Optimizing Wireless Sensor Network Deployment with Hybrid Genetic Algorithm

(Authors are hidden for blind review)

***Abstract*—Wireless Sensor Networks (WSNs) play a crucial role in various applications, including military applications, environmental monitoring, disaster management, and security systems. Efficient deployment of WSNs is essential for achieving m-connectivity and k-coverage, ensuring network reliability and effective target area monitoring. This paper presents a novel approach that combines the strengths of genetic algorithm with a problem-specific local improvement heuristic to efficiently explore the solution space. The proposed method, Hybrid Genetic Algorithm for Wireless Sensor Network deployment (HGA-WSN), demonstrates superior performance compared to existing algorithms, using fewer sensors to achieve connectivity and coverage requirements while converging more rapidly to optimal solutions. The effectiveness of HGA-WSN highlights its significant potential for optimizing WSN deployment across a wide range of applications.**

***Index Terms*—Wireless Sensor Networks, WSN deployment, Hybrid Genetic Algorithm, HGA-WSN, m-connectivity, k-coverage, optimization, local search strategies**

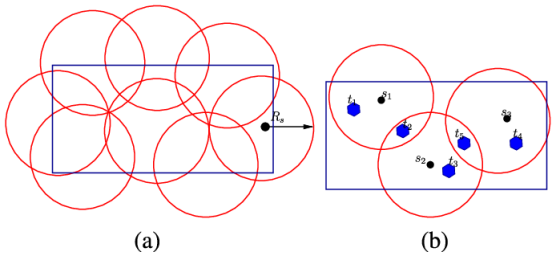
I. Introduction

Wireless Sensor Networks (WSNs) are a collection of small, low-power devices equipped with sensors, communication capabilities, and processing capabilities. These devices can be deployed in various environments to monitor and gather data about physical phenomena, such as temperature, humidity, or light. The collected data is transmitted wirelessly to a central node, where it can be analyzed and processed.

WSNs have a wide range of applications in various fields. In environmental monitoring, WSNs can be used to monitor air quality, water quality, and soil moisture. In healthcare, WSNs can be used to monitor patient vital signs and track the movement of patients and medical equipment in hospitals. In industrial automation, WSNs can be used to monitor and control manufacturing processes, track inventory, and monitor equipment conditions. WSNs can also be used in smart homes and buildings to monitor and control temperature, lighting, and security. Overall, WSNs offer a cost-effective and reliable solution for monitoring and gathering data in various environments.

One of the primary challenges in deploying WSNs is the placement problem. This refers to the problem of determining the optimal location for sensor nodes to ensure that the network provides adequate coverage, connectivity, and energy efficiency. The placement problem is a critical issue in WSN design since the effectiveness of the network depends largely on the location of the sensor nodes.

Several methodologies have been proposed to tackle the placement problem in WSNs, which involves determining the optimal placement of sensor nodes in a given area.



(a) (b)

Fig. 1. (a) Area Coverage vs (b) Target Coverage (source: [1])

These methodologies encompass various techniques, including evolutionary algorithms such as the Genetic algorithm [4], [8], [9], [11], [12], swarm intelligence techniques such as Bee Colony Optimization [3], and Ant Colony Optimization [15], as well as mathematical programming approaches such as integer-programming [13] and mixed-integer programming [15]. Integer-programming (IP) and mixed-integer programming (MIP) can provide optimal solutions for small to medium-sized instances of the placement problem. Section II includes a more detailed listing of the previous studies.

Coverage is one of the main objectives in the placement problem of WSNs, and it refers to the degree to which the sensing area is monitored by the sensors. In other words, it is a measure of how well the sensors are able to detect events or changes in the environment. Two main types of coverage problems are typically addressed in WSN placement: area coverage and target coverage.

Area coverage ensures that every point in the sensing region is covered by at least one sensor node. This type of coverage is vital in monitoring natural disasters and other applications requiring complete coverage of an area. The problem of area coverage can be solved by deploying sensor nodes uniformly or non-uniformly across the sensing region. However, uniform deployment may result in energy wastage, while non-uniform deployment may lead to coverage holes or redundancies.

