7-28-2010

A Multi-objective Evolutionary Algorithm to solve Complex Optimization Problems

Koyel Chaudhuri

To the University Council:

The Thesis Committee for Koyel Chaudhuri certifies that this is the final approved version of the following electronic thesis “A Multi-objective Evolutionary Algorithm to solve complex optimization problems.”

_____
Dipankar Dasgupta, Ph.D.
Major Professor

We have read this thesis and recommend its acceptance:

_____
Sajjan Shiva, Ph.D.

_____
Qishi Wu, Ph.D.

Accepted for the Graduate Council:

_____
Karen D. Weddle-West, Ph.D.
Vice Provost for Graduate Programs

A MULTI-OBJECTIVE EVOLUTIONARY ALGORITHM TO SOLVE COMPLEX OPTIMIZATION PROBLEMS

by

Koyel Chaudhuri

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Computer Science

The University of Memphis

August'2010

To my dearest mom who is the best mom in this world

and

To my late father whose blessing is always with me

ACKNOWLEDGEMENTS

I am heartily thankful to my advisor, Dr. Dipankar Dasgupta, whose guidance, support and encouragement from the incipient stage of this thesis to the final level helped me to successfully complete the thesis. I owe my deepest gratitude to him.

Lastly, I am indebted to many of my friends for supporting me in several aspects during many phases of my thesis. I would like to thank all of you for your support.

Abstract

Multi-objective optimization problem formulations reflect pragmatic modeling of several real-life complex optimization problems. In many of them the considered objectives are competitive with each other; emphasizing only one of them during solution generation and evolution incurs high probability of producing a one-sided solution, which is unacceptable with respect to other objectives. An appropriate solution to the multi-objective optimization problem is to investigate a set of solutions that satisfy all of the competing objectives to an acceptable extent, where no solution in the solution set is dominated by others in terms of objective optimization. In this work, we investigate well known Non-dominated Sorting Genetic Algorithm (NSGA-II), and Strength Pareto Evolutionary Algorithm (SPEA-II), to find Pareto optimal solutions for two real-life problems: Task-based Sailor Assignment Problem (TSAP) and Coverage and Lifetime Optimization Problem in Wireless Sensor Networks (CLOP). Both of these problems are multi-objective problems. TSAP constitutes five multi-directional objectives, whereas CLOP is composed of two competing objectives. To validate the special operators developed, these two test bed problems have been used. Finally, traditional NSGA-II and SPEA-II have been blended with these special operators to generate refined solutions of these multi-objective optimization problems.

Contents

List of Figures

List of Tables

1 Introduction

Several real-world design or decision making problems involve concurrent optimization of multiple objectives. Multi-objective optimization problems are lucidly different from single objective optimization problems. In many multi-objective optimization problems, objectives are conflicting in nature; their directions may oppose each other. A few examples: minimize cost, maximize number of cities traveled, minimum distance, and there are many more. Genetic algorithm is a potential meta-heuristic, which is specifically well-suited for these sorts of problems in which optimal solutions are sought so as to optimize all objectives of the problem. Traditional genetic algorithm is customized to fit into problem domains; problem specific crossover, mutation, selection, and local search and boundary search operators are utilized to generate optimum solution set.

1.1 Overview

1.1.1 Multi-objective Optimization Problem

The problems "Task-based Sailor Assignment Problem" (TSAP) and "Coverage and Lifetime Optimization of Wireless Sensor Network using Genetic Algorithm" (CLOP) enroot to the general multi-objective optimization problem. A general multi-objective optimization problem can be mathematically expressed in terms of objective functions and several equality and/or inequality constraints. It can be expressed as follows:

Maximize/ Minimize $f_i(x)$ where i=1,2,.....,N

Subject to

$g_j(x) <= 0; j = 1, 2, ..., J$

$h_k(x) > 0; k = 1, 2, ..., K$

The function $f_i(x)$ is the objective function where x is an N dimensional vector or vector with N decision variables. The solutions to a multi-objective optimization problem refer to a set of non-dominated solutions where no other existing solutions can dominate them with respect to all objectives. The set of non-dominated solutions is known as "Parent Optimal Solution", or "Pareto Front". The situation in general multi-objective optimization problem is explained in Figure 1. As the

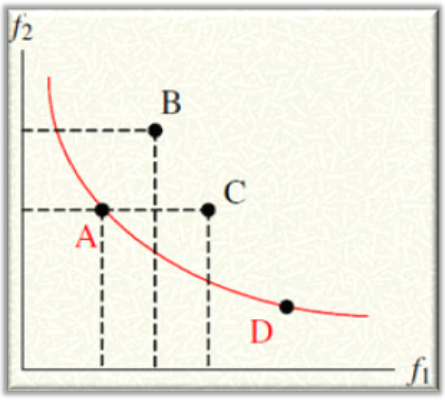


Figure 1: **Multi-objective Optimization Problem**

problem is composed of multiple objectives to be optimized, a solution that will balance all objectives rather prioritizing one and subduing other objectives will be suitable. Figure 1 represents the Pareto-optimal solutions for minimization problem. In Figure 1, both the objectives f1 and f2 are to be minimized. A, B, C, and D depicted in this figure represent the solution points over the objective space. Solution C, compared to other solutions, is poor, as the objectives are to be minimized. Solution B turns out to be good for objective f1, but produces negative results for objective f2. On the other hand, solution A is equally good with solution C as far as objective f2 is concerned, but solution A outperforms solution C with respect to objective f_1. Hence solution A is better solution than solutions B and C. On the contrary, solution D produces minimum value for objective f2, but not for objective f1. But both solutions A and D are shown to produce good results for one objective respectively which makes them incomparable. Therefore solutions A and D are non-dominated.

The parameter x is a p dimensional design or decision variable. Solutions to a multi-objective optimization problem are mathematically expressed in terms of non-dominated points. In a minimization problem, a solution vector $x^{(i)}$ is partially less than another solution vector $x^{(j)}$, i.e. $x^{(i)} \prec x^{(j)}$ when no value of $x^{(j)}$ is less than $x^{(i)}$ and at least one value of $x^{(j)}$ is strictly greater than $x^{(i)}$. If $x^{(i)}$ is partially less than $x^{(j)}$, then solution $x^{(i)}$ is said to dominate $x^{(j)}$ [1]. The member of this sort of solution vector, which is not dominated by any other solution, is defined as non-dominated. The maximization problems can also be described in the explanation above. The solutions to a multi-objective optimization problem are those non-dominated solutions, which are also Pareto-optimal solutions.

1.1.2 Genetic Algorithm

The genetic algorithm (GA) is a search heuristic that imitates the process of natural evolution. This heuristic is used to generate acceptable solutions to optimization and search problems. Genetic algorithm originates from evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. The flow chart of the algorithm is depicted in Figure 2.

