T.R.
ERCIYES UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING

DYNAMIC DEPLOYMENT OF WIRELESS SENSOR NETWORK WITH CONSTRAINTS BY USING ARTIFICIAL BEE COLONY ALGORITHM

Prepared by
AWS ABD AL KAREEM HAMEED ALTAEE

Advisor
Prof. Dr. Derviş KARABOĞA

MSc Thesis

July 2017
KAYSERİ

T.R.
ERCIYES UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
DEPARTMENT OF COMPUTER ENGINEERING

DYNAMIC DEPLOYMENT OF WIRELESS SENSOR NETWORK WITH CONSTRAINTS BY USING ARTIFICIAL BEE COLONY ALGORITHM

(MSc Thesis)

Prepared by
AWS ABD AL KAREEM HAMEED ALTAEE

Advisor
Prof. Dr. Derviş KARABOĞA

July 2017
KAYSERİ

SCIENTIFIC ETHICS DECLARATION

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

AWS ABD AL KAREEM HAMEED ALTAEE

SUITABILITY FOR GUIDE

The MSc thesis entitled **"Dynamic Deployment Of Wireless Sensor Network With Constraints By Using Artificial Bee Colony Algorithm"** has been prepared in accordance with Erciyes University Graduate Education and Teaching Institute Thesis Preparation and Writing Guide.

Prepared by

AWS ABD AL KAREEM HAMEED ALTAEE

Supervisor

Prof. Dr. Derviş KARABOĞA

Chairman of the Department of Computer Engineering

Prof. Dr. Derviş KARABOĞA

ACCEPTANCE AND APPROVAL PAGE

This study entitled **"Dynamic Deployment Of Wireless Sensor Network With Constraints By Using Artificial Bee Colony"** prepared by AWS ABD AL KAREEM HAMEED ALTAEE under the supervision of Prof. Dr. Derviş KARABOĞA was accepted by the jury as MSc. Thesis in Computer Engineering department.

13./07/2017

JURY:

Supervisor : Prof. Dr. Derviş KARABOĞA .....

Juror : Assoc. Prof. Dr. Bahriye AKAY .....

Juror : Assist. Prof. Dr. Gürkan YÜKSEK .....

APPROVAL:

That the acceptance of this thesis has been approved by the decision of the Institute's Board of Directors with the 18./7./2017 date and 2017/30-56 numbered decision.

19...../..7...../2017

Prof. Dr. Mehmet AKKURT

Director of the Institute

DYNAMIC DEPLOYMENT OF WIRELESS SENSOR NETWORK WITH CONSTRAINTS BY USING ARTIFICIAL BEE COLONY ALGORITHM

AWS ABD AL KAREEM HAMEED ALTAEE

Erciyes University, Graduate School of Natural and Applied Sciences
M. Sc. Thesis, July, 2017
Advisor: Prof. Dr. Derviş KARABOĞA

ABSTRACT

Wireless Sensor Networks (WSNs) are generally used for many private and military applications therefore; they have drawn important attention in recent and current years, due to the tiny size and low fabrication cost of the sensor nodes, power, bandwidth, memory, and other limited resources. Many research fields exist in wireless sensor networks, such as sensor network security, coverage, communication, etc. This work examines the major problem of dynamic deployment of WSNs by using an Artificial Bee Colony algorithm to optimize the coverage rate of wireless network of mobile sensor nodes. The position of the sensors in the sensing area is the most important part that affects the network coverage. Therefore, the main challenge of wireless sensor network deployment is positioning the sensor nodes in the area of interest to get maximum coverage. Within the dynamic deployment problem, sensors are firstly placed inside the area with random positions. If the sensors are mobile, they can change their positions by the usage of their information on other node positions. By those movements, the coverage rate is increased. This thesis focuses on the area coverage problem of mobile sensor nodes in wireless sensor network. The artificial bee colony algorithm is used for dynamic deployment of the mobile sensors to increase the coverage of wireless sensor networks. The artificial bee colony algorithm is implemented firstly to deploy mobile sensors in an area without any constraint or obstacle to get maximum coverage area. Then, the algorithm is utilized for an area with constraint and deployed the wireless sensor nodes by using two different methods. The first method depends on sensor coordinates. If a sensor coordinate is in the forbidden area, this sensor is redeployed outside the forbidden area. The second method depends on the fitness value of the sensors. If a sensor inside the forbidden area, the fitness takes a low value that produced from dividing the coverage rate over the number of sensors in the forbidden area +1, that means it is not a good solution.

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my thesis supervisor, Prof. Dr. Derviş KARABOĞA, for all his invaluable guidance, support and encouragement. He was always accessible and willing to help his students with their research. As a result, research life became smooth and rewarding for me. It has been my great pleasure and honor to have worked with him. I would also like to thank the members of my supervisory committee, Dr. Bahriye AKAY, Dr. Gürkan yüksek. I would also like to express my gratefulness to Dr. Beyza GÖRKEMLİ and all the faculty, staff and graduate students in the Department of Computer Engıneerıng, for their caring and support during my study at the Erciyes University. Finally, I would like to thank my family and friends for their encouragement and help, especially my parents who have devoted all their efforts in me, I am grateful for their continuing understanding and countless support throughout my entire life.

AWS ABD AL KAREEM HAMEED ALTAEE

July 2017, KAYSERİ

TABLE OF CONTENTS

DYNAMIC DEPLOYMENT OF WIRELESS SENSOR NETWORK WITH CONSTRAINTS BY USING ARTIFICIAL BEE COLONY ALGORITHM

CHAPTER 3

THEORIES AND METHODS

CHAPTER 4

SIMULATION RESULTS

CHAPTER 5

CONCLUSION

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

1.1. Introduction

Wireless sensor networks (WSN) have created powerful attention between researchers in the last few years because of their possible employment in a wide diversity of applications. Sensor nodes are inexpensive portable devices with limited processing power and energy resources. Sensor nodes can be utilized to gather information from the environment, also local process this data and transmit the sensed data back to the user. A wireless sensor network (WSN) consists of a set of self-organizing, lightweight sensor nodes that are working together to monitor physical or environmental conditions. Usually monitored parameters have information about temperature, sound, humidity, vibration, pressure and motion [1]. Each sensor node in a WSN is provided with a radio transmitter, various kinds of sensors, a battery unit, and a microcontroller. Wireless sensor networks are now applied in many industrial and public service areas embracing traffic monitoring, weather conditions monitoring, video surveillance, industrial automation and healthcare applications [1]. Because of the tiny size and low fabrication cost of the sensor nodes, power, bandwidth, memory, and other resources limit them. Many research fields exist in wireless sensor networks, such as sensor network security, coverage, communication, etc. In the area coverage problem, any sensor in the WSN covers a small area according to its range and the total covered area of the sensor network is obtained from the summation of covered regions of each sensor node. Maximizing the whole coverage area of the wireless sensor network is the main goal of the area coverage problem. The main purpose of the area coverage problem is certainly improving the performance of systems in different applications, such as target detection and tracking, monitoring the battlefield, homeland security, personal protection, and animal habit monitoring.

The position of the sensors is the most important part that instantly affects the network coverage. To get a large coverage, sensors are deployed in special ways. However, there are various conditions that sensors cannot be deployed in proper ways, such as in applications including areas of natural disasters or harsh environments. Also, if the area of interest is huge or has limited paths, many times, there is no ability to deploy sensors one by one in specific areas. They may alternatively be deployed completely at once from an aircraft or similar vehicles. If sensors are randomly deployed, the covered area originally obtained by the sensor network would not be considered as optimal coverage, as in the deterministic deployment. Figure (1.1) shows an example of a random deployment sensor networks, there are one hundred sensors deployed in a square sensing field. Every red circle represents a coverage range for one sensor and the white regions represent uncovered areas so in random deployment the density of the sensors is not distributed equally also the area of interest is not fully covered by the sensors.

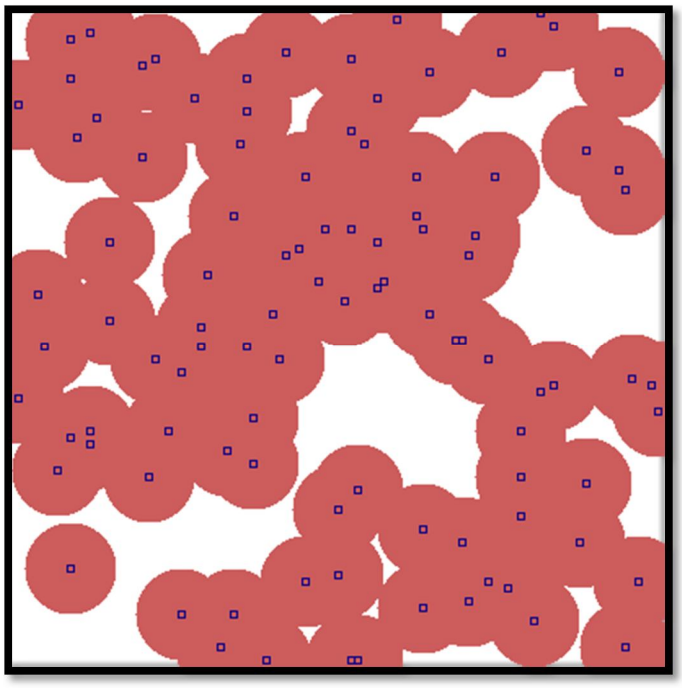


Figure 1.1. An example of random deployment of 100 sensors in a square area.

To enhance the coverage area of wireless sensor nodes, extravagant sensors are redeployed. Excess of the sensors makes the sensor networks have a larger density than normal ad-hoc networks. However, it is confirmed by [5] and [6] that only growing the sensor density cannot increase the coverage with 100% usually. Furthermore, it is expensive to preserve a large number of sensor networks. So, different methods have to

be implemented in order to avoid these problems and still improving coverage after the random deployment of the sensor nodes. The mobile sensors can be used for increasing the coverage instead of extending the sensing density in a wireless sensor network.

The main challenge of sensor network deployment is positioning the sensor nodes in the area of interest to get maximum coverage. In some kinds of networks, like global positioning systems (GPS), the sensors recognize their positions and might additionally study their neighbors' positions. Within the dynamic deployment problem, sensors are firstly placed inside the area with random positions. If the sensors are mobile, they can alternate their positions by the usage of their information of other positions. By those movements, the coverage rate is increased.

This thesis focuses on the area coverage problem of mobile sensor nodes in wireless sensor network. The artificial bee colony algorithm is used for dynamic deployment of the mobile sensors to increase the coverage of wireless sensor networks. Also, it is applied for deployment of wireless sensor nodes in areas with obstacles, to deploy sensors in a proper way to increase the coverage area.

1.1.1 Applications and uses of WSNs

Wireless sensor networks have some precise strengths which include low cost, the small size of nodes which might be quite easy to fabricate, inherent dispensed nature and ease of programming individual nodes to make the entire network works synchronously on user defined protocols. Numerous applications in this modern generation of science and technology make use of these strengths associated with WSNs [4].

i) WSNs in Military

One of the fundamental fields that ensure the effect of WSNs is their extensive application in military purposes [5]. WSNs are almost invented for different situations like monitoring, interference detection [6], cost evaluation, etc. Deployment of devices for military employment such as those listed typically has many detentions. One of those detentions, that deployment area cannot be attained by foot, in which case the devices require to be deployed from the air. Also, the spread nature of the WSN can be beneficial in a condition of fractional damage to the network. Mobile sensors or redeploying of sensors can enhance the network and repair the damages. Monitoring of

sensors reduces the necessity of human beings to be required in a real battlefield that's probably keeping many lives. Military forces can also use WSNs to protect important locations by locating the sensors at necessary locations. This makes a virtual field of protection in that region and can assist in recognizing invaders. WSNs can also be employed to observe the battlefield before and after military actions. They are also utilized for remote attacks in controlling resources such as homing rockets.

ii) WSNs in Environment

WSNs have many environmental application purposes also. The general risks like forest fires or floods could be catastrophic to the environment and require management efficiently. WSNs can be beneficial in these situations to estimate, limiting, or even the prevention damage caused. In matters such as forest fires, it is almost difficult for firefighters to certainly go into the distressed region. So, WSNs can be worked to determine the location of the source of the fire and also measure the area that affected by the fire and the potential progress of the fire into other areas. This can help administrators to be fully effective and thus limiting extra waste. WSNs could also be utilized to identify floods through the deployment of various types of sensors for measuring rainfall and water levels in water bodies [7]. WSNs could also be used to follow the movement of many kinds of animals and birds. That can stimulate researchers to understand the changing patterns of special types or monitor the status and health of threatened species. There are also many applications that use WSNs to examine the ground, greenhouse gasses, and atmosphere in an agricultural area and thus help in the improvement of agricultural production furthermore the health of the crop.

iii) WSNs in Industry and at home

As well as military and environmental purposes of WSNs, there occurs also an extended variety of industrial and household utilization. WSNs are very important in observing the health of vast machinery wherever manual reviews and activities are either too expensive or risky. Dangerous and difficult to access parts of these machines can be provided with a network of wıreless sensor nodes to observe cracks or unusual temperature and pressure and report them directly to the specialist in administration. WSNs have furthermore used in assisting people with their home requirements. These combine sensors and actuators located on internal devices to help users to manage their

houses better. WSNs are employed for general security by working as burglar detection systems to recognize burglars or visitors. WSNs can be programmed easily also efficiently deployed throughout the home and can be controlled by the homeowners from indoors the home.

1.2. Limitations and Challenges involved in WSNs

Despite the extensive use of WSNs in different fields, there are still unusual limitations occurs. It is quite necessary that the challenges posed due to these constraints are properly recognized and managed dramatically such that it won't influence the production and usability of WSNs. Posted below are some of the constraints of WSNs, attended by the challenges that are pretended due to these limitations.

1.2.1. Limitations

i) Sensor Lifetime

A common sensor node has a battery with the bounded capacity. The small size of the sensor and the absence of continual power supply indicates that sensors require being supplied with (small power source) that will slowly wasted depends on the number of activities (sensing, transmitting) that made. That power supply is connected to the sensors because of sensing, computation, and for frequently transmission of data. One time the battery is consumed, the sensor goes off and the most common deployment scenarios of WSNs indicate that these batteries can't be changed. Therefore the administrator of the WSN must be informed that any large amount of data transmission overheads need be avoided, if it is possible.

ii) Computational Power

Regularly, the sensors are provided with hardware to complete computations. Although this hardware's ability is limited because of space and expense limitations and doesn't have high computational functionalities of workstations or other types of wireless devices. Therefore, the sensors cannot produce high-intensity computations. Hence, complicated cryptographic encryption methods cannot be performed on these sensors. Thus, modifying a certain security mechanism in a WSN is quite critical to decrease the

computational load on the sensors also at the same time, holding the network safe from malicious nodes or adversaries.

ii) Connectivity and Compromise

An additional limitation of WSNs is that they are regularly deployed in antagonistic and remote regions where there is the intimidation of some nodes being destroyed or even worse, get negotiated by enemies. Usually they deploy new sensors in large quantities, but if the destruction of a large number of sensors and WSN can get cut. This means that the network is separated into two or more parts, which cannot communicate with each other. This leads to the loss of critical data and dramatically reduces the production of the WSN. In several situations the sensors might be compromised and might be stealing leading information or supplying incorrect information into the WSN, In several situations, the sensors might be compromised and might be stealing leading information or supplying incorrect information into the WSN, trying to damage it's running. This is a potential scenario and it is a challenge for the researchers to devise a solid and supple security mechanism.

1.2.2. Challenges

The above-cited limitations pose a few critical, demanding situations for researchers and it is very important that appropriate measures are taken to overcome them. One of them is the design assignment of scalability [2] and flexibility of the WSN. The conversation and authentication protocols need to be designed in this sort of way that including new nodes or changing the topology of the WSN need to now not disturb the steadiness of the system. Another layout task posed by means of WSNs is that the sensors proportion a lot of transmission as tons as viable to keep away from putting more burdens on some nodes and hence forcing them to speedy delete their strength source. A naive method to counter those boundaries could be to install a totally massive amount of sensors. But, this answer isn't always realistic due to the price concerned and additionally the undertaking in deploying the sort of a large number of sensors. Consequently, one of the critical WSN design demanding situations for researchers is to intelligently deploy the sensors in any such way that they maximize the productivity of the community, decrease the fee and at the identical time conquer the limitations of WSNs to offer a robust framework.

1.3. Purpose of this thesis

Wireless Sensor Networks play a major part in many real life scenarios and are being used widely for a wide variety of applications. Two of those are military, civilian security which calls for a frequent improvement on the stability, performance, and affordability of WSNs. We consider that deployment of sensors is a crucial aspect of the design of a WSN and can definitely contribute increasing the performance of WSN. In this thesis, an artificial bee colony algorithm is implemented to the dynamic deployment of mobile sensor networks to get a higher overall performance by deploying the sensors in a proper way to increase the coverage area of the network. The best performance of the algorithm indicates that it is able to be applied to the dynamic deployment of wireless sensor networks. The artificial bee colony algorithm is implemented firstly to deploy mobile sensors in an area without any constraints or obstacles to get maximum coverage area. Then the algorithm is utilized in an area with constraints and the artificial bee colony algorithm has deployed the wireless sensor nodes in two methods. The first method depends on sensor coordinates. If the sensor coordinates are inside the constraint area coordinates, the sensor is redeployed to the left or right outside the forbidden area. The second method depends on the fitness value of the sensor. If the sensor inside the constraint area, its fitness value will be low value, that means it is not a good solution. Iteration by Iteration the algorithm tries to out the sensors from the constraint region and improve the coverage rate in the area of interest. Chapter 2 of this thesis contains the related studies and the litreachere reviews. In chapter 3 we explain the idea of the artificial bee colony algorithm and how it Improve the solutions also, the simulation of the dynamic deployment of WSNs. In chapter 4 the result of the simulation is discussed. Chapter 5 includes the conclusion and the future research.

CHAPTER 2

RELATED WORK

This chapter explains the similar work and the literature review. We distribute the related optimization algorithms into two groups depending on their method of computation: distributed or centralized.

2.1. Distributed Optimization Algorithms for Sensor Redeployment

In the dynamic deployment of wireless sensor networks, distributed algorithms have been used. These algorithms are implemented for coverage optimization of mobile wireless sensor networks such as the virtual force associated algorithms [7]-[11].

A potential field method was submitted by Howard, et al., [7], for the coverage of sensor network optimization. Particularly, in [7], Therefore, each sensor is a point determined in the combined potential field generated by all the other sensors and constraints.. Any sensor will be impacted by a distribution force from the virtual potential field. The interactive force of a critical potential field of another sensor or obstacle is directly proportional to the square of length to that sensor or obstacle. The forces can be calculated as:

$$F_n = -k_n \sum_i \frac{1}{r_i^2} \cdot \frac{\vec{r_i}}{r_i} \quad (2.1)$$

$$F_o = -k_o \sum_i \frac{1}{r_i^2} \cdot \frac{\vec{r_i}}{r_i} \quad (2.2)$$

Where F_n Also F_o, indicate the force due to other nodes and obstacles correspondingly, r_i, is the directional vector pointing for the center sensor to other sensors or obstacles and r_i Is its value. The previous equation shows that both forces have a negative sign which means there are only repulsive forces. Sensors will be assigned weights so that

they can move dependıng on the virtual forces applied to them as they would in a real physical situation. It has been shown that balance state can be performed in a sensing field with borders. As a result, the sensors in the sensing field will reset themselves into a more identical configuration after being randomly deployed, and also, the network's coverage can be significantly improved.

Sheng, et al., [8] also applied the potential field algorithm. In their work, the potential field algorithm was employed to relocate the sink nodes that were provided by mobile robotic devices. After the relocation of the mobile sink nodes, many static sensor nodes could reach to the sink nodes so that the quality of performance is increased.

Yi Zou, et al., [9], introduced a virtual force algorithm. The main variation between [7] and [9] is that [9] has both repulsive and attractive forces between sensors, while [7] only has the repulsive force. The force depends on the interval between sensors also their relative position. The forces can be described as follows: →

$$\vec{F}_{ij} = \begin{cases} (k_A(d_{ij} - d_{th}), a_{ij} & if\ d_{ij} > d_{th} \\ 0 & if\ d_{ij} = d_{th} \\ (k_R/d_{ij}, a_{ij} + \pi) & if\ d_{ij} = d_{th} \end{cases} \quad (2.3)$$

Where d_{ij}, the length between sensors and a_{ij}, is the direction from the center node to other sensor nodes, and d_{th}, is a length threshold that has a close relative to the sensor's sensing range. The purpose of this algorithm is similar to obtain excellent overall coverage by relocating the mobile sensors. Various force models have been assumed, containing the impact of hot spots and obstacles form by Li et al., [10].