Target coverage, on the other hand, ensures that a specific target is covered by at least one sensor node. This type of coverage is critical in applications such as intrusion detection, where specific targets need to be monitored. Target coverage problems are more complex than area coverage problems, requiring a specific set of targets to be covered. Common models used to solve target coverage problems include k-coverage and m-connectivity, which ensure that a specific number of sensor nodes covers each target. Optimization methods such as genetic algorithms, particle swarm optimization,

and simulated annealing are typically used to determine the optimal placement of sensor nodes that satisfy the k-coverage and m-connectivity constraints.

Both area coverage and target coverage problems are important in WSN placement as they affect the effectiveness and efficiency of the network. Several approaches have been proposed to solve these problems, including optimization algorithms, mathematical models, and simulation tools. The ultimate goal is to achieve optimal placement of sensor nodes to ensure adequate coverage, connectivity, and energy efficiency while minimizing the cost and maximizing the network lifetime.

In this study, we present a hybrid genetic algorithm (HGA-WSN) that contains a problem-specific local search for the WSN placement problem that satisfies k-coverage and m-connectivity constraints. We compare our solution with the standard implementation of a genetic algorithm, a greedy heuristic, and a standard implementation of simulated annealing. The experiments confirm that HGA-WSN achieves significantly better results in terms of sensor count.

The remainder of this paper is structured as follows: Section II provides a summary of previous studies on the WSN placement problem. Section III describes the proposed hybrid solution. In Section IV, we present the experimental work and results. Finally, Section V provides concluding remarks and outlines possible further research opportunities.

II. Related Work

Numerous studies have been conducted in the past regarding Wireless Sensor Networks (WSNs), with different objectives, limitations, and strategies. For instance, in [2], the focus was on the energy efficiency of sensor nodes in WSNs, which was achieved through controlled placement of sensor nodes instead of random deployment.

In [3], Yarinezhad and Hashemi defined the coverage problem in three sub-titles: simple coverage, k-coverage, and q-coverage. They used Particle Swarm Optimization while taking into account these factors and network lifetime issues.

Carter and Ragade, in [4], aimed to develop a probabilistic model that ensures target coverage with the minimum number of sensor nodes, using the Genetic Algorithm.

Katti [5] utilized disjoint and non-disjoint cover sets, where different sets of sensors were cycled through to optimize energy consumption and increase network lifetime. The locations of sensors were predetermined.

The work in [6] focused on area coverage with a longer network lifetime by using the Territorial Predator Scent Marking Algorithm.

In another study [7], the focus was on target coverage and connectivity with multiple sinks. The problem was divided into two sub-problems: covering targets and connecting sensors with the sinks. One heuristic was proposed for covering targets, and two for connectivity. The authors managed to reduce the number of required sensor nodes.

In [8], the Genetic Algorithm was used to extend the WSN lifetime with k-coverage. An algorithm was implemented to select cover sets for the optimization of network lifetime. Similarly, Chand and Kumar [9] used an algorithm to create cover sets of sensors to maximize network lifetime.

Cardei et al. [10] selected cover sets for the maximization of network lifetime. They used LPMSC and Greedy-MSC heuristics, with a linear programming formulation and a greedy approach, respectively.

In [11], the aim was to find the minimum number of sensor nodes required while taking into account k-coverage and m-connectivity issues, which was achieved with a GA-based approach.

Graph decomposition, integer linear programming, and binary search were used in [13] to find an optimal spanning tree of a wireless sensor network, as highlighted in their conclusion.

In [14], the authors propose a clustering routing algorithm for wireless sensor networks (CHRA) that balances energy usage and extends network lifetime. Their approach considers both heterogeneous nodes and cluster heads, and their results show that CHRA effectively prolongs network lifetime and balances energy consumption.

Okdem and Karaboga [15] aim to maximize network lifetime while ensuring efficient data transmission in WSNs using a novel protocol based on the ACO. This protocol enables efficient multi-path data transmission and reliable communication even in the presence of node faults.

