Dynamic Sensor Deployment in Mobile Wireless Sensor Networks Using Multi-Agent Krill Herd Algorithm

by

AMIR ANDALIBY JOGHATAIE

B.Sc., Islamic Azad University Shahre Rey, 2011
M.Sc., Newcastle University, 2012

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Dynamic Sensor Deployment in Mobile Wireless Sensor Networks Using Multi-Agent Krill Herd Algorithm

by

Amir Andaliby Joghataie
B.Sc., Islamic Azad University Shahre Rey, 2011
M.Sc., Newcastle University, 2012

Supervisory Committee

Dr. T. Aaron Gulliver, Supervisor
(Department of Electrical and Computer Engineering)

Dr. Amirali Baniasadi, Departmental Member
(Department of Electrical and Computer Engineering)

ABSTRACT

A Wireless Sensor Network (WSN) is a group of spatially dispersed sensors that monitor the physical conditions of the environment and collect data at a central location. Sensor deployment is one of the main design aspects of WSNs as this affects network coverage. In general, WSN deployment methods fall into two categories: planned deployment and random deployment. This thesis considers planned sensor deployment of a Mobile Wireless Sensor Network (MWSN), which is defined as selectively deciding the locations of the mobile sensors under the given constraints to optimize the coverage of the network.

Metaheuristic algorithms are powerful tools for the modeling and optimization of problems. The Krill Herd Algorithm (KHA) is a new nature-inspired metaheuristic algorithm which can be used to solve the sensor deployment problem. A Multi-Agent System (MAS) is a system that contains multiple interacting agents. These agents are autonomous entities that interact with their environment and direct their activity towards achieving specific goals. Agents can also learn or use their knowledge to accomplish a mission. Multi-agent systems can solve problems that are very difficult or even impossible for monolithic systems to solve. In this work, a modification of KHA is proposed which incorporates MAS to obtain a Multi-Agent Krill Herd Algorithm (MA-KHA).

To test the performance of the proposed method, five benchmark global optimization problems are used. Numerical results are presented which show that MA-KHA performs better than the KHA by finding better solutions. The proposed MA-KHA is also employed to solve the sensor deployment problem. Simulation results are presented which indicate that the agent-agent interactions in MA-KHA improves the WSN coverage in comparison with Particle Swarm Optimization (PSO), the Firefly Algorithm (FA), and the KHA.

Contents

List of Tables

List of Figures

Glossary

ACO	Ant Colony Optimization
AI	Artificial Intelligence
APF	Artificial Potential Field
BBO	Biography-Based Optimization
BFGS	Broyden-Fletcher-Goldfarb-Shanno
CG	Computational Geometry-based
CS	Cuckoo Search
DE	Differential Evolution
ES	Evolutionary Strategies
FA	Firefly Algorithm
GA	Genetic Algorithm
HS	Harmony Search
IoT	Internet of Things
ISOGRID	Isometric GRID-based
KHA	Krill Herd Algorithm
MA-KHA	Multi-Agent Krill Herd Algorithm
MAS	Multi-Agent System
MEMS	Micro-Electro-Mechanical-System
MWSN	Mobile Wireless Sensor Network
NFE	Number of Function Evaluations
NP-hard	Non-deterministic Polynomial-time hard

PSO	Particle Swarm Optimization
SA	Simulated Annealing
SD	Standard Deviation
SF	Sensing Field
SGA	Stud Genetic Algorithm
SI	Swarm Intelligence
SKH	Stud Krill Herd
TS	Tabu Search
VD	Voronoi Diagram
VFA	Virtual Forced-based Approach
WSN	Wireless Sensor Network

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. T. Aaron Gulliver, for his continuous support, patience, and immense knowledge. I could not have imagined having a better advisor and mentor.

DEDICATION

To the ones whom I worship, **Mom** and **Dad**

and

to the one who has always stood beside me with love, **Parniy**.

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) are a group of spatially dispersed sensors that monitor the physical conditions of the environment and organize the collected data at a central location. Advances in wireless communications, digital electronics, and Micro-Electro-Mechanical-System (MEMS) have led to WSNs being extensively employed in many fields [1]. The advantages of WSNs in distributed self-organization and low energy consumption have made them an integral part of new technologies such as the Internet of Things (IoT) [2].

A wireless sensor network is composed of small low-power sensor nodes that are capable of sensing various physical phenomena such as sound, light, motion, and temperature. These devices process sensory data and send the results to a collection point. The collection point, called a sink or base station, is connected to a wired or wireless network. This forms an ad-hoc network and lets the sensor network carry out data processing [3], [4].

WSNs are fundamentally different from other wired and wireless networks as they have communication and energy constraints. Sensor deployment is another design aspect that influences almost all performance metrics, including network coverage,

network lifetime, and sensor connectivity. Hence, good sensor deployment is essential in every WSN since this reduces the energy consumption of the network [5].

Optimization algorithms can be used to provide solutions for the WSN deployment problem. Nonetheless, conventional deterministic optimization approaches are not suitable for most deployment problems as they are characterized by multiple design objectives and a large number of heterogeneous sensors. Such problems have been proven to be Non-deterministic Polynomial-time hard (NP-hard) [6], and the time required to find an optimal solution increases exponentially with the size of the problem [7]. For this reason, metaheuristic and Swarm Intelligence (SI) algorithms that belong to the discipline of Artificial Intelligence (AI) have become popular over the last decade [8]. SI-based optimization (swarm optimization) is a branch of metaheuristic algorithms inspired by the collective behavior of social swarms of creatures such as ants, bees, fireflies, worms, and birds. While individuals in swarms are relatively unsophisticated, their collective behavior leads to the self-organization of the whole system [9].

1.1 Objectives

In this thesis, the deployment problem in Mobile Wireless Sensor Networks (MWSNs) using swarm optimization is investigated. Here the sensor deployment problem and the corresponding coverage problem are interpreted as obtaining the maximum coverage in a given target field using a set of mobile wireless sensors, and are solved using Particle Swarm Optimization (PSO), Firefly Algorithm (FA), and Krill Herd Algorithm (KHA).

KHA is a swarm optimization method that is among the best SI algorithms. Nonetheless, it has not been applied for many applications because it is a relatively

new algorithm. PSO is a well-studied algorithm in the area of swarm optimization and is usually used as a benchmark. The FA is a nature-inspired SI method that can provide solutions for difficult optimization problems including NP-hard problems. In this thesis, PSO and FA have been chosen to compare with KHA because these algorithms have similar characteristics.

To improve the performance of the KHA, a modified krill algorithm based on multi-agent systems is proposed named Multi-Agent Krill Herd Algorithm (MA-KHA). The effectiveness of the proposed algorithms are examined using several benchmark global optimization problems. The MA-KHA is then utilized for the MWSN deployment problem and its performance is compared with other algorithms.

1.2 Contributions

The contributions of this thesis are as follows.

- A modified krill algorithm is proposed using multi-agent systems named MA-KHA. It is studied and compared with the KHA using several benchmark global optimization problems.
- The proposed MA-KHA is used to optimize MWSN sensor deployment. The results are compared with those obtained using PSO, FA, and KHA optimization methods.

1.3 Thesis Outline

The remainder of this thesis is organized as follows.

- Chapter 2 introduces deployment algorithms for MWSNs as well as the corresponding design factors. The fundamental concepts and algorithms used

throughout the thesis are reviewed (deployment algorithms, metaheuristic and SI algorithms, Genetic Algorithm (GA), PSO, FA, and KHA). The related work in the literature is also reviewed.

- Chapter 3 starts with an explanation of multi-agent systems and the corresponding design aspects. An agent-based modified KHA is then presented and compared with the original KHA using several benchmark global optimization problems.

- Chapter 4 considers the solution of the MWSN deployment problem using SI algorithms. Simulation results using MA-KHA for MWSN deployment are presented and compared with the corresponding results using the FA, PSO, and original KHA.

- In Chapter 5, some conclusions are drawn followed by suggestions for future work.

Chapter 2

Background

In this chapter, the foundations of the systems and algorithms used in this thesis are presented. Related work regarding swarm optimization algorithms and WSN optimization are also reviewed.

2.1 Sensor Deployment in WSNs

The coverage of WSNs is a direct result of the sensor deployment. This section presents the most popular deployment algorithms in the literature.

2.1.1 WSN Design Factors

2.1.1.1 Sensing Model

A sensor is a device that measures changes in a physical condition of the environment. The Sensing Field (SF) of a wireless sensor network is the area covered by the sensors. The sensing model describes the probability of target (or event) detection by the sensor. It is a function of the distance between the target and sensor. Two sensing models have been discussed in the literature: binary sensing and probabilistic sensing

[9]. In this thesis, a binary sensing model is used. This means assuming target j is at a point x_j within the SF and r_s is the sensing range of sensor S_i, the target is detected by S_i only if it is at a distance r_s or less from the sensor. The probability of target detection by sensor S_i is given by

$$P_{ij} = \begin{cases} 1 & \text{if } d_{ij} \leq r_s \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

There are three basic assumptions considered in this thesis.

- Sensors have the same sensing ability with the same sensing range.
- Sensors are interconnected, which means they exchange information with each other.
- Sensors can move.

2.1.1.2 Sensor Mobility and Coverage

Wireless sensor networks can be categorized into two types: static WSNs (referred to as just WSNs), and Mobile WSNs (MWSNs). MWSNs contain sensors with locomotive capability in addition to sensing, processing, and communication functions. Mobility provides sensors with the ability to self-deploy after a random initial deployment.

Coverage is a primary performance metric for WSN deployment. There are three types of WSN coverage: area coverage, point coverage, and barrier coverage [10]. The main objective of area coverage is to cover the entire sensing field. In point coverage the objective is to cover targets with known locations which can be considered as a special case of the area coverage problem. Barrier coverage uses a number of barriers to block parts of the SF. In this thesis, area coverage is studied with the goal of

achieving maximum coverage using mobile wireless sensor nodes.

2.1.2 Deployment Algorithms

WSN deployment algorithms can be categorized into four types: random, planned, incremental, and movement-assisted. The first three are static deployment methods, while the last is a dynamic approach.

Random deployment is used when the SF is inaccessible or when no prior knowledge is available (e.g., disaster zones and military applications). It is also utilized in the initial phase of movement-assisted deployment strategies where the locations of sensors are adjusted based on the outcome of the random placement [5]. When the SF is reachable, planned deployment is used in which case the locations of the senors are determined simultaneously. Incremental deployment strategies use a one-at-a-time centralized approach to place the sensors [11]. Each node determines its location using information provided by the previously deployed nodes. The merit of this algorithm is that it can optimize the node locations in each step, but the network initialization time is lengthy because the sensors are deployed iteratively.

Random and planned deployment methods suffer from inaccuracy because control over the actual sensor locations is limited. Furthermore, incremental deployment methods are complex and time-consuming. For this reason, movement-assisted deployment algorithms have been proposed. These algorithms place sensors by optimizing one or more WSN design objectives under specific application constraints. Typical objectives are maximizing the coverage, minimizing the power consumption, and reliable network connectivity. In moving-assisted deployment algorithms, sensors are first deployed randomly and then moved using knowledge of other node locations. The main moving-assisted approaches are given below.

- Computational Geometry-based (CG) algorithms, including the Voronoi Dia-

gram (VD) approach [12] and the Isometric GRID-based (ISOGRID) algorithm [13].

- Virtual Forced-based Approach (VFA) [14], [15].
- Artificial Potential Field (APF) technique [16].
- SI algorithms, which will be discussed in Section 2.2.3.2 [17]–[28].

In this thesis, the focus is on SI algorithms as they are capable of solving complex optimization problems effectively and efficiently.

2.2 Metaheuristic and Swarm Optimization

Optimization methods can be categorized in several ways. From one perspective, they are trajectory-based or population-based. Trajectory-based algorithms follow only one path. Hill-climbing and Simulated Annealing (SA) are examples of trajectory-based optimization approaches. On the contrary, population-based algorithms such as PSO employ multiple elements (called particles in PSO), to examine numerous paths simultaneously.

Optimization algorithms can be divided into deterministic or stochastic algorithms. Stochastic algorithms employ randomness while deterministic algorithms do not. Stochastic algorithms may provide different solutions each time they are executed, even with the same initial values. Genetic algorithms and PSO are examples of stochastic algorithms.

The algorithm search capabilities can be used for classification into local and global search algorithms. There are also mixed type methods, called hybrid algorithms, which utilize a combination of these characteristics [29].

2.2.1 Metaheuristic Algorithms

In general, heuristic means to discover using trial and error, and meta means higher level. In metaheuristic algorithms, a trade-off between diversification (global search) and intensification (local search) is utilized. These algorithms can provide solutions to difficult optimization problems in an acceptable amount of time [7]. However, there is no guarantee that the solutions found are optimal and convergence for most algorithms is not assured. There are two major components in any metaheuristic algorithm: exploration and exploitation. Exploitation (or intensification) means to locally focus on the search region of the current solutions, while exploration (also called diversification) generates diverse solutions by exploring the entire search space. The former ensures that the solutions found are locally the best, while the latter enables the algorithm to escape from local optima to increase the diversity of solutions. A proper balance between the two components can greatly affect the performance of the algorithm [30].

2.2.2 Swarm Intelligence

SI is a branch of metaheuristic search algorithms. SI was first used with cellular robotic systems [31] and has become increasingly popular over the last decade. It has found numerous applications in optimization (swarm optimization), science, and engineering.

SI algorithms focus on the collective behavior of the individuals and their interactions with the environment and each other. For example, SI mimics the foraging, mating, nest-building, and clustering of social groups like insects, colonies of ants, flocks of birds, and herds of animals.

All SI algorithms have the following properties:

- they comprise many individuals that are relatively homogeneous, and
- individuals interact based on simple behavioral rules.