A ranking system has also been implemented to the force pattern, so that the algorithm can be employed for differing application elements. Wong et al., [11] developed the virtual force algorithms by implementing a back-off delay time in the virtual force algorithm. The back-off delay time is utilized such that the sensors relocate them one at a time in each cycle of movement. So, any sensor will have the complete updated location data of the other sensors, containing the change ın position of previous sensors in the current cycle. By this technique, the movement of the sensor is smaller than when they use the aged location information.

It must be noted out that the virtual force related algorithms defined before needs that every sensor knows the specific or relative positions of all other sensors in the network. In other words, location information from the sensors is needed, and must somehow be communicated throughout the network.

Another deployment algorithm for randomly allocated mobile sensor networks were submitted by Wang et al.,[12], Pac et al.,[13], and Chang, et al.,[14]. These algorithms expect that every sensor knows just the corresponding locations of sensors that are in its transmission spectrum or its neighborhood.

Especially, Wang et al.,[12], used Voronoi diagrams to identify areas that are not covered or the sensing holes and decided the way the sensors should move to increase the coverage space. Three optimization algorithms have been produced: vector based optimization (VEC), Voronoi diagram based optimization (VOR), and Minimax optimization. The VOR and Minimax have related achievements that are greater than the VEC when implemented in a low mass of sensor nodes. These algorithms also adopted a virtual movement method. By implementing these methods, each sensor determines the expected movements of all neighboring sensors and follows these virtual movements to predict their future locations using the virtual locations. This prophecy technique is reproduced many times, and then the sensor shifts only once. By applying the virtual movement method, the whole distance of sensor relocation is quite limited, along with the energy consumed. The deployment algorithms above are not described to converge faster. However, the convergence speed is an important factor in real- time applications.

2.2. Centralized Optimization Algorithms for Sensor Redeployment

The effectiveness of the deterministic deployment of a sensor network is a function of how efficiently it uses the sensor nodes to cover the desired area without any uncovered spots. This problem is related to the Art Gallery problem [13]. The Art Gallery problem is that of placing the cameras of an art gallery so that the least amount of camera sensors can cover the entire art gallery. This problem is an optimization problem that is Non-deterministic, Polynomial-time hard. The video cameras in the Art Gallery problem are quite related to the sensors in wireless sensor networks. The central difference is that the video cameras have no limit of the visible range, as they can see as long as no wall or

obstacle blocks their line of sight. However, the sensor nodes always have a limited sensing range.

2.2.1. Genetic Algorithm

Genetic algorithms (GAs) are a class of evolutionary, optimization algorithms. Evolutionary optimization algorithms were presented firstly by Barricelli in 1957 [15]. GAs are heuristic search algorithms that turn up from the idea of natural evolution [16] [17]. They get the theory of chromosomes, inheritance, mutation, and crossover in real evolution. In a genetic algorithm, the input variables are used as chromosome vectors. Several combinations of initial variables will be created.

The genetic algorithm estimates the costs of the fitness function associated with the input variables and holds the chromosomes of the better fitness values. The GA applies mutation and crossover operation on the chromosomes and always holds the better solutions. Therefore, the better chromosomes are continuously inherited. GAs is very useful in getting the global maximum without being trapped in a local optimum. This is as the initial population includes the entire solution range, so that locally optimized values will be avoided. They are proper for determining nonlinear optimization problems and to find the global optimal amount of a fitness function. Genetic algorithms can be applied for solving optimization problems by NP-hard complexity. Also, deterministic deployment algorithms try to increase the lifetime of sensor networks by reducing consumption power. Some earlier advanced algorithms have used GAs to achieve optimum deployment of wireless sensor networks.

Jourdan, et al., [18] used a Multi-Objective Genetic Algorithm to optimize the coverage of wireless sensor network. In the sensor network in [18], there is only one sink node. All sensors need to transmit data to the sink node directly or through multi-hop communication. Data transmission is the main concern of energy consumption in [18]. So with different communication ranges, the data transmission will have a different number of hops, changing the energy consumed.

GAs is proper for optimizing the deployment positions of sensor nodes, and scheduling of groups of sensor. However, the main restriction of GAs is they have quite the high computational complexity; thus, they need powerful CPUs. A typical sensor node

usually does not have the capacity to do so, and a central node with powerful computational capabilities is always required for GAs.

2.2.2. Particle Swarm Optimization

Particle Swarm Optimizations (PSOs) were first introduced by Kennedy, et al., [20] in 1995. The idea of a PSO comes from the natural behavior of birds seeking food. When a group of birds is looking for food together, each bird will first look around in the area near it. Each bird will communicate with the other birds where it finds the most amount of food near its area. Thus, all the birds can know which areas have the most amount of food in their entire, collective feeding area. Birds will continue looking for food in nearby places, especially where the most amount of food had been found in the entire area. PSO algorithms simplify this concept of organized labor. Similar to GAs, a group of candidates will be generated in PSOs from the entire space. Each candidate is a set of vectors that contain the variables related to the problem. The group of candidates will evolve to the combination of personal best fitness and the group global best fitness. This is like the process of the birds seeking food. Compared to GAs, the PSO has the advantage that it is easier to program and implement, and it has fewer parameters to control. PSO has been used in coverage and energy optimizations for non-mobile wireless sensor networks. Wang, et al., [35] used a parallel particle swarm optimization algorithm for optimizing the energy used by sensors to track targets. The parallel PSO is used for maximizing the coverage and minimizing the energy consumption by turning off the sensors that are far away from the targets. The simulation results showed significant improvement in energy efficiency. The PSO has also been used for relocating mobile sensor nodes [36] - [39]. Bai, et al., [36], used PSO to optimally relocate sensors that are initially randomly deployed. The objective of their PSO is to optimize the coverage of the sensor network with the least amount of sensor movements. Specifically, they studied two different cases of sensors mobility. In the first case, the sensors can move unlimited distance. In the second case, the sensors modeled with limited mobility, which means they can only move within a certain maximum distance. The PSO algorithm improved the network's coverage in both cases, especially in the limited mobility model. Also, the energy spent for the sensors' movement was greatly reduced. Wang, et al.,[37], developed a new PSO algorithm, called VFCPSO, which combines the virtual force (VF) algorithms and co-evolutionary

particle swarm optimizations (CPSO). In this algorithm, virtual forces are used to update the evolving velocity of each candidate solution and different, cooperating swarms. The algorithm optimizes the coverage of a network that contains both static and mobile sensors. Compared to the traditional PSO, the VFCPSO can perform better with increasing dimensionality of the optimization problem and decrease the computation time of the VFCPSO. Simulation results showed that VFCPSO provides 10% coverage increased compared to the traditional PSO. Huang and Lu [38] also applied the PSO to the coverage optimization of hybrid, wireless sensor networks. Li, et al., [39] imported selection and mutation to the PSO and named their algorithm Particle Swarm Genetic Optimization (PSGO). Their algorithm aims at improving the density of nodes. Mobile nodes move to cover the uncovered areas (coverage holes). Their work demonstrated that by relocating only 5 of 100 sensors, the algorithm could increase the coverage by 6%. However, these algorithms do not focus on energy savings and prolonging the lifetime of sensor networks with limited energy resources.

2.3. Self-localization in Wireless Sensor Networks

Self-localization is the capacity to get one's place to admire to a few recognized upon reference, and it is very necessary for wireless sensor networks, that many applications need information about where the data is hosted from. And it's essential to the coverage optimization algorithms, many of relocation algorithms specified before require identifying at the limited the relevant locations of sensors, and they can't perform if the sensors have no self-localization direction. Consequently, sensor localization must turn out to be an active research subject material in the last few years.

There are two varieties of sensor self-localization techniques; one is fine-Grained Localization and the alternative one is Coarse-Grained localization. The first kind typically has much less error within the localization consequences than the second one. However, it commonly includes either better computational complexity or hardware expense.

A common Coarse-Grained Localization algorithm for wireless sensor networks is the distance Vector hop (DV-Hop) approach delivered by Niculescu [40]. In this algorithm, the distances among nodes are predicted via the smallest amount of hops they require to communicate, where a hop is a transmission link used among sensors that form element

of a greater segmented transmission direction. Since neither the hop distance between each sensor set is steady, nor are they related directions frequently the equator, this algorithm can simply determine the relative area that different nodes are in, and not their specific places. Several algorithms are improved to optimize the initial DV-Hop. Li, et al., [41] produced a weighted DV-Hop self-localization design that can enhance the efficiency of the conventional DV-Hop. Other enhanced algorithms are [42-44, 47-51].

Bulusu, et al., [47] included a reference-point meant, a low-cost self-localization algorithm for wireless sensor networks. In such algorithm, base stations, or other kind of bases, is placed at grid locations. They carry out beacon signals containing their position data to nearby sensors. Certain sensors handle this data to determine their individual position by considering that they are at the centroid, or ordinary place, of all the locations from which they get beacon signals. If a sensor node can detect from source nodes settled in $(x_1, y_1), (x_2, y_2), \ldots (x_m, y_n)$, the sensor node will self-locate it as:

$$\begin{cases} x_{est} = \frac{1}{n}\sum_{i=1}^{n} x_i \\ y_{est} = \frac{1}{n}\sum_{i=1}^{n} y_i \end{cases} \tag{2.3}$$

where (x_{est}, y_{est}), is the evaluation position determined by the sensor. Therefore, this process does not need the sensors to identify or use the authority of the beacon signal, as the precise range is not determined. The efficiency depends on the number of the reference points and the ranges between them. Moreover, the accuracy of this method improves because of the density of the reference points increases.

Xu, et al., [48] included a common localization algorithm meant on receiving signal strength indicators (RSSI). In this algorithm, three base stations are located in various corners of a sensing area that, repeatedly, send out beacon signals to nearby sensors. A signal intensity map is created and saved by the different sensor nodes. On the map, there are covering discrete relating points at positions of predetermined signal power that are known to the sensor nodes. The specific sensors will analyze the received signal powers of the three base stations with that of the reference nodes and attribute to themselves the location of the reference points with the most related states. In the Coarse Grained Localization schemes explained above, there is not much calculation

required. But, the Fine Grained localization is another technique. Fine-grained localization algorithms can be divided into three types: received signal strength (RSS) based, time of arrival (TOA) based, and direction of arrival (DOA) based. In RSS based localization algorithms, radio propagation model equations are used, as the received signal power between sensors or base stations are instantly associated with the distance between them. The basic idea of an RSS-based localization algorithm is the perception by the sensors of their locations. At the beginning, every sensor gets the beacon from the reference nodes (base stations etc). Then, it estimates the distance to each reference node. Finally, it determines the distance equations to obtain its own position. The main impediment of this kind of algorithm is that the parameters, mainly that of the path loss type (also called distance-power gradient), in the equations, are not simply achieved due to changes in the environment of sensing area and the possibility of channel noise in lognormal fading. Therefore, many algorithms are improved to optimize the solutions of the equations and reduce the localization error.

Li, [49] provided an RSS-based localization algorithm for unknown distance potential gradient. He applied the gradient method to reduce the calculation error. Simulation results give an important interest of this algorithm above ones, including fixed distance-power gradient values when the distance-power gradient is not correct. Related works are also done in [50].

MacDonald, et al., [51] examined the RSS-based localization algorithm by an isotropic transmitter. They also decided the distance-power gradient to be 2, that is the open location estimation. They also apply the gradient way to optimize their outcomes. They inquired their algorithm with localized, cell phone base stations at Lake Shore Drive in Chicago.

More important RSS-based efforts are involved in the following. Shi, et al., [52] adopt the steepest descent approach to improve the node location of the RSS and reduce the impact of the noise. Yao, et al., [53] applied Particle Swarm Optimization to enhance the localization efficiency. Jia, et al., [54] handled Genetic Algorithm to achieve the optimization. Amutha, et al., [55] acquired a hybrid algorithm that initially utilizes the hop calculation to estimate the locations of the sensor nodes and then does RSS to improve the result.

TOA is different distance-based localization algorithm. In that kind of algorithm, the distance is computed using the time variations between during the beacon signal is sent and while it is received. Radio waves develop quite fast, through expecting a perfect clock to define the positions correctly. Therefore, the TOA algorithms become a powerful value. The global positioning system (GPS) is a common employment of the TOA localization [56]. Mobile devices use the signal received from satellites to determine the length between the satellites and themselves. The satellites also transmit their personal position information; therefore, the mobile devices can measure and evaluate their own positions. A comprehensive review of TOA-based localization algorithms is conferred in [57]. Other TOA algorithms can be seen in [58-60].

DOA and AOA (angle of arrival) algorithms are adopted for localization sensor nodes. In these algorithms, different sensors have the capability to indicate the direction of DOA and AOA (angle of arrival) algorithms is adopted for localization sensor nodes. In these algorithms, different sensors have the capability to indicate the direction of beacon signal they got. This regularly needs antenna patterns or some other complex hardware like directional antennas. The Sensors use the position information of the reference nodes and the angle, or direction, at that, they got the beacon signal to determine triangulation equations and discover their places [59-62].

While sensor networks are implemented indoors, perfect self-localization matures significantly more complicated. This is essential because of multipath declining, or signal degeneration when transmitted through solid materials, like metal and concrete found in buildings. Several enhancements have been implemented to reduce the noise in indoor environments. Pu, et al., [65] applied time-series filters to reduce the noise in RSS localization algorithms. Wu, et al., [66] used radio map filters to decrease the noise. Chen, et al., [67] examined the obstacles' impact on the localization systems and obtained an error-based enhanced RSS localization algorithm.

The localization mistakes of wireless sensor networks applied indoors are at least two meters, in nowadays produced products. This is common if there are not any barriers or obstacles that need to be used in calculations. But, if a sensor node is next to a wall, that is around twenty cm dense on medium, the before presented localization methods are not effective to decide the location of a sensor according to the obstacle. By that level of

inaccuracy, many relocation methods that explained before cannot operate accurately for indoor sensor networks.

2.4. Voronoi Diagram

A Voronoi diagram is a technique of decomposing an area. Imagine there is a collection of N sensor nodes deployed in an area out of all obstacles, the Voronoi diagram will divide the whole area into N Subareas, and any Subarea has a particular node within it. The quality of Voronoi diagram is that every Subarea is included in the area nearest to the node inside it, as exposed to the additional nodes. The production of the Voronoi diagram needs the position data of each node. Voronoi diagrams are quite helpful in the coverage problem of wireless sensor networks. If any sensor covers its individual Voronoi Subarea, the whole sensing area can be covered. Voronoi diagrams are utilized for identifying uncovered sections. Aziz, et al., [68] adopted PSO to optimize the coverage of wireless sensor networks. Voronoi diagrams are utilized for determining if there are some uncovered spaces, so that, to compute the fitness function. Meguerdichian, et al., [69] studied the best-worst state coverage in target tracking in wireless sensor networks. A Voronoi diagram is applied for the worst case coverage study. The worst case coverage occurred during the target moved at the edge side of Voronoi Subarea. Boukerche, et al., [70] optimized the centralized Voronoi diagram method and decreased the estimate for coverage problem in wireless sensor networks.

Wang, et al., [12], studied on a Voronoi diagram for uncovering area discovery in wireless sensor networks. After identifying the uncovered area, sensors, drive moving in that direction, by this movement the coverage does increase. Wang, et al., [12], also improved a method in that the sensors constantly go to the middle of the Voronoi Subarea, therefore, the sensors will have a higher possibility to cover the complete Voronoi Subarea. Voronoi diagrams have also been utilized to optimize the coverage of wireless sensor networks. Li, et al., [71] adopted the Voronoi diagram to help in decreasing the count of sensors required to cover the maximum sensing field for mobile sensors.

CHAPTER 3

THEORIES AND METHODS

This chapter explains the principles and methods that are associated with the optimization of dynamic deployment of the wireless sensor network and the artificial bee colony algorithm. The deployment (WSNs) that implemented can directly increase or decrease the coverage of wireless sensor networks. Various sensing models are presented in this chapter, also the use of artificial bee colony algorithm to redeploy the mobile sensor nodes in an area with constraint.

3.1. Problem description

As explained before, (WSNs) represent a main part in many real life projects and are being used widely for an extensive diversity of applications. Two of those are military, civilian security which calls for an improvement on the stability, performance of WSNs. The dynamic deployment of sensors is a critical point that concerns the plan of a WSN and can develop the performance of WSN. This section of the thesis represents the problem of this thesis and how to solve it. The problem is how to make dynamic deployment for a number of mobile sensor nodes in a specific sensing area to get the maximum coverage field. The deployment is implemented in different types of areas. Firstly, all areas is allowed to deploy sensor nodes on it, it means there are no obstacles or constraints. The other case when the area of interest has constraints on it. There are many different techniques for deployment. However, this thesis explains how to implement the ABC algorithm to optimize the locations of sensor nodes to get the maximum coverage rate. An artificial bee colony algorithm is implemented to the dynamic deployment of mobile sensor networks to get a higher overall performance by deploying the sensors in a proper way to extend and enhance the coverage rate of the network. The good results of the algorithm indicate that it is able to be implemented to

the dynamic deployment of WSNs. The (ABC) algorithm is implemented firstly to deploy mobile sensors in an area without any constraints or obstacles to get maximum coverage area. Then the algorithm is utilized in an area with constraints and the ABC algorithm has deployed the wireless sensor nodes in two methods. The first method depends on sensor coordinates. When the sensor coordinates are inside the constraint area coordinates, the sensor is redeployed and shifted to the left or right outside the forbidden area. The second process depends on the fitness value of the new solution. While the sensor inside the forbidden area, its fitness value will be low that means it is not a good solution. Iteration by Iteration the algorithm works to out the sensors from the constraint region and improve the overall coverage area.

3.2. Swarm Intelligence

Swarm intelligence has emerged as studies due to several research scientists of associated domains in recent years. Bonabeau has explained the swarm intelligence as "any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies". Bonabeau et al. adjusted their perspective on common mites simply like termites, bees, wasps also other various ant kinds. The word swarm is related to indicate any controlled group of socializing causes or individuals. One example of the swarm is the bees teeming nearby their colony; although, the comparison can simply be spread to additional systems that have generally the same architecture. The other example of the swarm is an ant colony that its special agents are ants. Also, a group of flight birds is a swarm of birds. The intelligence of swarm behavior is a good source for researchers to discover and improve new optimization techniques that can find the best solution (global minimum or maximum). These techniques or swarm algorithms represent a simulation of the swarm intelligence behavior in real life. Like how to discover the direct path to the food source or to a destination point. These behaviors have converted into a mathematical expression to implement it to find better solutions for complex optimization problems. The ant colony algorithm is an example of converting the real life ant's steps of discovering the less distance path between the food source and the nest into mathematical steps that can improve the solutions of optimization problems. Also (PSO) particle swarm optimization algorithm that its main idea based on the common behavior of bird or fish matures. In 2005 the (ABC) algorithm is improved by

(Karaboga) [48], the main idea of this algorithm depends on the nature behavior of honeybees that search the area to find the best food source that depends on the distance and the quality (nectar). This algorithm is simple to implement. Also, it can be applied in many different optimization applications and it has great results.

3.3. Artificial Bee Colony (ABC) algorithms

Artificial Bee Colony algorithm was produced by Karaboga in 2005, its idea based on the intelligence of real honey bee colonies [51]. The ABC represents foraging and dance behavior of real life bees to produce global solutions for various optimization problems. The community of artificial bees involves of three combinations of bees, the first is employed bees, the second are onlookers, and third are scouts. The colony includes 50% of employed, scout bees and 50% of the onlookers. Every employed bee in the colony is allocated for only one food source. Therefore, a number of the food sources and the employed bees are the same nearby the hive. In ABC algorithm, the place of a food source denotes a feasible solution to the optimization problem also the nectar value of a food source match for the quality of the solution (fitness). For each the population, the number of producing solutions is similar to the number of employed and onlooker bees. In the first cycle, the ABC produces a randomly assigned primary population P $(G = 0)$ of SN solutions (food source positions), where SN specifies the volume of the population as in Eq 3.3. Any solution xi $(i = 1,2,\dots,SN)$ is a D-dimensional vector. Here, D is number of optimization parameters. After an initialization step, the population of the places (solutions) is managed to repeat sequences, $C = 1,2,\dots,MCN$ of the search techniques of the employed, the onlooker, and the scout bees. An employed bees provides an adjustment to the position (solution) in their memory based on local information (visual information) and test the value of nectar (fitness value) of the new source (new solution) as in Eq 3.2. Provided that the amount of nectar of the new solution higher than its previous one, the bees preserves the new solution value and delete the old value. Otherwise, they maintain the position of the previous one in their memory. When the employed bees perform the entire search manner, they share the food source locations and the nectar knowledge with other bees (onlooker bees) on the dance space. An onlooker bee estimates the nectar knowledge that got from all employed bees and determines a food source with a chance associated with its nectar value. Similar to the employed bee, onlooker bees provides a modification on the site in

their memory and measure the nectar value of the applicant source. Performing that the new nectar value is bigger than the old one, the bee records the new location and remove the previous one. An artificial onlooker bee takes a food source depending on the expected rate equated with that food source, *pi*, determined by the following expression (3.1):

$$pi = \frac{fiti}{\sum_{n=1}^{SN} fitn} \tag{3.1}$$

Where $fiti$ is the fitness value of the solution i which is equivalent to the nectar value of the food source in the location i and SN is the number of food sources that is equal to the number of employed bees (*BN*).

