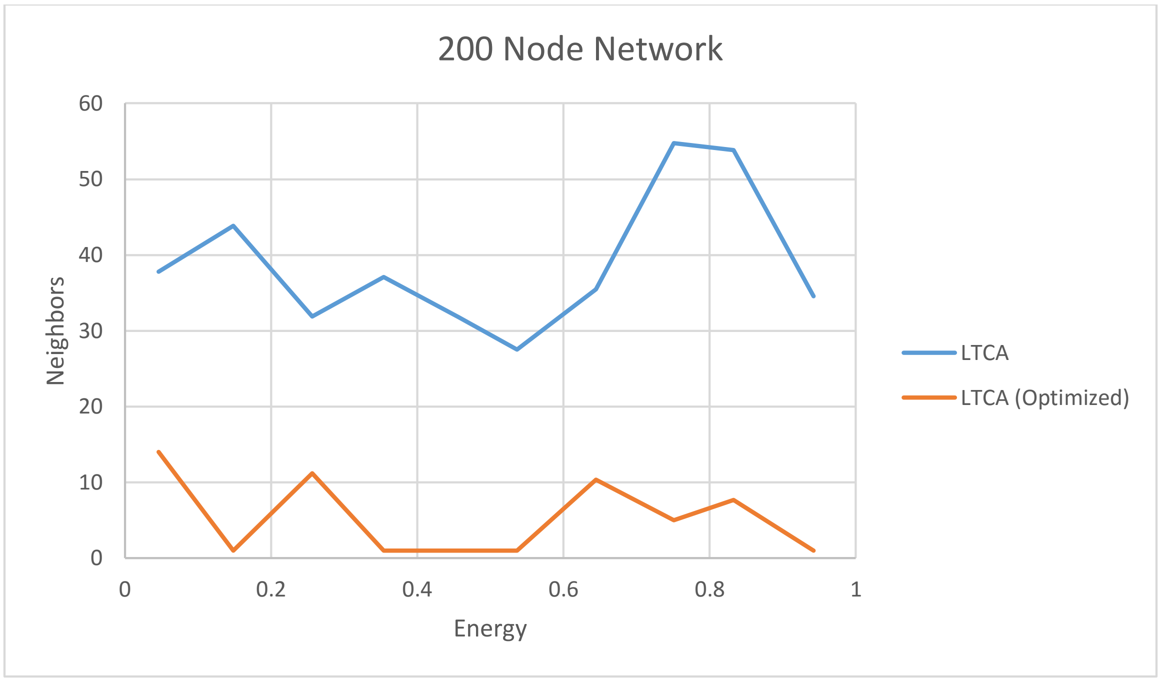


*Figure 6.3*

Finally, the 200 node network had a big difference in average neighbor count with the optimized version of the local topology algorithm having significantly less. This optimized version of the algorithm had less nodes on the low energy nodes compared to the original algorithm, but compared to the other nodes in its own topology it had the most. Overall the topology provided by the optimized algorithm is still superior since there is no node range where the original algorithm had any benefit in neighbor count. This shows that the improvement amount is dependent on the original topology of the network, but the figures shown were chosen since they offered results that were seen the most during testing.

## 6.2. Example

To demonstrate the improvement made by the optimization algorithm, here is an example on a small network of size 4. The layout of this network is shown in figure 6.4.

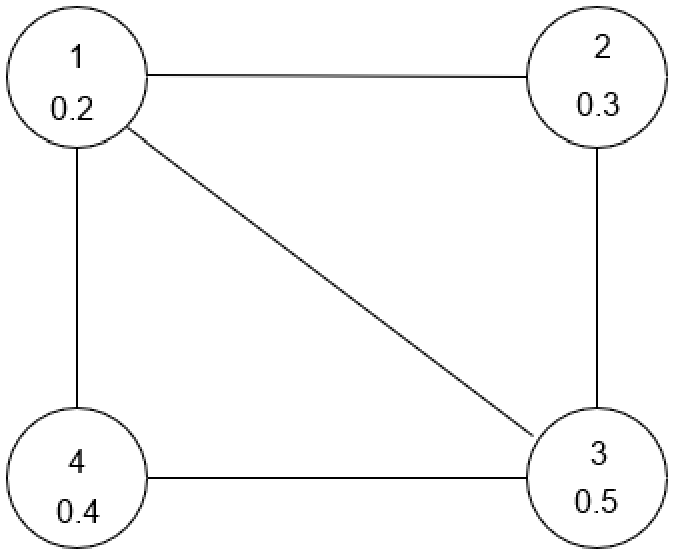


*Figure 6.4*

Each node has an id and below it, an energy level. These are the two values used in the algorithm. Edges between nodes indicate that the nodes are within communication range. The traditional local topology control algorithm would use the graph above without any modification as input, but the improved algorithm optimizes the id of each node based on the energy level of itself and its neighbors. Figure 6.2 shows the network after the optimization has been run on it.