The first metaheuristic algorithm were developed based on Evolutionary Strategies (ES) during the 1960s [32], [33]. Genetic Algorithms (GAs) were developed in the 1970s [34]. During the 1980s, Simulated Annealing (SA) [35] (inspired by the annealing process of metals) and Tabu Search (TS) were developed. TS was the first metaheuristic algorithm to use memory [36]. Ant Colony Optimization (ACO) [37] was introduced in 1992. This algorithm was inspired by the social interaction of ants using pheromones. In the same year, genetic programming was developed which laid the basis for machine learning [38]. In 1995, PSO was developed [39] and a year later, Differential Evolution (DE) was introduced [40]. The latter is a vector-based evolutionary algorithm and has been shown to outperform GAs in many applications. At the beginning of the 21st century, several metaheuristics algorithms were developed including the music-inspired Harmony Search (HS) algorithm [41], the honey bee algorithm [42], and the artificial bee colony algorithm [43]. In 2008, the firefly algorithm was introduced [44] followed by the efficient Cuckoo Search (CS) algorithm in 2009 [45]. In 2011, the bio-inspired KHA was introduced which competes with the best SI algorithms [46]. Table I gives a chronologically ordered list of metaheuristic algorithms. Figure 1 shows the classification of metaheuristic algorithms.

Every swarm optimization algorithm has a n-dimensional search space A^n. Let $A^n \subseteq R^n$, where R^n is the n-dimensional real space. Assuming a swarm population size of N, the swarm population (called population in general) is defined as

$$X = (x_1, x_2, \ldots, x_i, \ldots, x_N) \tag{2.2}$$

where x_i is the ith population element (called particle in PSO, firefly in FA, krill in

TABLE I. History of Metaheuristic Algorithms

Year	**Algorithm**
1963	Evolution strategies [33]
1966	Evolutionary programming [34]
1975	Genetic algorithms [34]
1983	Simulated annealing [35]
1986	Tabu search [36]
1992	Ant colony optimization [37]
1992	Genetic programming [38]
1995	Particle swarm optimization [39]
1996	Differential evolution [40]
2001	Harmony search algorithm [41]
2004	Honey bee algorithm [42]
2005	Artificial bee colony [43]
2008	Firefly algorithm [44]
2010	Cuckoo search [45]
2011	Krill herd algorithm [46]

KHA, and agent in a MAS) as the ith solution to the optimization problem given by

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{in}) \in A^n, i = 1, 2, \ldots, N \tag{2.3}$$

where x_{ij}, $j = 1, 2, \ldots, n$ is the jth element of x_i. A swarm optimization algorithm also needs an objective function $f()$ to measure the performance of the solutions. A minimization problem is then given by

$$\min_x \{f(x) | x \in X\} \tag{2.4}$$

where x is the population element variable. A solution x^* is said to be the global best solution for the minimization problem given above only if $f(x^*) \leq f(x)$ for all $x \in X$.

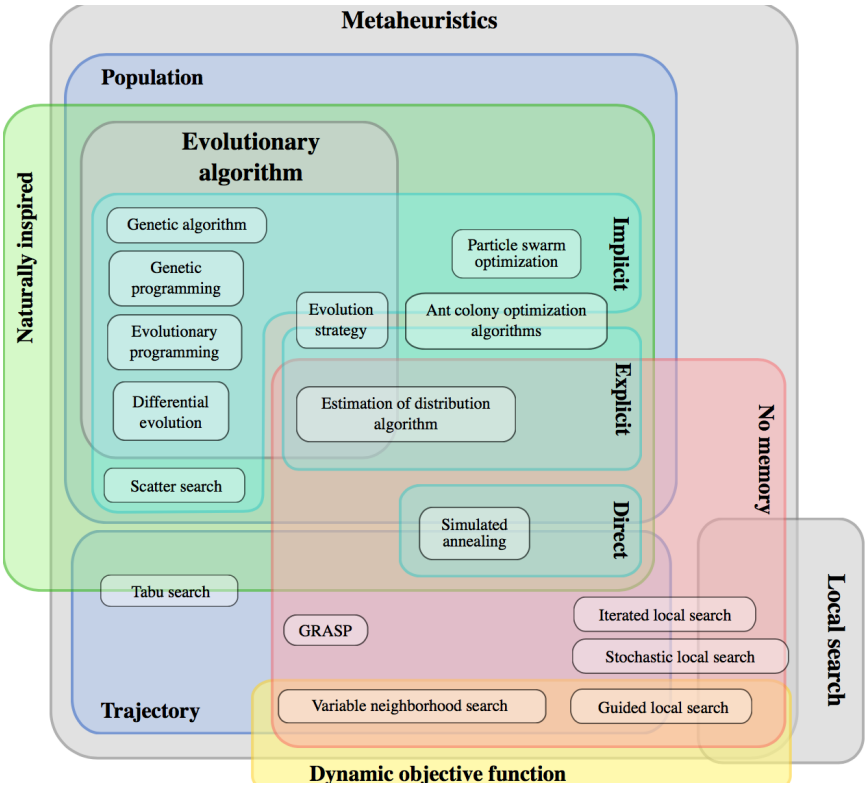


Figure 1. Classification of metaheuristic algorithms*.

2.2.2.1 Genetic Algorithms

GAs are optimization algorithms based on the mechanism of natural selection in evolution [47]. The use of crossover, recombination, mutation, and selection was proposed for the study of artificial systems in the 1970s. Since then, GAs have been used to solve both discrete and continuous optimization problems. These algorithms have been shown to be very efficient in solving multi-objective optimization problems for which deterministic optimization methods are not practical [48]. Many modern evolutionary algorithms have been inspired by GAs. All variants of genetic algorithms have the following three essential components [49]:

- encoding, which is a genetic representation of the candidate solutions,
- a fitness function (or cost function), which is a mathematical expression of the

*http:metah.nojhan.netpost200710/12Classification-of-metaheuristics

optimization objective, and

- stochastic genetic operators (crossover and mutation), to change the composition of the offspring. Crossover swaps a portion of two binary strings whereas mutation randomly alters the entries of a binary string.

There are also five steps in a GA [49].

1. Creating an initial population: an initial population of individuals (also called chromosomes) is created so that the search space of the problem is uniformly covered.
2. Population evaluation using the fitness function: individuals in the population are evaluated using the fitness function. The weakest individuals are eliminated from the population.
3. Parent selection: parent chromosomes are selected.
4. Offspring production: the genetic operators are applied to the chosen parents to produce an offspring population. Crossover is the primary genetic operator which randomly pairs parents to produce offspring. Mutation generates a new individual by randomly altering part of a selected parent. The offspring population is then evaluated using the fitness function.
5. Final selection: the best individuals from both the parent and offspring populations are chosen to form a new population.

Each iteration of steps 2 to 5 (also called a generation), results in a new population. By repeating these steps the algorithm converges to a population which hopefully represents an optimum solution to the problem. The termination conditions can be determined by several factors including Number of Function Evaluations (NFE), number of iterations, time, or reaching a specific solution or threshold. In this thesis,

both NFE and number of iterations are used as the termination conditions for all algorithms. The GA pseudo-code is shown in Algorithm 1.

Step	Algorithm 1 Genetic Algorithm
1.	Set $t = 0$
2.	Initialize the parameters and a random initial population X_0
3.	Evaluate X_0 with the fitness function $f()$
4.	**while** (termination conditions are not met) do
5.	Apply the genetic operators to the population
6.	Evaluate the resulting population with $f()$
7.	Select and update population X
8.	$t = t + 1$
9.	**end while**

2.2.2.2 Particle Swarm Optimization

PSO is a stochastic optimization method based on models of fish school and bird flock movements. The idea is to utilize the swarm population (particles) that move stochastically across the search space of the optimization problem.

The mathematical framework of PSO is as follows. Each particle retains its best position during the search and is denoted by $m_i = (m_{i1}, m_{i2}, \ldots, m_{in}) \in A^n$ for particle i. The set $M = (m_1, m_2, \ldots, m_N)$ contains the best positions of the particles. The best position of all particles is called the global best and is denoted by $g = (g_1, g_2, \ldots, g_n) \in A^n$. The global best and best positions play a crucial role in PSO to move the particles in the search space. The other parameter is the velocity $v_i = (v_{i1}, v_{21}..., v_{in}), i = 1, 2, \ldots, N$ which updates the particle positions in each iteration. The velocity of particle i is defined as

$$v_i(t+1) = wv_i(t) + c_1R_1\big(m_i(t) - x_i(t)\big) + c_2R_2\big(g(t) - x_i(t)\big) \tag{2.5}$$

where w is the inertia weight, R_1 and R_2 are random variables which are uniformly

distributed between 0 and 1, and c_1 and c_2 are the weights of the cognitive and social components, respectively. Note that t and $t+1$ represent the current and next iterations, respectively. The velocity is obtained by combining the cognitive component (the distance of particle i from its best position m_i), the social component (the distance of particle i from the global best g), and the current value of the velocity. The updated particle in the next iteration is given by

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{2.6}$$

where $x_i(t)$ and $x_i(t+1)$ are particle i in the current and next iterations, respectively, and $v_i(t+1)$ is the velocity of particle i at iteration $t+1$.

Algorithm 2 gives the PSO algorithm in pseudo-code. When the termination conditions are met, the global best particle g in the last updated population is the solution of the PSO algorithm.

Step	**Algorithm 2 Particle Swarm Optimization**
1.	Set $t = 0$
2.	Set inertia weight, personal learning, and global learning coefficients
3.	Initialize a random population X_0 using (2.2) and (2.3), and set $M = X_0$
4.	Evaluate X_0 with the fitness function $f()$ and define g as the global best position
5.	**while** (termination condition not met) do
6.	Update population X for each particle using (2.5) and (2.6)
7.	Evaluate X, update M, and recalculate g
8.	$t = t + 1$
9.	**end while**

2.2.2.3 Firefly Algorithm

The FA [44] is a stochastic SI algorithm. It is based on the following three fundamental rules:

- fireflies are attracted to each other regardless of their sex,

- attractiveness is proportional to the brightness and decreases as the distance from other fireflies increases, and
- the brightness of a firefly is defined by its performance based on the objective function.

The main characteristic of fireflies is their flashing light. These lights are courtship signals for the purpose of mating, and fireflies prefer the ones that are brighter [50]. In the FA, the movement of fireflies is a function of the flashing light behavior of others, which means fireflies move in the direction of brighter fireflies (local search). Randomization enables the FA to explore the search space and helps avoid being trapped in local minima [51].

2.2.2.3.1 Formulation and Implementation of the Firefly Algorithm

The firefly algorithm is based on the light intensity which follows an inverse square law. Attractiveness is a relative measure of the light from the perspective of the other fireflies and is defined as

$$\beta = \beta_0 e^{-\gamma r_{i,j}^2} \tag{2.7}$$

where β_0 is the initial attractiveness at the source ($r = 0$), γ is the light absorption coefficient, and β is the firefly attractiveness at distance $r_{i,j}$, which is the distance between fireflies i and j given by

(2.8)

The movement of the ith firefly towards a brighter firefly j is formulated as

$$x_i(t+1) = x_i(t) + \beta(x_j(t) - x_i(t)) + \alpha\epsilon_i \quad \text{if} \quad L_j \geq L_i \tag{2.9}$$

where L_i and L_j are the light intensities of ith and jth fireflies, respectively. The light intensity of firefly i representing the solution x_i is given by $L_i = f(x_i)$. Note that

for a minimization problem, a brighter firefly corresponds to a solution with lower fitness. ϵ_i is a random value that can be drawn from a Gaussian, Levy, or uniform distribution, and α is the random walk coefficient. In order to have a good global search at the beginning of the iterative process and a better local search in later iterations, a damping parameter α_{damp} is defined (step 12 in Algorithm 3) [50].

The parameter γ has a crucial impact on the convergence of the algorithm and typically is between 0.1 to 10 [44], depending on the problem. Following the mathematical framework in (2.2) and (2.3) for population initialization, Algorithm 3 gives the steps of the FA in pseudo code.

Step	**Algorithm 3 Firefly Algorithm**
1.	Set $t = 0$
1.	Set light absorption, attractiveness, and random walk coefficients
2.	Initialize a population X_0 using (2.2) and (2.3),
	evaluate it using the fitness function $f()$
3.	**while** (termination condition not met) do
4.	for $i = 1$ to N
5.	for $j = 1$ to N
6.	if $(L_j \geq L_i)$
7.	Move firefly i towards the brighter firefly j using (2.9)
8.	end if
9.	end for
10.	end for
11.	Evaluate population X with the fitness function $f()$
12.	Determine a new value for the random walk coefficient using $\alpha = \alpha_{damp}\alpha$
13.	$t = t + 1$
14.	**end while**

2.2.2.4 Krill Herd Algorithm

The grouping behavior of many marine animals is non-random. This has led to the investigation of the underlying mechanisms of these creatures, including feeding behavior, reproduction, and defense and protection [52]. Among marine animals, Antarctic krill have been studied extensively, but there are many uncertainties about

the factors that determine krill herding. However, conceptual frameworks have been proposed to explain the ecology, distribution, and formation of krill herds [53], [54]. The KHA is a biologically-inspired metaheuristic optimization algorithm which is based on krill herding behavior. It has been shown to outperform several state-of-the-art SI algorithms [46], [55]–[58].

2.2.2.4.1 The Lagrangian Model of Krill Herding

The fitness function in KHA is a multi-objective function which is used to minimize the distance of each krill from the food location and the highest density of the herd. Depending on the value of the objective function, the position of each krill is governed by the following three actions [46]:

1. induced movement by the other krill (I_i),
2. foraging activity (F_i), and
3. random (or physical) diffusion (D_i).

For the ith krill $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ in an n-dimensional search space, the krill movement is defined by a Lagrangian model given by

$$\frac{dx_i}{dt} = I_i + F_i + D_i \tag{2.10}$$

The details of each motion are as follows.

1. Induced motion: For the ith krill, the induced motion is given by

$$I_i = I^{\max}\alpha_i + \omega_n I_i^{old} \tag{2.11}$$

where $I^{\max}$ is the maximum induced speed, ω_n is the inertia weight in the range [0,1],

and I_i^{old} is the previous induced movement. The direction of motion α_i is defined as

$$\alpha_i = \alpha_i^{local} + \alpha_i^{target} \tag{2.12}$$

where α_i^{local} and α_i^{target} are the local effect provided by the neighbors and the target (global best krill) effect, respectively. The local effect α_i^{local} can be formulated as

$$\alpha_i^{local} = \sum_{j=1}^{NN} \hat{k}_{i,j}\hat{r}_{i,j} \tag{2.13}$$

where NN is the number of neighbors and $\hat{r}_{i,j}$ is the normalized distance of the ith krill from krill j in its neighborhood given by

$$\hat{r}_{i,j} = \frac{x_j - x_i}{\left\|x_j - x_i\right\| + \varepsilon} \tag{2.14}$$

where ε is a small positive number added to the denominator to avoid singularity. $\hat{k}_{i,j}$ denotes the normalized fitness of krill i from krill j defined as

$$\hat{k}_{i,j} = \frac{f(x_i) - f(x_j)}{k^{worst} - k^{best}} \tag{2.15}$$

where k^{best} is the lowest fitness belonging to the global best krill given by $k^{best} = f(g)$ and k^{worst} is the highest fitness belonging to the worst krill.