In order to provide a competitor food location from the old food source in memory, the ABC uses the following expression (3.2):

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \tag{3.2}$$

Where $k \in \{1,2,...,SN\}$ and $j \in \{1,2,...,D\}$ are randomly determined indexes. While k is chosen randomly, its value has not to be equal *i.* φ_{ij}, and it is a random value between [-1, 1]. It regulates the generation of neighbor food sources nearby x_{ij} and denotes the ratio of two food places visually by a bee. In equation (3.2) it can be seen that when the difference between x_{ij} and x_{kj}, uncertainty is reduced, the disturbance in the x_{ij}, a position also decreases. Therefore, while the search progresses, give the optimum solution in the search range, the step length is adaptively decreased. If a parameter rate generated by this process passes its proposed limit, the parameter can be set to an admissible value. The food source that the nectar is rejected by the bees is renewed with a different food source by Scouts. In ABC, This is reproduced by randomly generating the position and changing it to an abandoned one. In ABC, Which states that a position cannot be developed more through a selected number of cycles, it is assumed that this food source is abandoned. The use of a decided quantity of cycles is an initial control parameter of the ABC algorithm that is called "limit" for stopping the search. Suppose that the abundant source is *xi,* and $j \in \{1,2, ..., D\}$, and then the Scout obtains a new food source to change it with x_i. Eq. (3.3) shows this process bellow:

$$x_i^j = x_{min}^j + rand(0,1)(x_{max}^j - x_{min}^j) \quad (3.3)$$

After every candidate source position in, v_{ij} is generated and then estimated by the artistic bee, comparing its performance with the old ones. Whenever the current food source has nectar equal or better than the previous source, it is changed with the old in memory. Otherwise, the old one is kept in memory. In other words, greedy selection mechanisms used as a process of choice between the old and the candidate one. As in the above explanation, it's clear that there are four control parameters used in the ABC: the number of food sources equal to the number of employed or onlooker bees (*SN*), the amount of the limit, and the maximum number of cycles (*MCN*).

3.4. Dynamic deployment of wireless sensor networks with artificial bee colony algorithm

As explained before, the ABC algorithm can be considered as one of the new swarm intelligence algorithms encouraged by the intelligent foraging behavior of honey bees is used for the dynamic deployment problem of WSNs. The approach of the optimization method is to extend the coverage value of the network, as in Eq. (3.4).

$$\text{CR} = \frac{\cup c_i}{\text{A}}, \; i \in S, \quad (3.4)$$

Where c_i , is the coverage rate of sensor i, *S,* is the set of nodes, and A, is the total measurement of the sensing area. The success of a sensor network essentially depends on the locations of the sensors in the area that located. Sensors must locate to increase sensing information that can be collected in the area of interest. In the static class of the problem, after the sensors first positioned, there will not be any further movement in the sensors. But, the dynamic class of the problem, sensors can move to new positions coordinately in the field of interest. Also, the coverage of WSNs highly depended on the sensor detection model that's used in the network, if it is a binary detection model or probabilistic detection model [52]. In this thesis, the binary detection model is used, that pretends no uncertainty in the detection of the sensor. We more pretend that there are *k* sensors in the random deployment step; every sensor has the same detection range *r*, and sensor *Si* is placed at the point (x_i, y_i). For any point *P* at (*x, y*), the Euclidean distance between s_i, and *P,* is $d(s_i, P)$. The binary sensor model is shown in Eq. (3.5).

$$c_{xy}(s_i) = \begin{cases} 1, & if\ d(s_i, P) < r \\ 0, & otherwise \end{cases}, \quad (3.5)$$

Where $c_{xy}(s_i)$, is the coverage of a grid point *P* by sensor s_i.

As mentioned before, the location of a food source means a feasible solution to the optimization problem and the nectar value of a food source match for the quality (fitness) of the related solution. Therefore, each sensor node position in the task area represents a possible food source in the algorithm. The coverage ratio of the network (the total coverage area), identifies to the fitness rate (nectar) of the solution. In the ABC model, colonies of artificial bee, where bees try to reach the best solution, and bees are classified into three groups: employed bees, onlookers, and scouts. An onlooker bee waits in the hive for an employed bee to learn the information about food sources. The employed bee makes some movements like dance inside the hive to teach the onlookers the positions of existing food sources to be able to get the food, a bee that goes to make random searches is called a scout. In ABC algorithm, there are many steps that can be taken to get the best solution as follows:

1. Initialize the parameters: sensor detection radius *r*, the size of the sensing area *A*, the number of mobile sensors *m*, colony size *cs*, the maximum number of iterations *MaxCycle*, and limit for scout *l*.

2. Deploy *m* sensors randomly for each food source *xi* of employed bees using Eq. (3.6).

$$x_{ij} = \min_j + rand(0,1)(\max_j - \min_j) \quad (3.6)$$

3. Evaluate the population.

4. *c* = 0.

5. Repeat.

6. Produce new solutions v_i, in the neighborhood of x_i for the employed bees using Eq.

(3.7).

$$v_{ij} = x_{ij} + \varphi_{ij}(x_{ij} - x_{kj}) \quad (3.7)$$

Here, k is a solution in the neighborhood of i, φ is a random number in the range [-1,1], and j is the randomly selected mobile sensor's position.

7. Check v_{ij} for staying in the bounds of the area.

8. Apply the greedy selection process between x_i and v_i.

9. Calculate probability values p_i for solutions x_i by means of their fitness values using Eq. (3.8).

$$p_i = \frac{0.9 \text{ x } fit_i}{fit_{best}} + 0.1 \quad (3.8)$$

10. Produce the new solutions,v_i, for the onlooker bees from solutions x_i, selected depending on p_i, and evaluate them.

11. Apply the greedy selection process for the onlookers between x_i and v_i.

12. Memorize the best solution achieved thus far.

13. Determine the abandoned solution; if it exists, replace it with a new randomly produced solution using Eq. (3.6).

14. $c = c + 1$.

15. Until $c = MaxCycle$.

The product solution represents an array that contains m elements. Figure 3.1 displays the solution set. The solution array elements are (x, y) positions of mobile sensors in the network.

Sensor number	1	2	3	4	i	……..	m
Position	(x1,y1)	(x2,y2)	(x3,y3)	(x4,y4)	(xi,yi)	…….	(xm,ym)

Figure 3.1. Solution array

3.5. Sensing Models

There are a couple sets of sensors in the physical environment. The first model of the sensor is related to the data at the position that it is located in, like temperature, humidity, and pressure sensors. The second type of sensor has a specific set of detectors such as motion detectors and video camera sensors. This model is used for area monitoring and safety surveillance. In the real applications, the sensing area of sensors may be not the same because of the constraints in the sensing field, such as obstacles or rain and snow. The sensing range of every sensor is regularly pretended to be a circular shape in the sensing area. Generally, there are two various models of sensing that used for simulating the achievement of sensors [9]: the binary and probability models.

The difference between binary sensing model and probability sensing model is in the probability model if a target is in a particular region, the target can be recognized with a definite probability value of (0) to (1). However, in the binary model, the target detection is equal either (0) that means the target is undetected or (1) that means the target is detected.

In this thesis, the binary sensing model is used. Figure 3.2 shows the difference between the two models.

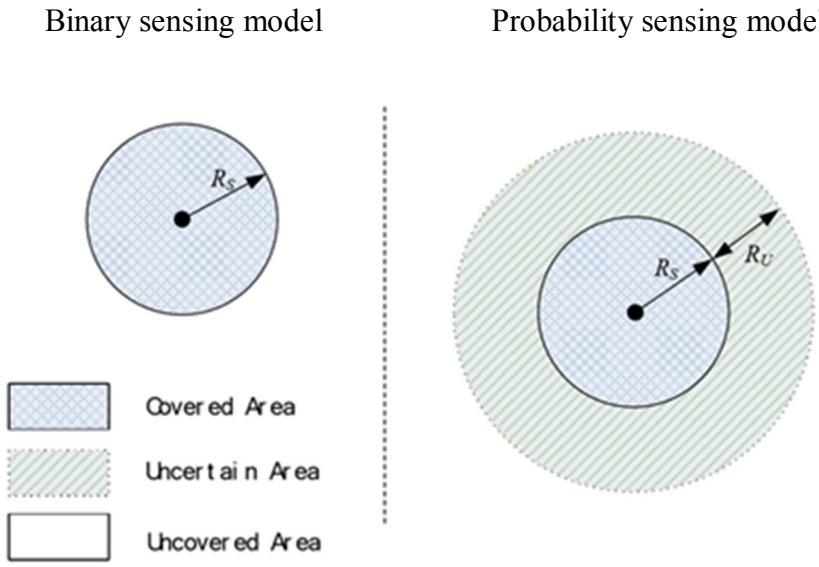


Figure 3.2. Differences between binary sensing model and probability sensing model.

3.5.1. Binary sensing model

S is a sensor node at position (x_s,y_s), we pretend that the sensing field of the sensor *S* is a circular shape with a radius R_s and centered at (x_s,y_s), R_s is named the radius sensor node *S*.

In the binary detection model, the sensor S is able for detecting the target that found inside its sensing area by a probability equals 1, but it cannot recognize every target that is farther of its sensing area. Therefore, within the binary model, a sensor can identify the target with a probability of 1 if the range between the target region and the sensor is smaller than the sensing radius R_s. But, if the length between a target and the sensor is more than R_s, the sensor will have zero probability of identifying it.

Consider a target *T* is placed at coordinates (x_t,y_t), the probability that the target *T* will be identified by sensor *S* in a binary model can be shown in the following equation:

$$p_{sb} = \begin{cases} 1, & D_{TS} \le R_s \\ 0, & otherwise \end{cases} \quad (3.9)$$

Where p_{sb} represents the sensing probability in binary model, D_{TS} is the distance between target T and sensor *S* which can be estimated as:

$$D_{TS} = \sqrt{(x_s - x_t)^2 + (y_s - y_t)^2} \quad (3.10)$$

There is no transition range in the binary model. A small variation in positions may cause a completely different detection output. The binary model is the clearest model for sensing.

3.5.2. Probability sensing model

The probability model differs from the binary model, that there is a transitional range that through a sensor clearly can or cannot identify a target. Where will be a certain range for every sensor, in which the sensor cannot determine if a target can be detected? In that area, the target will have a probability to be discovered from 0 to 1. The probability model has two significant distances for a sensor, the first one is R_s that is similar to the binary model. If the space between the target and the sensor is lower than

or equals R_s, the sensor can detect the target with a probability of 1. The second crucial distance is R_U, which represent the uncertain possible range. If the range between the target and the sensor is within the range between R_s and R_s+ R_U, the probability that the target will be identified by the sensor is similar to the field between them. If the space between the target and the sensor is further than R_s+ R_U, the target will not be recognized by the sensor. The mathematical expression of probability model is as shown below:

$$p_{sb} = \begin{cases} 1 & D_{TS} \leq R_s \\ e^{-\lambda a^{\beta}} & R_s < D_{TS} \leq R_s + R_U \\ 0 & R_s + R_U < D_{TS} \end{cases} \quad (3.11)$$

Where a = D_{TS} - R_s, and λ and β are constants belong to the sensors' hardware features.

3.6. Coverage Calculation

The coverage calculation is only one model that matches both of the sensing models that are explained before. The term coverage ratio (R_{Coverage}) is represents the rate of an area that can be covered by sensors cooperatively (A_{Covered}) over the entire sensing area (A_{Total}) as in Eq. 3.12.

$$R_{Coverage} = \frac{A_{Covered}}{A_{Total}} \quad (3.12)$$

The greatest coverage rate is 1. In this thesis, we will use a percentage to represent the coverage value.

The part of an area is covered if there is one sensor node covers that part or by several sensors joins detections. Imagine that there are N sensor nodes in the complete sensing area; the common detection probability for a particular field is calculated as in Eq. 3.13.

$$p_{Cover} = 1 - \prod_{s=1}^{s=n} (1 - p_s) \quad (3.13)$$

Where p_s can be found by the equation (3.9) and (3.11) according to the model that be used.

In the binary model, p_{Cover} will be equals 1 or 0, that means the field will be covered or not. But, in the probability model, p_{Cover} will be each value started from 0 to 1. An entrance (p_{th}) is needed for deciding if a region is covered. If p_{Cover} is greater than (p_{th}), the area is supposed to be covered, oppositely, the region is not covered. The amount of (p_{th}) depends on the purpose specification.

Because of the random locations of sensors, the field they cover will be different. One way for determining an abnormal field is the Monte Carlo method. The Monte Carlo approach is a mathematical method for getting statistical results for the problems that are hard to obtain or do not have analytical results. In the Monte Carlo approach, random individuals will be estimated regularly. If a satisfactory number of individuals are estimated, the results will be pretty near to the required value of the result [79] [80].

In the Monte Carlo approach, the range of an unusual field can be measured in the subsequent method. First, a regular space that encircles the target field is chosen. Second, we randomly select a number of points in the regular area and referee if all is in the target range. Then, the coverage rate is estimated as the number of points within the target range over the number of all individual points. The field will be the coverage rate multiplied by the area of the regular space.

In our simulation, we did not test randomly when computing the coverage. But, we managed the sensing areas as a grid and managed every grid point as an individual point for estimating the coverage.

CHAPTER 4

SIMULATION RESULTS

This chapter shows the simulation results of WSN, consists of 100 mobile sensor nodes and all the sensors have the same characteristics, the detection range for each sensor is 7 meters. The whole region of interest is 10.000 square meters, with 100 meters length and 100 meters width. Computer software is used for simulating the dynamic deployment of the specific WSN. That software needs high performance computer with CPU CORE i7 and RAM 8 GB, to implement an ABC algorithm to maximize the coverage area of the network. The dynamic deployment is performed in two cases. The first case is without constraints (there is no any obstacle in the sensing region). And the second case is with constraints that mean not all the area is allowed to be deployed with the sensor nodes. In this case, the algorithm will deploy the sensor nodes out of the obstacle region and this will be done by using two techniques. The first technique depends on the coordinates of the sensor node. The second technique depends on the fitness value of the produced solution.

4.1. Dynamic Deployment of WSN Without Constraints

In this case, there is no any obstacle in the area of interest. The fitness function equals the coverage rate as in Eq. (1.4).

$$fitness = \frac{A_{Covered}}{A_{Total}} \tag{4.1}$$

Where $A_{Covered}$ is the area that covered by sensor nodes, A_{Total} the total area.

If the position of the sensor goes beyond the area the assignments (4.2) are employed.

$$\begin{Bmatrix} if & x_i < x_{min} \ then \ x_i = x_{min} \\ if & x_i > x_{max} \ then \ x_i = x_{max} \\ if & y_i < y_{min} \ then \ y_i = y_{min} \\ if & y_i > y_{max} \ then \ y_i = y_{max} \end{Bmatrix} \quad (4.2)$$

Where x_{min}, y_{min} is the lower boundary of the sensing area, x_{max}, y_{max} is the upper boundary of the sensing area.

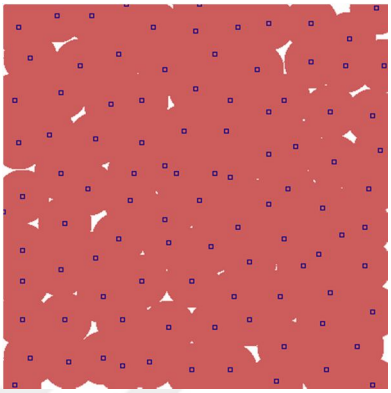
The number of cycles equals 1000; there are different values for colony size changes from 10, 20, 30, 40, 60, 80 and 100. The α value determines the value of the limit for the scout bees by Eq. 4.3.

$$l = \frac{D*CS}{2} * \alpha \quad (4.3)$$

Where l is the limit, D is the dimension of the problem, CS is the colony size, α is Alpha value.

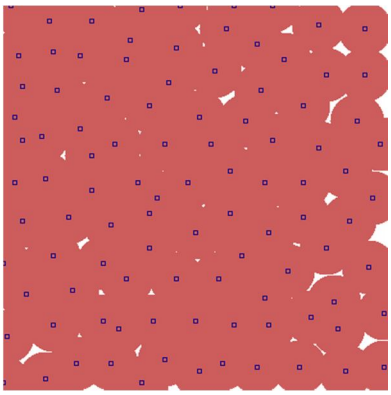
The α value changes from 0.25, 0.5, 1, 1.5 and 2, for example, in the first stage the colony size is 10 and the α is 0.25, other control parameters are the same, the algorithm runs for 30 times, every run consists of 1000 iterations. The results of 30 runs are collected and analyzed to find the best solution (maximum coverage value), Worst solution, the Mean value and the Standard deviation of the results, these values is set into a table to be able to compare the current results with the next results. Next, the colony size is still 10 and α value will change to 0.5, and the same procedure is taken. When the result of each α value change is obtained, the colony size will set to 20 and the same steps are done until colony size equals 100. When all results are collected, the result will be analyzed to see in which case the coverage is the maximum rate. This case represents the best deployment for the WSN. Table (4.1) shows the best result for 30 runs when colony size equals 10 and α equals 0.25.

Table 4.1. The best result for 30 runs when colony size equals 10 and α equals 0.25.

Colony size	10	Best solution 
α	0.25	
Best solution	0.9797	
Worst solution	0.9688	
Mean value	0.974886667	
Standard deviation	0.002941467	

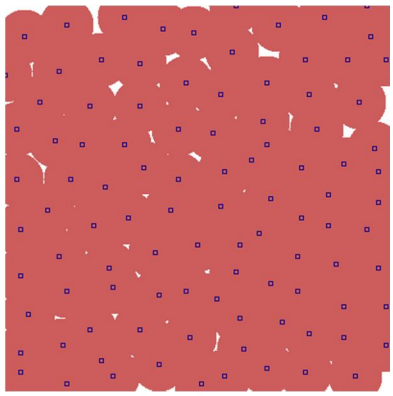
As shown in Table 4.1, when colony size is 10 and α is 0.25, the best solution is 0.9797, the Worst solution is 0.9688, the Mean value calculated by finding the summation of all results and divided by 30 (the number of results), Standard deviation represents the stability of the algorithm that means if there is not a huge difference between results the Standard deviation value will decrease. In Table 4.2, all the control parameters are the same, but the α value is changed to 0.5 as shown in Table 4.2:

Table 4.2. The best result for 30 runs when colony size equals 10 and α equals 0.5.

Colony size	10	Best solution 
α	0.5	
Best solution	0.9804	
Worst solution	0.9705	
Mean value	0.974976667	
Standard deviation	0.002566452	

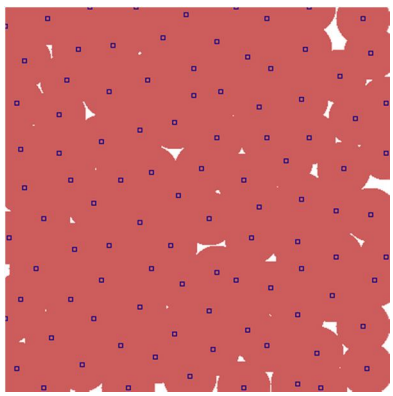
In Table 4.2, as you can see when α is 0.5 the coverage is increased and the Standard deviation is decreased. In Table 4.3, α equals 1 and all control parameters are the same.

Table 4.3. The best result for 30 runs when colony size equals 10 and α equals 1.