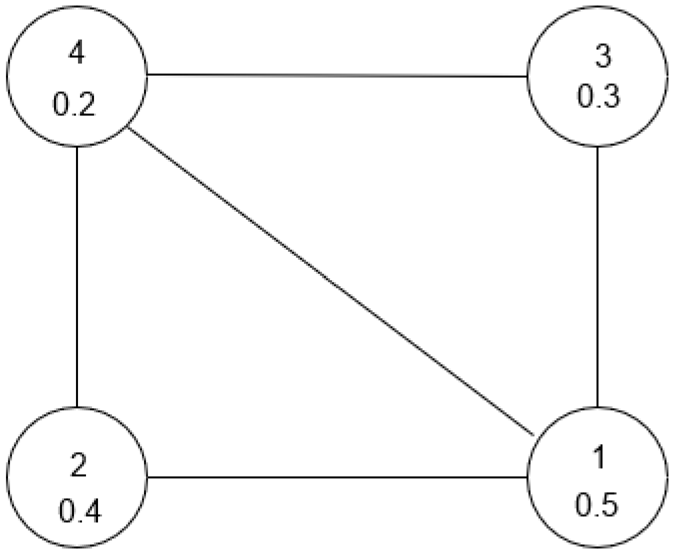


*Figure 6.5*

As you can see, the nodes remain the same except for the id swap that occurred. This allows the local topology control algorithm to provide a better result. If both networks are used as input, figure 6.3 shows the results for both.

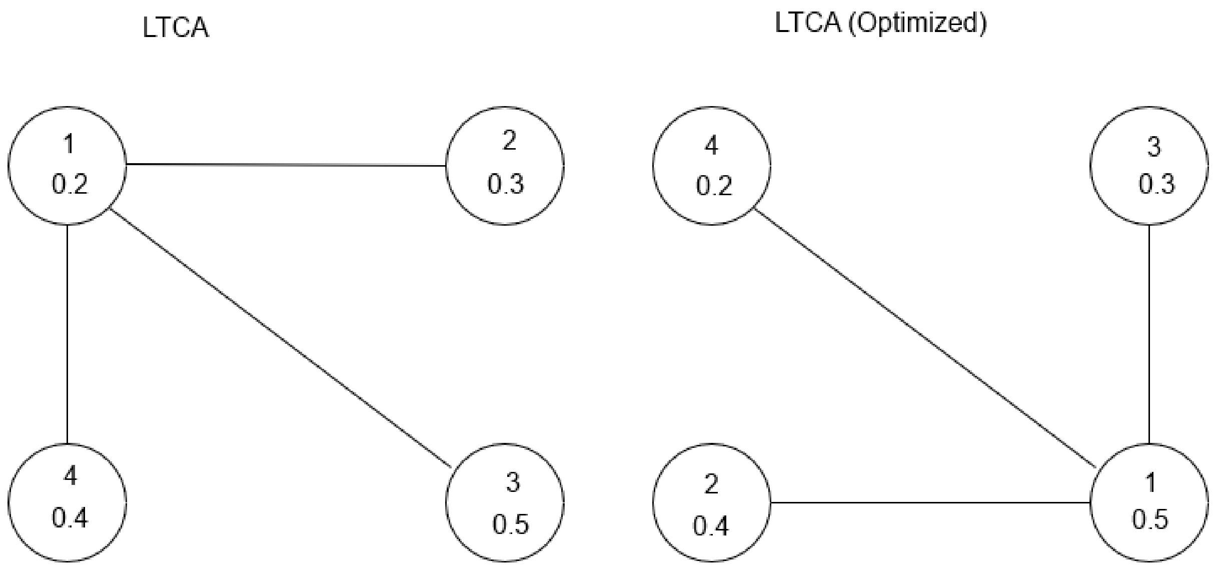


*Figure 6.6*

Both results provide a connect subgraph for the network that eliminates redundant edges, but the optimized version is structured better in terms of node energy. On the network found by the traditional local topology control algorithm, the node with the lowest energy has the most neighbors which will deplete the already limited power supply. The node with 0.5 energy has one connection and will last the longest, but that does not matter since other nodes will die first and break the connectivity of the graph. The optimized version takes the node with greatest energy and uses it to run the most connections allowing the other nodes to run longer by better distributing performance requirements. This organization of the network provides better performance by increasing the lifetime of the network.