GA starts with an initial set of randomly generated solutions called population, where each solution is enciphered into the form of a chromosome. The process by which a population encodes a solution to an optimization problem is called genetic representation, which plays a crucial role in EA. Genetic representation can encode appearance, behavior, and physical qualities of individuals. Designing a good genetic representation that is expressive and evolvable is a hard problem in evolutionary computation. In each generation, the fitness of every individual solution in the population is evaluated first. Once evaluation of each individual is done, multiple

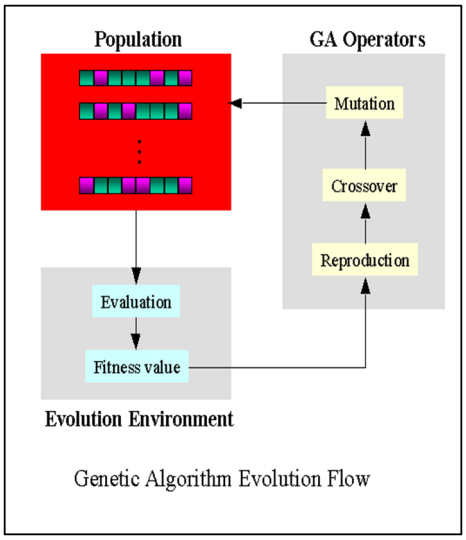


Figure 2: **Genetic Algorithm**

individuals are stochastically selected from the current population based on their fitness. Then the selected individuals are subjected to several genetic operators such as crossover and mutation to form a new population. The new population is then used in the next iteration of the algorithm. In general, the algorithm terminates when either a maximum number of generations has been produced, or another satisfactory criteria has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

The process of simple generational genetic algorithm is described below.

1. Generate the initial population of individual solution
2. Evaluate the fitness of each individual in that population
3. Repeat the following steps on this generation until termination criteria have been satisfied
   (a) Select the best-fit individuals for reproduction
   (b) Generate new individuals through crossover and mutation operations to produce offsprings

(c) Evaluate the individual fitness of new individuals

(d) Replace least-fit population with new individuals

1.2 Related Work

1.2.1 Genetic Algorithm in Sailor Assignment Problem

The domain of application of evolutionary techniques to solve multiobjective optimization problems has been enriched with many successful researches in the last decade. The intrinsic use of a population of candidate solutions is fruitful because it was not used by other search techniques when applied to problems with multiple objectives. Recently, the performance of the experiments has been focused, and significant concentration has been devoted to the performance study of these evolutionary techniques in problems with multiple objectives. To investigate thoroughly, mostly optimization problems with more than three conflicting objectives have been studied; this helps us to see that the performance of the evolutionary techniques is severely thwarted, with the increment in the dimension of the objective space. A recent study [2] shows that for problems with more than ten objectives, a purely random search may perform favorably when compared with an evolutionary technique. From past researches, it has become quite evident that for single objective problems the performance of evolutionary algorithms can often be improved through the inclusion of a local search operator. The conglomeration of evolutionary algorithm with local searches are known by the names of Memetic Algorithms (MAs) [3], Hybrid Evolutionary Algorithms [4], [5], [6], and Genetic Local Search [7]. These hybrids of evolutionary and local search algorithms have been shown to provide state-of-the-art performance on a wide range of hard single objective combinatorial optimization

problems [8], [7], [9] and have also shown to provide a good performance on multiobjective optimization problems [10], [11], [12], [6], [13], [14].

In [15], authors applied genetic algorithms to solve the single objective Sailor Assignment Problem (SAP) and compared the results thus obtained with an existing algorithm, the GaleShapley algorithm [16]. The Gale-Shapley algorithm is a quadratic time algorithm for finding an optimal set of stable matches in the Stable Marriage Problem (SMP), and produces a set of marriages that are stable and optimal with respect to the preferences of one group in $O(n^2)$ steps. It has been proven that with the absence of the one objective of minimizing the total Permanent Change of Cost (PCS) in SAP, the results produced by the Gale-Shapley algorithm are an optimal set of assignments with respect to the preferences of the sailors and commanders; such a match may not necessarily involve all sailors. Based on these findings, this work investigated an alternative genetic algorithm approach to the SAP in order to allow outside constraints to influence the quality of the match. This work also explored the capability of the genetic algorithm (GA) to generate multiple acceptable solutions to SAP through niching techniques. The solutions produced by the GA were significantly better with respect to solution diversity/coverage and fulfilling the objectives of SAP when compared to the Gale-Shapley Algorithm. Due to the scalarization of the objective functions (converting multi-objective SAP to a single objective problem), the impact of a change in a weight vector on the solution is unpredictable. This advises for multiple runs of the algorithm with different parameters settings to get a broad view of the solution space. For this reason, diversity (coverage) of the solutions is highly dependent upon the ability of the algorithm to find Pareto optimal solutions, as well as the proper selection of parameters settings used.

For those problems to be surmounted, the authors in [17] augmented their previous work to straightway integrate each objective into the optimization problem. Thus, two well-known multi-objective evolutionary algorithms, NSGAII [18] and

SPEA2 [19] were developed to solve the multi-objective Sailor Assignment Problem. The results produced by these algorithms fell short of diversity when compared to CHC (a single objective genetic algorithm), repeated for many weight vectors. Then, a local search operator was designed and incorporated into the MOEAs with the sole intention to emphasize the importance of finding diverse solutions across the Pareto set approximation.

In [17], [15], rigorous experiments have been performed using evolutionary and hybrid algorithms on single and multi-objective versions of the SAP. In [20], a hybrid genetic algorithm came up with an efficient SAP solver, the Kuhn-Munkres algorithm [21, 22, 23] which was utilized to further improve performance. The Kuhn-Munkres algorithm solves linear assignment problems in $O(n^3)$ time, and was selected because single objective versions of SAP with small modifications can be represented as linear assignment problems. However, the main pitfall of this approach lies in the fact that it suffers high time complexity with increase in problem size, even though it produces very good solutions. The work [20] examined an approach to find more diversified solutions, which prompts for a combination of solutions produced by perfect solver KM with MOEA, in what is termed "informed initialization". According to [20], the informed initialization can be defined as the introduction of approximations of extreme and internal points of the optimal Pareto front in the initial population of MOEAs. These approximations, called informed initial solutions, are found using a high quality evolutionary or local search algorithm or a perfect deterministic solver (Kuhn-Munkres) algorithm on single objective problems. The work [20] also investigates the effect of these informed initialized solutions into MOEA by tuning the parameter of the number of informed initial solutions into the population of the MOEA while keeping the population size of MOEA constant.

1.2.2 Genetic Algorithm in Wireless Sensor Network

In the realm of wireless sensor networks, several researchers have been focusing on applying heuristic algorithms in the last five years. According to [24], regarding the wireless sensor network, researchers have to be ambidextrous to deal with many objective functions that are non-linear in nature. They also all need to be optimized simultaneously without losing generality. This scenario, in turn, prompts for concentrating on finding a set of non-dominated solutions that will be close enough to the optimal solutions in an acceptable computational time frame. In spite of there being several approaches to handle these criteria, many works have been performed using Genetic Algorithm (GA) [25] one of the robust heuristic algorithms that is appropriate for multi-objective problems. GA is solely intended for imitating the natural evolution process in computation domain by assigning a fitness value to each of the solutions in a single solution set, depending upon its degree of optimization of multiple objective functions. Then it applies the theory of ”survival of the fittest” among the solutions in the solution set. In the frame of GA, the solutions are represented in the form of a chromosome, and initially they are created randomly to fill up the population in the solution set. In the next step, the solutions undergo several genetic operators, from crossover to mutation, through selection of solutions to produce better solutions out of the randomly generated ones. The booming utilization of GA in wireless sensor network in [26], led to the construction of several GA-based application specific approaches to wireless sensor network by coming up with a single fitness function [27], [28], [29], [30].