In this study, we improve the standard GA algorithm by hybridizing it with a problem-specific local search. The details of the algorithm along with the problem formulation are explained in the following section.

III. Methodology

This section presents the methodology for our Wireless Sensor Network (WSN) framework. Targets are uniformly distributed across a two-dimensional plane, with an array of predetermined potential positions for placing sensor nodes. The targets and sensor nodes, once assigned to their respective positions, are fixed and immobile. A target is regarded as covered if it falls within any sensor node's sensing range. A critical feature of our framework is that a single sensor node has the capacity to detect multiple targets, optimizing the use of resources.

A. Formal Definition

Terminology:

- T, P, and S represent N target points, K potential positions, and M sensor nodes, respectively.
- R_{com} and R_{sen} are the communication and sensing ranges.
- $dist(\tau_i, s_j)$ is the distance between τ_i and s_j.
- $Cov(\tau_i)$ and $TCov(s_i)$ are sensor nodes covering τ_i and target points covered by s_i, respectively.
- $Com(s_i)$ denotes sensor nodes within s_i's communication range.

The coverage and connectivity issue: given N targets and K potential positions, find the minimal set of positions for sensor nodes to ensure k-coverage and m-connectivity.

Boolean variables b_{ij} and c_{ij} denote if τ_i is covered by s_j and if s_i is directly connected to s_j, respectively. q_i indicates if a sensor is at p_i.

The Linear Programming Problem (LPP) is:

$$\text{Minimize } Z = \sum_{i=1}^{K} q_i$$

$$\text{Subject to}$$

$$\sum_{j=1}^{M} b_{ij} \geq k, \forall i, 1 \leq i \leq N$$

$$\sum_{j=1, j\neq i}^{M} c_{ij} \geq m, \forall i, 1 \leq i \leq M$$

Constraints ensure k-coverage for each target and m-connectivity for each sensor node.

B. Fitness Function

The proposed method seeks to identify the fewest sensor positions from a given set that ensure k-coverage of all targets and m-connectivity of the deployed sensors, given set values for m and k. A fitness function guides the algorithm to feasible solutions. It comprises:

1) *Minimizing the selected potential positions:* Assuming M positions are chosen from the K potential positions, the first goal can be written as:

$$\text{Goal 1: Minimize } F_1 = \frac{M}{K}$$

2) *k-coverage of targets:* $Cov(\tau_i)$ denotes the sensor nodes covering τ_i. The coverage cost of a target τ_i is defined as:

$$CovCost(\tau_i) = \begin{cases} 0, & \text{if } \mid Cov(\tau_i) \mid \geq k \\ k - Cov(\tau_i), & \text{else.} \end{cases} \tag{1}$$

Thus, our second goal is:

$$\text{Goal 2: Minimize } F_2 = \frac{1}{N \times k} \sum_{i=1}^{N} CovCost(\tau_i)$$

3) *m-connectivity of sensor nodes:* $Com(s_i)$ symbolizes the sensor nodes within s_i's communication range. The connectivity cost $ConnCost(s_i)$ for a sensor node s_i is:

$$ConnCost(s_i) = \begin{cases} 0, & \text{if } \mid Com(s_i) \mid \geq m \\ m - Com(s_i), & \text{else.} \end{cases} \tag{2}$$

Hence, our third goal is:

$$\text{Goal 3: Minimize } F_3 = \frac{1}{M \times m} \sum_{i=1}^{M} ConnCost(s_i)$$

These goals may conflict. Ensuring k-coverage and m-connectivity requires more potential positions, conflicting with our first goal. Each objective's equations are normalized to yield values between 0 and 1. We apply the Weighted Sum Approach (WSA) [12] to construct a multi-objective fitness function. This classic method multiplies each objective by a weight value W_i and adds the results together into a unified scalar objective function, defined as follows:

$$\text{Fitness Value} = W_1 \times F_1 + W_2 \times F_2 + W_3 \times F_3$$

$$\text{Fitness Value} = W_1 \times \frac{M}{K} + W_2 \times \frac{1}{N \times k} \sum_{i=1}^{N} \text{CovCost}(\tau_i) + W_3 \times \frac{1}{M \times m} \sum_{i=1}^{M} \text{ConnCost}(s_i) \tag{3}$$

As Goal 2 and Goal 3 are the original problem's constraints, we give them a sufficiently large weight (W_2 and $W3$) to ensure that infeasible solutions always have worse fitness values than the feasible.