To determine NN in (2.13), a sensing distance d_s is determined around a krill as shown in Figure 2, which is defined as

$$d_{s,i} = \frac{1}{5N} \sum_{j=1}^{N} \left\|x_i - x_j\right\| \tag{2.16}$$

where N is the number of krill (krill population size). Using (2.16), two krill are neighbors if the distance between them is less than the sensing distance.

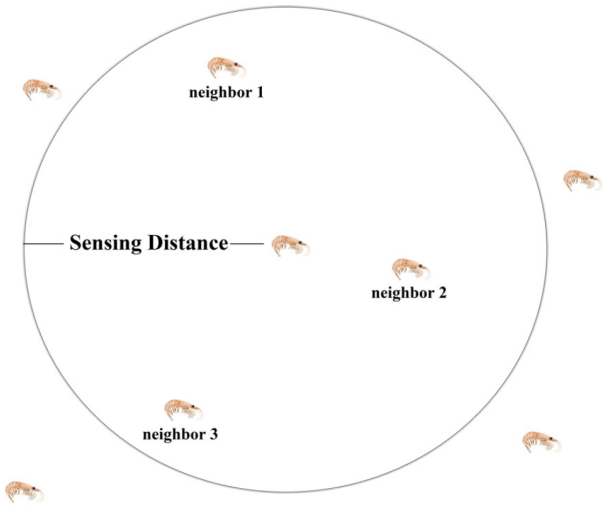


Figure 2. The sensing region around a krill [46].

The effect of the global best krill g on the ith krill is

$$\alpha_i^{target} = C_g \hat{k}_{i,g} \hat{r}_{i,g} \tag{2.17}$$

where $\hat{r}_{i,g}$ is the normalized distance of the ith krill from the global best krill and $\hat{k}_{i,g}$ is the normalized fitness of krill i from the global best krill. C_g is an adaptive coefficient for the target effect and is given by

$$C_g = 2\left(rand(0,1) + \frac{t}{t_{\max}}\right) \tag{2.18}$$

where $rand(0,1)$ is a random value with uniform distribution between 0 and 1. t and $t_{\max}$ are the current and maximum iteration numbers, respectively.

2. Foraging activity: The foraging motion is given by

$$F_i = V_f \beta_i + \omega_f F_i^{old} \tag{2.19}$$

and

$$\beta_i = \beta_i^{food} + \beta_i^{best} \tag{2.20}$$

where V_f is the foraging speed, ω_f is the inertia weight, F_i^{old} is the previous foraging motion, β_i^{food} is the food attraction, and β_i^{best} is the best position effect of the ith krill. The food attraction is given by

$$\beta_i^{food} = C_f \hat{k}_{i,f} \hat{r}_{i,f} \tag{2.21}$$

where $\hat{r}_{i,f}$ is the normalized distance of the ith krill from x_f as the center of the food given by

$$x_f = \frac{\sum_{k=1}^{N} \frac{x_k}{f(x_k)}}{\sum_{k=1}^{N} \frac{1}{f(x_k)}} \tag{2.22}$$

and $\hat{k}_{i,f}$ is the normalized fitness of krill i from the center of the food. C_f is an adaptive coefficient for the food attraction which is given by

$$C_f = 2\left(1 - \frac{t}{t_{\max}}\right) \tag{2.23}$$

Similar to the PSO algorithm, krill i, $i = 1, 2, \ldots, N$, retains its best position during the search denoted by $m_i = (m_{i1}, m_{i2}, \ldots, m_{in})$. The set $M = (m_1, m_2, \ldots, m_N)$ contains the best positions of the krill. The best position effect of the ith krill is then given by

$$\beta_i^{best} = \hat{k}_{i,m_i} \hat{r}_{i,m_i} \tag{2.24}$$

where $\hat{r}_{i,m_i}$ and $\hat{k}_{i,m_i}$ are the normalized distance and fitness of the ith krill from m_i, respectively.

3. Random diffusion: The physical diffusion is a random process given by

$$D_i = D^{\max}\left(1 - \frac{t}{t_{\max}}\right)\delta \tag{2.25}$$

where $D^{\max}$ is the maximum diffusion and δ is uniform random directional vector with elements in range [-1,1]. The ith krill in the next iteration is given by

$$x_i(t+1) = x_i(t) + \Delta t \frac{dx_i}{dt} \tag{2.26}$$

where $\frac{dx_i}{dt}$ was given in (2.10) and

$$\Delta t = C_t \sum_{j=1}^{n} \left(UB_j - LB_j\right) \tag{2.27}$$

where LB_j and UB_j are the lower and upper bounds of the jth variable, respectively, which are determined by the optimization problem. C_t is the time interval constant in the range [0,2].

2.2.2.4.2 Genetic Operators in Enhanced KHA

To improve the performance of the KHA, crossover and mutation genetic operators can be incorporated into the algorithm [46]. The crossover operator is controlled by the crossover probability

$$C_r = 0.2\hat{k}_{i,m_i} \tag{2.28}$$

To apply crossover to the ith krill $x_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{in})$, a uniformly distributed random vector $rand_i^C(0,1) = (c_{i1}, c_{i2}, \ldots, c_{ij}, \ldots, c_{in})$ with elements in the

range [0,1] is used. Then, applying crossover to the jth element of krill i gives

$$x_{ij} = \begin{cases} x_{rj} & c_{ij} \leq Cr \\ x_{ij} & \text{otherwise} \end{cases} \tag{2.29}$$

which means x_{ij} is replaced by the corresponding element of a uniform randomly chosen krill $x_r = (x_{r1}, x_{r2}, \ldots, x_{rj}, \ldots, x_{rn})$, $r \in \{1, 2, \ldots, i-1, i+1, \ldots, N\}$, if $c_{ij} \leq Cr$.

Adaptive mutation is also applied to the elements of x_i using a random vector $rand_i^M(0,1) = (mu_{i1}, mu_{i2}, \ldots, mu_{ij}, \ldots, mu_{in})$ with uniform distribution and elements in the range [0,1]. By applying mutation to the jth element of x_i, it is changed to

$$x_{ij} = \begin{cases} g_j + \mu(x_{pj} - x_{qj}) & mu_{ij} < Mu \\ x_{ij} & \text{otherwise} \end{cases} \tag{2.30}$$

where mu_{ij} is the jth element of $rand_i^M(0,1)$, and x_{pj} and x_{qj} are the jth elements of two uniform randomly chosen krill, respectively, where $p, q \in \{1, 2, \ldots, i-1, i+1, \ldots, N\}$. g_j is the jth element of the global best krill and μ is a random coefficient in the range [0,1]. The mutation operator is controlled by a mutation probability given by

$$Mu = 0.05/\hat{k}_{i,m_i} \tag{2.31}$$

Assuming it is a minimization problem, $\hat{k}_{i,m_i}$ for both C_r and Mu in (2.28) and (2.31) is

$$\hat{k}_{i,m_i} = \frac{f(x_i) - f(m_i)}{k^{worst} - k^{best}} \tag{2.32}$$

In [46], four KHA types were proposed:

- KH I: KHA without any genetic operators,

- KH II: KHA with just the crossover operator,
- KH III: KHA with just the mutation operator, and
- KH IV: KHA with both crossover and mutation operators.

Following the same mathematical framework of population initialization as in (2.2) and (2.3), Algorithm 4 describes the KH algorithms.

Step	Algorithm 4 Krill Herd Algorithm
1.	Set $t = 0$
2.	Set the maximum diffusion and induced speeds as well as the foraging speed
3.	Initialize a random population X_0 using (2.2) and (2.3) and set $M = X_0$
4.	Evaluate X_0 with the fitness function $f()$ and set g as the global best krill
5.	**while** (termination condition not met) do:
6.	Motion calculation according to (2.10), (2.11), (2.19), and (2.25)
7.	Update the krill population (2.26)
8.	Apply the genetic operators for KHA II, KHA III, or KHA IV
9.	Update and evaluate the krill population X with the fitness function $f()$
10.	Update M and recalculate g
10.	$t = t + 1$
11.	**end while**

2.2.3 Related Work

2.2.3.1 KHA Variants, Adjustments, and Applications

Several KHA variants have been proposed, including discrete krill herd [59], binary krill herd [60], fuzzy krill herd [61], and multi-objective krill herd [62]. Moreover, several adjustments have been made to the KH algorithm to obtain improved and hybrid schemes [63]. In [64], chaos theory was used in the KHA optimization process and is called CKH. An improved version of KHA with opposition-based learning was proposed in [65]. An improved krill herd algorithm was introduced in [66] using a new Levy flight distribution and elitism scheme to update the motion calculation. In [67], a multi-stage krill herd algorithm was proposed in which separate exploration

and exploitation stages were employed. An efficient Stud Krill Herd (SKH) method which combines KHA with the Stud Genetic Algorithm (SGA) was proposed in [68]. In this approach, instead of using stochastic selection, the best individual (stud) provides its direction information to the other krill with the help of GA.

Several adaptive variants of KHA have been proposed in the literature. In [69], an adaptive technique was proposed which changes the positions of current solutions towards the global optimum according to the fitness function. A hybrid metaheuristic algorithm called CSKH, which is a combination of cuckoo search and KHA, was introduced in [70]. In [71], KHA with a migration operator was employed for Biography-Based Optimization (BBO). The idea of differential evolution was incorporated into KHA (DEKH) in [72]. In [73], quantum-behavior PSO was proposed in combination with KHA. In [74], several global optimization problems were solved with a hybrid simulated annealing-based KHA. A hybrid Monkey Algorithm (MA) was proposed for KHA (MAKHA) in [75]. To combine the advantages of FA and KHA, a firefly-based hybrid krill algorithm was suggested in [76]. HSKH is another hybrid method that incorporates harmony search into KHA [77].

2.2.3.2 WSN Swarm Optimization

In this section, related work in the area of WSN optimization is reviewed, especially coverage maximization using swarm optimization algorithms. As mentioned before, metaheuristic and SI algorithms have been widely used for sensor deployment. For instance, genetic algorithms in [17]–[19] and PSO in [20], [21] were used to determine the coverage. In [22], the firefly algorithm was used to solve the MWSN coverage problem.

The KHA was used in [23] to maximize the sensor network lifetime for clustering algorithms. In [24], KHA was employed to select cluster heads to efficiently decrease

cluster energy consumption and balance the network energy consumption. The KHA has not yet been considered for the MWSN coverage problem.

There are several WSN coverage optimization studies that employed metaheuristic algorithms. In [25] and [26], artificial bee colony was proposed for WSN deployment, and ant colony optimization was proposed to deploy a grid-based static WSN in [27]. In [28], a probabilistic sensing model for sensors with line-of-sight-based coverage was proposed to solve the sensor placement problem. Three schemes were proposed to optimize the deployment of static WSNs, including simulated annealing, the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, and covariance matrix adaptation evolution.

Chapter 3

Multi-Agent Krill Herd Algorithm

KHA was designed based on the imitation of krill herding behavior. The global optimization analysis in [46] showed that this algorithm performs well compared to eight other algorithms. However, recent studies have shown that KHA has some weaknesses [78]. In this chapter, these weaknesses are discussed and the proposed multi-agent krill herd algorithm (MA-KHA) is introduced. We also show that it can perform better than KHA in solving global optimization problems.

3.1 Multi-Agent Systems

Agents are autonomous entities that act on the environment and direct their activities towards achieving specific goals. Multi-Agent Systems (MASs) are computational systems that consist of multiple agents and their environment. An agent in a MAS could be software, a robot, or a human, with three basic characteristics.

1. Autonomy: agents are autonomous and self-aware.
2. Local view: agents do not have an understanding of the entire search space.
3. Decentralization: there is no designated controlling agent.

Multi-agent systems have self-organization as well as self-direction capabilities which allows them to solve difficult problems.

Agent-based computation has been employed in the field of distributed artificial intelligence and other branches of computer science [79]–[82]. According to [82] and [83], agents exist in a lattice-like environment with the following properties.

- Agents sense their external environment and interact with neighboring agents.
- Agents are designed to achieve particular goals for specific purposes.
- Agents have reactive behavior meaning that they can respond to changes based on their learning ability.

To obtain a solution to a problem, agents cooperate and compete with their neighbors simultaneously. Using agent-agent interactions, a MAS can find good solutions with fast convergence [84]. To achieve this, the lattice and local environment should be defined as well as the behavioral rules. Agents interact with their neighbors to diffuse information to the entire lattice.

3.1.1 Definition of the Lattice and Local Environment

In a MAS, every agent is a candidate solution and has a fitness value for the optimization problem. Agents exist in a lattice-like environment and lie on lattice points. In Figure 3, each circle represents an agent, and the numbers in the circles denote the positions of the agents.

The agent lattice has size $L = L_{size} \times L_{size}$, where L_{size} is an integer and L is the total number of elements in the optimization algorithm (e.g. krill in KHA, particles in PSO, or fireflies in FA). If agent $\alpha_{ij}, i, j = 1, 2, \ldots, L_{size}$, is located at lattice-point

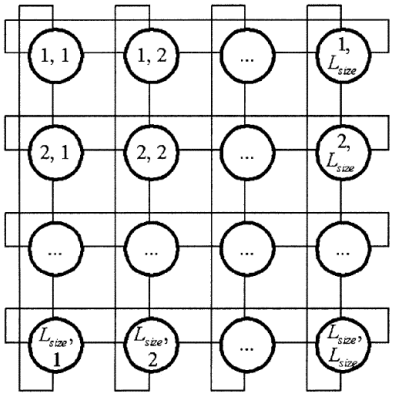


Figure 3. The lattice environment of multi-agent systems.