Colony size	10	Best solution
α	1	
Best solution	0.9805	
Worst solution	0.9711	
Mean value	0.97567	
Standard deviation	0.002327746	

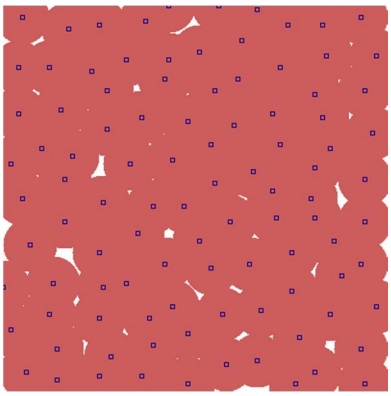
In Table 4.3, as you can see when α is 1, the coverage is increased and the Standard deviation is decreased, that means the result is improved. In Table 4.4, α equals 1.5 and all control parameters are the same.

Table 4.4. The best result for 30 runs when colony size equals 10 and α equals 1.5.

Colony size	10	Best solution
α	1.5	
Best solution	0.9811	
Worst solution	0.9713	
Mean value	0.975856667	
Standard deviation	0.002275529	

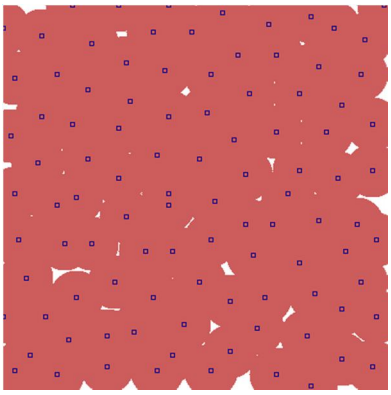
As shown In Table 4.4, when α is 1.5 the coverage is increased and the Standard deviation is decreased, that means the result is improved. In Table 4.5, α equals 2 and all control parameters are the same.

Table 4.5. The best result for 30 runs when colony size equals 10 and α equals 2.

Colony size	10	Best solution
α	2	
Best solution	0.9827	
Worst solution	0.9732	
Mean value	0.97631333	
Standard deviation	0.0020741	

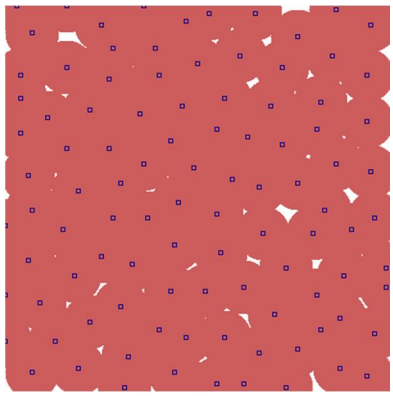
As shown In Table 4.5, when α is 2 the coverage is increased and the Standard deviation is decreased, that means the result is improved. Now let's change the colony size to 20 and set the α value to 0.25 and run the algorithm for 30 times, Table 4.6 shows the best results.

Table 4.6 The best result for 30 runs when colony size equals 20 and α equals 0.25.

Colony size	20	Best solution
α	0.25	
Best solution	0.9816	
Worst solution	0.9592	
Mean value	0.976823333	
Standard deviation	0.004202272	

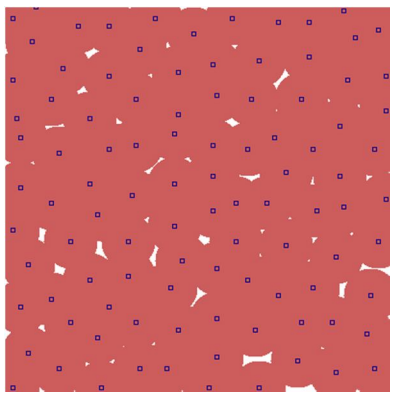
When the results in Table 4.6 are compared with the results in Table 4.1, it shows that when the colony size is increased, the result will be better, with the same α value. Let's set the α value to 0.5, Table 4.7 shows the results.

Table 4.7. The best result for 30 runs when colony size equals 20 and α equals 0.5.

Colony size	20	Best solution
α	0.5	
Best solution	0.9822	
Worst solution	0.9733	
Mean value	0.977576667	
Standard deviation	0.002504056	

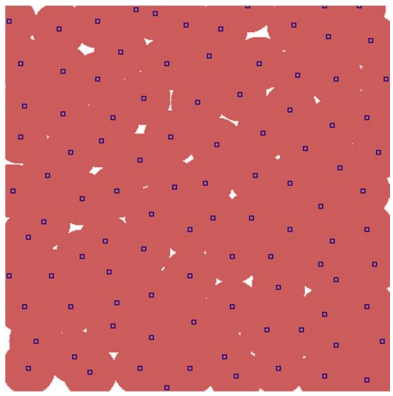
As you can see, the results in Table 4.7 are better than the results in Table 4.6, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Now let's set the α value to 1 without changing other parameters, Table 4.8 shows the results.

Table 4.8. The best result for 30 runs when colony size equals 20 and α equals 1.

Colony size	20	Best solution
α	1	
Best solution	0.9826	
Worst solution	0.9737	
Mean value	0.978263333	
Standard deviation	0.002131647	

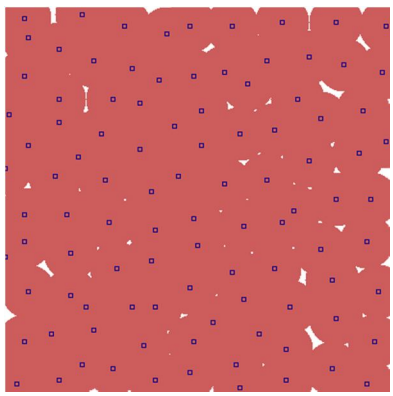
As you can see, the results in Table 4.8 are better than the results in Table 4.7, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Now let's set the α value to 1.5 without changing other parameters, Table 4.9 shows the results.

Table 4.9. The best result for 30 runs when colony size equals 20 and α equals 1.5.

Colony size	20	Best solution
α	1.5	
Best solution	0.983	
Worst solution	0.974	
Mean value	0.978326667	
Standard deviation	0.001915037	

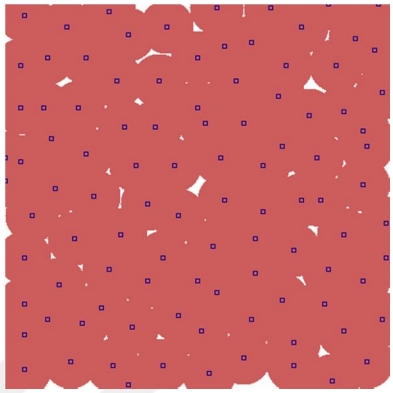
As you can see, the results in Table 4.9 are better than the results in Table 4.8, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Now let's set the α value to 2 without changing other parameters, Table 4.10 shows the results.

Table 4.10. The best result for 30 runs when colony size equals 20 and α equals 2.

Colony size	20	Best solution
α	2	
Best solution	0.9833	
Worst solution	0.9745	
Mean value	0.978643333	
Standard deviation	0.001863716	

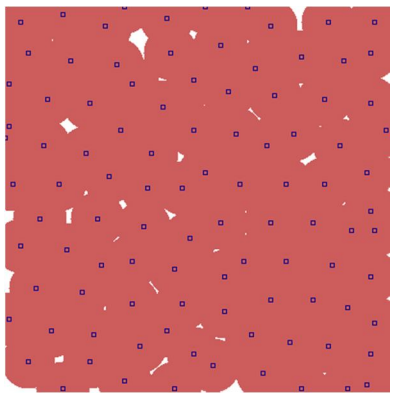
As you can see, the results in Table 4.10 are better than the results in Table 4.9, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Now let's set the colony size to 30 and α value to 0.25 without changing other parameters, Table 4.11 shows the results.

Table 4.11. The best result for 30 runs when colony size equals 30 and α equals 0.25.

Colony size	30	Best solution 
α	0.25	
Best solution	0.9809	
Worst solution	0.9759	
Mean value	0.978876667	
Standard deviation	0.005513891	

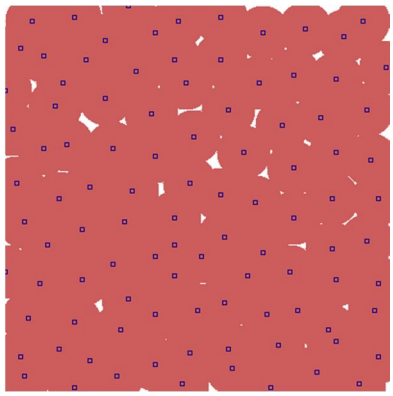
When the results in Table 4.11 are compared with the results in Table 4.6, it shows that when the colony size is increased, the result will be better, with the same α value. Let's set the α value to 0.5, Table 4.12 shows the results.

Table 4.12. The best result for 30 runs when colony size equals 30 and α equals 0.5.

Colony size	30	Best solution 
α	0.5	
Best solution	0.9816	
Worst solution	0.9577	
Mean value	0.97904	
Standard deviation	0.002115083	

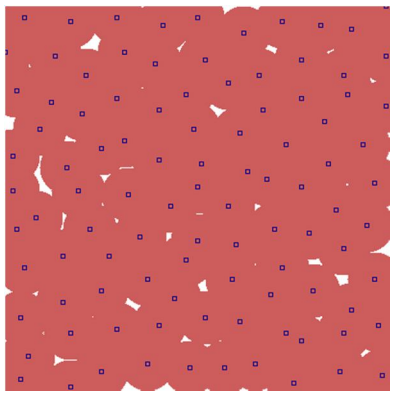
As you can see, the results in Table 4.12 are better than the results in Table 4.11, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Let's set the α value to 1 without changing other parameters, Table 4.13 shows the results.

Table 4.13. The best result for 30 runs when colony size equals 30 and α equals 1.

Colony size	30	Best solution
α	1	
Best solution	0.9821	
Worst solution	0.9751	
Mean value	0.977566667	
Standard deviation	0.001793375	

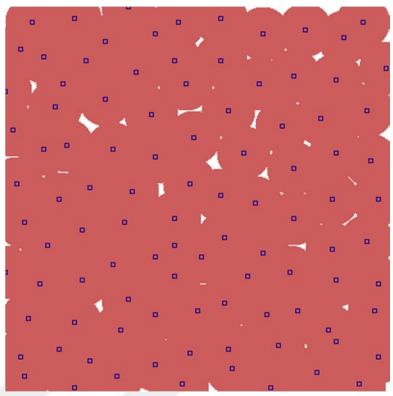
As you can see, the results in Table 4.13 are better than the results in Table 4.12, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Let's set the α value to 1.5 without changing other parameters, Table 4.14 shows the results.

Table 4.14. The best result for 30 runs when colony size equals 30 and α equals 1.5.

Colony size	30	Best solution
α	1.5	
Best solution	0.9822	
Worst solution	0.9753	
Mean value	0.978536667	
Standard deviation	0.00148039	

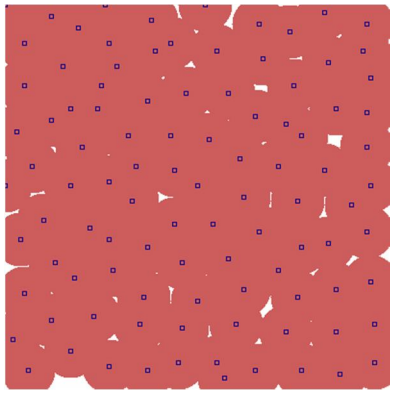
As shown in Table 4.14, the results are better than the results in Table 4.13, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Let's set the α value to 2 without changing other parameters, Table 4.15 shows the results.

Table 4.15. The best result for 30 runs when colony size equals 30 and α equals 2.

Colony size	30	Best solution 
α	2	
Best solution	0.984	
Worst solution	0.9759	
Mean value	0.978876667	
Standard deviation	0.001465076	

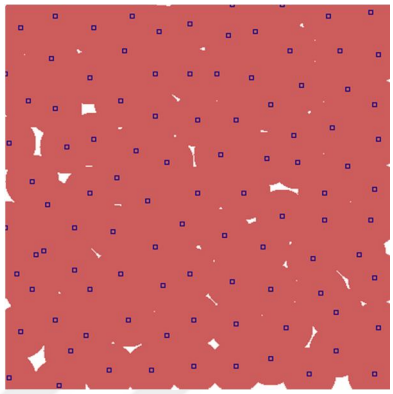
As you can see, the results in Table 4.15 are better than the results in Table 4.14, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Now let's set the colony size to 40 α value to 0.25 without changing other parameters, Table 4.16 shows the results.

Table 4.16. The best result for 30 runs when colony size equals 40 and α equals 0.25.

Colony size	40	Best solution 
α	0.25	
Best solution	0.9816	
Worst solution	0.9752	
Mean value	0.978786667	
Standard deviation	0.002124582	

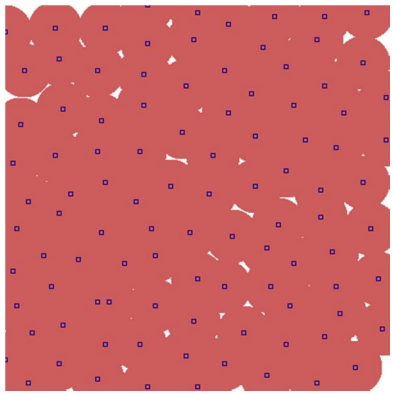
When the results in Table 4.16 are compared with the results in Table 4.11, it shows that when the colony size is increased, the result will be better with the same α value. Let's set the α value to 0.5, Table 4.17 shows the results.

Table 4.17. The best result for 30 runs when colony size equals 40 and α equals 0.5.

Colony size	40	Best solution
α	0.5	
Best solution	0.9823	
Worst solution	0.9755	
Mean value	0.979066667	
Standard deviation	0.0020517	

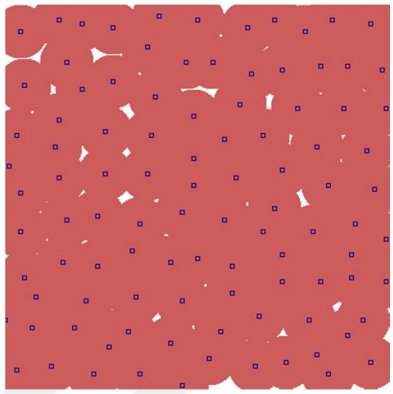
As shown In Table 4.17, when α is 0.5 the coverage is increased and the Standard deviation is decreased, that means the result is improved. Now let's set the α value to 1 and run the algorithm for 30 times, Table 4.18 shows the best results.

Table 4.18. The best result for 30 runs when colony size equals 40 and Alp α ha equals 1.

Colony size	40	Best solution
α	1	
Best solution	0.9826	
Worst solution	0.9758	
Mean value	0.979083333	
Standard deviation	0.001742402	

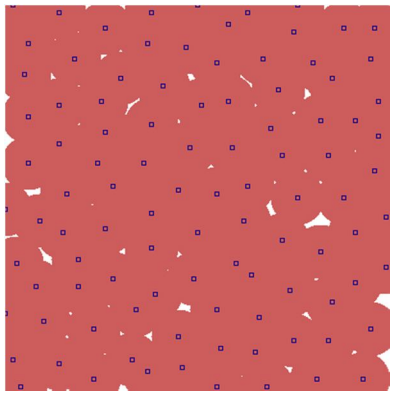
As shown in Table 4.18, the results are better than the results in Table 4.17, let's set the α value to 1.5 without changing other parameters, Table 4.19 shows the results.

Table 4.19. The best result for 30 runs when colony size equals 40 and α equals 1.5.

Colony size	40	Best solution
α	1.5	
Best solution	0.9831	
Worst solution	0.9760	
Mean value	0.97917	
Standard deviation	0.001516514	

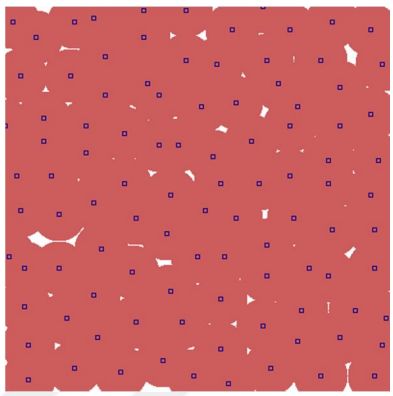
As shown in Table 4.19, the results are better than the results in Table 4.18, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Let's set the α value to 2 without changing other parameters, Table 4.20 shows the results.

Table 4.20. The best result for 30 runs when colony size equals 40 and α equals 1.5.

Colony size	40	Best solution
α	2	
Best solution	0.985	
Worst solution	0.9767	
Mean value	0.979646667	
Standard deviation	0.00147071	

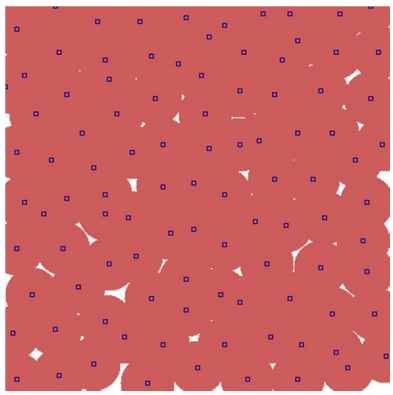
As you can see, the results in Table 4.20 are better than the results in Table 4.19, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Now let's set the colony size to 60 and α value to 0.25 without changing other parameters, Table 4.21 shows the results.

Table 4.21. The best result for 30 runs when colony size equals 60 and α equals 0.25.

Colony size	60	Best solution 
α	0.25	
Best solution	0.9829	
Worst solution	0.9703	
Mean value	0.979613333	
Standard deviation	0.004138777	

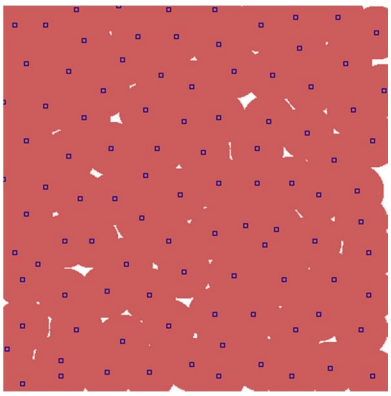
When the results in Table 4.21 are compared with the results in Table 4.16, it shows that when the colony size is increased, the result will be better, with the same α value. Let's set the α value to 0.5, Table 4.22 shows the results.

Table 4.22. The best result for 30 runs when colony size equals 60 and α equals 0.5.

Colony size	60	Best solution 
α	0.5	
Best solution	0.9841	
Worst solution	0.9766	
Mean value	0.979696667	
Standard deviation	0.001899089	

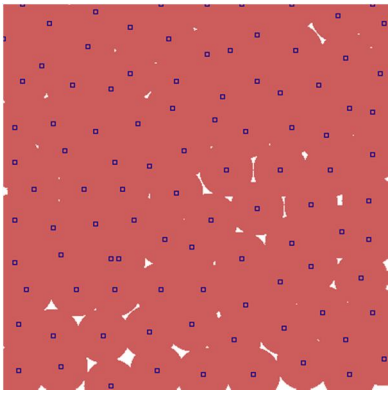
As shown in Table 4.22, the results are better than the results in Table 4.21, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Let's set the α value to 1 without changing other parameters, Table 4.23 shows the results.

Table 4.23. The best result for 30 runs when colony size equals 60 and α equals 1.

Colony size	60	Best solution
α	1	
Best solution	0.9842	
Worst solution	0.9767	
Mean value	0.97974	
Standard deviation	0.001858921	

As shown in Table 4.23, the results are better than the results in Table 4.22, let's set the α value to 1.5 without changing other parameters, Table 4.24 shows the results.

Table 4.24. The best result for 30 runs when colony size equals 60 and α equals 1.5.

Colony size	60	Best solution
α	1.5	
Best solution	0.9851	
Worst solution	0.9768	
Mean value	0.979933333	
Standard deviation	0.001618712	

As shown in Table 4.24, the results are better than the results in Table 4.23; also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Let's set the α value to 2 without changing other parameters, Table 4.25 shows the results.

Table 4.25. The best result for 30 runs when colony size equals 60 and α equals 2.

Colony size	60	Best solution
α	2	
Best solution	0.9856	
Worst solution	0.9776	
Mean value	0.980356667	
Standard deviation	0.001403858	

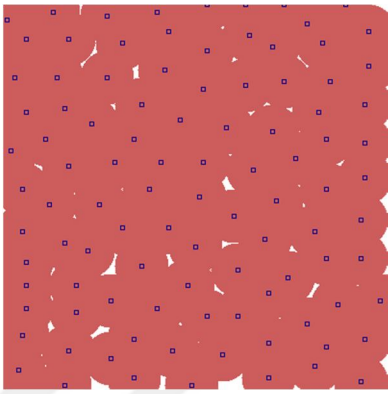
As you can see, the results in Table 4.25 are better than the results in Table 4.24, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Now let's set the colony size to 80 α value to 0.25 without changing other parameters, Table 4.26 shows the results.