C. Genetic Algorithm

The Genetic Algorithm (GA) is a computational method inspired by biology that mimics natural selection in order to discover optimal solutions for difficult problems [16]. Drawing on principles from evolutionary biology, GAs operate by maintaining a population of candidate solutions, which are evolved iteratively through selection, crossover, and mutation operations. The algorithm begins with an initial population of randomly generated solutions. During each iteration or generation, a new set of offspring solutions is produced by applying genetic operators to the current population. Algorithm 1 demonstrates a simple GA implementation.

The following sections explain how this study adapts the various components of GA in the proposed solution:

1) Solution Representation: One of the challenges of GA is choosing an appropriate encoding scheme for candidate solutions. This significantly impacts the algorithm's effectiveness.

We use binary representation, where each candidate solution is represented as a binary string of fixed length, composed of 0s and 1s. Each character in the string indicates the existence of a sensor node at a certain potential position. Formally, let $P = p_1, p_2, ..., p_n$ be the set of potential positions, where each position p_i is represented as a coordinate tuple (x_i, y_i). Given a binary string $S = s_1 s_2 ... s_n$ representing a candidate solution, $s_i = 1$ denotes that there is a sensor at p_i.

2) Selection Process: This process favors fitter individuals, allowing them to pass their genetic material to the next generation. Various selection methods, such as rank selection, roulette wheel selection, and tournament selection, can be used to choose parents based on their fitness.

Algorithm 1 Genetic Algorithm

```
Initialize: pop_size, mut_rate
Generate an initial population
Evaluate population fitness
while (not termination_state) do
    Initialize empty offspring_pop
    while size of offspring_pop < pop_size do
        Select parents by fitness
        Apply crossover
        Apply mutation with probability mut_rate
        Add offspring to offspring_pop
    end while
    Evaluate offspring_pop fitness
    Replace population with offspring_pop
    Next termination_state
end while
return best_individual
```

Algorithm 2 HGA-WSN

```
Require: mig_period
Initialize: pop_size, imp_rate, mut_rate
Generate an initial population
while not termination_state do
    Select parents by fitness
    Apply crossover to each parent pair
    Apply mutation with probability mut_rate
    Apply problem-specific local search with probability imp_rate
    Get migration to the population in every mig_period
    Add each offspring solution to the population
    Apply survival policy
    Next termination_state
end while
return best_individual
```

In the proposed algorithm, the roulette wheel selection algorithm is used. Roulette wheel selection is a stochastic method that assigns selection probabilities based on individuals' fitness. Fitter individuals have higher chances of being chosen, while less fit ones still have a chance to contribute, ensuring diversity.

3) Crossover Operator: The crossover operator plays an essential role in the genetic algorithm by merging pairs of parent solutions to produce offspring. Crossover techniques, such as single-point or multi-point crossover, recombine the parents' features to create diverse offspring that inherit traits from both parents. These techniques facilitate the generation of new solutions that could potentially outperform their predecessors, ultimately driving the population toward improved fitness levels.

Single-point crossover operator is used in the proposed algorithm which is a widely-used technique, that selects a random crossover point in the parent binary strings and exchanges the segments after this point between the two parents.

4) Mutation Operator: The mutation operator is responsible for introducing small, random changes to candidate solutions to maintain diversity and facilitate exploration of the search space. Different types of mutation operators can be employed depending on the solution representation and the nature of the problem, such as bit-flip mutation, swap mutation, and k-bit flip mutation.

D. Proposed Algorithm

We propose a hybrid Genetic Algorithm (HGA-WSN) powered with a problem-specific local search that delivers the optimal solution faster than the standard implementation of GA (a detailed comparison is taking place in Section IV). Algorithm 2 presents the pseudo-code of the algorithm.