## 7. Conclusion

After testing was completed on the sleep/wake scheduling algorithms, multiple patterns emerged. The first being that the minimum dominating set algorithm provided more disjoint sets when compared to the max first algorithm. This allowed for more opportunities to improve the sets using the local search, fixed depth and variable depth algorithms. The results show that these algorithms performed better, on average, when ran on the output sets from the minimum dominating set algorithm because of the amount of disjoint sets it provided compared to the max first algorithm. Of these three algorithms, the variable depth algorithm offered the best improvement for the lifetime of the network, providing the highest lifetime on every test network. The local search and fixed depth algorithm were comparable, with the fixed depth algorithm performing slightly better on average. The slowest algorithm, in regard to running time was the variable depth search which was very slow on the larger network tests when faced with more and larger sets. This is the main drawback to the algorithm, since the better results that it provides, requires more time to run. The local search algorithm performed the fastest, with the fixed depth algorithm having a slightly slower running time. These patterns appear to scale with the network size and density of the networks, so more nodes or a higher density should provide greater improvements, but also increase the running time of the improvement algorithms. All algorithms tested (local search, fixed depth and variable depth) gave an improved or at least equal network lifetime when compared to the lifetime of the original sets (minimum dominating set and max first).

The local topology control algorithm was also improved and showed promising results for improving the lifetime of wireless sensor networks. The optimization algorithm running before the local topology control algorithm organized the network in a more efficient manner compared to the control tests. The resulting network topologies had more high energy nodes with more connections compared to lower energy nodes when possible. These results show that this optimization technique could be used as a method of improving the lifetime of wireless sensor networks using a local topology control algorithm.

Overall the algorithms tested provided good results which satisfied the goals that were set at the beginning of this project. The local search, fixed depth, variable depth and local topology control algorithms all increased the network lifetime to varying degrees. The only algorithm which did not accomplish this consistently was the max first algorithm which did not perform as well as the minimum dominating set algorithm.

## References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks,* vol. 38, no. 4, pp. 393-422, 2012.

[2] J. Yick, B. Mukherjee and D. Ghosal, "Wireless sensor network survey," *Computer networks,* vol. 52, no. 12, pp. 2292-2330, 2008.

[3] S. Roundy, D. Steingart, L. Frechette, P. Wright and J. Rabaey, "Power sources for wireless sensor networks," in *Wireless sensor networks*, Springer, 2004, pp. 1-17.

[4] M. Cardei and D.-Z. Du, "Improving wireless sensor network lifetime through power aware organization," *Wireless Networks,* vol. 11, no. 3, pp. 333--340, 2005.

[5] T. Moscibroda and R. Wattenhofer, "Maximizing the lifetime of dominating sets," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, 2005, pp. 8--pp.

[6] K. Islam, S. G. Akl and H. Meijer, "A Constant Factor Localized Algorithm for Computing Connected Dominating Sets in Wireless Sensor Networks," in *Parallel and Distributed Systems, 2008. ICPADS'08. 14th IEEE International Conference on*, 2008, pp. 559--566.

[7] J. Yu, N. Wang, G. Wang and D. Yu, "Connected dominating sets in wireless ad hoc and sensor networks--A comprehensive survey," *Computer Communications,* vol. 36, no. 2, pp. 121--134, 2013.

[8] K. Islam, S. G. Akl and H. Meijer, "Maximizing the lifetime of wireless sensor networks through domatic partition," in *Local Computer Networks, 2009. LCN 2009. IEEE 34th Conference on*, 2009, pp. 436--442.

[9] R. Battiti, M. Brunato and F. Mascia, Reactive search and intelligent optimization, University of Trento, 2007.

[10] B. W. Kernighan and S. Lin, "An Efficiant Heuristic Procedure for Partitioning Graphs," *Bell system technical journal,* vol. 49, no. 2, pp. 291--307, 1970.

[11] M. Yagiura, T. Yamaguchi and T. Ibaraki, "A variable depth search algorithm for the generalized assignment problem," in *Meta-heuristics*, 1999, pp. 459--471.

[12] P. Santi, "Topology control in wireless ad hoc and sensor networks," *ACM computing surveys (CSUR),* vol. 37, no. 2, pp. 164--194, 2005.

[13] R. Rajaraman, "Topology control and routing in ad hoc networks: A survey," *ACM SIGACT News,* vol. 33, no. 2, pp. 60--73, 2002.

[14] K. Islam and S. G. Akl, "Localized Topology Control Algorithm with no Geometric Information for Ad hoc Sensor Networks," in *Sensor Technologies and Applications, 2008. SENSORCOMM'08. Second International Conference on*, 2008, pp. 65--72.