Table 4.26. The best result for 30 runs when colony size equals 80 and α equals 0.25.

Colony size	80	Best solution
α	0.25	
Best solution	0.984	
Worst solution	0.9773	
Mean value	0.980906667	
Standard deviation	0.002044494	

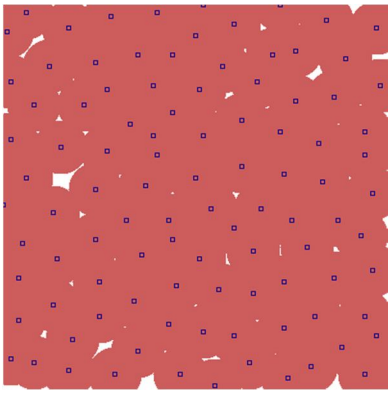
When the results in Table 4.26 are compared with the results in Table 4.21, it shows that when the colony size is increased, the result will be better, with the same α value. Let's set the α value to 0.5, Table 4.27 shows the results.

Table 4.27. The best result for 30 runs when colony size equals 80 and α equals 0.5.

Colony size	80	Best solution
α	0.5	
Best solution	0.9843	
Worst solution	0.9775	
Mean value	0.98105	
Standard deviation	0.001827092	

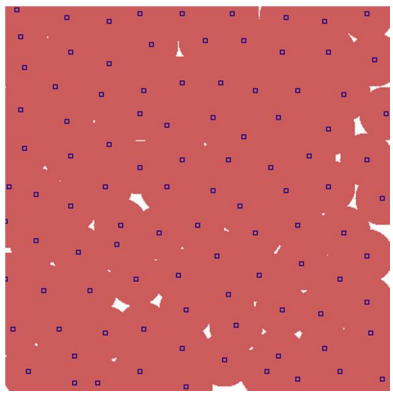
As shown in Table 4.27, the results are better than the results in Table 4.26, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Let's set the α value to 1 without changing other parameters, Table 4.28 shows the results.

Table 4.28. The best result for 30 runs when colony size equals 80 and α equals 1.

Colony size	80	Best solution
α	1	
Best solution	0.9845	
Worst solution	0.9783	
Mean value	0.981103333	
Standard deviation	0.001789317	

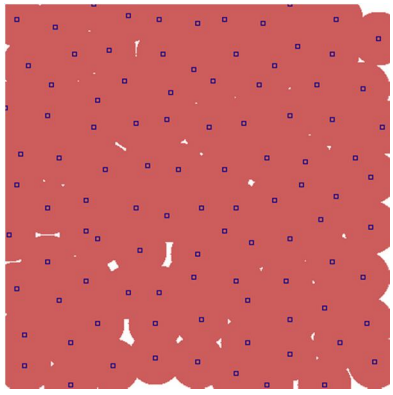
As shown in Table 4.28, the results are better than the results in Table 4.27, let's set the α value to 1.5 without changing other parameters, Table 4.29 shows the results.

Table 4.29. The best result for 30 runs when colony size equals 80 and α equals 1.5.

Colony size	80	Best solution 
α	1.5	
Best solution	0.9859	
Worst solution	0.9787	
Mean value	0.981323333	
Standard deviation	0.001403383	

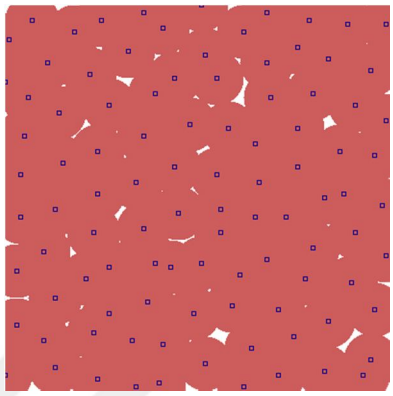
As shown in Table 4.29, the results are better than the results in Table 4.28, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Let's set the α value to 2 without changing other parameters, Table 4.30 shows the results.

Table 4.30. The best result for 30 runs when colony size equals 80 and α equals 2.

Colony size	80	Best solution 
α	2	
Best solution	0.9865	
Worst solution	0.9787	
Mean value	0.98142	
Standard deviation	0.001297925	

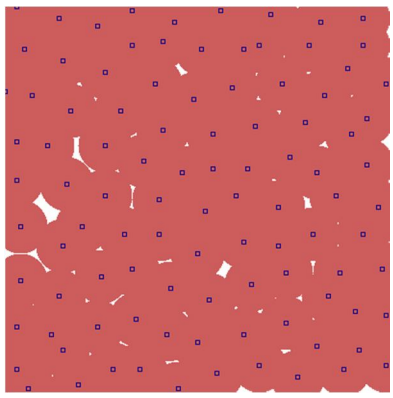
As you can see, the results in Table 4.30 are better than the results in Table 4.29, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Now let's set the colony size to 100, α value of 0.25 without changing other parameters, Table 4.31 shows the results.

Table 4.31. The best result for 30 runs when colony size equals 100 and α equals 0.25.

Colony size	100	Best solution
α	0.25	
Best solution	0.9841	
Worst solution	0.9773	
Mean value	0.980906667	
Standard deviation	0.002044494	

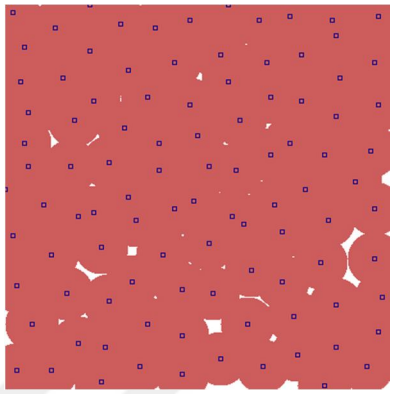
When the results in Table 4.31 are compared with the results in Table 4.26, it shows that when the colony size is increased, the result will be better, with the same α value. Let's set the α value to 0.5, Table 4.32 shows the results.

Table 4.32. The best result for 30 runs when colony size equals 100 and α equals 0.5.

Colony size	100	Best solution
α	0.5	
Best solution	0.9843	
Worst solution	0.9775	
Mean value	0.98105	
Standard deviation	0.001827092	

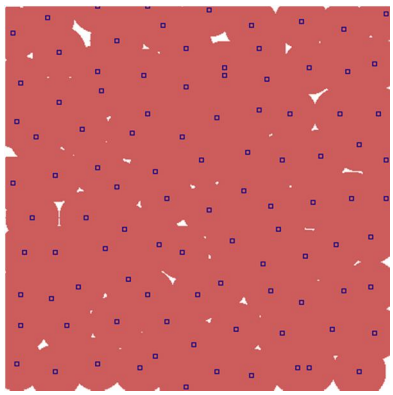
As shown in Table 4.32, the results are better than the results in Table 4.31, also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Let's set the α value to 1 without changing other parameters, Table 4.33 shows the results.

Table 4.33. The best result for 30 runs when colony size equals 100 and α equals 1.

Colony size	100	Best solution
α	1	
Best solution	0.9845	
Worst solution	0.9783	
Mean value	0.981103333	
Standard deviation	0.001789317	

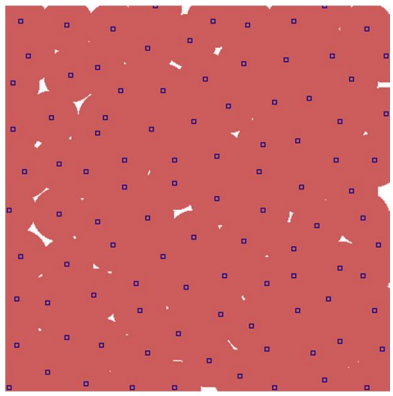
As shown in Table 4.33, the results are better than the results in Table 4.32, let's set the α value to 1.5 without changing other parameters, Table 4.34 shows the results.

Table 4.34. The best result for 30 runs when colony size equals 100 and α equals 1.5.

Colony size	100	Best solution
α	1.5	
Best solution	0.9859	
Worst solution	0.9787	
Mean value	0.981323333	
Standard deviation	0.001403383	

As shown in Table 4.34, the results are better than the results in Table 4.33; also the Standard deviation is decreased that means there is not a big difference between the obtained results in each round. Let's set the α value to 2 without changing other parameters, Table 4.35 shows the results.

Table 4.35. The best result for 30 runs when colony size equals 100 and α equals 2.

Colony size	100	Best solution 
α	2	
Best solution	0.9872	
Worst solution	0.9787	
Mean value	0.98142	
Standard deviation	0.001297925	

As shown in Table 4.35, the best result when the colony size equals 100 and the α value equals 2.

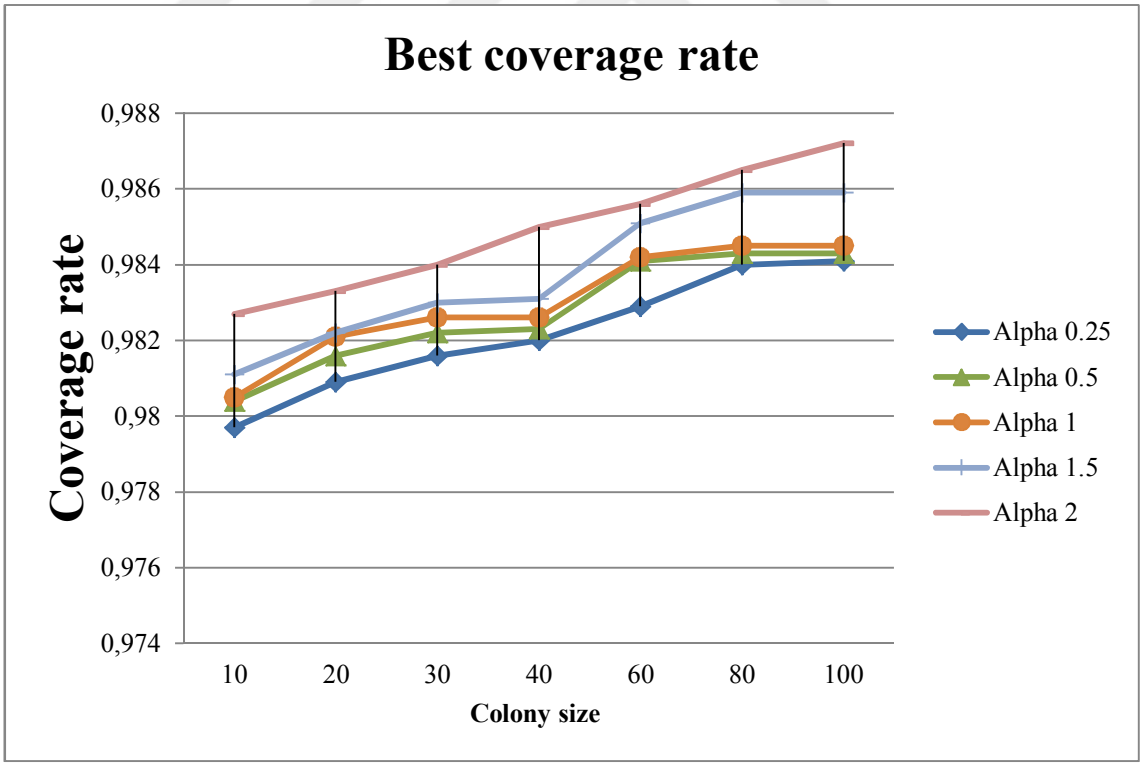


Figure 4.1. The best coverage rate for different values of colony size.

Figure 4.1 shows the plot of all results that obtained. To see how the solution is improved by increasing the colony size. The x-axis represents the colony size values

(10, 20, 30, 40, 60, 80, and 100) and the y-axis represent the coverage rate. Each curve line represents one of the α values (0.25, 0.5, 1, 1.5, and 2). When colony size is 10 and α is 0.25 the coverage is 0.9797, this value can be increased by increasing the colony size and the α value. When colony size equals 100 and α equals 2, the coverage rate is equal 0.9872, this rate is quite near to 1, as mentioned before the maximum possible coverage rate that can be obtained is 1.

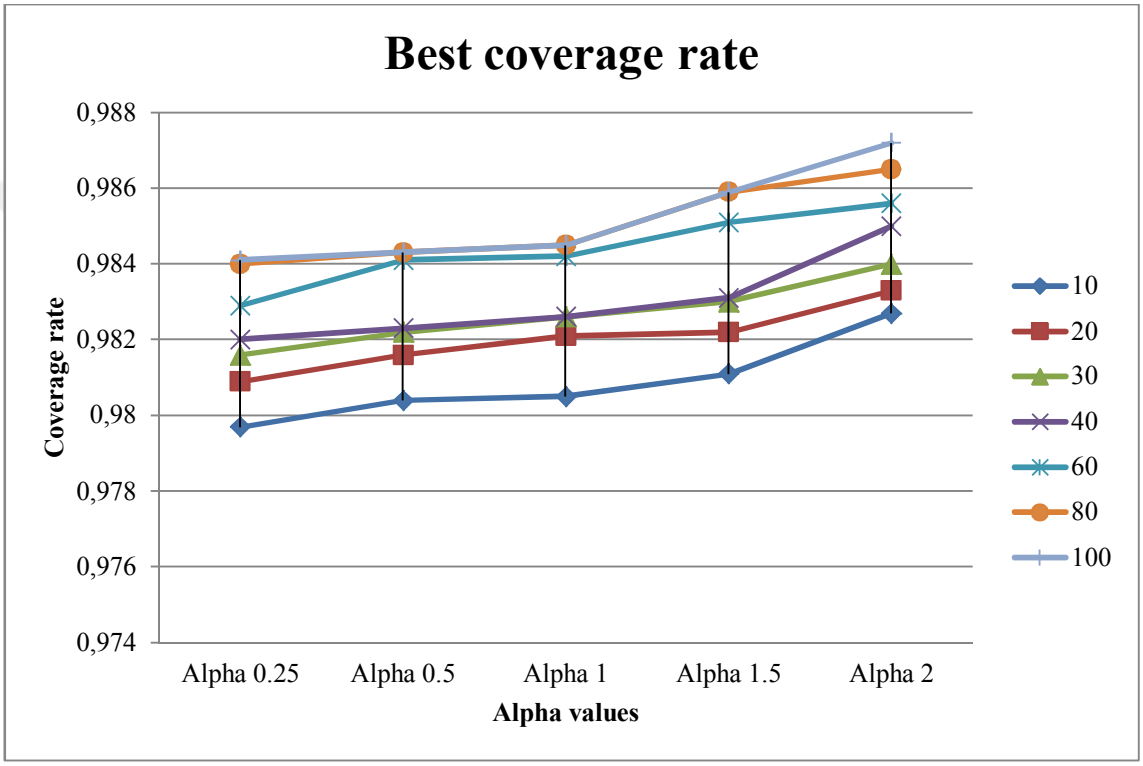


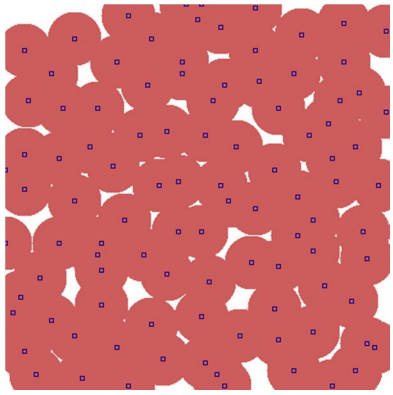
Figure 4.2. The best coverage rate for different values of α.

Figure 4.2 shows the plot of all results that obtained. To see how the solution is improved by increasing the α value. The x-axis represents the α values (0.25, 0.5, 1, 1.5, and 2) and the y-axis represents the coverage rate, Each curve line represents one of the coverage rate when colony size values (10, 20, 30, 40, 60, 80, and 100) respectively. When α is 0.25 the coverage rate is 0.9797, this value can be increased by increasing the α value. When colony size equals 100 and α equals 2, the coverage rate is equal 0.9872, this rate is quite near to 1, as mentioned before the maximum possible coverage rate that can be obtained is 1.

4.2. Dynamic Deployment of WSN Evaluation of Solutions

This part focuses on how the solution is improved iteration by iteration for the same number of population. When the number of cycles is increased, the number of population is increasing too. Also, the increment in the number of food source increases the number of populations. When the number of food source equals 10 and the number of cycles equals 1000, the population number is 20.000. The population number value is kept without change in this part. Therefore, if the number of food source is 20, the number of cycles should equal 500 to get the same population number (20.000). Also, when the number of food source is 40 the number of cycles should equal 250. And when food source is 80 the number of cycles should equal 125. When food source is 100 the number of cycles should equal 100. To understand how the algorithm behaves when the food source is 100 and the number of cycles is 100, the algorithm runed for 30 run tımes, the best result is shown in Table 4.36.

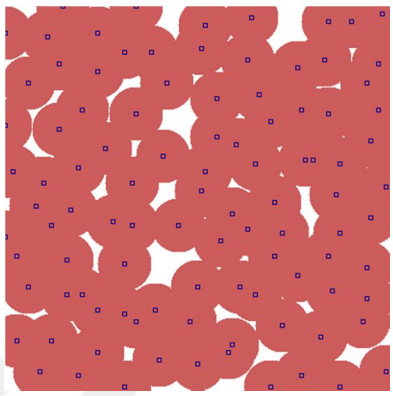
Table 4.36. The best result for 30 runs when food source equals 100 and number of cycles equal 100.

Food source	100	Best solution 
Number of cycles	30	
Number of Runs	30	
Best solution	0.9212	
Worst solution	0.8838	
Mean value	0.898366	
Standard deviation	0.007259965	

As shown in Table 4.36, while the food source is 100 the results are not good, because of the number of cycles are 100, and it is not enough to get a good coverage rate. Let's decrease the food source to 80 and set the iteration number to 125 and run the algorithm for 30 run tımes, the best result is shown in Table 4.37.

Table 4.37. The best result for 30 runs when food source equals 80 and number of cycles equal 125.

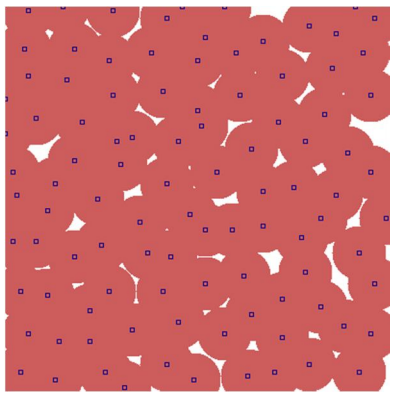
Food source	80	Best solution
Number of cycles	125	
Number of Runs	30	
Best solution	0.9216	
Worst solution	0.8831	
Mean value	0.896482	
Standard deviation	0.007194126	



As shown in Table 4.37, while the food source is 80 the best result is 0.9216, it is little better than the result in Table 4.36, because of the number of cycles is increased to 125, Let's decrease the food source to 40 and set the iteration number to 250 and run the algorithm for 30 run tımes, the best result is shown in Table 4.38.

Table 4.38. The best result for 30 runs when the number of food source = 40 and the number of cycles equal 250.

Food source	40	Best solution
Number of cycles	250	
Number of Runs	30	
Best solution	0.9551	
Worst solution	0.9264	
Mean value	0.93612	
Standard deviation	0.00461221	



As shown in Table 4.38, when the food source is 40 the best result is 0.9551, it is better than the result in Table 4.37, because of the number of cycles is doubled to 250, Let's decrease the food source to 20 and set the iteration number to 500 and run the algorithm for 30 run tımes, the best result is shown in Table 4.39.

Table 4.39. The best result for 30 runs when the food source equals 20 and number of cycles equal 500.

Food source	20	Best solution
Number of cycles	500	
Number of Runs	30	
Best solution	0.9696	
Worst solution	0.953	
Mean value	0.959394667	
Standard deviation	0.003215019	

As shown in Table 4.39, when the food source is 20 the best result is 0.9696, it is better than the result in Table 4.38, because of the number of cycles is doubled to 500, Let's decrease the colony size to 10 and set the iteration number to 1000 and run the algorithm for 30 run tımes, the best result is shown in Table 4.40.

Table 4.40. The best result for 30 runs when the food source equals 10 and number of cycles equals 1000.

Food source	10	Best solution
Number of cycles	1000	
Number of Runs	30	
Best solution	0.9834	
Worst solution	0.9697	
Mean value	0.97558	
Standard deviation	0.002715059	

As shown in Table 4.40, while the food siurce is 10 the best result is 0.9834, it is better than the result in Table 4.39, because of the number of cycles is doubled to 1000, Figure 4.3 shows the curve for each case.