Note that, the fitness values of solutions are implicitly calculated whenever they are generated. Thus, in the algorithm listing, it is not mentioned as a separate step.

The parents are selected using inverse roulette wheel selection favoring the fitter individuals. Selected parents then are passed to the crossover operator using the single-point crossover technique. With the probability of mut_rate each offspring solution is mutated using a simple mutation operator, in which, some randomly selected bits are set to zero.

Each solution returned from the mutation operator undergoes a heuristic search. This heuristic improves the solutions with the probability of imp_rate. Thereafter, depending on the iteration count, the population accepts offspring, and, depending on the iteration count, immigrant solutions are put into the population. The immigrant solutions are generated randomly to boost diversity. Finally, to keep the population size fixed, some individuals, selected by roulette wheel selection according to their fitness, are removed from the population.

1) Problem-Specific Local Search: The proposed heuristic simply turns on sensors that are contributing to either the connectivity of an insufficiently connected sensor or the coverage of an insufficiently covered target. The pseudo-code is listed in Algorithm 3.

Firstly, it calculates each target's coverage value. If the coverage is achieved for target t, i.e. the current k value is greater than or equal to the k-constraint of the problem, it skips that target. If the coverage is not achieved, it looks at all of the sensor nodes that can cover that target. The sensor nodes are turned on the k value is updated accordingly. This operation is done for all of the targets. If there is no sensor to turn on, then the sensors that are violating the connectivity constraint are detected and their neighbors are turned on one by one until it gets connected enough. Here the neighbors with higher connectivity are favored to avoid new connectivity violations introduced by newly turned-on sensors.

Note that the heuristic only turns on sensors. This is because finding redundant sensors takes too much time without achieving significant improvements in the offspring solutions. As shown in 3, turning off sensors only benefits the first parameter which has a small weight, 0.1, compared to the summation of the other two which is, 10.0. Therefore, the heuristic only

Algorithm 3 Local Heuristic

Require: K, M, *targets*, *turned_off_sensors*, *turned_on_sensors*

sensors_to_turn_on = list of sensors to turn on
for all $t \in$ *targets* **do**
  Calculate k for t
  if calculated $k \geq K$ **then**
    continue
  end if
  for all *idle* $\in$ *turned_off_sensors* **do**
    if *idle* can cover t **then**
      Add *idle* to *sensors_to_turn_on*
    end if
  end for
  if *sensors_to_turn_on* is $\emptyset$ **then**
    for all $s \in$ *turned_on_sensors* **do**
      if Calculated $m \geq M$ **then**
        continue
      end if
      Add the neighbor with the highest connectivity to *turned_on_sensors*
    end for
  end if
end for
return *sensors_to_turn_on*

involves turning on sensors whilst the mutation operator is turning off sensors. Thus the balance is achieved and the population maintains both diverse and optimal solutions.

IV. Experimental Work

The experiments are conducted on a system equipped with an Intel i5 processor, a 1.60GHz CPU, 8 GB RAM, and Microsoft Windows 10 as the platform.

The sensing field is assumed 600 $\times$ 600 square meters in area and the sensing range and the communication range are assumed 50 and 100 meters, respectively. Additionally, the weight values for the sensor, the coverage, and the connectivity objective are 0.1, 5, and 5, respectively. These values are predefined and never changed throughout the execution of the experiments. The number of potential sensors is also predefined as 576 since we have a field of 600 $\times$ 600 square meters and each potential sensor is located evenly spaced 25 meters away from other sensors. Therefore, 600 / 25 = 24, and 24 $\times$ 24 equals 576 potential positions.

Different problem instances are created for different values of m, k, and target values. These m and k values could take 1, 2, or 3 whereas the target count value could take 100, 200, or 300 as a value. A problem instance is one of the combinations of these values and it is ensured that one problem instance is created for each combination.