The work in [29], demonstrated the efficiency of an approach based on GA to solve a communication optimization problem in wireless sensor network between sensor nodes and the base station. Due to the huge communication distance between sensor nodes and the base station, it is highly likely for the nodes to have their energies quickly depleted, thereby hampering the total lifetime of the network itself. The work

in [29] clusters the nodes, depending upon the minimum acceptable distance to the sink/base station, thus increasing the network lifetime. This work finally comes up with a set of cluster heads which are responsible for collecting data from its neighboring sensor nodes and conveying it to the base station. In this way it can reduce the rate of sensor nodes' energy depletion, thereby helping them to concentrate on their original purpose of sensing and collecting information from the terrain into which they are deployed. This work, even though depicting a novel approach and coming up with an appropriate set of solutions, lacks the complexity of dealing with multiple objectives simultaneously, as well as consideration of several application specific constraints.

Another work described in [31] attempts to amalgamate application specific requirements with network characteristics in the performance measure of GA. Initially, the algorithm locates the operational modes of the nodes in order to meet the application specific requirements, along with minimization of energy consumption by the network. Particularly, network design is explored in terms of active sensor nodes placement, and clustering and communication range of sensors; performance estimation includes, together with connectivity and energy-related characteristics, some application-specific properties like uniformity and spatial density of sensing points. Hence, the implementation of the proposed methodology results in an optimal design scheme, which specifies the operation mode for each sensor. This work is an excellent example of how genetic algorithm can be utilized to solve multiple objectives in wireless sensor network, but leaves room for future improvement; perhaps dealing with the design and implementation of heuristic techniques for optimal routing of dynamically selected cluster-in-charge sensor nodes through some multi-hop communication protocols, and algorithms could possibly be developed for dynamic integration of battery capacity.

The authors in [24] concentrated to present a multi-objective optimization methodology for self-organizing, adaptive wireless sensor network design and energy

management, keeping in mind the application-specific requirements, communication constraints and energy-conservation characteristics. This work depicted the novelty of the development of an integrated GA approach, both in the degrees of freedom of network characteristics and of application-specific requirements represented in the performance metric of the GA. This work is dedicated to find a dynamic sequence of operation modes for each sensor, i.e., a sequence of wireless sensor network designs, which will lead to maximization of network lifetime in terms of number of data collection (measuring) cycles. This is achieved by implementing the algorithm repeatedly in order to develop a dynamic network design that adapts to new energy-related information concerning the status of the network after each measuring cycle or at predefined time intervals.

According to [32], GAs and other learning algorithms use prior network knowledge to reduce energy consumption. Turgut et al. [27] used a GA to improve the performance of clustering algorithms in mobile ad-hoc networks in their work. This work represents wireless sensor network in the form of chromosomes. This encoded information is used to obtain the best solution (network topology) by applying biological operations such as crossover and mutation. In this approach, the best solution is based on the fitness function which defines the fate of an individual chromosome. In another direction, Jin et al. [29] used GA for reduced energy consumption in WSNs. In their work, a GA allows the formation of a number of pre-specified independent clusters, which assists in decreasing the total minimum communication distance. The results of this work depict that the number of cluster-heads is about 10% of the total number of nodes. The pre-specified cluster formation also reduces the communication distance by 80% as compared to the direct transmission distance.

Ferentinos et al. [33] extends the ideas proposed by Jin et al. [29] by improving the GA fitness function. They mostly concentrated on the optimization

properties of a genetic algorithm. The characteristics used in the design of the fitness function of a GA incorporate the status of sensor nodes, network clustering with the selection of appropriate cluster-heads, and the choice between two signal ranges from the normal sensor nodes. This improves the uniformity of measurement points and it reduces energy consumption. Hussain and Matin [34], [35] came up with a hierarchical cluster-based routing (HCR) protocol where nodes self-organize into clusters, and each cluster is managed by a set of associates called head-sets. Hussain et al. [32] improved the HCR protocol by using a GA to determine the number of clusters, the cluster heads, the cluster members, and the transmission schedules. A GA is used at the base station, which provides energy efficient solutions to the optimizer. This helps the base station in selecting the best cluster formation that will give minimum energy consumption during run time. After rolling through the analysis of the current network conditions, the base station applies the GA after every iteration, or set of transmissions. According to [32], the optimizer at the base station selects the best solution based on the acquired knowledge through the GA fitness function. The proposed fitness function is based on parameters such as energy consumption, number of clusters, cluster size, direct distance to sink, and cluster distance. Upon completion of iteration, the optimizer improves its decisions by receiving feedback, which is then used to adjust the weights of the parameters of the fitness function for the next iteration. In this paper, we provide thorough and rigorous investigation of the results and discuss the design and implementation issues of simulation.

The rest of this paper is organized as follows: Next section presents the Motivation behind applying MOEA to Task based Sailor Assignment problem (TSAP) and Coverage and Lifetime Optimization Problem in wireless sensor network (CLOP) in detail. Section II presents the problem statement which describes the problems of interest ‘TSAP’ and ‘CLOP’ in detail. In order to describe TSAP in detail, its root, Generalized Assignment Problem (GAP) is explored thoroughly and presented in

detail in this section. Proceeding with the flow of 'GAP', a more specific assignment problem, 'Linear Assignment Problem' (LAP), is described. This flow ends with the detailed portrayal of the problem 'TSAP'. The next subsection of this Section II sketches the problem of 'Coverage and Lifetime Optimization Problem' (CLOP) in detail. Section III then gives a brief overview of the multi-objective evolutionary algorithms utilized in this work, which encompasses Non-dominated Sorting Genetic Algorithm (NSGA-II) and Strength-Pareto-Evolutionary-Algorithm (SPEA-II). Section IV presents the proposed approaches and describes how they are applied to the problems of interest 'TSAP' and 'CLOP'. Specifically, several aspects including the manner the chromosome/solution are represented, fitness measures and genetic operators are portrayed, and details of experimental results are depicted with analysis and comparative study. Also in this section, special operators such as Local Search and Boundary Search, which combine with the algorithms, are described. In this work, a graphical user interface has also been implemented and clearly depicted in this section. Section V reports the conclusion and describes future work in this domain.

1.3 Research Issues

The sole motivation behind this work was to depict that a problem-specific genetic algorithm, if designed carefully, is sufficient to generate the Pareto front of a multi-objective optimization problem. This approach utilized two special operators, 'Local Search' and 'Boundary Search', and achieved solution diversity in the problem. Many real-life optimization problems can be expressed in terms of multi-objective optimization problems, and from the research works in past ten years, it has also been demonstrated that with the increase in problem specifications such as number of objectives, constraints etc., the time and computational complexity rise. It is highly unlikely that the best optimal solutions will be reached through these more

complicated problems with increasing conflicting objectives and constraints. Furthermore, most of these problems turn out to be NP-complete, for which exhaustive search is highly unrealistic. In this situation, to satisfy the need for the optimal balanced acceptable solutions, the approach of using multi-objective evolutionary algorithm proves beneficial.