(i, j), then its neighbors are defined as $N_{ij} = \{\alpha_{i^1 j}, \alpha_{ij^1}, \alpha_{i^2 j}, \alpha_{ij^2}\}$, where

$$i^1 = \begin{cases} i-1 & i \neq 1 \\ L_{size} & i = 1 \end{cases} \qquad j^1 = \begin{cases} j-1 & j \neq 1 \\ L_{size} & j = 1 \end{cases}$$

$$i^2 = \begin{cases} i+1 & i \neq L_{size} \\ 1 & i = L_{size} \end{cases} \qquad j^2 = \begin{cases} j+1 & j \neq L_{size} \\ 1 & j = L_{size} \end{cases} \tag{3.1}$$

Hence, the local environment is defined as the four neighbors around an agent. For instance, in a lattice environment of size 25 ($L_{size} = 5$), the four neighbors of agent α_{11} are $N_{11} = \{\alpha_{51}, \alpha_{15}, \alpha_{21}, \alpha_{12}\}$.

3.2 Multi-Agent Design for KHA Optimization

This section explains how KHA and MAS are integrated to form a multi-agent krill herd algorithm. In MA-KHA, agents compete and cooperate with their neighbors in a local environment. Using KHA, the information exchange among agents in the lattice is accelerated. Finally, self-learning is applied to the best agent found to strengthen the local search.

3.2.1 Agent Behavioral Strategies

Each agent in MA-KHA has some specific behavior and aims to diffuse its information to the lattice. To do so, three operators are employed, namely competition and cooperation, KHA, and self-learning. These operators are explained below.

3.2.1.1 Competition and Cooperation Operator

With this operator, each agent interacts with its neighbors in a form of competition and cooperation. Suppose that agent $\alpha_{ij} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ is located at lattice-point (i, j) and $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ is the best local agent with minimum fitness (for a minimization problem), in the neighborhood given by

$$\forall \varepsilon \in N_{ij} \rightarrow f(\beta) \leq f(\varepsilon) \tag{3.2}$$

where ε is any of the four neighbors of α_{ij} and $f()$ is the fitness function of the minimization problem. Agent $\alpha_{i,j}$ is the winner in its neighborhood if it satisfies

$$f(\alpha_{ij}) \leq f(\beta) \tag{3.3}$$

In this case, $\alpha_{i,j}$ remains unchanged and its location in the search space will not be changed. Otherwise, it is moved towards the best local agent β. The new agent $New_{ij} = (\alpha'_1, \alpha'_2, \ldots, \alpha'_n)$ is determined using one of the following two strategies [84].

- Strategy I:

  With this strategy, a heuristic crossover is applied to each element of α_{ij} to get New_{ij}. If α_k is the kth element of α_{ij}, the kth element of $New_{i,j}$ is

$$\alpha'_k = \beta_k + rand(0, 1)(\beta_k - \alpha_k) \quad k = 1, 2, \ldots, n \tag{3.4}$$

where $rand(0,1)$ is a random function with uniform distribution in the range [0,1]. Then, boundary checking is done on α'_k, $k = 1,2, \ldots, n$, to make sure it is still within the range

$$\alpha'_k \leq x_k^{\min}, \quad \text{then} \quad \alpha'_k = x_k^{\min} \tag{3.5}$$

$$\alpha'_k \geq x_k^{\max}, \quad \text{then} \quad \alpha'_k = x_k^{\max} \tag{3.6}$$

where $x^{\min} = (x_1^{\min}, x_2^{\min}, \ldots, x_n^{\min})$ is the lower bound vector, and $x^{\max} = (x_1^{\max}, x_2^{\max}, \ldots, x_n^{\max})$ is the upper bound vector of the search space defined by the optimization problem.

- Strategy II:

  In this strategy a metaheuristic mutation updates the elements of α_{ij} to obtain New_{ij}. First, vector $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$ which was defined above is normalized as

$$\beta'_k = \frac{\beta_k - x_k^{\min}}{x_k^{\max} - x_k^{\min}} \quad k = 1, 2, \ldots, n \tag{3.7}$$

  Then, using a uniform distribution two integers i_1 and i_2 are randomly chosen such that

$$1 \leq i_1 \leq n, \quad 1 \leq i_2 \leq n, \quad \text{and} \quad i_1 \leq i_2$$

  New'_{ij}= $(\gamma'_1, \gamma'_2, \ldots, \gamma'_n)$ is determined as

$$New'_{ij} = (\beta'_1, \ldots, \beta'_{i_1-1}, \beta'_{i_2}, \beta'_{i_2-1}, \ldots, \beta'_{i_1+1}, \beta'_{i_1}, \beta'_{i_2+1}, \beta'_{i_2+2}, \ldots, \beta'_n) \tag{3.8}$$

  and

$$\alpha'_k = x_k^{\min} + \gamma'_k(x_k^{\max} - x_k^{\min}) \quad k = 1, 2, \ldots, n \tag{3.9}$$

  Using (3.9) $New_{ij} = (\alpha'_1, \alpha'_2, \ldots, \alpha'_n)$ is obtained from New'_{ij}.

Strategy I employs a deep search and emphasizes exploitation, whereas strategy II emphasizes exploration. Strategy II is employed more at the beginning of the iterative process to better explore the search space. Later, strategy I is used more to have a better local search. This is done using P_c and P_m which are the probabilities of employing the first and second strategies, respectively.

KHA provides excellent local search by considering various motion characteristics. However, studies have shown that the global exploration capability of KHA is not as effective [68]. For this reason, KHA might become trapped in a local minima for some optimization problems, e.g. the multi-modal fitness landscape [85]. In [46], an attempt was made to solve this issue by adding crossover and mutation operators to the basic KHA (KHA I). However, the integrity of the system can be degraded if there is a poor balance between exploration and exploration. In MA-KHA, instead of having crossover and mutation, the global search is reinforced by initially using strategy II more (P_m large and P_c small). Then to regain the balance, the local search is fortified by using strategy I more (P_m small and P_c large) in later iterations and the self-learning operator (explained in Section 3.2.1.3).

3.2.1.2 Integrated KHA Operator

The krill herd algorithm not only optimizes the objective function but can also speed up the information exchange between agents. Since agents can only sense their local environment, information is transferred from the local environment (N_{ij}) to the entire lattice quite slowly. Thus, KHA can be used to make this process faster.

3.2.1.3 Self-Learning

Each agent can learn from itself to enhance the local search. Motivated by the use of GA in [84] for local search, a small-scale KHA (mini-KHA) is used here for the

Step	Algorithm 5 Self-learning Operator in KHA
1.	Set the global best agent of the KHA operator as $\alpha_{ij}^g = g(t)$
2.	Initialize $sNew\alpha$ as the $sL-1$ mini-agents within a radius sR around α_{ij}^g
3.	Form the self-learning mini-lattice $s\alpha$ of size sL
4.	Perform the competition and cooperation operation on the mini-agent lattice
5.	Perform mini-KHA on the mini-agent lattice
6.	Evaluate each mini-agent with the fitness function to yield a solution η^g
7.	Update $g(t+1) = \eta^g$ if $f(\eta^g) \leq f(\alpha_{ij}^g)$

self-learning operator. It has a lattice containing $sL = sL_{size} \times sL_{size}$ mini-agents, where sL is the number of mini-agents in the self-learning lattice and sL_{size} is a small integer. The mini-agents are defined as

$$s\alpha = \begin{cases} \alpha_{ij}^g & k = 1 \\ sNew\alpha_k & k = 2, 3, \ldots, sL \end{cases} \tag{3.10}$$

where $\alpha_{ij}^g = \{\alpha_1^g, \alpha_2^g, \ldots, \alpha_n^g\}$ is the global best krill $\alpha_{i,j}^g = g(t)$ located at (i, j), and $sNew\alpha_k = (e_{k1}, e_{k2}, \ldots, e_{kn})$ is the kth mini-agent around $\alpha_{i,j}^g$ with radius sR. The jth, $j = 1, 2, \ldots, n$, element of $sNew\alpha_k$ is given by

$$e_{kj} = \begin{cases} x_j^{\min} & \alpha_j^g \times rand(1 - sR, 1 + sR) < x_j^{\min} \\ x_j^{\max} & \alpha_j^g \times rand(1 - sR, 1 + sR) > x_j^{\max} \\ \alpha_j^g \times rand(1 - sR, 1 + sR) & \text{otherwise} \end{cases} \tag{3.11}$$

Thus, the self-learning mini-lattice contains the best krill $\alpha_{i,j}^g$ and $sL-1$ mini-agents.

After forming the mini-lattice, competition and cooperation is iteratively performed on the mini-agents followed by mini-KHA. If an agent with a lower fitness than α_{ij}^g is found during self-learning, g is replaced by that agent (η^g). Algorithm 5 describes the self-learning procedure.

Figure 4 illustrates MA-KHA with all three operators. In each iteration compe-

tition and cooperation is applied to the population. Using strategies I or II, meta-heuristic mutation and crossover are applied to each agent. Then, KHA is employed to speed up the information exchange between the agents and optimize the problem. The global best agent of the KHA operator is then taken as $\alpha_{i,j}^{g}$ and used with the self-learning operator to enhance the local search.

3.2.2 Simulation and Numerical Results

Benchmark test functions are artificial problems used to assess the behavior of an algorithm. They may contain a single global minimum, several local minima with single or multiple global minima, narrow valleys, flat surfaces, or other shapes. The reliability and efficiency of optimization algorithms can be validated using these functions.

In this section, the proposed algorithm is examined using a set of benchmark test functions. For each benchmark, the characteristics of the test function are explained, and the simulation results of the algorithms are provided. The results for MA-KHA are compared with KHA I, KHA II, and KHA IV. According to Table 1 in [46], algorithms KH I to IV have rank 5, 1, 6, and 2, respectively, among 12 algorithms. In this thesis, the KHA parameters are as follows: foraging speed $V_f = 0.02$ ms^{-1}, maximum diffusion speed $D^{\max} = 0.005$ ms^{-1}, maximum induced speed $I^{\max} = 0.01$ ms^{-1}, time interval constant $C_t = 0.5$, maximum number of iterations $t_{\max} = 250$, population size $N = 25$, and maximum number of function evaluations $NFE_{\max} = 15,000$. Moreover, the motion induced inertia weight (ω_n) and foraging motion inertia weight (ω_f) are both set to 0.9 at the beginning of the search to emphasize exploration. To encourage exploitation, they are linearly decreased to 0.1 as given by

$$\omega_f = \omega_n = 0.1 + 0.8(1 - t/t_{\max}) \tag{3.12}$$

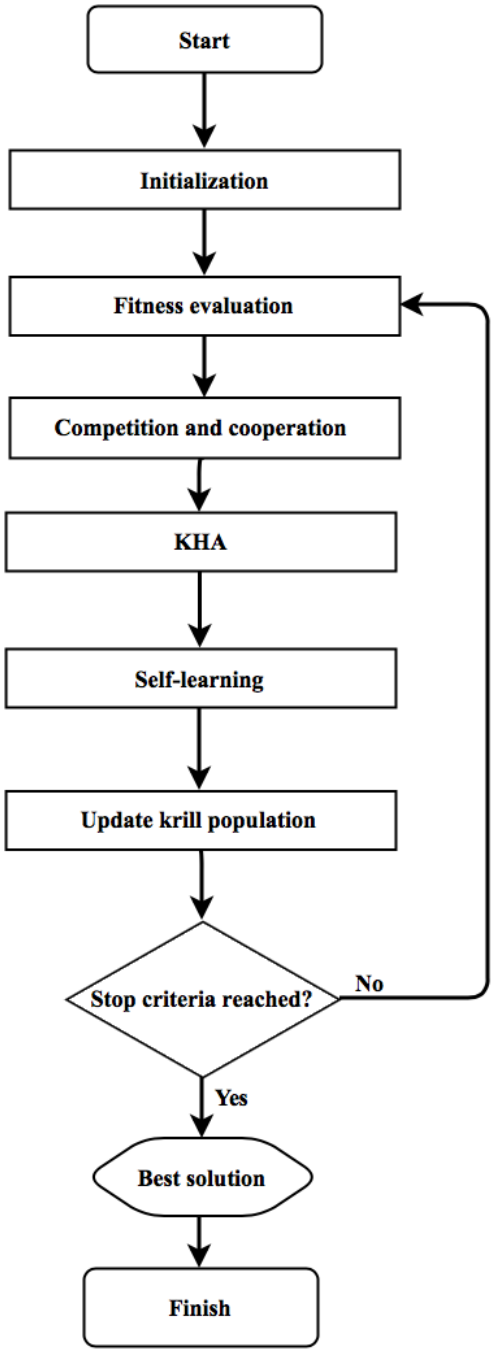


Figure 4. The MA-KHA flowchart.

TABLE II. KHA Parameters

V_f	$D^{\max}$	$I^{\max}$	C_t	$N = L$	L_{size}	$\omega_n = \omega_f$
0.02 ms^{-1}	0.005 ms^{-1}	0.01 ms^{-1}	0.5	25	5	[0.9, 0.1]

TABLE III. MA-KHA Parameters

Competition and Cooperation		Self-Learning					
P_m	P_c	sIt	sL	sL_{size}	sR	sP_m	sP_c
[0.7, 0.2]	[0.2, 0.6]	5	9	3	0.25	[0.7, 0.2]	[0.2, 0.6]

where t and $t_{\max}$ are the iteration and maximum iteration numbers. Table II gives the KHA parameters.

Table III gives the MA-KHA parameters including competition and cooperation (P_c and P_m) and self-learning. sIt is the maximum number of iterations in self-learning. sL and sL_{size} denote the size of the mini-lattice, sR is the radius of the circle that mini-agents are created around $\alpha_{i,j}^{g}$, and sP_c and sP_m are the probability of employing strategies I and II in self-learning, respectively. For the KHA operator in MA-KHA, the parameters given in Table II are used.

The test functions were chosen to have diverse properties. Table IV gives the five benchmark test functions that are used in this thesis. The search space boundaries $[x^{\min}, x^{\max}]$ and dimensions n are given in the third column. They are also given in Appendix A in detail.