The target points are created randomly. Assume the target count is 200. In this case, 200 target points in which each target can have point values between 0 and 600 are created. To make sure that these points are not gathered in a favorable

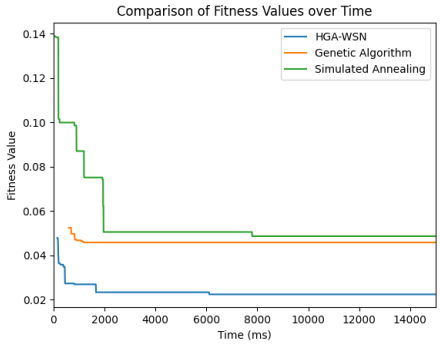


Fig. 2. Comparison of the algorithms by time

area, each combination of m, k, and target count values is created 5 times. In summary, the total number of problem instances is 3 $\times$ 3 $\times$ 3 $\times$ 5 = 135.

Due to the stochastic nature of the algorithms, in both parameter tuning and comparison experiments, each experiment is repeated 10 times on each problem instance, and the averages of the results are taken.

A. Parameter Tuning

There are three parameters within the HGA-WSN: population count, improvement rate, and immigrant count, and the values {30, 60, 100}, {0.2, 0.4, 0.6, 0.8}, and {6, 12, 24, 30} are tested respectively. The values for the population count, improvement rate, and immigrant count that are observed to perform optimal performance are 60, 0.8, and 24, respectively. These values will be used in Section IV-B where comparison tests will be discussed.

B. Comparison Experiments

To determine the time required for the algorithms to converge to an optimal solution (local or global) we analyze the progress of the algorithms over time. As shown in Figure 2, after about 10 seconds, all algorithms converge and cease to make significant improvements on the solution they offer. Therefore we run each instance of the comparison experiments for 15 seconds to allow the algorithms to converge. Figure 2 also demonstrates that HGA-WSN reaches better solutions faster than the other algorithms. Note that, Greedy Algorithm is omitted in this figure since it does not run against time.

Three different algorithms are compared against HGA-WSN to evaluate their performance in wireless sensor networks (WSNs): Standard Genetic Algorithm [11], Simulated Annealing [17], and Greedy Algorithm [11]. All of the algorithms are implemented in Java.

Figure 3 presents the comparison of the four algorithms. In the figure, there are 9 different problem instances where m and k values can take 1, 2, 3 and the total number of target points is 300. At the beginning of the chapter, it is mentioned that there are 27 different problem instances. However, to fit the data into one visible figure, the problem instances with 100 and 200 target points are omitted.

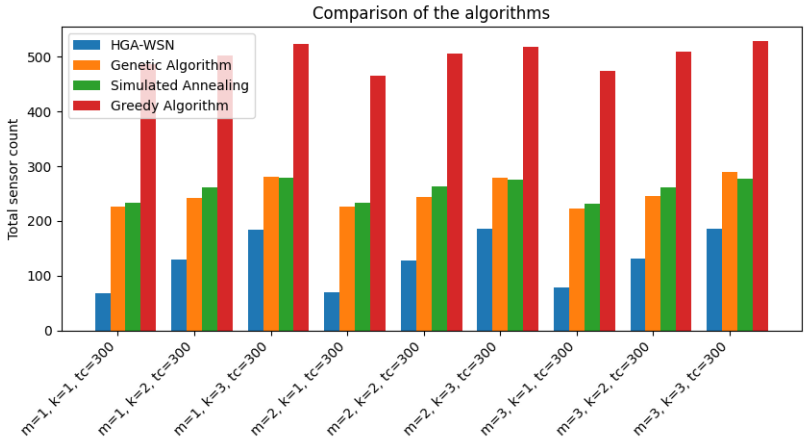


Fig. 3. Comparison of the algorithms

V. Conclusion

This paper introduces a Hybrid Genetic Algorithm (HGA-WSN) for optimizing wireless sensor network deployment, aiming to achieve both m-connectivity and k-coverage. The proposed algorithm combines the strengths of genetic algorithms and local search strategies to efficiently explore the solution space. Extensive experiments are conducted on various problem instances, including different values for m, k, and target counts.