The problem-specific motivation of utilizing evolutionary algorithms in solving complex real life multi-objective problems mostly depends on the complicated nature of the problems, as well as the overhead associated to the problems of this type. According to the United States Navy personnel policies, approximately every three years sailors serving on active duty need to be reassigned to different tasks. As a result, at any given time there exists a sizable population of sailors to be reassigned to available tasks. The Navy's goal is to identify sailor and task matches that maximize some overall criteria of "satisfiability" of sailors and commanders, and is referred to as the Sailor Assignment Problem (SAP). In this work, a fine-grained and more complex version of SAP, referred to as 'Task based Sailor Assignment Problem' (TSAP), was studied. In 'TSAP', sailors have to be assigned to different tasks in different time slots. In this case, instead of assigning a single task to a sailor, a sailor gets assigned to multiple tasks, which are distributed in a certain number of time frames over a day. Thus, the problem can be boiled down to a simplified frame: each day is divided into several time-slots, each time-slot consisting of several tasks, and sailors need to be assigned to particular tasks in that slot. Accordingly, in this 'TSAP' problem, a sailor can be assigned to different tasks in each time shift; in the case of SAP, a sailor cannot be assigned to two different tasks at the same time.

In reality, the efficacy of a possible solution to TSAP is not decided by a single metric; instead, it depends on a number of attributes that contribute to determine whether a solution is acceptable. Typically, each of these attributes is represented by a distinct objective function. Along with the objective functions, the problem also

embraces several constraints to be satiated. However, constructing the metric requires detailed knowledge of the structure of each objective function as well as the constraints associated with them. Thus, it can be difficult for a decision maker to specify an appropriate set of parameters to obtain a viable solution.

With the volumetric increase in problem specifications, the exhaustive search approach remains incapable of producing appropriate solutions for TSAP. This approach suffers from high time complexity issue, which in real-life situations is inadmissible. Moreover, the 'Task based Sailor Assignment Problem' is reduced to 'General Assignment Problem', which in turn is an NP-hard problem. Therefore, it is eventually inefficient to exploit the features of exhaustive search methodologies to solve TSAP. Genetic Algorithm plays a crucial role in this situation to overcome the difficulties incurred by the complicated nature of the problem. GA, after executing a huge number of evaluations of the problem, comes up with a set of optimal solutions in a reasonable amount of time. The final solutions, even though they cannot be claimed as the best solutions, are close to optimal.

Another challenging multi-objective optimization problem of 'Layout Optimization of wireless sensor network' [36] inspires me to dig into the problem because of its exigent demands of optimizing competing objectives while maintaining complicated constraints. Wireless Sensor Networks (WSNs) generally consist of a large number of cost-effective, low-power, multi-functional sensor nodes that are small in size and communicate over short distances [24]. Their mechanical, electronic, communication restrictions, and to some extent application-specific requirements, determine their structure and characteristics. In the field of wireless sensor networks, random deployment of sensor nodes are commonplace. However there are several current applications that incorporate some guidelines and insights that lead to the fabrication of an optimal design topology in terms of application-specific requirements and network infrastructure restrictions.

One of the major challenges in the layout of wireless sensor networks is the fact that energy resources of tiny sensor nodes are drastically restricted compared to wired networks. Due to the hostile nature of deployment terrain, recharging or replacing the energy source of the sensors in the network may be difficult or impossible, causing severe impact on the communication and processing time among all sensors in the network. Changing only a few sensor nodes' batteries between data processing and communication while keeping other sensor nodes unaltered might cause asynchronies in the network, which could make the deployment unsuccessful overall. Even failure of regular sensors may not harm the overall functioning of a wireless sensor network, since neighboring sensors can take over, provided that their density is high. Therefore, the optimal key parameter is network lifetime, or the time until the network gets exhausted.

Several of the latest military operations depicted the restrictions of surveillance missions performed by high-altitude platforms (UAV, U2, satellite), even when equipped with powerful sensors. Most of these restrictions are intrinsic to this type of long-distance surveillance and cannot be resolved with improvement in state-of-the-art sensor network domain research. The application needs information from the ground level, which in turn requires close-range observation and rejects the surveillance mission performed by the high-altitude platforms utilized previously. The close-range observation is performed by using remote sensing devices such as Wireless Sensor Networks (WSN) placed in the areas of interest. As these missions will be performed in hostile areas, the deployment of these sensors needs to be performed without human intervention, as through deployment from an aircraft. Once all the sensor nodes are placed on the ground, their data is transmitted back to the home base station to provide the necessary situational alertness.

The deployed sensor nodes gratify two fundamental functions: sensing data, and communicating that data to the home base station. Depending upon the structure

and characteristics of sensor nodes, sensing data is categorized into several types, such as acoustic, seismic, optical, chemical etc., and the communication is performed wirelessly. The challenge lies in the fact that their tiny size and energy storage capacity prevent them from relaying their gathered information directly to the base station. In order to make the mission successful, the sensor nodes need to transmit their data to a high-energy communication node (HECN) generally known as 'Base Station' (BS) which in turn is responsible for providing the transmission relay to an aircraft or a satellite. Therefore, the main objective of the sensor node deployment is to transmit their data to this node, either directly or via hops, using nearby sensors as communication relays.

The aircraft responsible for sensor node deployment has a limited payload, hence it is impracticable to randomly drop thousands of sensor nodes over an area of interest or else deterministically drop millions of sensors in a defined area, hoping the necessary communication connectivity would automatically be set up. Moreover, the hostile and rough nature of the ambient terrain such as wooded areas, mountains, country borders, hilly areas etc., hinder sensor nodes from executing their objectives; sensing and communicating. An additional possibility with aerial deployment is that airdrops might incur uncertainty in the final placement of the sensor nodes. Thus, the mission must be performed with a fixed maximum number of sensors. These criteria stir up the motivation to create an arrangement that automates the sensor nodes deployment process, keeping those requirements in mind.

Due to the limited number sensor nodes, it is highly unlikely to drop sensor nodes deterministically over the terrain. The overhead expenses involved in deploying sensor nodes in a hostile environment such as borders between countries, military environments, or rough terrains (seismic area, lava mountain, hilly area etc.) include several factors such as transportation cost, payload limitations of aircrafts etc. thereby making the entire procedure expensive as well as time consuming. Removing human

intervention from the process of node deployment is demanded, due to multiple reasons, e.g. hostility of environments, difficulty of access, need for sustained presence, and close-up presence for certain missions. All of these make the need for unmanned automated systems significant. To overcome these hurdles, Genetic Algorithm is chosen to produce an optimal set of solutions in an acceptable time frame.

2 Problem Statement

In this work, multi-objective optimization using evolutionary algorithms was used to solve the Task based Sailor Assignment Problem (TSAP), which is a more complex version of SAP, where sailors have to be assigned to different jobs (tasks) in different time slots. In TSAP, instead of assigning a single task to each sailor, each sailor gets assigned to multiple tasks, which are distributed in a certain number of time frames. Thus, it may be considered that each day is divided into several shifts, and sailors need to be assigned to particular tasks in each shift. Accordingly, a sailor can be assigned to different tasks in each time shift. The problem of 'TSAP' mainly originates from 'General Assignment Problem (GAP)' which was a more generic version of the assignment problem.