Tables IV to VIII present the numerical results for the Ackley, Grewank, Rastrigin, Rosenbrock, and sphere global optimization benchmark functions, respectively. For

TABLE IV. Benchmark Problems

	Benchmark Test Function	Search Space	Global Minima
F1	Ackley	$[-32, 32]^n$	0
F2	Griewank	$[-600, 600]^n$	0
F3	Rastrigin	$[-5.12, 5.12]^n$	0
F4	Rosenbrock	$[-30, 30]^n$	0
F5	Sphere	$[-100, 100]^n$	0

each function, MA-KHA is compared with KHA I, KHA II, and KHA IV for $n = 10$, 20, and 30. Results were obtained for 50 trials for each benchmark function. The best, worst, and average run values with the corresponding Standard Deviation (SD) are given.

The Ackley function [86] is a continuous, non-separable, multimodal global optimization benchmark problem with many minor local minima and one narrow steep global minimum valley. This is considered a challenging problem for algorithms with poor global search capability. The reason is that weak exploration leads to the algorithm getting stuck at a local minima and never reaching the global minimum.

TABLE V. Simulation Results for the Ackley Problem

Function	Dim.	Criteria	KHA I	KHA II	KHA IV	MA-KHA
Ackley	10	best	0.004	4.49E-04	1.00E-03	4.441E-15
		worst	5.899	0.484	0.575	1.51E-14
		average/ SD	2.23/ 1.952	0.287/ 0.006	0.298/ 0.006	8.349E-15/ 2.552E-15
	20	best	0.058	0.001	0.003	1.537E-13
		worst	8.793	4.03	2.016	2.13E-11
		average/ SD	3.995/ 1.739	0.772/ 1.153	0.841/ 0.825	4.557E-12/ 5.108E-12
	30	best	0.076	0.001	0.003	1.009E-10
		worst	10.329	2.815	5.095	1.982E-09
		average/ SD	4.631/ 2.159	0.457/ 0.796	1.200/ 1.358	6.939E-10/ 5.557E-10

Table V presents the optimization results for the MA-KHA and krill algorithms. This shows that the proposed multi-agent approach outperforms the other algorithms. It is able to escape from the local minima because it is equipped with competition and cooperation and self-learning operators. For example, for $n = 20$, MA-KHA obtained a solution with average fitness 4.557E-12, but KHA II (the best krill algorithm) reached 0.772 on average. Note that a smaller search space results in better solutions. For instance, MA-KHA found a solution with average fitness 8.349E-15 for $n = 10$

and 6.939E-10 for $n = 30$.

The Griewank function [86] is a continuous, non-separable, multimodal function with numerous widespread local minima. The non-separability feature of this benchmark makes it challenging for algorithms with good global search capabilities. Table VI presents the optimization results for MA-KHA and the krill algorithms. This shows that MA-KHA is better than the other algorithms on average. For example, with $n = 20$, MA-KHA reached 0.091, whereas KHA II obtained 0.131 on average.

TABLE VI. Simulation Results for the Griewank Problem

Function	Dim.	Criteria	KHA I	KHA II	KHA IV	MA-KHA
Griewank	10	best	0.060	0.043	0.060	0.052
		worst	0.367	0.474	0.339	0.365
		average/ SD	0.144/ 0.072	0.140/ 0.101	0.160/ 0.071	0.108/ 0.079
	20	best	0.108	0.061	0.092	0.071
		worst	0.361	0.252	0.320	0.222
		average/ SD	0.215/ 0.057	0.131/ 0.050	0.154/ 0.060	0.091/ 0.024
	30	best	0.310	0.143	0.152	0.126
		worst	1.023	0.330	0.439	0.234
		average/ SD	0.480/ 0.163	0.224/ 0.049	0.228/ 0.073	0.208/ 0.043

The Rastrigin function [86] is an extremely multimodal non-separable function with several regularly distributed local minima. In this problem, the area that contains the global minimum is very small in comparison with the search space. The consecutive sharp direction changes in this benchmark function made it one of the most difficult test functions for every optimization algorithms. Table VII presents the optimization results for MA-KHA and krill algorithms. This shows that the performance of MA-KHA is better than that of the other krill algorithms for $n = 10$. However, For $n = 20$ and $n = 30$, the performance of MA-KHA is very close to KHA II but still better than KHA I and KHA IV. For example, with $n = 10$, KHA I, KHA

II, KHA IV, and MA-KHA obtained solutions with fitness 6.329, 4.473, 4.864, and 4.021 on average, respectively, and for $n = 20$, solutions with average fitness 14.909, 12.374, 13.977, and 13.641, respectively, were obtained.

TABLE VII. Simulation Results for the Rastrigin Problem

Function	Dim.	Criteria	KHA I	KHA II	KHA IV	MA-KHA
Rastrigin	10	best	2.985	1.900	1.995	1.796
		worst	12.945	10.945	14.920	12.921
		average/ SD	6.329/ 2.548	4.473/ 2.177	4.864/ 4.078	4.021/ 3.129
	20	best	6.967	4.240	4.962	4.957
		worst	28.515	21.138	29.851	39.285
		average/ SD	14.909/ 5.404	12.374/ 5.113	13.977/ 4.819	13.641/ 6.574
	30	best	11.390	8.962	10.948	9.854
		worst	75.117	49.100	37.812	44.200
		average/ SD	24.658/ 13.721	19.509/ 9.621	21.297/ 7.511	20.631/ 14.181

The Rosenbrock function [86] is a conventional unimodal test problem. In this function the global minimum lies in a narrow valley. Even though the valley is quite easy to find, convergence to the global minimum is difficult. The reason is that it has a flat surface which does not provide algorithms with much information to direct the search towards the minima. Table VIII presents the optimization results for the Resenbrock function. The average results show that MA-KHA is better than the other algorithms. For instance, the average results for $n = 20$ show that MA-KHA is more than two times better than KHA II, three times better than KHA IV, and almost six times better than KHA I.

The sphere function is a continuous, convex, separable test function [86] which has no local minima except for the global solution. Algorithms with more focus on local search such as FA will have difficulty with this problem since a poor global search leads to slow movement towards the global minimum. On the other hand,

TABLE VIII. Simulation Results for the Rosenbrock Problem

Function	Dim.	Criteria	KHA I	KHA II	KHA IV	MA-KHA
Rosenbrock	10	best	21.632	5.478	6.189	4.740
		worst	775.735	625.431	619.543	206.724
		average/ SD	114.672/ 187.020	51.767/ 139.019	66.519/ 149.227	17.587/ 44.523
	20	best	17.969	12.226	14.625	12.061
		worst	316.987	443.540	250.320	75.286
		average/ SD	122.805/ 98.124	44.921/ 98.020	60.491/ 60.989	21.069/ 12.791
	30	best	40.099	21.118	23.151	17.185
		worst	841.535	447.800	314.010	38.486
		average/ SD	151.499/ 169.829	88.116/ 103.719	86.175/ 92.939	28.638/ 2.377

a poor local search results in never finding the global minimum. Table IX gives the numerical results for the sphere function. This shows that MA-KHA provides a significant performance improvement in comparison with the other algorithms. For instance, for $n = 20$, the average solution found by MA-KHA is almost half and one-third that of KHA II and KHA IV, respectively. The reason is that using a multi-agent method improves the global search at the beginning so it moves faster towards the global minimum.

TABLE IX. Simulation Results for the Sphere Problem

Function	Dim.	Criteria	KHA I	KHA II	KHA IV	MA-KHA
Sphere	10	best	0.195	5.150E-05	7.00E-05	2.250E-26
		worst	1.622	4.560E-04	5.590E-04	3.350E-22
		average/ SD	0.824/ 0.119	1.322E-04/ 1.113E-04	1.740E-04/ 9.209E-05	4.651E-23/ 9.497E-23
	20	best	0.324	1.290E-03	5.700E-03	1.990E-17
		worst	2.15223	0.004	8.190E-03	6.970E-15
		average/ SD	1.0879/ 0.296	2.27E-03/ 6.713E-03	3.980E-03/ 0.001	1.466E-15/ 2.045E-15
	30	best	0.531	2.940E-03	0.00302	2.740E-13
		worst	1.965	0.019	0.013	9.810E-12
		average/ SD	1.200/ 0.261	0.001/ 4.58E-03	0.008/ 0.002	2.419E-12/ 2.777E-12

3.2.2.1 Discussion

In order to compare MA-KHA with the krill herd algorithms for the minimization problem, the average results for $n = 20$ are normalized using

$$A_{ij} = 1 - \frac{a_{ij} - a_i^{\min}}{a_i^{\max} - a_i^{\min}} \tag{3.13}$$

where i and j denote benchmark test function and algorithm numbers, respectively ($i = 1, 2, 3, 4, 5$ and $j = 1, 2, 3, 4$). For the ith benchmark test function, a_{ij} is the solution of the jth algorithm, A_{ij} is the corresponding normalized values (score), and $a_i^{\min}$ and $a_i^{\max}$ are the worst and best solutions, respectively. In order to have an overall view of the algorithms, the algorithm with the best performance has score 1, while the one with the worst performance has score 0 [87]. Table X shows that MA-KHA

TABLE X. Normalized Average Results

Function	KHA I	KHA II	KHA IV	MA-KHA
Ackley	0.000	0.807	0.789	1.000
Griewank	0.000	0.678	0.491	1.000
Rastrigin	0.000	1.000	0.368	0.500
Rosenbrock	0.000	0.765	0.612	1.000
Sphere	0.000	0.998	0.996	1.000
Total Score	0.000	4.309	3.306	4.500

has the highest score among the four algorithms. In order to have a fair comparison, all algorithms have been tested with the same maximum number of iterations and function evaluations. These results illustrate and confirm the robustness of the multi-agent based krill algorithm for global optimization problems. MA-KHA obtained score 1 for all the test functions but one. It is better than KHA II which is the best krill herd algorithm. For the Ackley function, MA-KHA obtained a solution with average fitness 4.557E-12 versus KHA II which only reached 0.772. For the Griewank function, MA-KHA reached 0.091 whereas KHA II obtained 0.131, on average. However, for

TABLE XI. Runtime Analysis of the Algorithms

Function	KHA I	KHA II	KHA IV	MA-KHA
Ackley	47.6	62.6	63.3	63.5
Griewank	48.4	53.5	56.1	57.5
Rastrigin	53.1	57.2	68.2	70.4
Rosenbrock	49.5	51.7	59.3	60.3
Sphere	53.9	52.0	54.8	55.0
Total	252.5	276.9	301.6	306.7

the Rastrigin function, solutions with average fitness 14.0909, 12.374, 13.977, and 13.641 were obtained with KHA I, KHA II, KHA IV, and MA-KHA, respectively. The consecutive sharp direction changes in this function makes it difficult for optimization algorithms to pass the local minima and reach the global optimum. However, MA-KHA is close to KHA II. For the Rosenbrock function, MA-KHA produce less than half of the value with KHA II, providing an average fitness of 21.069. For the sphere function, MA-KHA obtained a much better solution that of KHA II. In general, MA-KHA has shown to be better able to escape from the local minima because it is equipped with competition and cooperation and self-learning operators.

Table XI gives the runtime for KHA I, KHA II, KHA IV, and MA-KHA for $n = 20$ and 50 runs. KHA I which has no genetic operator has the lowest runtime of 252.534 s. KHA II which only uses the crossover genetic operator has a runtime of 276.896 s. KHA IV with both crossover and mutation genetic operators has a runtime of 301.568 s. The proposed multi-agent krill herd algorithm required 306.703 s. Instead of crossover and mutation, MA-KHA employs competition and cooperation, including the two strategies using heuristic crossover and mutation, and self-learning. Table XI shows that the runtime of MA-KHA is very close to that of KHA IV, but the performance is better than KHA II and much better than KHA IV.

Chapter 4

MWSN Sensor Deployment Using Swarm Optimization

Sensor deployment is a significant topic in wireless sensor networks since it has a crucial effect on coverage, connectivity, and energy consumption. The goal of sensor deployment is to attain the best network coverage. In this chapter, the MWSN optimization problem is first defined. Then, the KHA and MA-KHA for the MWSN coverage problem are described, following by simulation results.

4.1 Sensor Deployment Using Swarm Algorithms

Sensor deployment is arranging sensors to meet some specific conditions or preferences. Sensors can be mobile or stationary. While stationary sensors are fixed in their positions, mobile sensors can move around. Depending on the type of WSN, the maximal coverage problem can be defined in several ways. The model used in this thesis is based on [88], in which all sensors are mobile with the same range r_s. Having N sensor nodes, the SF is defined as a grid of size $m \times n$, where the size of each grid is set to 1 unit. Thus, grid point $G(x, y)$ is detected by sensor s_i, $i = 1, 2, \ldots, N$,

with probability

$$P(x, y, s_i) = \begin{cases} 1 & d(G(x, y), s_i) \leq r_s \\ 0 & \text{otherwise} \end{cases} \tag{4.1}$$

where $d(G(x, y), s_i)$ is the Euclidean distance between the location of sensor s_i at (x_i, y_i) and grid point $G(x, y)$. Grid point $G(x, y)$ is covered if

$$P(x, y, S) = 1 - \Pi_{i=1}^{N}(1 - P(x, y, s_i)) = 1 \tag{4.2}$$

where $S = \{s_1, s_2, \ldots, s_N\}$ is the set of sensors. The total covered area is then

$$F = \sum_{x=1}^{n} \sum_{y=1}^{m} P(x, y, S) \tag{4.3}$$

To define a fitness function to evaluate the covered area the ratio between the total covered area F and the total number of grid points is used

$$f = \frac{F}{m \times n} \tag{4.4}$$

Applying SI algorithms to MWSN sensor deployment is possible because it is a maximization problem. The goal is to determine the best sensor distribution, which corresponds to the maximum fitness f. Table XII gives the relationship between the SI algorithms in a global optimization problem and the sensor deployment problem.

TABLE XII. Corresponding Parameters in Sensor Deployment and SI Algorithms

SI Algorithm	Sensor deployment problem
N number of swarm elements	N number of sensor nodes
each solution of the algorithm	a sensor distribution pattern
n dimensions in each solution	n sensor position coordinates
fitness of the solution	coverage of the sensing field
solution with minimum fitness	sensor distribution with maximum coverage

4.2 Simulation Results

To demonstrate the performance of the proposed multi-agent approach in KHA, it is compared with PSO, FA, and KHA. The best sensor distribution after an initial random configuration is simulated. There are Table XIII gives the parameters of the MWSN coverage problem that is considered here.