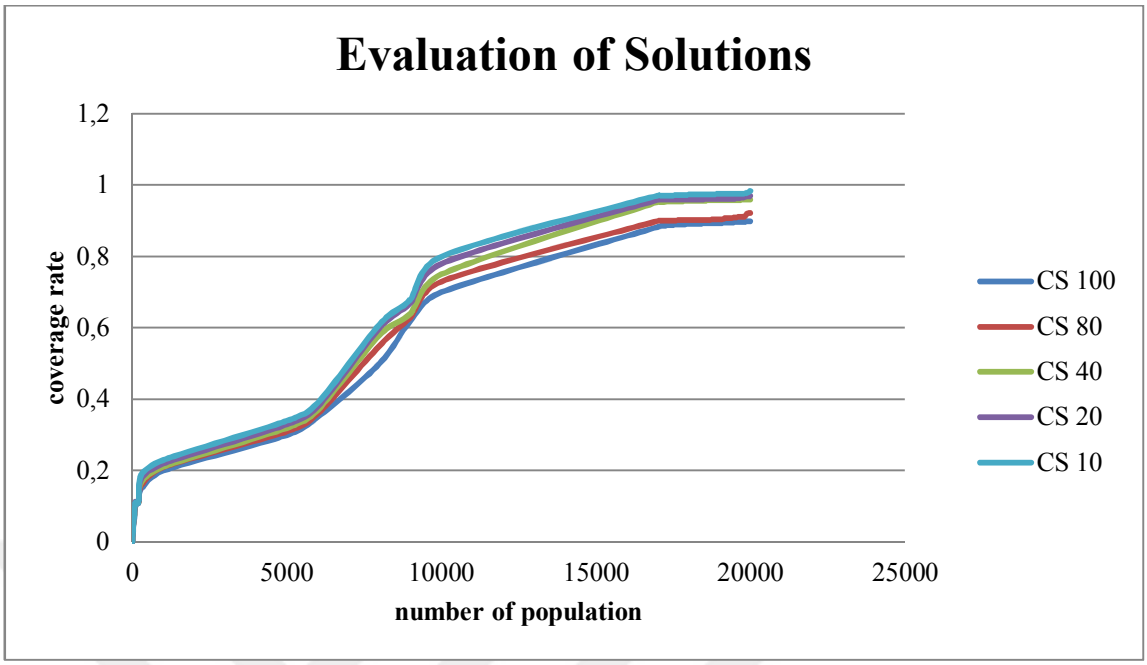


Figure 4.3. The population number starts from 0 to 20.000.

The x-axis represents the population number starts from 0 to 20.000 and the y-axis represents the coverage rate. The best solution obtained when the number of food source equals 10 and the number of cycles equals 1000. When food sources equals 100 and number of cycles equals 100, the worst solution is obtained. The solution is directly affected by the number of cycles.

4.3. Dynamic Deployment of WSN on Area With Constraints

In this part, the ABC algorithm is implemented to make the dynamic deployment of mobile sensor nodes on area with constraints or obstacles. The constraints may be a mountain, lake or any obstacle that prevents the sensors to be deployed on it. Therefore, the ABC algorithm must deploy the sensors out of constraint region. That deployment can be done by two different methods, the first one that depends on the coordinates of the constraint region. In each cycle, every new produced position of the mobile sensor that it's coordinates inside the forbidden area, the algorithm shift that sensor to the nearest age, to the left or right outside the forbidden area. That means, from the first cycle to the end of the cycles there is no any sensor in the obstacle area. The second method depends on the fitness value of the produced solution, when the coordinates of the produced solution are in the forbidden area the counter is increased by 1. The ABC

algorithm counts the number of sensors in the forbidden area and divides the coverage rate by that number +1, to get the fitness value. When the counter is not 0, that means it is not a good solution. Cycle by cycle the ABC algorithm deploys all mobile sensor nodes outside the obstacle area to get best possible coverage rate.

4.3.1. Direct constraint handling method: mapping

The Constraints and obstacles are not desirable in the sensing area, because it impedes the deployment of wireless sensors in the sensing area. Therefore, the sensors should be deployed away from these obstacles. Suppose that we have a square area with a length of 100 meters and there is a square obstacle in the center of that area, its length equals 1/3 of the whole zone length. And we have 100 wireless sensors, each of which has a sensing range of 7 meters. What is required is to deploy these sensors away from the obstacle area using the ABC algorithm. The deployment depends on the coordinates of the new solution, if the coordinates of the produced solution within the forbidden area, the algorithm shift that sensor to the nearest age, to the left or right outside the forbidden area. To create new solutions away from the forbidden area to increase the coverage rate within the area of interest. A simulation for that scenario was done by implementing the ABC algorithm. The colony size equals 20 and the number of cycles equals 1000, the algorithm was run for 30 run tımes. The results obtained depending on the assignment of (4.2) and new assignments of (4.4).

$$\begin{Bmatrix} if & x_i > w_{min} \\ if & x_i < w_{max} \\ if & y_i > h_{min} \\ if & y_i < h_{max} \end{Bmatrix} then \begin{Bmatrix} if\ \ x_i > y_i\ then\ \ x_i = \left(W * \frac{\frac{f+1}{2}}{f}\right) + 1 \\ if\ \ x_i \leq y_i\ then\ \ x_i = \left(W * \frac{\frac{f-1}{2}}{f}\right) - 1 \end{Bmatrix} \tag{4.4}$$

Where w_{min}, h_{min} is the lower boundary of the constraint, w_{max}, h_{max} is the upper boundary of the constraint, W is the whole width of the area, f is constant in this case equals 3.

The fitness is calculated depending on the equation (4.5) as shown bellow:

$$fitness = \frac{A_{Covered}}{A_{Total} - A_{constraint}} \tag{4.5}$$

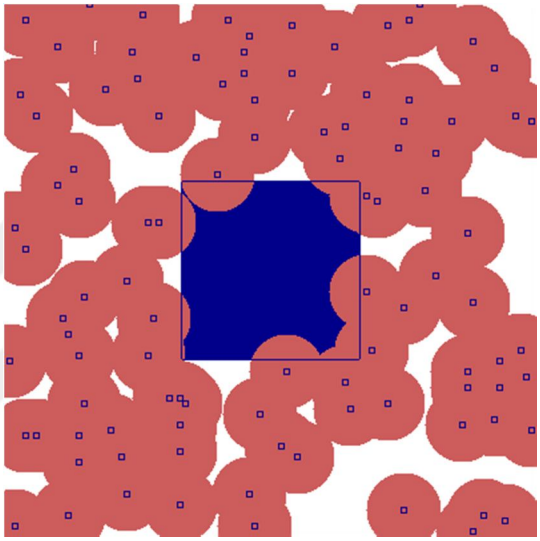
Where $A_{Covered}$ is the covered area, A_{Total} is the total area, $A_{constraint}$ is the forbidden area.

In cycle 0 all results were collected and analyzed to get the best solution, worst solution, mean value and the standard deviation. Table 4.41 shows the best results at cycle 0.

Table 4.41. The best deployment results depend on coordinates at cycle 0.

Results depend on coordinates		
Maximum Cycles	1000	
Colony size	20	
Cycle number	0	
Results	**Coverage**	**Fitness**
Best solution	0.835259791	0.835259791
Worst solution	0.785994838	0.785994838
Mean value	0.810545019	0.810545019
Standard deviation	0.011320934	0.011320934

Best solution



As shown in Table 4.41 the best coverage rate equals 0.835259791, it equals the fitness value, because the calculations depend on the coordinates of the new solution. Also, in the right of the same table there is the image of the best solution, the blue square in the center of the image represents the obstacle area and the blue small squares, each one represents a mobile sensor node the red circles around the sensors represents the sensing range for every sensor. There is no any sensor in the obstacle area. Because the algorithm neglects any solution with the coordinates inside the obstacle region.

Table 4.42. The best deployment results depend on coordinates at cycle 50.

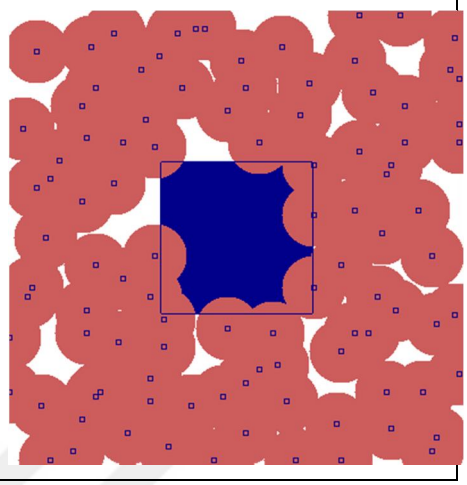
Results depend on coordinates			Best solution
Maximum Cycles	1000		
Colony size	20		
Cycle number	50		
Results	**Coverage**	**Fitness**	
Best solution	0.901357872	0.901357872	
Worst solution	0.87072158	0.87072158	
Mean value	0.890281674	0.890281674	
Standard deviation	0.007668663	0.007668663	

Table 4.42 shows the best results at cycle number 50, the best coverage rate for 30 run tımes equals 0.901357872, this result is better than the previous one. Also the standard deviation is less than the previous result, that means the results are stable and near the mean value. Also, there is no any sensor in the obstacle area, the white spots represent the uncovered area, the sensors deployed better than the previous case outside the obstacle area. The next case when the cycle number is 100 and it is shown in Table 4.43.

Table 4.43. The best deployment results depend on coordinates at cycle 100.

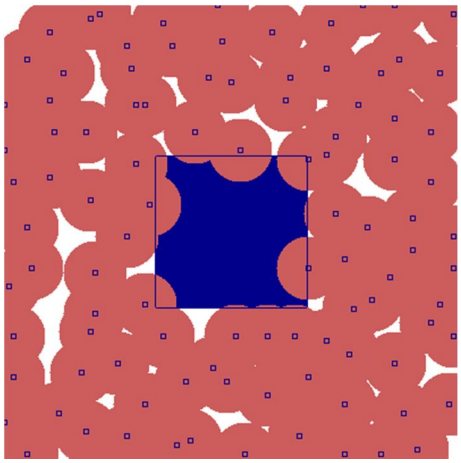
Results depend on coordinates			Best solution
Maximum Cycles	1000		
Colony size	20		
Cycle number	100		
Results	**Coverage**	**Fitness**	
Best solution	0.937380765	0.937380765	
Worst solution	0.909662215	0.909662215	
Mean value	0.921662365	0.921662365	
Standard deviation	0.006729828	0.006729828	

Table 4.43 shows the best results at cycle number 100, the best coverage rate for 30 run tımes equals 0.937380765, this result is better than the previous one (0.901357872).

Also the standard deviation 0.006729828 is less than the previous result (0.007668663), that means the results are stable and near the mean value. Also, there is no any sensor in the obstacle area; the white spots represent the uncovered area, the sensors deployed better than the previous case outside the obstacle area. The next case when the cycle number is 500 and it is shown in Table 4.44.

Table 4.44. The best deployment results depend on coordinates at cycle 500.

Results depend on coordinates		
Maximum Cycles	1000	
Colony size	20	
Cycle number	500	
Results	**Coverage**	**Fitness**
Best solution	0.981146897	0.981146897
Worst solution	0.972730333	0.972730333
Mean value	0.977458572	0.977458572
Standard deviation	0.002493989	0.002493989

Best solution

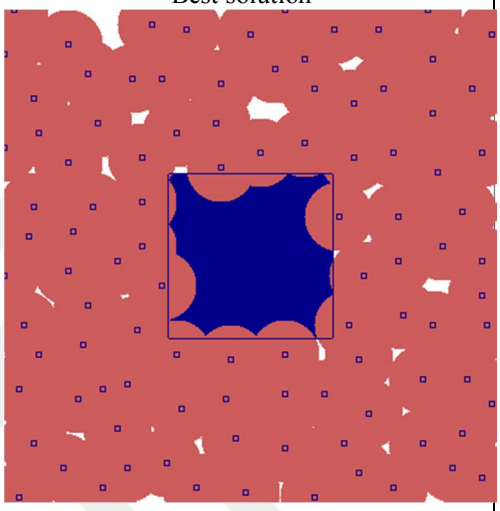


Table 4.44 shows the best results at cycle number 500, the best coverage rate for 30 run tımes equals 0.981146897, this result is better than the previous one (0.937380765). Also the standard deviation 0.002493989 is less than the previous result (0.006729828), that means the results are stable and near the mean value. Also, there is no any sensor in the obstacle area; the white spots represent the uncovered area, the sensors deployed better than the previous case outside the obstacle area. The next case when the cycle number is 1000 and it is shown in Table 4.45.

Table 4.45. The best deployment results depend on coordinates at cycle 1000.

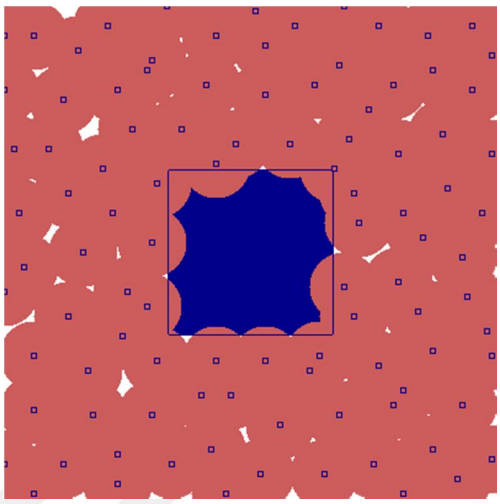
Results depend on coordinates			Best solution
Maximum Cycles	1000		
Colony size	20		
Cycle number	1000		
Results	**Coverage**	**Fitness**	
Best solution	0.990910111	0.990910111	
Worst solution	0.98148356	0.98148356	
Mean value	0.986937493	0.986937493	
Standard deviation	0.00232925	0.00232925	

Table 4.44 shows the best results at cycle number 500, the best coverage rate for 30 run tımes equals, 0.990910111 this result is better than the previous one (0.981146897) and it is quite near to 1. Also the standard deviation 0.00232925 is less than the previous result (0.002493989), that means the results are stable and near the mean value. Also, there is no any sensor in the obstacle area, the white spots represent the uncovered area and it is narrow spots that means, the sensors deployed better than the previous case outside the obstacle area. All the best solutions for the coverage for a are analyzed and plotted in Figure 4.4.

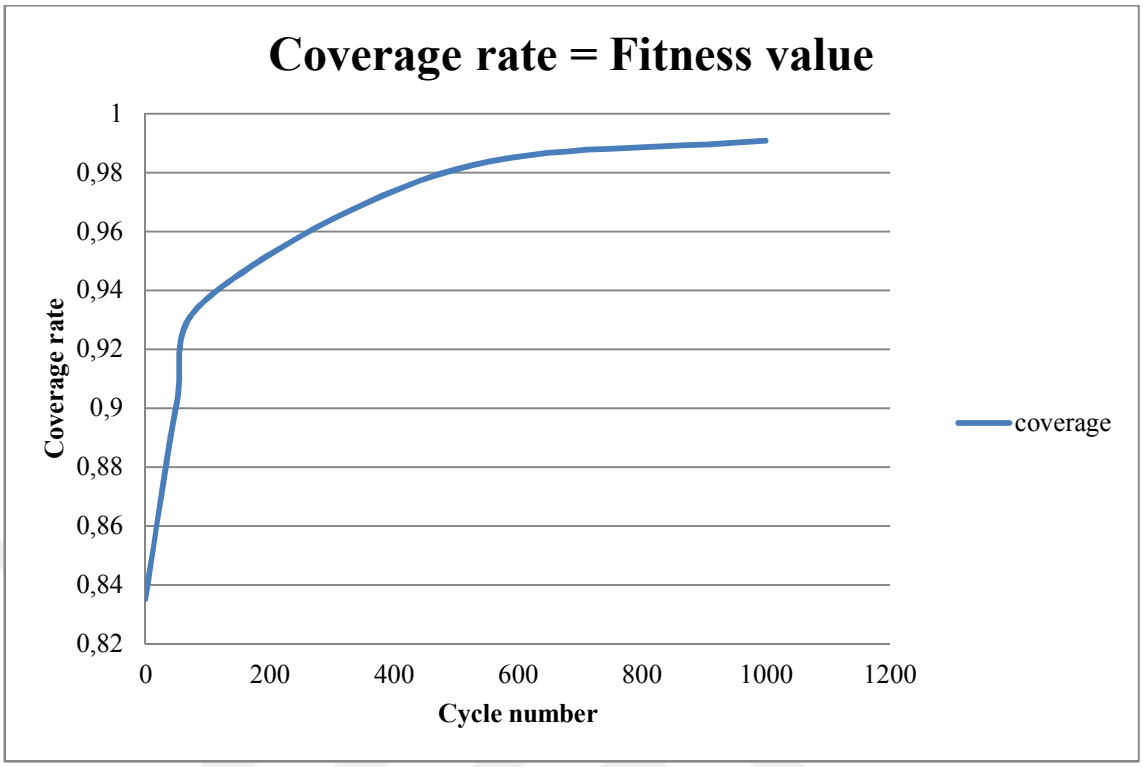


Figure 4.1. The best coverage rates for mobile sensors with obstacles depending on sensor coordinates.

Figure 4.4 shows the best coverage rates for 100 mobile sensor nodes deployed in the area with constraint (obstacle) by implementing the ABC algorithm. The x-axis represents the cycle number from 0 to 1000 and the y axis represents the coverage rate that its maximum obtained value near 1.

4.3.2. Indirect constraint method: penalty method

As mentioned before, the Constraints and obstacles are not wanted in the sensing area, because it slows down the deployment of wireless sensors in the sensing area. Therefore, the sensors should be deployed outside these obstacles. We have the same scenario of section 4.3.1, a square area with a length of 100 meters and there is a square obstacle in the center of that area, it's length equals 1/3 of the whole region length. And we have 100 mobile sensor nodes, each of them has a sensing range of 7 meters. What is required is to deploy these sensors away from the obstacle area using the ABC algorithm. The deployment depends on the fitness value of the new positions, the algorithm counts the number of the sensors inside the obstacle area, if there is many sensors in the obstacle area, the fitness value of that solution is low, that means it is not

a good solution and in the next cycle the ABC algorithm tries to create a new solution away from the obstacle area to increase the coverage rate within the area of interest. Depending on the assignment of 4.6 and 4.7.

$$\left.\begin{matrix} if & x_i > & w_{min} \\ if & x_i < & w_{max} \\ if & y_i > & h_{min} \\ if & y_i < & h_{max} \end{matrix}\right\} then\ \{n = n + 1\} \tag{4.6}$$

Where w_{min}, h_{min} is the lower bound of the constraint, w_{max}, h_{max} is the upper bound of the constraint.

$$fitness = \frac{coverage\ rate}{1 + n} \tag{4.7}$$

Where $fitness$, is the fitness function, $coverage\ rate$, is the ratio between the covered area to the total feasible area, n, is the number of sensors in the obstacle area.

A simulation for that scenario was done by implementing the ABC algorithm. The colony size equals 20 and the number of maximum cycles equals 1000, the algorithm was run for 30 run times, in cycle 0 all results were collected and analyzed to get the best solution, worst solution, mean value and the standard deviation. Table 4.46 shows the best results at cycle 0.

Table 4.46. The best deployment results depend on the fitness value at cycle 0.

Results depend on fitness function			Best solution
Maximum Cycles	1000		
Colony size	20		
Cycle number	0		
Results	**Coverage**	**Fitness**	
Best solution	0.801593536	0.100199192	
Worst solution	0.128754723	0.069658543	
Mean value	0.734699934	0.093815969	
Standard deviation	0.116486367	0.017437971	

Table 4.46 shows the best results for 30 run tımes of the ABC algorithm at cycle number 0. As shown in the image, there are 7 sensors inside the obstacle area, that means it is not a good deployment because of the best coverage rate equals 0.801593536, while fitness value equals 0.100199192. According to function 4.2 the fitness value equals the coverage rate divided by 1 + the number of sensors in the obstacle area. Therefore, the coverage rate 0.801593536 divided by 8, so that the fitness value is lower than the coverage rate. In the next cycles the ABC algorithm has to produce a new better solution with lower or no sensors in the obstacle area. To see the produced solution step by step, we get the results in a small period of cycles to understand the performance of the ABC algorithm. Table 4.47 shows the best deployment results depend on the fitness value at cycle 9.

Table 4.47. The best deployment results depend on the fitness value at cycle 9.