2.1 General Assignment Problem

The General Assignment Problem (GAP) is a generalization of the assignment problem that originally stepped in to the computational world as the problem of scheduling parallel machines with costs. GAP can be described as the following:

Given a set of bins and a set of items that have a different size and value for each bin, the objective of the problem is to pack a maximum-valued subset of items into the bins. GAP can also be exemplified in a job assignment scenario: a number of

agents n and a number of jobs m to be performed by the agents are provided. The conditions applied to this problem are: any agent can perform any job, but each agent has a budget, and the sum of resources required for jobs assigned to it cannot exceed its budget. When an agent is assigned to perform a task, it incurs some costs and resources associated with it. The solution to this problem is to find an assignment in which all agents stay within their budgets, and the total cost of the assignment is minimized.

Let b_i be the budget of agent i, let R_{ij} be the resources and C_{ij} be the cost incurred when agent i is assigned to perform task j, then the GAP can be expressed as the following linear integer program.

$minimize \sum_{i=1}^{n} \sum_{j=1}^{m} C_{ij} x_{ij}$

with respect to the constraints

$\sum_{j=1}^{m} R_{ij} \leq b_i \forall i \in 1, 2, ..., n.$

$\sum_{j=1}^{m} \leq 1 \forall i \in 1, 2, ..., n.$

$\sum_{i=1}^{n} x_{ij} = 1 \forall j \in 1, 2, ..., m.$

$x_{ij} \in 0, 1 \forall i \in 1, 2, ..., n, j \in 1, 2, ..., m.$

2.2 Linear Assignment Problem

The Linear Assignment Problem (LAP) is the simplest form of the assignment problem, in which the number of agents and number of tasks are equal and every agent can perform every task, but each agent can perform only one task. Mostly in assignment problems, it has been noticed that the most crucial objective ”cost” needs to be optimized. Then the problem is equivalent to the problem of finding an optimum weight vertex matching in an n n cost-weighted complete bipartite graph, as shown in Figure 3.

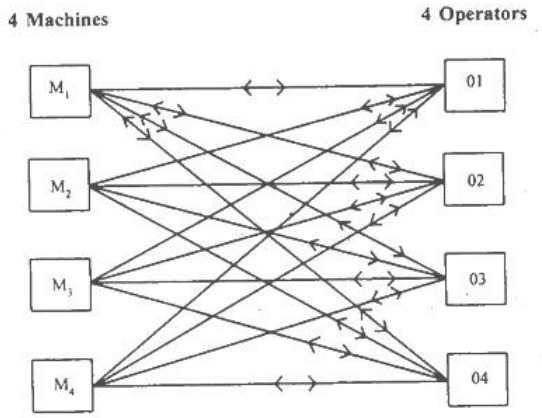


Figure 3: **Linear Assignment Problem**

Linear Assignment Problem can be formulated in a very simple way as follows: given a set of agents $A = \{a_1, a_2, ..., a_n\}$ and a set with the same number of tasks $T = \{t_1, t_2, ..., t_n\}$ and the cost function $C : A \times T \rightarrow R$, find a bijection/matching $m : A \rightarrow T$ such that the cost function: $\sum_{a \in A} C(a, m(a))$ is minimized. Usually the cost function is described as square real-valued matrix C with elements $C_{ij} = C(a_i, t_j)$. The overall linear assignment problem can be framed in the way depicted below.

$minimize \sum_{i=1}^{n} \sum_{j=1}^{m} C_{ij} x_{ij}$

with respect to the constraints

$\sum_{i=1}^{n} x_{ij} = 1 \forall j \in 1, 2, ..., n.$

$\sum_{j=1}^{n} x_{ij} = 1 \forall i \in 1, 2, ..., n.$

$x_{ij} \in 0, 1 \forall i, j \in 1, 2, ..., n.$

$x_{ij} = 1$ if agent i is assigned to perform task j.

$x_{ij} = 0$ otherwise.

2.3 Sailor Assignment Problem

According to US Navy's policy, [15] every three years, each sailor in the Navy is required to change jobs. As a result, at any given time, there is a sizable population of sailors who require assignments to a new jobs. The problem the Navy faces is to find a set of assignments, also called a match/sailor-task assignment, of sailors to jobs which keep the sailors happy and maintain fleet readiness while minimizing the cost of implementing those assignments. Many factors go into determining a good set of assignments. The Navy must ensure that not only are the sailors and commanders satisfied with the assignments, but also that each match can be implemented within a fixed budget. If the resulting cost is too expensive, then the Navy cannot adequately maintain its other priorities. However, if the assignment is overly-focused on cost, it may be that the sailors are unhappy with their assigned jobs. This could lead to a decrease in morale, followed by decreased retention rates and other problems.

Formally, the problem may be defined as follows.

$maximize \sum_{i=1}^{n} \sum_{j=1}^{m} F_{ij} d_{ij}$

subject to

$\sum_{i=1}^{n} d_{ij} \leq 1 \forall j \in 1, 2, ..., m$ and,

$\sum_{j=1}^{n} d_{ij} \leq 1 \forall i \in 1, 2, ..., n.$

where F_{ij} denotes the fitness of assigning sailor i to job j and D is an assignment matrix such that

$d_{ij} = 1$: Sailor i is assigned to job j

$d_{ij} = 0$: Otherwise

2.4 Task based Sailor Assignment Problem

Assignment problems, in general, were introduced as the problem of scheduling parallel machines with costs [37]. Sailor Assignment Problem is one sort of assignment problem in which sailors are to be fit into tasks assigned for them. Now in my work, I have chosen a more complicated version of the problem called 'Task based Sailor Assignment Problem', which encounters four competing objectives along with three different constraints. This problem is specifically designed for naval bases of the United States Navy, where there are a limited number of sailors, and even in a day the same task may have to be done by several different sailors. So depending on the requirement, a day is divided into different time shifts where different tasks have to be completed by sailors. However, the same task may not be repeated the next day or the next time shift, requiring the sailor to shift to another task. So this shows that a sailor may have to do different tasks in a day, but the same sailor cannot perform more than one task in the same time shift. The goal here to is minimize the number of sailors working on the naval base, along with all other objectives of SAP which are described below.

In short, instead of assigning a single job or task to a sailor, a sailor has to perform multiple tasks, which are distributed throughout a certain number of time frames. Thus, each day is divided into several shifts, and sailors need to be assigned to particular tasks that are required to be completed in those shifts. Accordingly, a sailor can be assigned different tasks in different shifts. Similar to SAP, here each sailor is examined for a variety of qualifications and constraints to determine valid matches for the tasks, and a score is computed determining the extent to which the sailor is a valid match. The training score encompasses many factors, including the pay grades of the sailor and the proposed job, and the amount of training required for the sailor to be able to perform the duties of the proposed job, among others. In addition, the monetary cost of assigning the sailor to the proposed job is computed.