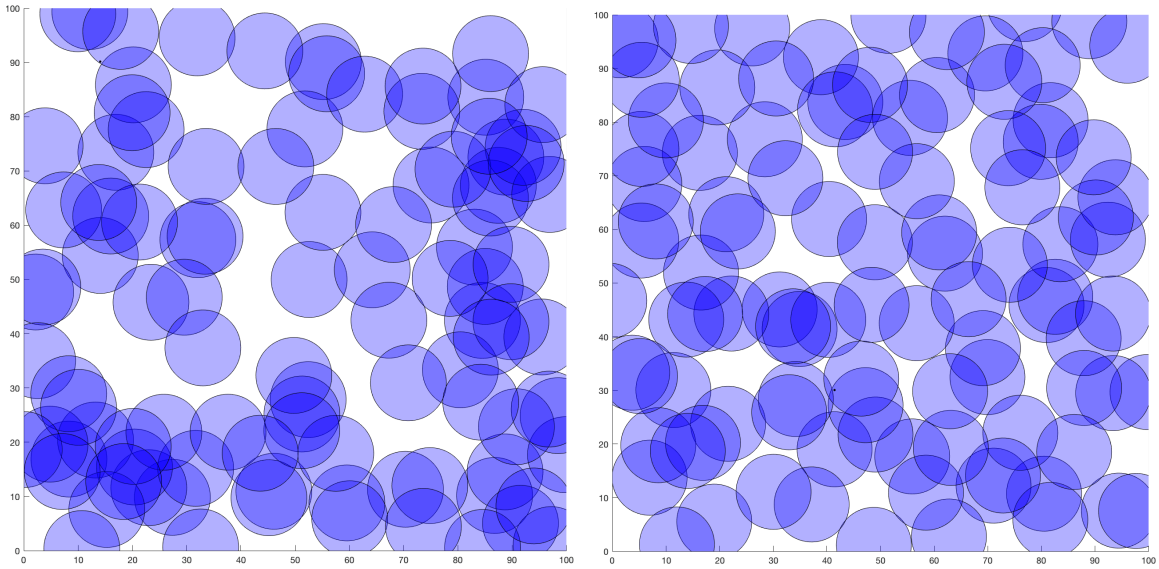
TABLE XIII. Parameters for the Sensor Deployment Problem

mobile sensor population	search space	sensing radius	sensing field
100	$[0, 100]^{100}$	3, 5, and 7	100×100

The parameters for the PSO and FA were selected based on [39] and [44], respectively. The FA parameters are as follows: light absorption coefficient $\gamma = 1$, initial attractiveness $\beta_0 = 2$, mutation vector coefficient $\alpha = 0.2$, damping parameter $\alpha_{damp} = 0.99$, and random value ε follows a uniform distribution. For the PSO algorithm, the inertia weight w, personal learning, and global learning parameters are set based on [89] as follows: inertia weight $w = 0.7298$, and cognitive and social components $c_1 = c_2 = 1.4962$. The KHA and MA-KHA settings are the same as those in the previous chapter. In order to have a fair comparison, the number of sensors and their sensing radius are the same for all algorithms. For all algorithms $t_{\max} = 500$ with 50 runs. They were run on a macOS platform with a Core i5 CPU at a frequency of 2.7 GHz with 8 GB RAM, using Matlab R2016b.

For each algorithm, part (a) in the figures shows the initial random distribution and part (b) indicates the final sensor distribution after a specific number of iterations and with the same maximum NFE. The stopping criteria was set to 500 iterations or 15,000 NFEs.

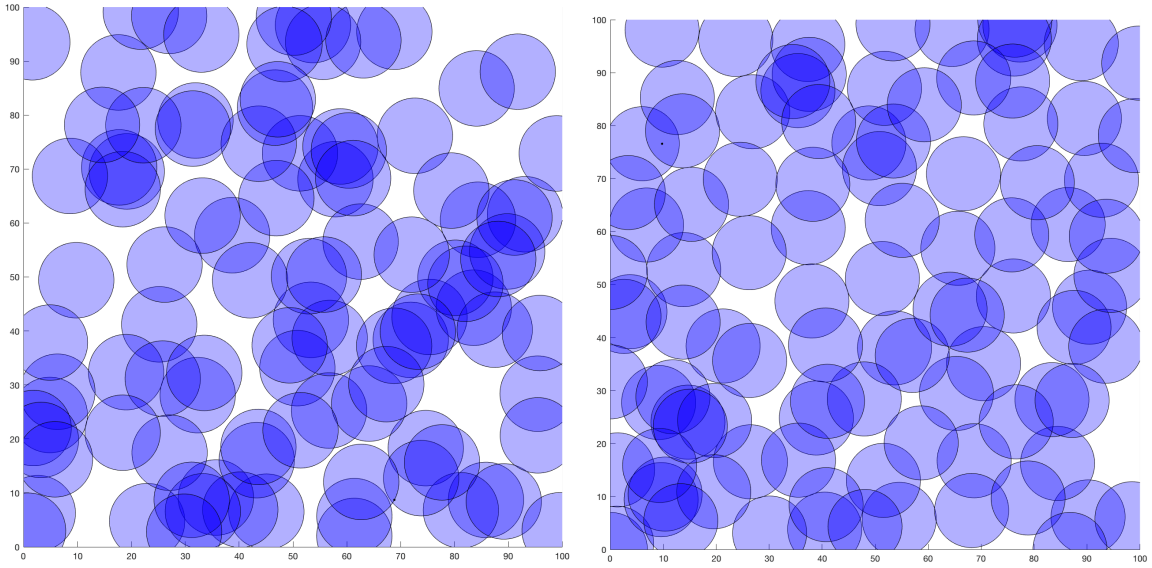
Figures 5a and 5b show the coverage for the initial random sensor distribution and the sensor distribution after 500 iterations of the PSO algorithm. The PSO algorithm benefits from a simple implementation with short convergence time. The PSO algorithm increased the coverage from 77.0% to 89.4% on average over 50 runs.



(a) (b)

Figure 5. Sensor distribution using PSO: (a) initial and (b) after 500 iterations.

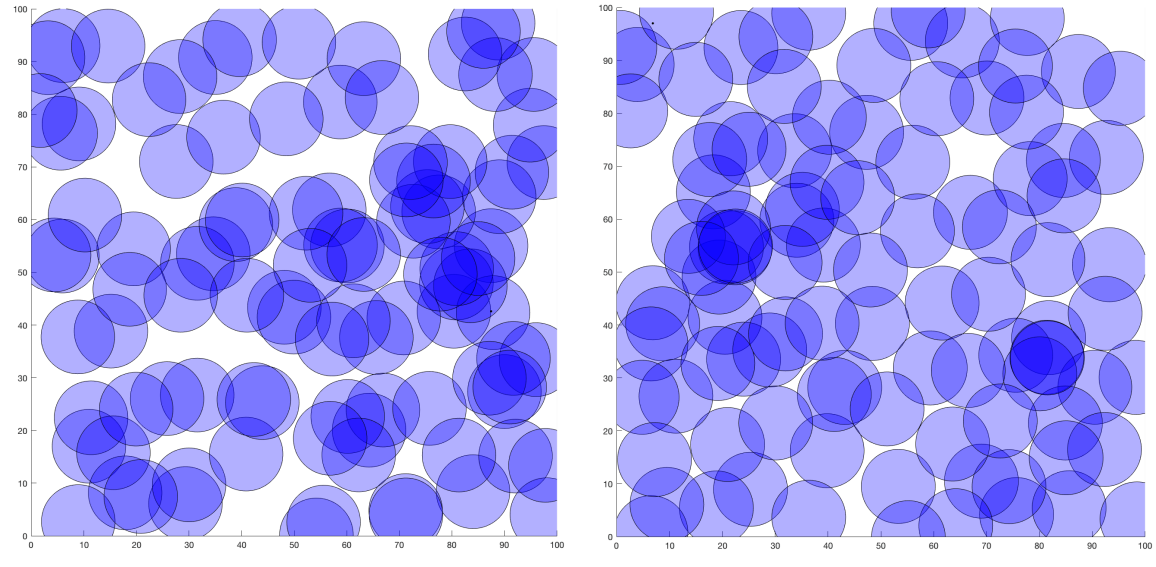
Figures 6a and 6b show the coverage for the initial random sensor distribution and final sensor distribution using FA. For this algorithm, the stopping criteria hits the maximum of 15,000 NFEs first, so it terminated after 33 iterations. The solutions found by FA shows that coverage was increased on average from 77.7% to 90.5%. FA is a slow algorithm because it compares each firefly with the entire population in each iteration resulting in many function evaluations.



(a) (b)

Figure 6. Sensor distribution using FA: (a) initial and (b) after 33 iterations.

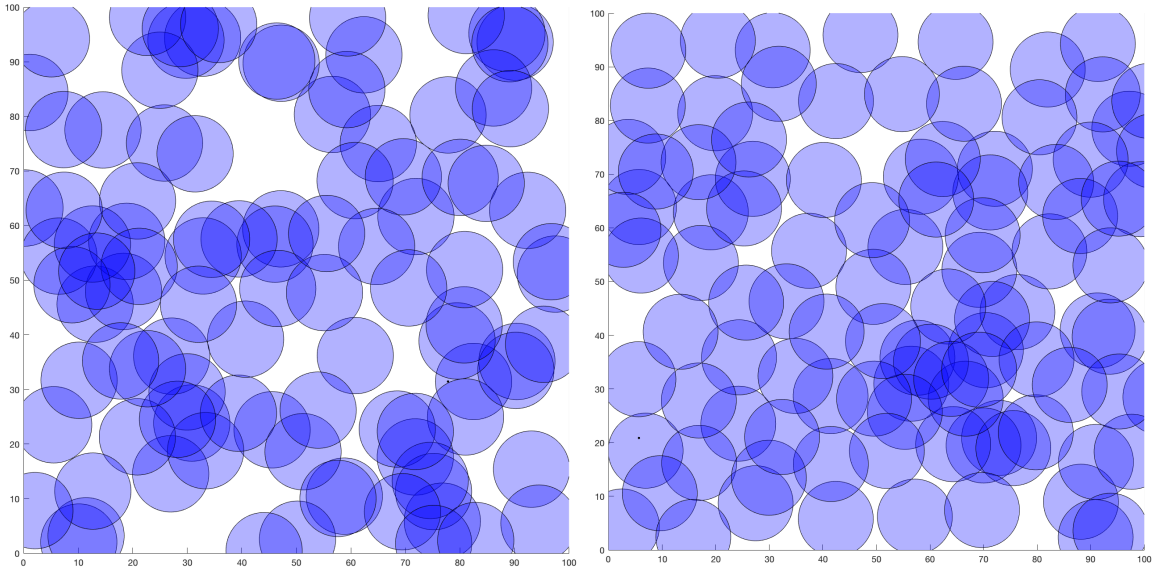
Figures 7a and 7b show the coverage for the initial random sensor distribution and sensor distribution using KHA I after 500 iterations. KHA I is as good as FA as it increases the coverage from 77.1% to 90.7% on average.



(a) (b)

Figure 7. Sensor distribution using KHA I: (a) initial and (b) after 500 iterations.

Figures 8a and 8b show the performance of KHA II for the coverage problem. Using the crossover genetic operator resulted in an increase in MWSN coverage from 78.0% to 91.9% on average.



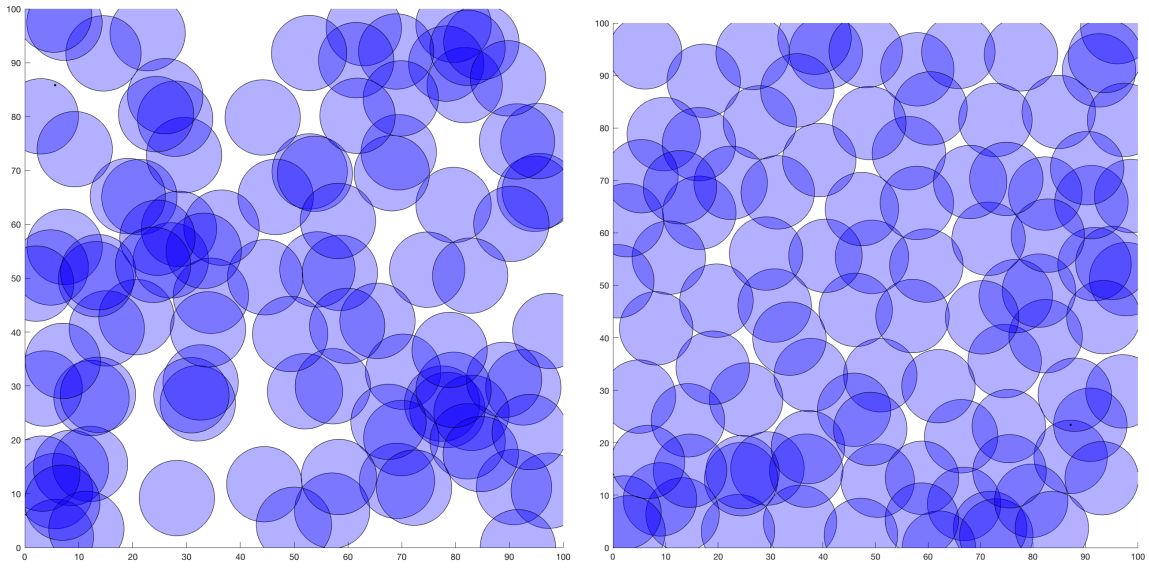
(a) (b)

Figure 8. Sensor distribution using KHA II: (a) initial and (b) after 500 iterations.

The performance of MA-KHA is given in Figures 9a and 9b. The average results for 50 runs show that MA-KHA improves the coverage from 77.6% to 94.5% after 500 iterations.

Tables XIV, XV, and XVI give the average coverage (50 runs) with sensing radius $r = 7$, 5, and 3, respectively, before (random sensor distribution) and after using the PSO, FA, KHA I, KHA II, and MA-KHA SI algorithms. The same initial sensor distributions and parameters as well as stopping criteria (maximum 500 iteration or 15,000 NFEs) were used. Each table also gives the runtime for each algorithm in seconds per run.