Results depend on fitness function			Best solution
Maximum Cycles	1000		
Colony size	20		
Cycle number	9		
Results	**Coverage**	**Fitness**	
Best solution	0.8038379530	0.1148339932	
Worst solution	0.7436875771	0.1239479295	
Mean value	0.754699934	0.107814276	
Standard deviation	0.106486367	0.015212338	

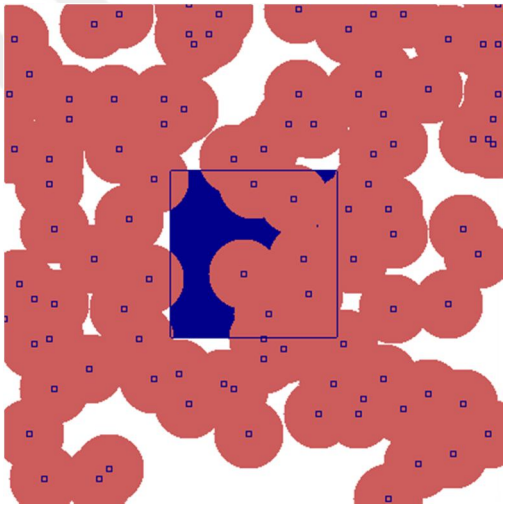


Table 4.47 shows the best results for 30 run tımes of the ABC algorithm at cycle number 9. As shown in the image, the number of sensors in the obstacle area is decreased to 6, that means after 9 cycles the algorithm improved the solution. Also, the coverage rate has increased to 0.8038379530, while fitness value equals 0.1148339932. According to function 4.2 the fitness value equals the coverage rate divided by 1 + the number of sensors in the obstacle area. Therefore, the coverage rate 0.8038379530 divided by 7, so that the fitness value is lower than the coverage rate. To see the produced solution step by step, we get the results in a small period of cycles to

understand the performance of the ABC algorithm. Table 4.48 shows the best deployment results depend on the fitness value at cycle 24.

Table 4.48. The best deployment results depend on the fitness value at cycle 24.

Results depend on fitness function		
Maximum Cycles	1000	
Colony size	20	
Cycle number	24	
Results	**Coverage**	**Fitness**
Best solution	0.8850858489	0.2950286163
Worst solution	0.7856581752	0.0982072719
Mean value	0.834699934	0.278233311
Standard deviation	0.906486367	0.302162122

Best solution

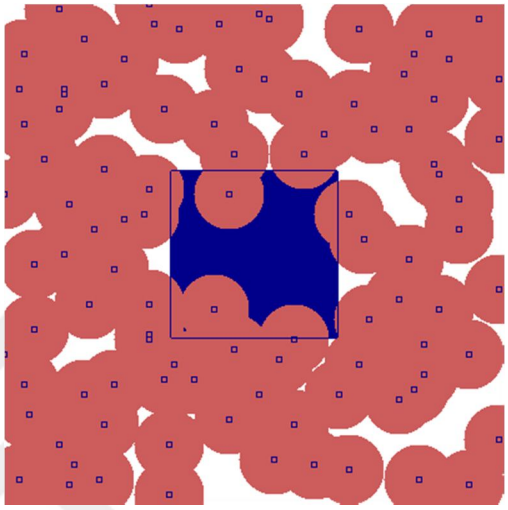


Table 4.48 shows the best results for 30 run tımes of the ABC algorithm at cycle number 24. As shown in the image, the number of sensors in the obstacle area is decreased to 2, that means after 24 cycles the algorithm improved the solution. Also, the coverage rate has increased to 0.8850858489, while fitness value equals 0.2950286163. According to function 4.2 the fitness value equals the coverage rate divided by 1 + the number of sensors in the obstacle area. Therefore, the coverage rate 0.8038379530 divided by 3, so that the fitness value is lower than the coverage rate. To see the produced solution step by step, we get the results in a small period of cycles to understand the performance of the ABC algorithm. Table 4.49 shows the best deployment results depend on the fitness value at cycle 49.

Table 4.49. The best deployment results depend on the fitness value at cycle 49.

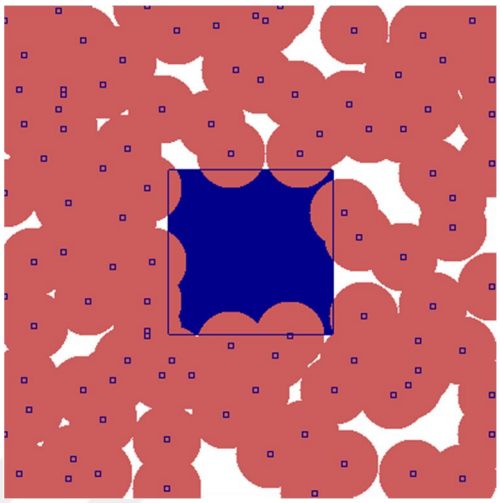
Results depend on fitness function			Best solution
Maximum Cycles	1000		
Colony size	20		
Cycle number	49		
Results	**Coverage**	**Fitness**	
Best solution	0.905285602	0.905285602	
Worst solution	0.815396701	0.137769798	
Mean value	0.860752628	0.260650507	
Standard deviation	0.021723841	0.13904842	

Table 4.49 shows the best results for 30 run tımes of the ABC algorithm at cycle number 49. As shown in the image, the number of sensors in the obstacle area is decreased to 0, that means in 49 cycles the algorithm improved the solution. Also, the coverage rate has increased to 0.905285602, and the fitness value equals 0.905285602 the same coverage rate. According to function 4.2 the fitness value equals the coverage rate divided by 1 + the number of sensors in the obstacle area. Therefore, the coverage rate 0.905285602 divided by 1, so that the fitness value equals the coverage rate. The ABC algorithm out all the sensors from the obstacle area and optimize he solution after 50 cycles. Table 4.50 shows the best deployment results depend on the fitness value at cycle 99.

Table 4.50. The best deployment results depend on the fitness value at cycle 99.

Results depend on fitness function			Best solution
Maximum Cycles	1000		
Colony size	20		
Cycle number	99		
Results	**Coverage**	**Fitness**	
Best solution	0.939064078	0.939064078	
Worst solution	0.45516777	0.280439906	
Mean value	0.89353982	0.648738762	
Standard deviation	0.085409794	0.260898352	

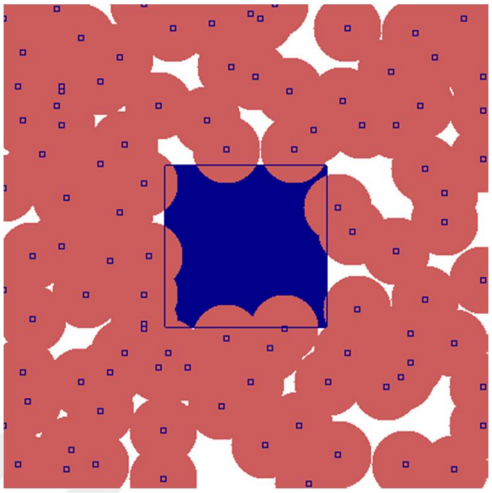


Table 4.50 shows the best results for 30 run tımes of the ABC algorithm at cycle number 49. As shown in the image, the number of sensors in the obstacle area is decreased to 0, that means in 49 cycles the algorithm improved the solution. Also, the coverage rate has increased to 0.939064078, and the fitness value equals 0.939064078 the same coverage rate. According to function 4.2 the fitness value equals the coverage rate divided by 1 + the number of sensors in the obstacle area. Therefore, the coverage rate 0.939064078 divided by 1, so that the fitness value equals the coverage rate. The ABC algorithm out all the sensors from the obstacle area and optimize he solution after 100 cycles. Table 4.51 shows the best deployment results depend on the fitness value at cycle 499.

Table 4.51. The best deployment results depend on the fitness value at cycle 99.

Results depend on fitness function			Best solution
Maximum Cycles	1000		
Colony size	20		
Cycle number	499		
Results	**Coverage**	**Fitness**	
Best solution	0.98810459	0.98810459	
Worst solution	0.972505892	0.972505892	
Mean value	0.977249093	0.977249093	
Standard deviation	0.003452336	0.003452336	

Table 4.51 shows the best results for 30 run tımes of the ABC algorithm at cycle number 499. As shown in the image, the number of sensors in the obstacle area is decreased to 0, that means in 499 cycles the algorithm improved the solution. Also, the coverage rate has increased to 0.98810459, and the fitness value equals 0.98810459 the same coverage rate. According to function 4.2 the fitness value equals the coverage rate divided by 1 + the number of sensors in the obstacle area. Therefore, the coverage rate 0.98810459 divided by 1, so that the fitness value equals the coverage rate. The ABC algorithm out all the sensors from the obstacle area and optimize he solution after 500 cycles. Table 4.52 shows the best deployment results depend on the fitness value at cycle 999.

Table 4.52. The best deployment results depend on the fitness value at cycle 99.

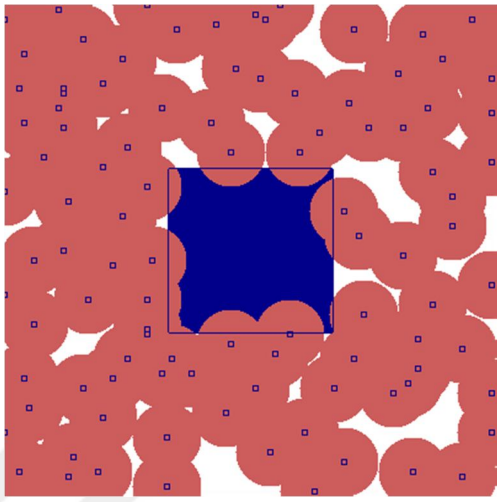
Results depend on fitness function			Best solution
Maximum Cycles	1000		
Colony size	20		
Cycle number	999		
Results	**Coverage**	**Fitness**	
Best solution	0.992256761	0.992256761	
Worst solution	0.980473572	0.980473572	
Mean value	0.986589608	0.986589608	
Standard deviation	0.002582467	0.002582467	

Table 4.52 shows the best results for 30 run times of the ABC algorithm at cycle number 999. As shown in the image, the number of sensors in the obstacle area is decreased to 0, that means in 999 cycles the algorithm improved the solution. Also, the coverage rate has increased to 0.992256761, that's quite near to 1 and the fitness value equals 0.992256761 the same coverage rate. According to function 4.2 the fitness value equals the coverage rate divided by 1 + the number of sensors in the obstacle area. Therefore, the coverage rate 0.992256761 divided by 1, so that the fitness value equals the coverage rate. The ABC algorithm out all the sensors from the obstacle area and optimize he solution after 1000 cycles. Figure 4.5 shows the best coverage rates that depend on the fitness value from cycle 0 to 999.

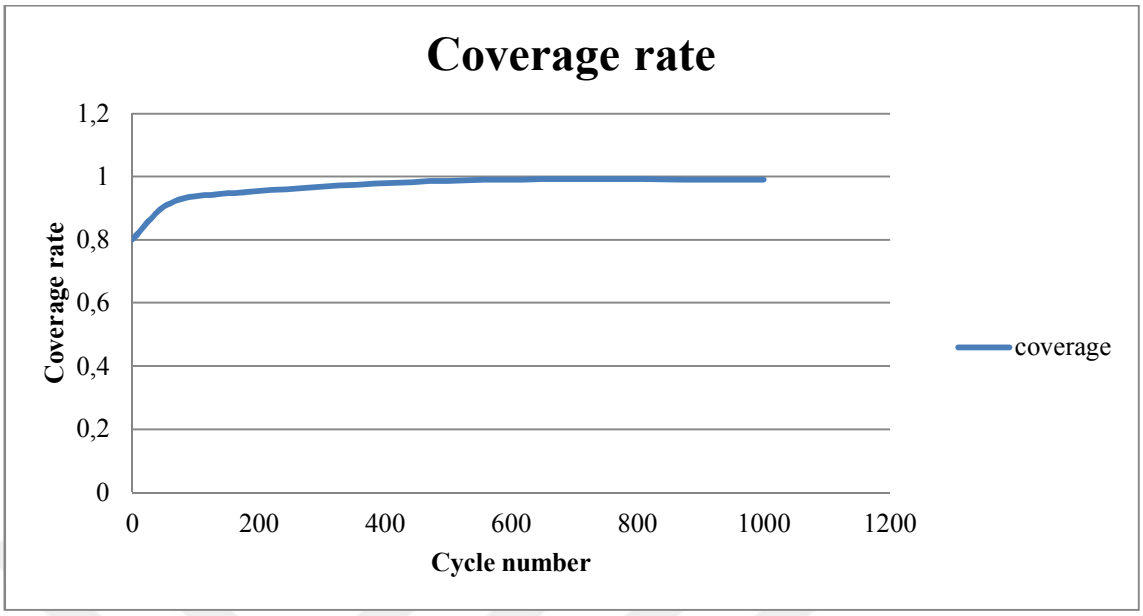


Figure 4.5. The best coverage rates that depend on the fitness value for 1000 cycles.

Figure 4.6 shows the coverage rate for 100 mobile sensor nodes deployed in the area with constraint (obstacle) by implementing the ABC algorithm with specific control parameters. The x-axis represents the cycle number from 0 to 1000 and the y axis represents the coverage rate. In cycle 0 the coverage initial value equals 0.8, cycle by cycle the ABC algorithm improved the solutions to be near 1 at cycle number 1000. Figure 4.6 shows the change in fitness value from cycle 0 to 999.

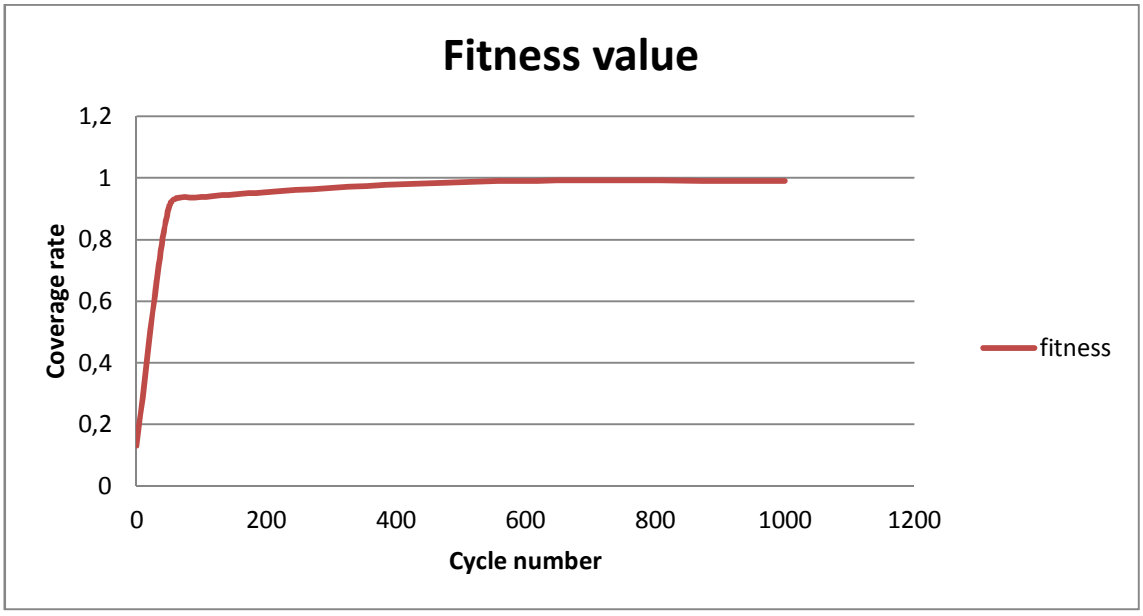


Figure 4.6. The change in the fitness value for 1000 cycles.

Figure 4.7 shows the change in fitness value for 1000 cycles. The x-axis represents the cycle number from 0 to 1000 and the y axis represents the fitness value that its maximum obtained value near 1. At cycle 0, the fitness value started from 0.1 while in Figure 4.6 the coverage rate started from 0.8. That caused by the 7 sensors that deployed in the obstacle area, as explained before in function 4.2. At cycle 50 fitness value started to equal the coverage rate because all sensors deployed out of the obstacle area. Figure 4.7 shows both the fitness value and the coverage rate.

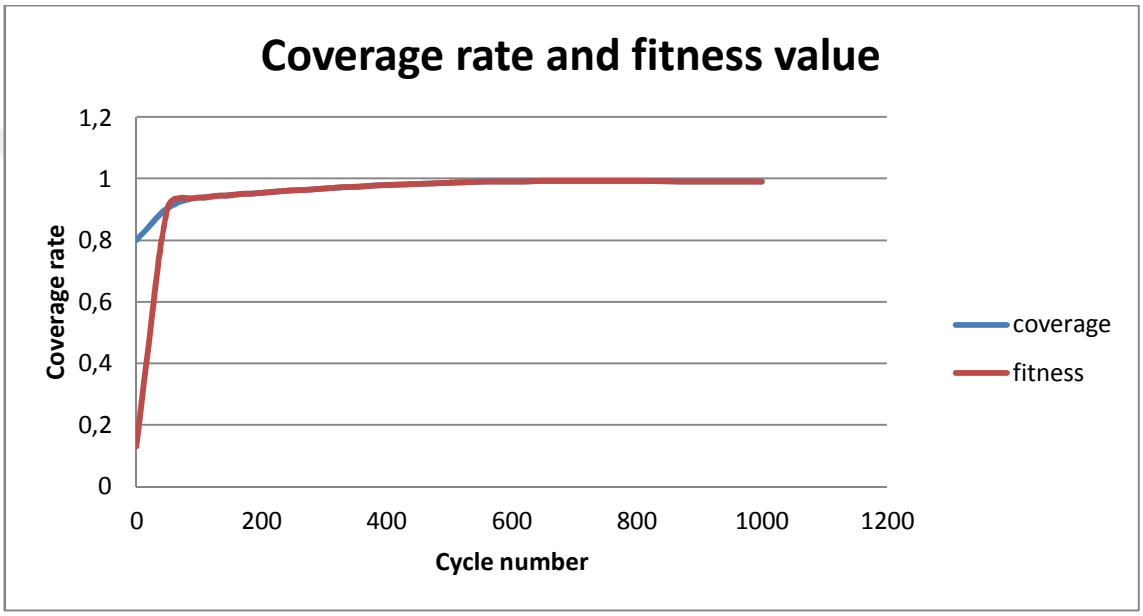


Figure 4.7. Shows both the fitness value and the coverage rate.

Figure 4.7 shows both the fitness value and the coverage rate of 1000 cycles. The x-axis represents the cycle number from 0 to 1000 and the y axis represents the fitness value and the coverage rate that its maximum obtained value near 1. At cycle 0, the fitness value started from 0.1 while the coverage rate started from 0.8. That caused by the 7 sensors that deployed in the obstacle area, as explained before in function 4.2. At cycle 50 fitness value started to equal the coverage rate because all sensors deployed out of the obstacle area.

4.4. The difference between Dynamic deployment of WSN on area with constraints depending on fitness value and the coordinates

From the previously obtained results, there is no big difference between the two methods. Both of them produced a high coverage rate near to 1. Figure 4.8 shows the best obtained results for the both two methods for 30 run times and 1000 cycle number.

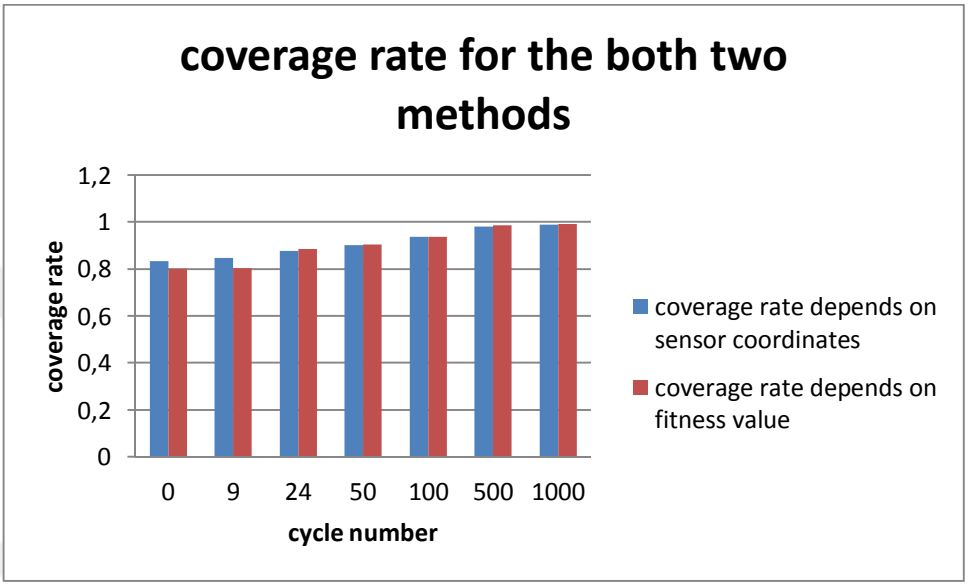


Figure 4.8. The best obtained coverage rate for the both two methods.