After all candidates have been identified, the sailors are allowed to rate those tasks for which they are qualified. The final sets of solutions must be in compliance with Naval rules and regulations at reasonable costs to the Navy. This work has emphasized assigning sailors to only those tasks for which they are suitable and have the capacity and resources to perform. The problem of TSAP can be mathematically expressed as the following.

Let n be the number of sailors, m be the number of task classes and t be the number of time slots. Any eligible sailor can be assigned to any task in a time slot. Each sailor has his own capacity and consumes some resources for doing some tasks assigned to him. The problem is to find sailor-task assignment for each time slot in such a way that minimizes the necessary number of sailors and fulfills the previous objectives of SAP, namely, maximizing total training score, the sailor preference score, the commander preference score, and also minimizing the cost. Additionally, several constraints also are involved in this problem:

- The sum of the resources for a sailor-task assignment over given time should not exceed its capacity
- The same sailor cannot be assigned to multiple tasks in one time slot
- The number of sailors working in one time slot should be equal to the number of tasks required to be done in that time slot.

Let cap_i be the capacity of sailor i, R_{ij} be the resources and C_{ij} be the cost that incurs when agent i is assigned to perform task j. Let y_{jk} be the requirement of number of class j tasks to be performed in time slot k. TSAP can be formally expressed as the following multidimensional integer linear program:

Minimize $F(x)$ =($f_1(x)$, $f_2(x)$, $f_3(x)$, ..., $f_N(x)$)

where x = $(x_{ijk})_{i,j,k}$; f_{obj} = 1,2,.....,N are the objectives to minimize is defined as

$f_{obj}(x)$=$\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{t} x_{ijk} C_{ij}^{obj}$

where C_{ij}^{obj} represents the cost of assigning sailor i to task j over timeslot t determined by objective obj; and $F(x)$ is subject to the constraints depicted below:

$\sum_{j=1}^{m} x_{ijk} R_{ij} \leq cap_i; \forall i \in \{1, 2, ..., n\}, \forall k \in \{1, 2, ..., t\}.$

$\sum_{j=1}^{m} x_{ijk} \leq 1 \forall i \in \{1, 2, ..., n\}, \forall k \in \{1, 2, ..., t\}.$

$\sum_{i=1}^{n} x_{ijk} = y_{ijk}$

$x_{ijk} \in 0, 1; \forall i \in \{1, 2, ..., n\}, \forall j \in \{1, 2, ..., m\}, \forall k \in \{1, 2, ..., t\}$

The generalized assignment problem has been depicted to be NP-hard. In addition, recently, it was shown that it is $e/(e-1) - \epsilon$ hard to approximate for every ϵ. Therefore, it can be proved that TSAP is NP-complete as explained here. Since GAP does not have any time dependency in it, to reduce an instance of GAP to a TSAP problem, consider a single time slot (having entire tasks to be done by sailors in GAP). Here no sailor will repeat in each time slot, and the sum of the resources for sailor-task assignment will not exceed its capacities. Since there is only one time slot covering each task, it is ensured that all the tasks are assigned to different sailors. All the constraints are then satiated, and GAP instance is reduced to TSAP instance. Therefore by reduction process, it can be inferred that the solution for TSAP and GAP will be identical.

In this research work, a 'Multi-objective Evolutionary Algorithm' (MOEA) is proposed to solve the TSAP. The MOEA will produce a set of nondominated solutions which will not only optimize the objectives of the 'TSAP' but also fulfill the constraints associated with it.

2.5 Coverage and Lifetime Optimization in Wireless Sensor Network

Wireless Sensor networks (WSNs) consist of a huge number of tiny, cost-effective, power-efficient, multi-functional energy-constraint sensor nodes that can communicate over a specific distance [38]. A sensor node, popularly known as a 'mote', is particularly capable of performing sensing information from its surroundings, gathering that information, processing it and transmitting that information to other connected nodes in the network. The structure of the network entirely depends on the node's electronic, mechanical and communication restrictions, along with application-specific requirements [24]. In wireless sensor networks, sensor nodes are generally deployed either deterministically or randomly based on the application of interest [39]. A wireless sensor network is generally designed to be deployed in hostile environments such as battlefields, military environments, deserts, mountainous and border areas etc. for area monitoring, intrusion detection, environmental monitoring etc. Deployment in a war-front or hazardous area is generally random, whereas a deterministic deployment is preferred in amicable environments [31].

Wireless sensor nodes can be deployed in hostile environments in different ways, e.g. sparse deployment, dense deployment, etc. But in my work, mainly sparse deployment in a hostile environment is emphasized. The deployed sensor nodes primarily accomplish two fundamental functions: sensing surroundings (optical, seismic, acoustic, chemical etc.) and communicating to the base station/sink, which is done wirelessly. Due to energy-constraints of the sensor nodes, it is simply impossible for all sensor nodes to communicate to the sink for elongated period of time. This can shorten the lifetime of the node, thereby reducing the entire lifespan of the network. Consequently, another node of high energy needs to relay the information to a high

altitude or an aircraft. This calls for the existence of a base station, which plays a significant role in relaying information of all sensor nodes to high altitudes or to an aircraft or satellite. The aircraft that is responsible for deploying sensors is of limited payload, so it is highly unlikely to supply an unlimited number of sensor nodes over the area of interest.

Hostile environments are not conducive for human intervention to provide battery replacement in the sensor nodes. Hence it has become the primary requirement for the sensor network design to plan the position of the sensor nodes in such a way that the lifetime of the entire network is maximized. On the other hand, due to the limited number of sensor nodes in design, the mission of covering the area as much as possible has turned out to be an additional objective. Regarding the characteristics of sensor nodes, the sensing range and communication range incur one constraint in this design model. The constraint referred to as the connectivity constraint keeps a restriction on the sensor node placement in the terrain, which is primarily to maintain the connections among all the sensor nodes. The objective of maximizing area coverage always tends to place sensor nodes far away from the sink, so as to cover maximum area. But if the distance between two nodes becomes so large that it disconnects any one of them from the entire network, then the information from the sensor node will not be able to reach the base station, thereby making the sole purpose of the wireless sensor network lost. These issues call for simultaneous optimization of more than one non-linear design optimization while keeping connectivity constraint satisfied.

3 Multi-objective Evolutionary Algorithms

3.1 NSGA-II

NSGA-II stands for Non-dominated Sorting Genetic Algorithm. The main motivation behind this algorithm comes from the pitfalls of state-of-the-art classical methods. The idea behind the non-dominated sorting procedure is that a ranking selection method is used to maintain the stable subpopulation of good solutions. It differs from simple genetic algorithm only by the way it performs in selection operator; the crossover and mutation operators are unaltered. The Non-Dominated Sorting Genetic Algorithm II [18] exploits the notion of a domination level to assign fitness. All non-dominated individuals are assigned level zero. Upon removing those individuals from the population, the newly non-dominated solutions are assigned level one. The process continues until all solutions have been assigned a non-domination level. NSGA outperforms SPEA in terms of maintaining a better diversity of solutions and convergence in the obtained non-dominated Pareto front. Like SPEA2, NSGA-II then employs a density estimation technique to determine which solutions may be removed with the least impact on diversity. According to [18], the algorithm flows as depicted in Figure 4. The way the algorithm NSGA-II selects individual solution, draws a distinct boundary between NSGA-II and other state-of-the-art genetic algorithms. The crossover and mutation operator performs the same way as in all other GAs.