Tables XIV and XV show that MA-KHA provides a higher coverage than PSO, FA, and KHA I and II for $r = 7$ and $r = 5$. MA-KHA has a slightly higher runtime than other algorithms. All algorithms stopped at iteration 500 except FA which hit



(a) (b)

Figure 9. Sensor distribution using MA-KHA: (a) initial and (b) after 500 iterations.

TABLE XIV. Average MWSN Coverage Optimization Results for $r = 7$

Algorithm	Before (random distribution)	After	Time (s)
PSO	77.0%	89.4%	50.9
FA	77.7%	90.5%	50.5
KHA I	77.1%	90.7%	50.4
KHA II	78.0%	91.9%	55.9
MA-KHA	77.6%	94.5%	61.2

the $NFE_{\max}$ first and stopped at iteration 33. Table XVI results show that KHA II and MA-KHA provided the highest coverage on average which is slightly more than the PSO algorithm. A sensing radius of $r = 3$ is too small to cover the sensing field with 100 mobile sensors and there is little overlap between the sensors. Thus, a larger sensing radius is recommended.

Table XVII gives the the average coverage (50 runs) with $r = 7$ employing PSO, FA, KHA I, KHA II, and MA-KHA using a maximum of 100, 250, and 500 iterations and a maximum of 3,000, 7,500, and 15,000 NFEs, respectively. Again, all the algorithms hit the maximum number of iterations except FA which stopped at 10 iterations for $NFE_{\max} = 3,000$ and 19 iterations for $NFE_{\max} = 7,500$, respectively.

TABLE XV. Average MWSN Coverage Optimization Results for $r = 5$

Algorithm	Before (random distribution)	After	Time (s)
PSO	55.6%	66.9%	61.3
FA	55.9%	63.8%	65.8
KHA I	56.0%	65.8%	65.2
KHA II	55.5%	67.2%	70.0
MA-KHA	56.3%	69.9%	76.5

TABLE XVI. Average MWSN Coverage Optimization Results for $r = 3$

Algorithm	Before (random distribution)	After	Time (s)
PSO	25.1%	28.4%	75.2
FA	24.7%	26.9%	85.0
KHA I	25.6%	27.6%	80.7
KHA II	25.1%	28.5%	83.3
MA-KHA	25.8%	28.5%	89.0

4.2.1 Discussion

In this chapter, the FA and PSO algorithms were chosen to compare with KHA and MA-KHA in a MWSN sensor deployment problem. The reason is that KHA utilizes the main components of the FA and PSO algorithms. Table XVIII shows that there are similarities between KHA and the other two algorithms in terms of the movement components. PSO is designed based on three major components, best particle positions, global best particle, and random walk. The firefly algorithm utilizes two major components to find a solution. As well as using a random walk, this algorithm compares a solution with all better solutions (brighter fireflies) and moves it

TABLE XVII. Average MWSN Coverage Optimization for Different Number of NFEs and $r = 7$

Algorithm	$t_{\max} = 100/$ $NFE_{\max} = 3,000$	$t_{\max} = 250/$ $NFE_{\max} = 7,500$	$t_{\max} = 500/$ $NFE_{\max} = 15,000$
PSO	82.3%	86.8%	89.4%
FA	88.0%	89.4%	90.5%
KHA I	83.0%	87.3%	90.7%
KHA II	87.6%	90.2%	91.9%
MA-KHA	85.5%	90.1%	94.5%

towards them (neighbor effect). Using crossover and mutation operators reinforces the global search of KHA by providing relatively balanced exploration and exploitation. Thus, the strength of KHA lies in combining the movement components of the PSO and firefly algorithms assisted with the genetic operators. In the proposed multi-agent krill herd algorithm, providing the KHA with competition and cooperation and self-learning instead of the genetic algorithm operators resulted in a better global search and a higher coverage for the MWSN sensor deployment problem.

TABLE XVIII. KHA Motion and Effective Components

Motion	Effective component	Equivalent in other algorithms
induced motion	neighboring krill	FA
	global best krill	PSO
foraging motion	best krill position	PSO
	food center position	-
diffusion motion	random walk	FA/PSO
genetic operators	crossover/ mutation	GA

Chapter 5

Conclusions

In this thesis, swarm intelligence algorithms, specifically the Krill Herd Algorithm (KHA), were considered to solve the dynamic sensor deployment problem in mobile wireless sensors networks. A modified multi-agent based KH algorithm, named MA-KHA, was introduced. The MA-KHA and KHA were applied to five benchmark global optimization problems and the results compared. The simulation results showed that MA-KHA is better than KHA I, KHA II, and KHA IV as it provides better solutions.

After introducing MA-KHA and providing its performance for several global optimization problems, it was used to solve the coverage problem for mobile wireless sensor networks. To do so, PSO, FA as well as KHA I and KHA II were implemented for the MWSN problem and the performance compared with that of the proposed method. The results obtained show that MA-KHA achieves a higher coverage. The average coverage for $r = 7$ is 94.5% with MA-KHA in comparison with 91.9%, 90.7%, 90.5%, and 89.4% using KHA II, KHA I, FA, and PSO, respectively.

KHA I utilizes five major components to find a solution for an optimization problem, neighboring krill, global best krill, best krill position, food center position, and random walk. KHA II to KHA IV also employ genetic operators. In the proposed al-

gorithm, competition and cooperation and self-learning operators were used instead of genetic operators. The competition and cooperation operator strengthens the exploration while self-learning focuses on exploitation. The results for the MWSN coverage and global optimization problems show that the multi-agent approach can help the krill algorithm provide better solutions by balancing exploitation and exploration.

In this thesis, parameter selection for MA-KHA was done based on the KHA algorithm parameters to have a fair comparison. As future work, parameter selection can be studied in detail since there are a number of parameters that can be adjusted. A more advanced self-learning operator can also be designed. For instance, instead of using the best agent, a random one can be used in the self-learning to increase randomness. Depending on the optimization problem, different metaheuristic algorithms can be chosen for the self-learning. For instance a mini-FA can be utilized in self-learning to provide a more powerful local search. There are also several algorithms that can be hybridized with KHA. For example, hybrid KHA and PSO can be implemented to solve optimization problems such as battery consumption and coverage for WSNs.

Appendix A

Benchmark Global Optimization Problems

- F1: Ackley function (Figure 10)

  This function is widely used for testing optimization algorithms. It is characterized by a nearly flat outer region, and a large hole at the center. This poses a risk for optimization algorithms to be trapped in one of its many local minima [86]. The Ackley function with $-32 \leq x_i \leq 32$ is given by

$$f(x_0, x_1, \ldots, x_n) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2} - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$$

- F2: Griewank function (Figure 11)

  This function has many widespread local minima which are regularly distributed [86]. The Griewank function is given by

$$f(x_0, x_1, \ldots, x_n) = 1 + \frac{1}{4,000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}), -600 \leq x_i \leq 600$$

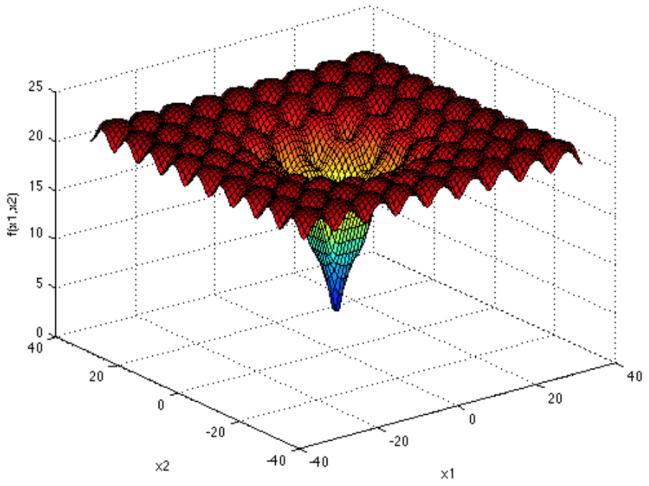


Figure 10. The Ackley function

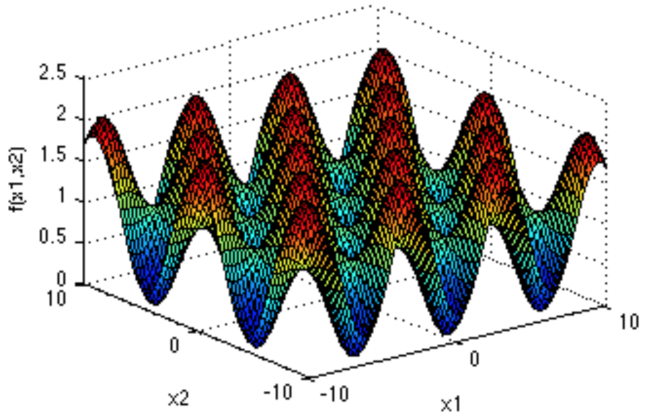


Figure 11. The Griewank function

- F3: Rastrigin function (Figure 12)

  This is a highly multimodal function with several local minima. However, the locations of the minima are regularly distributed [86]. The Rastrigin function is given by

$$f(x_0, x_1, \ldots, x_n) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i)), -5.12 \leq x_i \leq 5.12$$

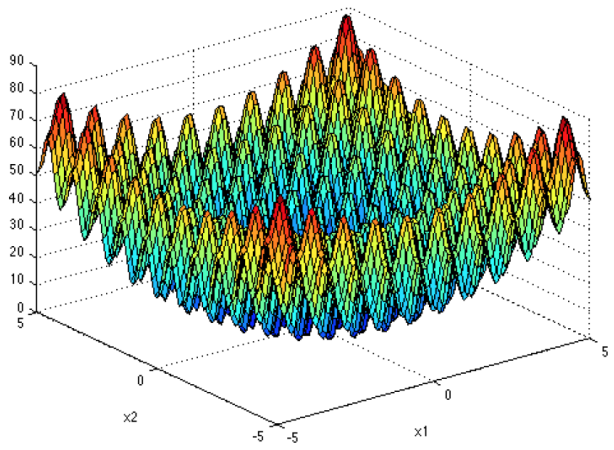


Figure 12. The Rastrigin function

- F4: Rosenbrock function (Figure 13)

  This function is a popular unimodal test problem, in which the global minimum lies in a narrow valley. However, even though the valley is quite easy to find, convergence to the minimum is difficult [86].

  The Rosenbrock function is given by

$$f(x_0, x_1, \ldots, x_n) = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2), -30 \leq x_i \leq 30$$

- F5: sphere function (Figure 14)

  This function is a continuous, convex and unimodal test function. It has several local minima but one global minimum [86]. The sphere function is given by

$$f(x_0, x_1, \ldots, x_n) = \sqrt{\sum_{i=1}^{n} x_i^2}, -100 \leq x_i \leq 100$$

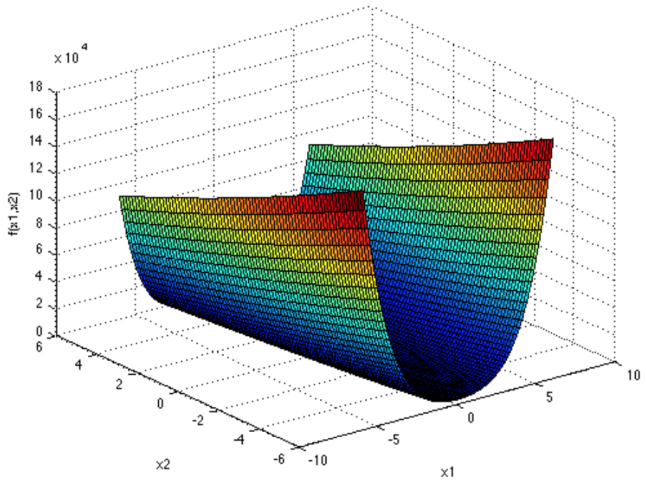


Figure 13. The Rosenbrock function

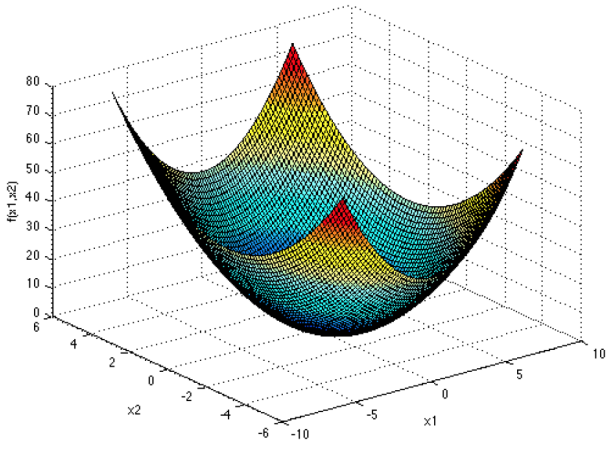


Figure 14. The sphere function

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] L. Mainetti, L. Patrono, and A. Vilei, "Evolution of wireless sensor networks towards the internet of things: A survey," in *Proc. International Conference on Software, Telecommunications and Computer Networks*, 2011, pp. 1–6.

[3] V. Potdar, A. Sharif, and E. Chang, "Wireless sensor networks: A survey," in *Proc. IEEE International Conference on Advanced Information Networking and Applications Workshops*, 2009, pp. 636–641.

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

[5] J. Chen, E. Shen, and Y. Sun, "The deployment algorithms in wireless sensor net works: A survey," *Information Technology Journal*, vol. 8, no. 3, pp. 293–301, 2009.

[6] K. Chakrabarty, S. S. Iyengar, H. Qi, and E. Cho, "Grid coverage for surveillance and target location in distributed sensor networks," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1448–1453, 2002.

[7] D. S. Deif and Y. Gadallah, "Classification of wireless sensor networks deployment techniques," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 834–855, 2014.

[8] J. Krause, J. Cordeiro, R. S. Parpinelli, and H. S. Lopes, "A survey of swarm algorithms applied to discrete optimization problems," *Swarm Intelligence and Bio-inspired Computation: Theory and Applications*, pp. 169–191, 2013.

[9] M. Hefeeda and H. Ahmadi, "A probabilistic coverage protocol for wireless sensor networks," in *Proc. IEEE International Conference on Network Protocols*, 2007, pp. 41–50.