HGA-WSN consistently outperforms the other algorithms, using fewer sensors to attain m-connectivity and k-coverage, and converging to optimal solutions more rapidly. The effectiveness of HGA-WSN highlights its potential for wireless sensor network deployment in various applications, such as environmental monitoring, disaster management, and security systems. As a follow-up to this study, one can explore the applicability of HGA-WSN in large-scale networks, the integration of other heuristics, alternative solution representations, adaptive parameters, and dynamic environments. Additionally, parallelizing the HGA-WSN algorithm could be investigated, leveraging the power of parallel computing to further enhance the optimization process and improve the adaptability and scalability of HGA-WSN.

References

[1] Tripathi, Abhishek Gupta, Hari Dutta, Tanima Mishra, Rahul Shukla, Kaushal Jit, Somporn. (2018). Coverage and Connectivity in WSNs: A Survey, Research Issues and Challenges. IEEE Access. 6. 26971-26992. 10.1109/ACCESS.2018.2833632.

[2] Slijepcevic, S., Potkonjak, M. (2001, June). Power efficient organization of wireless sensor networks. In ICC 2001. IEEE international conference on communications. Conference record (Cat. No. 01CH37240) (Vol. 2, pp. 472-476). IEEE.

[3] Yarinezhad, R., Hashemi, S. N. (2020). A sensor deployment approach for target coverage problem in wireless sensor networks. Journal of Ambient Intelligence and Humanized Computing, 1-16.

[4] Carter, B., Ragade, R. (2009, February). A probabilistic model for the deployment of sensors. In 2009 IEEE Sensors Applications Symposium (pp. 7-12). IEEE.

[5] Katti, A. (2022). Target coverage in random wireless sensor networks using cover sets. Journal of King Saud University-Computer and Information Sciences, 34(3), 734-746.

[6] Zainol Abidin, H., Din, N. M. (2013). Sensor node placement in wireless sensor network based on territorial predator scent marking algorithm. International Scholarly Research Notices, 2013.

[7] Hanh, N. T., Le Nguyen, P., Tuyen, P. T., Binh, H. T. T., Kurniawan, E., Ji, Y. (2018, January). Node placement for target coverage and network connectivity in WSNs with multiple sinks. In 2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC) (pp. 1-6). IEEE.

[8] Elhoseny, M., Tharwat, A., Farouk, A., Hassanien, A. E. (2017). K-coverage model based on genetic algorithm to extend WSN lifetime. IEEE sensors letters, 1(4), 1-4.

[9] Chand, S., Kumar, B. (2018). Genetic algorithm-based meta-heuristic for target coverage problem. IET Wireless Sensor Systems, 8(4), 170-175.

[10] Cardei, M., Thai, M. T., Li, Y., Wu, W. (2005, March). Energy-efficient target coverage in wireless sensor networks. In Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies. (Vol. 3, pp. 1976-1984). IEEE.

[11] Gupta, S. K., Kuila, P., Jana, P. K. (2016). Genetic algorithm approach for k-coverage and m-connected node placement in target based wireless sensor networks. Computers Electrical Engineering, 56, 544-556.

[12] Konak A, Coit DW, Smith AE. Multi-objective optimization using genetic algorithms: a tutorial. Reliab Eng Syst Saf 2006;91(9):992–1007.

[13] Ma, X., Zhu, X., Chen, B. (2017, March). Exact algorithms for maximizing lifetime of wsns using integer linear programming. In 2017 IEEE Wireless Communications and Networking Conference (WCNC) (pp. 1-6). IEEE.

[14] Li, C., Bai, J., Gu, J., Yan, X., Luo, Y. (2018). Clustering routing based on mixed integer programming for heterogeneous wireless sensor networks. Ad Hoc Networks, 72, 81-90.

[15] Okdem, S., Karaboga, D. (2006, June). Routing in wireless sensor networks using ant colony optimization. In First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06) (pp. 401-404). IEEE.

[16] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. University of Michigan Press.

[17] Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. (1983). Optimization by Simulated Annealing. Science, 220(4598), 671-680. doi:10.1126/science.220.4598.671