3.2 SPEA-II

According to the authors of [19], the algorithm SPEA-II originates from SPEA (Strength Pareto Evolutionary Algorithm) [40]. Starting with an initial population and

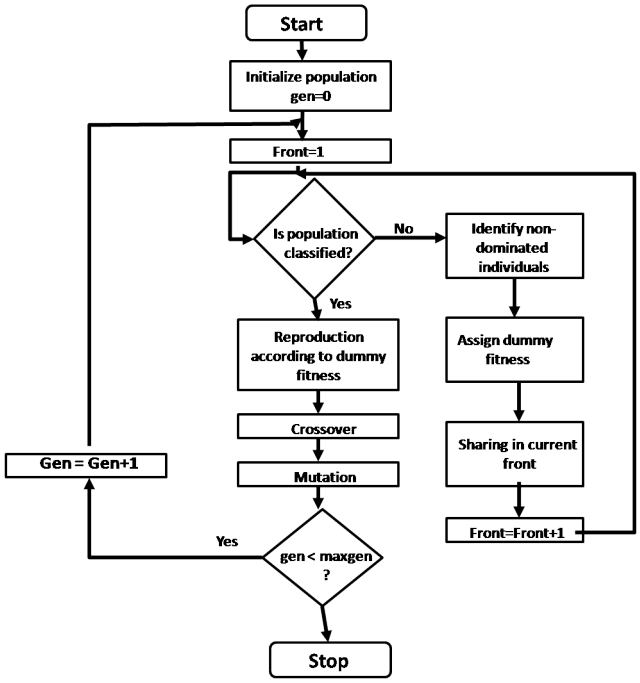


Figure 4: **Non dominated Sorting Genetic Algorithm**

an empty archive the following steps are performed per iteration. First, all non-dominated population members are copied to the archive; any dominated individuals or duplicates (regarding the objective values) are removed from the archive during this update operation. If the size of the updated archive exceeds a predefined limit, further archive members are deleted by a clustering technique which preserves the characteristics of the non-dominated front. Afterwards, fitness values are assigned to both archive and population members:

- Each individual i in the archive is assigned a strength value $S(i) \in [0; 1)$, which at the same time represents its fitness value $F(i)$. $S(i)$ is the number of population members j that are dominated by or equal to i with respect to the objective values, divided by the population size plus one.

- The fitness $F(j)$ of an individual j in the population is calculated by summing the strength values $S(i)$ of all archive members i that dominate or are equal to j, and adding one at the end.

The next step represents the mating selection phase where individuals from the union of population and archive are selected by means of binary tournaments. As the fitness is to be minimized here, each individual in the archive has a higher chance to be selected than any population member. Finally, after recombination and mutation, the old population is replaced by the resulting offspring population. Although SPEA performed well in different comparative studies ([19]; [41]), there is still room for improvement as recent studies ([42]; [43]) have shown. In particular, the following issues have been identified as potential weaknesses of SPEA:

Fitness Assignment: Individuals that are dominated by the same archive members have identical fitness values. That means in the case when the archive contains only a single individual, all population members have the same rank independent of whether they dominate each other or not. As a consequence, the selection pressure is decreased substantially, and in this particular case SPEA behaves like a random search algorithm.

Density Estimation: If many individuals of the current generation are indifferent, or, do not dominate each other, none or very little information can be obtained on the basis of the partial order defined by the dominance relation. In this situation, which is very likely to occur in the presence of more than two objectives, density information has to be used in order to guide the search more effectively. Clustering makes use of this information, but only with regard to the archive and not to the population.

Archive Truncation: Although the clustering technique used in SPEA is able to reduce the non-dominated set without destroying its characteristics, it may lose outer solutions. However, these solutions should be kept in the archive in order to obtain a good spread of nondominated solutions.

To overcome the pitfalls of SPEA, a new revised version SPEA-II was designed. The algorithm works as follows:

Algorithm:

Input: N (Population size)

$\overline{N}$

T (maximum number of generations)

Output: A (non-dominated set of solutions)

Step1: **Initialization:** Generate an initial population $\mathbf{P}_0$ and create the empty archive (external set) $\mathbf{P}_0 = \emptyset$;. Set $t = 0$.

Step2: **Fitness assignment:** Calculate fitness values of individuals in P_t and P_t (Refer to Section 'Fitness Assignment').

Step3: **Environmental selection:** Copy all nondominated individuals in P_t and P_t to P_{t+1}. If size of P_{t+1} exceeds N then reduce P_{t+1} by means of the truncation operator, otherwise if size of P_{t+1} is less than N then fill P_{t+1} with dominated individuals in P_t and P_t (Refer to Section Environmental Selection).

Step4: **Termination:** If $t \geq T$ or another stopping criterion is satisfied then set A to the set of decision vectors represented by the nondominated individuals in P_{t+1}. Stop.

Step5: **Mating selection:** Perform binary tournament selection with replacement on P_{t+1} in order to fill the mating pool.

Step6: **Variation:** Apply recombination and mutation operators to the mating pool and set Pt+1 to the resulting population. Increment generation counter $(t = t + 1)$ and go to Step 2.

Compared to SPEA, SPEA2 uses a fine-grained fitness assignment strategy which incorporates density information to make it more robust.

3.2.1 Fitness Assignment

To avoid the situation in which individuals dominated by the same archive members have identical fitness values, with SPEA2 for each individual both dominating and dominated solutions are taken into account. In detail, each individual i in the archive P_t and the population P_t is assigned a strength value $S(i)$, representing the number of solutions it dominates:

$$S(i) =\mid j \mid j \in P_t + \overline{P_t} \bigwedge i \succ j \mid$$

where $\mid . \mid$ denotes the cardinality of a set, + stands for multiset union and the symbol $\succ$ corresponds to the Pareto dominance relation. On the basis of the S values, the raw fitness $R(i)$ of an individual i is calculated:

$$R_i = \sum_{j \in P_t + \overline{P_t}, j \succ i} S(j)$$

In the aforementioned way, raw fitness is determined by the strengths of its dominators in both archive and population. As fitness is to be minimized here, i.e., $R(i) = 0$ corresponds to a nondominated individual, while a high $R(i)$ value means that i is dominated by many individuals (which in turn dominate many individuals). Although the raw fitness assignment provides a sort of niching mechanism based on the concept of Pareto dominance, it may fail when most individuals do not dominate each other. Therefore, additional density information is incorporated to discriminate between individuals having identical raw fitness values. The density estimation technique used in SPEA2 is an adaptation of the k-th nearest neighbor method (Silverman 1986), where the density at any point is a (decreasing) function of the distance to the k-th nearest data point. In this point, only the inverse of the distance to the k-th nearest neighbor is taken as the density estimate. To be more precise, for each individual i the distances (in objective space) to all individuals j in archive and population are calculated and stored in a list. After sorting the list in increasing order, the k-th element gives the distance sought, denoted as σ_i^k . As a common setting, we use k equal to the square root of the sample size (Silverman 1986), thus,

$k = \sqrt{N + \overline{N}}$. Afterwards, the density $D(i)$ corresponding to i is defined by

$$D(i) = 1/(\sigma_i^k + 2)$$

In the denominator, two is added to ensure that its value is greater than zero and that $D(i) < 1$. Finally, adding $D(i)$ to the raw fitness value $R(i)$ of an individual i yields its fitness $F(i)$: $F(i) = R(i) + D(i)$ The run-time of the fitness assignment procedure is dominated by the density estimator (O($M^2 logM$)), while the calculation of the S and R values is of complexity O(M^2), where $M = N + \overline{N}$.