[10] M. Cardei, M. T. Thai, Y. Li, and W. Wu, "Energy-efficient target coverage in wireless sensor networks," in *Proc. IEEE International Joint Conference of the Computer and Communications Societies*, 2005, pp. 1976–1984.

[11] A. Howard, M. J. Matarić, and G. S. Sukhatme, "An incremental self-deployment algorithm for mobile sensor networks," *Autonomous Robots*, vol. 13, no. 2, pp. 113–126, 2002.

[12] X. Wang, Y. Yang, and Z. Zhang, "A virtual rhomb grid-based movement-assisted sensor deployment algorithm in wireless sensor networks," in *Proc. International Multi-Symposiums on Computer and Computational Sciences*, 2006, pp. 491–495.

[13] M.-l. Lam and Y.-h. Liu, "Isogrid: An efficient algorithm for coverage enhancement in mobile sensor networks," in *Proc. IEEE International Conference on Intelligent Robots and Systems*, 2006, pp. 1458–1463.

[14] X. Yu, W. Huang, J. Lan, and X. Qian, "A novel virtual force approach for node deployment in wireless sensor network," in *Proc. IEEE International Conference on Distributed Computing in Sensor Systems*, 2012, pp. 359–363.

[15] ——, "A van der waals force-like node deployment algorithm for wireless sensor network," in *Proc. International Conference on Mobile Ad-hoc and Sensor Networks*, 2012, pp. 191–194.

[16] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," *Distributed Autonomous Robotic Systems*, vol. 5, pp. 299–308, 2002.

[17] D. B. Jourdan and O. L. de Weck, "Layout optimization for a wireless sensor network using a multi-objective genetic algorithm," in *Proc. IEEE Vehicular Technology Conference*, 2004, pp. 2466–2470.

[18] D. B. Jourdan and O. L. de Weck, "Multi-objective genetic algorithm for the automated planning of a wireless sensor network to monitor a critical facility," in *Proc. International Conference on Sensors, and Command, Control, Communications, and Intelligence Technologies for Homeland Security and Homeland Defense III*, 2004, pp. 565–576.

[19] Y. Yoon and Y.-H. Kim, "An efficient genetic algorithm for maximum coverage deployment in wireless sensor networks," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1473–1483, 2013.

[20] X. Wang, S. Wang, and J.-J. Ma, "An improved co-evolutionary particle swarm optimization for wireless sensor networks with dynamic deployment," *Sensors*, vol. 7, no. 3, pp. 354–370, 2007.

[21] N. Kukunuru, B. R. Thella, and R. L. Davuluri, "Sensor deployment using particle swarm optimization," *International Journal of Engineering Science and Technology*, vol. 2, no. 10, pp. 5395–5401, 2010.

[22] E. Tuba, M. Tuba, and M. Beko, "Mobile wireless sensor networks coverage maximization by firefly algorithm," in *Proc. International Conference Radioelektronika*, 2017, pp. 1–5.

[23] M. Shopon, M. A. Adnan, and M. F. Mridha, "Krill herd based clustering algorithm for wireless sensor networks," in *Proc. IEEE International Workshop on Computational Intelligence*, IEEE, 2016, pp. 96–100.

[24] P. Jiang, Y. Feng, F. Wu, S. Yu, and H. Xu, "Dynamic layered dual-cluster heads routing algorithm based on krill herd optimization in UWSNs," *Sensors*, vol. 16, no. 9, pp. 1379–1384, 2016.

[25] C. Ozturk, D. Karaboga, and B. Gorkemli, "Artificial bee colony algorithm for dynamic deployment of wireless sensor networks," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 20, no. 2, pp. 255–262, 2012.

[26] ——, "Probabilistic dynamic deployment of wireless sensor networks by artificial bee colony algorithm," *Sensors*, vol. 11, no. 6, pp. 6056–6065, 2011.

[27] X. Liu, "Sensor deployment of wireless sensor networks based on ant colony optimization with three classes of ant transitions," *IEEE Communications Letters*, vol. 16, no. 10, pp. 1604–1607, 2012.

[28] V. Akbarzadeh, C. Gagné, M. Parizeau, M. Argany, and M. A. Mostafavi, "Probabilistic sensing model for sensor placement optimization based on line-of-sight coverage," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 2, pp. 293–303, 2013.

[29] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[30] A. H. Gandomi, X.-S. Yang, S. Talatahari, and A. H. Alavi, "Metaheuristic algorithms in modeling and optimization," in *Metaheuristic Applications in Structures and Infrastructures*, Elsevier, 2013.

[31] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Robots and Biological Systems: Towards a New Bionics*, Springer, 1993, pp. 703–712.

[32] I. Rechenberg, "Evolution strategy: Natures way of optimization," in *Optimization: Methods and Applications, Possibilities and Limitations*, Springer, 1989, pp. 106–126.

[33] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial Intelligence Through Simulated Evolution*. Wiley, 1966.

[34] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992.

[35] S. Kirkpatrick *et al.*, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[36] F. Glover and M. Laguna, "Tabu search," in *Handbook of Combinatorial Optimization*, Springer, 2013, pp. 3261–3362.

[37] V. Maniezzo, M. Dorigo, and A. Colorni, "Distributed optimization by ant colonies," in *Proc. European Conference on Artificial Life*, 1991, pp. 124–142.

[38] J. R. Koza, "Genetic programming as a means for programming computers by natural selection," *Statistics and Computing*, vol. 4, no. 2, pp. 87–112, 1994.

[39] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. IEEE International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.

[40] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[41] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.

[42] S. Nakrani and C. Tovey, "On honey bees and dynamic server allocation in internet hosting centers," *Adaptive Behavior*, vol. 12, no. 3-4, pp. 223–240, 2004.

[43] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[44] X.-S. Yang, *Nature-inspired Metaheuristic Algorithms*. Luniver Press, 2010.

[45] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. World Congress on Nature & Biologically Inspired Computing*, 2009, pp. 210–214.

[46] A. H. Gandomi and A. H. Alavi, "Krill herd: A new bio-inspired optimization algorithm," *Communications in Nonlinear Science and Numerical Simulation*, vol. 17, no. 12, pp. 4831–4845, 2012.

[47] C. Darwin and M. J. Adler, *On the Origin of Species by Means of Natural Selection*. Broadview Press, 2003.

[48] N. F. McPhee, R. Poli, and W. B. Langdon, *Field Guide to Genetic Programming*. Lulu Enterprises, 2008.

[49] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Optimization*. Wiley, 2000.

[50] S. L. Tilahun and J. M. T. Ngnotchouye, "Firefly algorithm for discrete optimization problems: A survey," *KSCE Journal of Civil Engineering*, vol. 21, no. 2, pp. 535–545, 2017.

[51] I. Fister, I. Fister Jr, X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm and Evolutionary Computation*, vol. 13, pp. 34–46, 2013.

[52] E. E. Hofmann, A. E. Haskell, J. M. Klinck, and C. M. Lascara, "Lagrangian modelling studies of antarctic krill (euphausia superba) swarm formation," *ICES Journal of Marine Science*, vol. 61, no. 4, pp. 617–631, 2004.

[53] S. Nicol, "Living krill, zooplankton and experimental investigations: A discourse on the role of krill and their experimental study in marine ecology," *Proc. of the International Workshop on Understanding Living Krill for Improved Management and Stock Assessment Marine and Freshwater Behavior and Physiology*, vol. 36, no. 4, pp. 191–205, 2003.

[54] E. Murphy, D. Morris, J. Watkins, and J Priddle, "Scales of interaction between antarctic krill and the environment," in *Antarctic Ocean and Resources Variability*, Berlin: Springer, 1988, pp. 120–130.

[55] A. H. Gandomi and A. H. Alavi, "An introduction of krill herd algorithm for engineering optimization," *Journal of Civil Engineering and Management*, vol. 22, no. 3, pp. 302–310, 2016.

[56] H. Faris *et al.*, "Optimizing feedforward neural networks using krill herd algorithm for e-mail spam detection," in *Proc. IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies*, 2015, pp. 1–5.

[57] A. Mohammadi, M. S. Abadeh, and H. Keshavarz, "Breast cancer detection using a multi-objective binary krill herd algorithm," in *Proc. Iranian Conference on Biomedical Engineering*, 2014, pp. 128–133.

[58] L. M. Abualigah, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "A krill herd algorithm for efficient text documents clustering," in *Proc. IEEE*

Symposium on Computer Applications & Industrial Electronics, 2016, pp. 67–72.

[59] G.-G. Wang, S. Deb, and S. M. Thampi, "A discrete krill herd method with multilayer coding strategy for flexible job-shop scheduling problem," *Intelligent Systems Technologies and Applications*, pp. 201–215, 2016.

[60] D. Rodrigues, L. A. Pereira, J. P. Papa, and S. A. Weber, "Binary krill herd approach for feature selection," in *Proc. International Conference on Pattern Recognition*, 2014, pp. 1407–1412.

[61] E. Fattahi, M. Bidar, and H. R. Kanan, "Fuzzy krill herd (FKH): An improved optimization algorithm," *Intelligent Data Analysis*, vol. 20, no. 1, pp. 153–165, 2016.

[62] H. V. H. Ayala, E. H. Segundo, V. C. Mariani, and L. S. Coelho, "Multiobjective krill herd algorithm for electromagnetic optimization," *IEEE Transactions on Magnetics*, vol. 52, no. 3, pp. 1–4, 2016.

[63] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and D. Gong, "A comprehensive review of krill herd algorithm: Variants, hybrids and applications," *Artificial Intelligence Review*, pp. 1–30, 2017.

[64] G.-G. Wang, L. Guo, A. H. Gandomi, G.-S. Hao, and H. Wang, "Chaotic krill herd algorithm," *Information Sciences*, vol. 274, pp. 17–34, 2014.

[65] G.-G. Wang, S. Deb, A. H. Gandomi, and A. H. Alavi, "Opposition-based krill herd algorithm with cauchy mutation and position clamping," *Neurocomputing*, vol. 177, pp. 147–157, 2016.

[66] L. Guo, G.-G. Wang, A. H. Gandomi, A. H. Alavi, and H. Duan, "A new improved krill herd algorithm for global numerical optimization," *Neurocomputing*, vol. 138, pp. 392–402, 2014.

[67] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and S. Deb, "A multi-stage krill herd algorithm for global numerical optimization," *International Journal on Artificial Intelligence Tools*, vol. 25, no. 02, p. 1 550 030, 2016.

[68] G.-G. Wang, A. H. Gandomi, and A. H. Alavi, "Stud krill herd algorithm," *Neurocomputing*, vol. 128, pp. 363–370, 2014.

[69] I. N. Trivedi *et al.*, "Adaptive krill herd algorithm for global numerical optimization," in *Advances in Computer and Computational Sciences*, Springer, 2017, pp. 517–525.

[70] G.-G. Wang, A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "A new hybrid method based on krill herd and cuckoo search for global optimisation tasks," *International Journal of Bio-Inspired Computation*, vol. 8, no. 5, pp. 286–299, 2016.

[71] G.-G. Wang, A. H. Gandomi, and A. H. Alavi, "An effective krill herd algorithm with migration operator in biogeography-based optimization," *Applied Mathematical Modelling*, vol. 38, no. 9, pp. 2454–2462, 2014.

[72] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and G.-S. Hao, "Hybrid krill herd algorithm with differential evolution for global numerical optimization," *Neural Computing and Applications*, vol. 25, no. 2, pp. 297–308, 2014.

[73] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and S. Deb, "A hybrid method based on krill herd and quantum-behaved particle swarm optimization," *Neural Computing and Applications*, vol. 27, no. 4, pp. 989–1006, 2016.

[74] G.-G. Wang, L. Guo, A. H. Gandomi, A. H. Alavi, and H. Duan, "Simulated annealing-based krill herd algorithm for global optimization," *Abstract and Applied Analysis*, vol. 2013, 2013.

[75] A. M. Khalil, S.-E. K. Fateen, and A. Bonilla-Petriciolet, "MAKHA new hybrid swarm intelligence global optimization algorithm," *Algorithms*, vol. 8, no. 2, pp. 336–365, 2015.

[76] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and Y.-Q. Dong, "A hybrid meta-heuristic method based on firefly algorithm and krill herd," in *Handbook of Research on Advanced Computational Techniques for Simulation-Based Engineering*, IGI Global, 2016, pp. 505–524.

[77] G. Wang, L. Guo, H. Wang, H. Duan, L. Liu, and J. Li, "Incorporating mutation scheme into krill herd algorithm for global numerical optimization," *Neural Computing and Applications*, vol. 24, no. 3-4, pp. 853–871, 2014.

[78] Q. Li and B. Liu, "Clustering using an improved krill herd algorithm," *Algorithms*, vol. 10, no. 2, pp. 56–67, 2017.

[79] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson Education Limited, 2016.

[80] N. R. Jennings, K. Sycara, and M. Wooldridge, "A roadmap of agent research and development," *Autonomous Agents and Multi-Agent Systems*, vol. 1, no. 1, pp. 7–38, 1998.

[81] J. Ferber, *Multi-agent systems: An introduction to distributed artificial intelligence*. Addison-Wesley, 1999.

[82] J. Liu, *Autonomous agents and multi-agent systems: Explorations in learning, self-organization and adaptive computation*. World Scientific, 2001.

[83] J. Liu, H. Jing, and Y. Y. Tang, "Multi-agent oriented constraint satisfaction," *Artificial Intelligence*, vol. 136, no. 1, pp. 101–144, 2002.

[84] W. Zhong, J. Liu, M. Xue, and L. Jiao, "A multiagent genetic algorithm for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 34, no. 2, pp. 1128–1141, 2004.

[85] G. P. Singh and A. Singh, "Comparative study of krill herd, firefly and cuckoo search algorithms for unimodal and multimodal optimization," *International Journal of Intelligent Systems and Applications*, vol. 6, no. 3, pp. 35–49, 2014.

[86] J. Momin and X.-S. Yang, "A literature survey of benchmark functions for global optimization problems," *Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[87] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[88] M. Abo-Zahhad, S. M. Ahmed, N. Sabor, and S. Sasaki, "Coverage maximization in mobile wireless sensor networks utilizing immune node deployment algorithm," in *Proc. IEEE Canadian Conference on Electrical and Computer Engineering*, 2014, pp. 1–6.

[89] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.