As shown in Figure 4.8, the x-axis represents the cycles from 0 to 1000 and the y-axis represents the coverage rate. The blue lines represent the coverage rate that depends on the first method (sensor coordinates), the other lines represent the coverage rate that depends on the second method (fitness value). At cycle 0 the first method result is better than the second one. Also, in cycle 9. But in cycle 24,50,100,500 and 1000 the obtained results in the second methods is better than the first method, that is mean the second method that depends on the fitness value is better than the first method that depends on sensor coordinates when the cycle number more than 10.

Table 4.53. The summary of all obtaned results.

Table number	Colony size	α	Best solution	Worst solution	Mean value	Standard deviation
4.1	10	0,25	0.9797	0.9688	0.974886667	0.002941467
4.2	10	05	0.9804	0.9705	0.974976667	0.002566452
4.3	10	1	0.9805	0.9711	0.97567	0.002327746
4.4	10	1,5	0.9811	0.9713	0.975856667	0.002275529
4.5	10	2	0.9827	0.9732	0.97631333	0.0020741
4.6	20	0.25	0.9816	0.9592	0.976823333	0.004202272
4.7	20	0.5	0.9822	0.9733	0.977576667	0.002504056
4.8	20	1	0.9826	0.9737	0.978263333	0.002131647
4.9	20	1.5	0.983	0.974	0.978326667	0.001915037
4.10	20	2	0.9833	0.9745	0.978643333	0.001863716
4.11	30	0.25	0.9809	0.9759	0.978876667	0.005513891
4.12	30	0.5	0.9816	0.9577	0.97904	0.002115083
4.13	30	1	0.9821	0.9751	0.977566667	0.001793375
4.14	30	1.5	0.9822	0.9753	0.978536667	0.00148039
4.15	30	2	0.984	0.9759	0.978876667	0.001465076
4.16	40	0.25	0.9816	0.9752	0.978786667	0.002124582
4.17	40	0.5	0.9823	0.9755	0.979066667	0.0020517
4.18	40	1	0.9826	0.9758	0.979083333	0.001742402
4.19	40	1.5	0.9831	0.9760	0.97917	0.001516514
4.20	40	2	0.985	0.9767	0.979646667	0.00147071
4.21	60	0.25	0.9829	0.9703	0.979613333	0.004138777
4.22	60	0.5	0.9841	0.9766	0.979696667	0.001899089
4.23	60	1	0.9842	0.9767	0.97974	0.001858921
4.24	60	1.5	0.9851	0.9768	0.979933333	0.001618712
4.25	60	2	0.9856	0.9776	0.980356667	0.001403858
4.26	80	0.25	0.984	0.9773	0.980906667	0.002044494
4.27	80	0.5	0.9843	0.9775	0.98105	0.001827092
4.28	80	1	0.9845	0.9783	0.981103333	0.001789317
4.29	80	1.5	0.9859	0.9787	0.981323333	0.001403383
Table number	**Colony size**	**α**	**Best solution**	**Worst solution**	**Mean value**	**Standard deviation**
4.30	80	2	0.9865	0.9787	0.98142	0.001297925
4.31	100	0.25	0.9841	0.9773	0.980906667	0.002044494
4.32	100	0.5	0.9843	0.9775	0.98105	0.001827092
4.33	100	1	0.9845	0.9783	0.981103333	0.001789317
4.34	100	1.5	0.9859	0.9787	0.981323333	0.001403383
4.35	100	2	0.9872	0.9787	0.98142	0.001297925
Table number	**Food source**	**Number of cycles**	**Best solution**	**Worst solution**	**Mean value**	**Standard deviation**
4.36	100	100	0.9212	0.8838	0.898366	0.007259965
4.37	80	125	0.9216	0.8831	0.896482	0.007194126
4.38	40	250	0.9551	0.9264	0.93612	0.00461221
4.39	20	500	0.9696	0.953	0.959394667	0.003215019
4.40	10	1000	0.9834	0.9697	0.97558	0.002715059

Table 4.54. (continuation) The summary of all obtaned results.

Table	Best coverage rate	Worst coverage rate	Best fitness value	Worst fitness value	Mean value coverage rate/ fitness value	SD for Coverage rate/ fitness value
4.41	0.835259791	0.785994838	0.835259791	0.785994838	0.810545019	0.011320934
4.42	0.901357872	0.87072158	0.901357872	0.87072158	0.890281674	0.007668663
4.43	0.937380765	0.909662215	0.937380765	0.909662215	0.921662365	0.006729828
4.44	0.981146897	0.972730333	0.981146897	0.972730333	0.977458572	0.002493989
4.45	0.990910111	0.98148356	0.990910111	0.98148356	0.986937493	0.00232925
4.46	0.801593536	0.128754723	0.100199192	0.069658543	0.734699934 0.093815969	0.116486367 0.017437971
4.47	0.8038379530	0.7436875771	0.1148339932	0.1239479295	0.75469993 0.10781427	0.10648636 0.01521233
4.48	0.8850858489	0.7856581752	0.2950286163	0.0982072719	0.834699934 0.278233311	0.906486367 0.302162122
4.49	0.905285602	0.815396701	0.905285602	0.137769798	0.860752628 0.260650507	0.021723841 0.13904842
4.50	0.939064078	0.45516777	0.939064078	0.280439906	0.89353982 0.648738762	0.085409794 0.260898352
4.51	0.98810459	0.972505892	0.98810459	0.972505892	0.977249093 0.977249093	0.003452336 0.003452336
4.52	0.992256761	0.980473572	0.992256761	0.980473572	0.986589608 0.986589608	0.002582467 0.002582467

CHAPTER 5

CONCLUSION

5.1 Conclusion

In conclusion, WSNs is an attractive topic nowadays that can be used in different real life applications. Dynamic deployment is one of the most important challenges that affects the success of WSNs, by affecting the coverage rate of the wireless sensor network. Using optimization techniques to solve the dynamic deployment problem to optimize the coverage rate of the wireless mobile sensor nodes can be considered as an easy way to get a best possible coverage rate. ABC algorithm is an amazing optimization technique that can obtain the optimal solution for the problem in just a few steps. The control parameters of the ABC algorithm directly affect the results. For example, when increasing the number of cycles and the colony size, the obtained solutions are getting better. The solutions represent a best possible position for all mobile sensors in the sensing area. Sometimes the sensing area has an obstacle or constraints. The constraints may be a mountain, lake or any obstacle that prevents the sensors to be deployed on it. Therefore, the ABC algorithm must deploy the sensors out of the constraint region. That deployment can be done by two methods, the first one that depends on the coordinates of the constraint region. In each cycle, every new produced position of the mobile sensor that it's coordinates inside the constraint region, the ABC algorithm shifts that sensor to the left or to the right outside the forbidden region. That means, from the first cycle to the end of the cycles there is no any sensor in the obstacle area. The second method depends on the fitness value of the produced solution, when the coordinates of the produced solution are in the forbidden area the counter is increased by 1. The ABC algorithm counts the number of sensors in the forbidden area and divide the obtained coverage rate on that number +1, to get the fitness value. When the counter is not 0, that means it is not a good solution. Cycle by cycle the ABC

algorithm deploys all mobile sensor nodes outside the forbidden area to get best possible coverage rate. The obtained results show that the ABC algorithm is powerful optimization tool to solve optimization problems. Also, it can be applied for the dynamic deployment of WSNs to optimize the coverage rate.

5.2 Future works

The deployment of WSNs is one of the most important issues that determines the success of the WSNs. This work focused on the dynamic deployment of wireless mobile sensor nodes with constraints by using the ABC algorithm. In future we intend to use the quadcopters (Drones) with the ABC algorithm to make the predetermined deployment of WSNs for stationary sensor nodes. The ABC algorithm chooses the optimal positions for the WSN and send that information to the flight controller of the drone to deploy the sensors in their determined positions. Also, it is possible to make the dynamic deployment of stationary sensor nodes by using drones and the ABC algorithm.

REFERENCES

1. A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler and a. J. Anderson, (2002). "Wireless sensor networks for habitat monitoring," in First ACM Workshop on Wireless Sensor Networks and Applications, Atlanta, GA, Sept.

2. P.-J. Wan and C.-W. Yi, (2006). "Coverage by randomly deployed wireless sensor networks," **IEEE Transactions on Information Theory, 52**(6),. 2658-2669.

3. B. Wang, K. C. Chua, V. Srinivasan and W. Wang, (2007). "Information Coverage in Randomly Deployed Wireless Sensor Networks," **IEEE Transactions on Wireless Communications, 6**(8), 2994-3004.

4. A. Howard, M. Mataric and G. Sukhatme, (2002). "Mobile Sensor Network Deployment using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem," in the **6th International Symposium on Distributed Autmomous Robotics System (DARS)**, Fukuoka, Japan, Jun 25-27.

5. P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh and a. D. Rubenstein, (2002). "Energyefficient computing for wildlife tracking: Design tradeoffs and early experiences with **ZebraNet**," in Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X), San Jose, CA, Oct.

6. T. Wong, T. Tsuchiya and T. Kikuno, (2004). "A self-organizing technique for sensor placement in wireless micro-sensor networks," in **18th International Conference on Advanced Information Networking and Applications**, Fukuoka, Japan, March 26-29.

7. W. Sheng, G. Tewolde and S. Ci, (2006). "Micro Mobile Robots in Active Sensor Networks: Closing the Loop," in Proceeding of **IEEE International Conference on intelligent Robots and Systems (RSJ)**, Beijing, China, October 9-15.

8. Y. Zou and K. Chakrabarty, (2003). "Sensor Deployment and Target Localization Based on Virtual Fources," in IEEE Societies Twenty-Second Annual Joint Conference of the IEEE Computer and Communications **(INFOCOM)**, San Fransisco, USA, April 1-3.

9. S. Li, C. Xu, W. Pan and Y. Pan, (2005). "Sensor deployment optimization for detecting maneuvering targets," in **8th International Conference on Information Fusion**, PA, USA, July 25-29.

10. T. Wong, T. Tsuchiya and T. Kikuno, (2004). "A self-organizing technique for sensor placement in wireless micro-sensor networks," in **18th International Conference on Advanced Information Networking and Applications**, Fukuoka, Japan, March 26-29.

11. G. Wang, G. Cao and T. F. Porta, (2006). "Movement-Assisted Sensor Deployment," **Transactions on mobile computing, 5**, 640-652.

12. M. R. Pac, A. M. Erkmen and I. Erkmen, (2006). "Scalable Self-Deployment of Mobile Sensor Networks: A Fluid Dynamics Approach," in Proceeding of IEEE International Conference on intelligent Robots and Systems **(RSJ)**, Beijing, China, October 9-15.

13. R.-S. Chang and S.-H. Wang, (2008). "Self-Deployment by Density Control in Sensor Networks," **IEEE transactions on vehicular technology**, 57, 1745-1755.

14. O'Rourke and Joseph, (1987). "Art Gallery Theorems and Algorithms," New York: Oxford University Press.

15. Barricell and N. Aall, (1957). "Symbiogenetic evolution processes realized by artificial methods," Methods, pp. 143-182.

16. F. Alex, (1957). "Simulation of genetic systems by automatic digital computers. I. Introduction," **Aust. J. Biol. Sci, 10**, 484-491,.

17. A. Fraser and D. Burnell, Computer Models in Genetics, New York: McGrawHill, 1970.

18. J. Jia, J. Chen, G. Chang, J. Li and a. Y. Jia, "Coverage Optimization based on Improved NSGA-II in Wireless Sensor Network," **IEEE International Conference on Integration Technology**, pp. 614-618, 2007.

19. X. Wang, J. Ma, S. Wang and D. Bi, "Distributed Energy Optimization for Target Tracking in Wireless Sensor Networks," **IEEE Transactions on Mobile Computing, vol. 9**, pp. 73-86.

20. X. Bai, S. Li, C. Jiang and Z. Gao, "Coverage Optimization in Wireless Mobile Sensor Networks," **5th International Conference on Wireless Communications, Networking and Mobile Computing**, pp. 1-4, Beijing, China, September 24-26, 2009.

21. J. Li, K. Li and a. W. Zhu, "Improving sensing coverage of wireless sensor networks by employing mobile robots," Proceedings of the International Conference on Robotics and Biomimetics **(ROBIO)**, pp. 899–903, Sanya, China, December 15-18, 2007.

22. X. Wang, S. Wang and a. J.-J. Ma, "An Improved Co-evolutionary Particle Swarm Optimization for Wireless Sensor Networks with Dynamic Deployment," Sensors, vol. 7, p. 354–370, 2007.

23. Z. Huang and T. Lu, "A particle swarm optimization algorithm for hybrid wireless sensor networks coverage," **IEEE Symposium on Electrical & Electronics Engineering**, pp. 630-632, Kuala Lumpur, Malaysia, June 24-27, 2012.

24. N. D and N. B, "DV-based positioning in ad hoc networks," **Telecommunication Systems**, pp. 267-280, 2003.

25. J. Li, J. Zhang and L. Xiande, "A Weighted DV-Hop Localization Scheme for Wireless Sensor Networks," Scalable Computing and Communications Eighth International Conference on Embedded Computing, pp. 269-272, 25-27 **Sept Dalian**, China, September 25-27, 2009.

26. N. Bulusu, J. Heidemann and D. Estrin, "GPS-less low-cost outdoor localization for very small devices," **IEEE Personal Communications**, pp. 28-34, 2000.

27. J. MacDonald, D. Roberson and D. Ucci, "Location Estimation of Isotropic Transmitters in Wireless Sensor Networks," **IEEE Military Communications Conference (MILCOM)**, pp. 1-5, Washington, DC, October 23-25, 2006.

28. Z. Zhao-yang, G. Xu, L. Ya-peng and S.-s. Huang, "DV-Hop Based SelfAdaptive Positioning in Wireless Sensor Networks," in **5th International Conference on Wireless Communications, Networking and Mobile Computing**, Marrakech, Morocco, October 12-14, 2009.

29. L. Brito, Y. Liu and Y. Garcia, "An improved error localization on DV-Hop scheme for wireless sensors networks," **3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)**, vol. 2, pp. 80-84, Chengdu, China, August 20-22, 2010.

30. C. Xiaoyan, H. Ting and G. Mingjie, "Improved DV-Hop Algorithm Based on Self-Localization of WSN," **1st International Conference on Information Science and Engineering (ICISE)**, pp. 4026-4029, Nanjing, China, December 26-28, 2009.

31. K. Xu, M. Chen and Y. Liu, "A Novel Localization Algorithm Based on Received Signal Strength Indicator for Wireless Sensor Networks," **International Conference on Computer Science and Information Technology**, pp. 249-253, Singapore, August 29- September 2, 2008.

32. Q. Shi, H. Huo, T. Fang and R. Li, "Using steepest descent method to refine the node positions in wireless sensor networks," **IET Conference on Wireless, Mobile and Sensor Networks**, pp. 1055-1058, Shanghai, China, December 12-14, 2007.

33. J. Yao, J. L, L. Wang and Y. Han, "Wireless Sensor Network Localization Based on Improved Particle Swarm Optimization," **International Conference on Computing, Measurement, Control and Sensor Network**, pp. 72-75, Taiyuan, China, July 7-9, 2012.

34. J. Huanxiang, W. Yong and T. Xiaoling, "Localization algorithm for mobile anchor node based on genetic algorithm in wireless sensor network," **International Conference on Intelligent Computing and Integrated Systems**, pp. 40-44, Guilin, China, October 22-24, 2010.

35. B. Amutha and M. Ponnavaikko, "Locate: A hybrid localization scheme for wireless sensor networks," **IEEE Symposium on Industrial Electronics & Applications**, vol. 1, pp. 295-300, Kuala Lumpur, Malaysia, October 4-6, 2009.

36. K. Yu, Y. Guo and M. Hedley, "TOA-based distributed localisation with unknown internal delays and clock frequency offsets in wireless sensor networks," **IET Signal Processin**g, Vols. 3, Issue 2, pp. 106-118 , 2009.

37. Z. Ma and K. Ho, "TOA localization in the presence of random sensor position errors," 2011 **IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**, pp. 2468-2471, Prague, Czech Republic, May 22-27, 2011.

38. E. Xu, Z. Ding and S. Dasgupta, "Source Localization in Wireless Sensor Networks From Signal Time-of-Arrival Measurements," **IEEE Transactions on Signal Processing**, vol. 59 Issue: 6 , pp. 2887-2897 , 2011.

39. J. Xu, M. Ma and C. L. Law, "AOA Cooperative Position Localization," **IEEE Global Telecommunications Conference (GLOBECOM)**, pp. 1-5, New Orleans, LA, November 30- December 4, 2008.

40. C.-C. Pu and W.-Y. Chung, "Mitigation of Multipath Fading Effects to Improve Indoor RSSI Performance," **IEEE Sensors Journal, Vols. 8**,Issue: 11 , pp. 18841886 , 2008.

41. Y. Wu, J. Hu and Z. Chen, "Radio Map Filter for Sensor Network Indoor Localization Systems," **5th IEEE International Conference on Industrial Informatics, vol. 1**, pp. 63-68, Vienna, Austria, July 23-26, 2007.

42. W. Chen, T. Mei, L. Sun, Y. Liu, Y. Li, S. Li, H. Liang and M.-H. Meng, "Error analyzing for RSSI-based localization in wireless sensor networks," **7th World Congress on Intelligent Control and Automation**, pp. 2701-2706, Chongqing, China, June 25-27, 2008.

43. N. Aziz, A. Mohemmed and M. Alias, "A wireless sensor network coverage optimization algorithm based on particle swarm optimization and Voronoi diagram," **International Conference on Networking, Sensing and Control,** pp. 602- 607, Okayama City, Japan, March, 26-29, 2009.

44. S. Meguerdichian, F. Koushanfar, M. Potkonjak and M. Srivastava, "Coverage problems in wireless ad-hoc sensor networks," Proceedings. **IEEE INFOCOM.** Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1380-1387, Anchorage, Alaska, April 22-26, 2001.

45. A. Boukerche and X. Fei, "A Voronoi Approach for Coverage Protocols in Wireless Sensor Networks," **IEEE Global Telecommunications Conference (GLOBECOM)**, pp. 5190-5194, Washington, DC, November 26-30, 2007.

46. G. Wang, G. Cao and T. F. Porta, "Movement-Assisted Sensor Deployment," Transactions on mobile computing, vol. 5, pp. 640-652, 2006.

47. J. Li, R.-c. Wang, H.-p. Huang and L.-j. Sun, "Voronoi Based Area Coverage Optimization for Directional Sensor Networks," **Second International Symposium on Electronic Commerce and Security**, vol. 1, pp. 488-493, Nanchang, China, May 22-24, 2009.

48. D. Karaboga, "An idea based on honey bee swarm for numerical optimization", Technical Report-TR06, **Erciyes University**, Engineering Faculty, Computer Engineering Department, 2005.

49. D. Karaboga, C. Ozturk, "A novel clustering approach: Artificial Bee Colony (ABC) algorithm", **Applied Soft Computing**, Vol. 11, pp. 652-657, 2011.

50. D. Karaboga, C. Ozturk, "Neural networks training by artificial bee colony algorithm on pattern classification", **Neural Network World**, Vol. 19, pp. 279-292, 2009.

51. B. Basturk, D.Karaboga, An Artificial Bee Colony (ABC) Algorithm for Numeric function Optimization, **IEEE Swarm Intelligence Symposium** 2006, May 1214, 2006, Indianapolis, Indiana, USA.

52. Ozturk, C.; Karaboga, D.; Gorkemli, B. Artificial Bee Colony Algorithm for Dynamic Deployment of Wireless Sensor Networks. **Turk. J. Elec. Engi. Compu. Sci.** 2011, in press

CURRICULUM VITAE

Name and surname: AWS ABD AL KAREEM HAMEED ALTAEE

Nationality: İraq

Birth date and place: 1983

Marital status: Bağdat

Cell phone: 0 537 309 65 12

E-mail: masterthesisofaws@gmail.com

Correspondence Address: Fevzi Çakmak Mah. Hilal Cad. No:27/7

Kocasinan / KAYSERİ

EDUCATION

Degree	Institution	Date of graduation
MSc	Erciyes University	----------
License	Bağdat University	2006
High school	Al-Tameem	2002

FOREIGN LANGUAGE

Arabic

English