3.2.2 Environmental Selection

The archive update operation in SPEA2 differs from the one in SPEA in two respects:

- The number of individuals contained in the archive is constant over time.
- The truncation method prevents boundary solutions being removed.

During environmental selection, the first step is to copy all nondominated individuals, i.e. those which have a fitness lower than one, from archive and population to the archive of the next generation: $\overline{P_{t+1}} = i \mid i \in P_t + \overline{P_t} \bigwedge F(i) < 1$.

If the nondominated front fits exactly into the archive ($\mid \overline{P_t} \mid = \overline{N}$) the environmental selection step is completed. Otherwise, there can be two situations: Either the archive is too small ($\mid \overline{P_t} \mid < N$) or too large ($\mid \overline{P_t} \mid > N$). In the first case, the best ($\overline{N} - \mid \overline{P_t} \mid$) dominated individuals in the previous archive and population are copied to the new archive. This can be implemented by sorting the multiset $(P_t + \overline{P_t})$ according to the fitness values and copy the first ($\overline{N} - \mid \overline{P_t} \mid$) individuals i with $F(i) \geq 1$ from the resulting ordered list to $\overline{P_t}$. In the second case, when the size of the current nondominated (multi)set exceeds N, an archive truncation procedure is

invoked which iteratively removes individuals from $\overline{P_t}$ until $\overline{P_{t+1}} = N$. Here, at each iteration that individual i is chosen for removal for which $i \leq_d j; \forall j \in \overline{P_{t+1}}$ with

$$i \leq_d j :\Leftrightarrow \forall 0 < k < |\overline{P_{t+1}}| : \sigma_i^k = \sigma_j^k \bigvee$$
$$\exists 0 < k < |\overline{P_{t+1}}| : [(\forall 0 < l < k : \sigma_i^l = \sigma_j^l) \bigwedge \sigma_i^k < \sigma_j^k]$$

where σ_i^k denotes the distance of i to its k-th nearest neighbor in $\overline{P_{t+1}}$. In other words, the individual which has the minimum distance to another individual is chosen at each stage; if there are several individuals with minimum distance, the tie is broken by considering the second smallest distances and so forth. The average time complexity is approximately $O(M^2 \log M)$ where $M = (N + \overline{N})$.

4 Proposed Approach for Task based Sailor Assignment Problem

In this section, an MOEA, based on NSGA-II , used to solve Task based Sailor Assignment Problem is presented in detail.

4.1 Chromosome Representation

One of the crucial decisions to be made when using a genetic algorithm to a particular problem is how to encode solutions to the problem in a way that is aligned to genetic search. In this work, we utilize an integer encoding methodology. A chromosome consists of a set of integers, each representing a sailor assigned to perform a particular task. Each of the sailors and tasks are given a unique integer number. The length of the chromosome is equal to the total number of tasks that should be assigned to sailors along all the time slots. Each integer is chosen from a

subset of possible numbers, precisely those representing valid sailors for that task. In this way, we try to ensure that only those sailors qualified to perform a task may be assigned. However, it may be possible that in the middle of the search process, no sailor can be assigned to a task, then a '-1' is assigned meaning that this task has not yet been assigned.

A linear chromosome representation is used as follows: the week is divided into number of days d which are subdivided into k time slots. Each time slot contains a variable number of tasks to be done by eligible sailors. Figure 5 shows the representation of chromosome where each day is divided into multiple time slots and tasks contained in each time slot are done by the eligible sailors; here, t_m;j represents the task class m and its j-th instance to be done by sailor S_i.

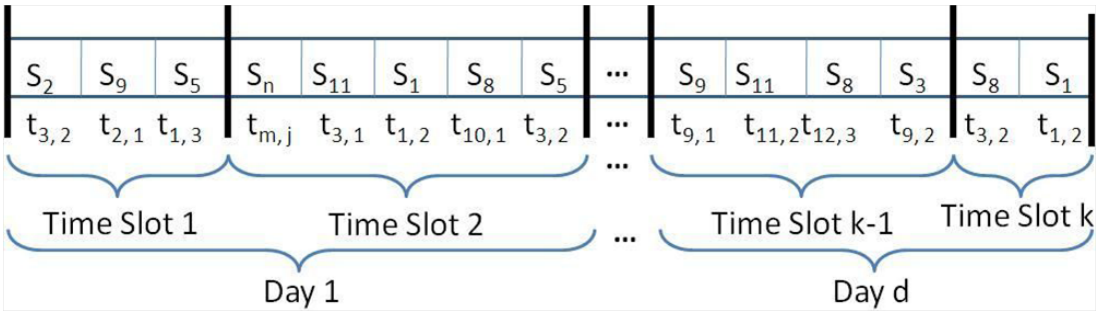


Figure 5: **Chromosome Representation**

Unwrapping a chromosome, C, is simply performed by assigning sailor S_i to a corresponding task on location i of the chromosome, which occurs at the corresponding time slot k. It is important to note that tasks are associated with corresponding sailors but they are not explicitly specified in the chromosome, which contains sailors only. Therefore it is necessary to map the sailors to tasks in the decoding process. Note that in Figure 3 a random crossover or mutation could easily result in a violation of the constraints. The second and third constraints of TSAP (in the section below) ensure that feasible solutions do not assign two different tasks to the

same sailor in a particular time slot, that the sailor's capacity will not be exceeded, and that each task is assigned a sailor to perform it. However, the proposed encoding cannot prevent violations of the constraints from occurring. Note also that it is possible for a sailor or task to be left unassigned.

4.2 Objective and Fitness Measure

As mentioned in Section 'Problem Statement', the objective functions of this entire problem are following:

- Minimize the number of sailors assigned to navy tasks.
- Minimize the sailor-task assignment cost:
  $\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{t} x_{ijk} C_{ij}$
- Maximize the training score (TS) of a particular sailor:
  $\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{t} x_{ijk} TS_{ij}}{\sum_{i=1}^{n} \max_{j=1}^{m} TS_{ij}}$
- Maximize the sailor preference (SR) of specific tasks:
  $\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{t} x_{ijk} SR_{ij}}{\sum_{i=1}^{n} \max_{j=1}^{m} SR_{ij}}$
- Maximize the commander preference (CR):
  $\frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{t} x_{ijk} CR_{ij}}{\sum_{i=1}^{n} \max_{j=1}^{m} CR_{ij}}$

Even though some of the objectives are to be maximized, the multi-objective optimization process used in this work is represented in such a way to minimize all of the five objectives defined above, which are assumed to be normalized in [0, 1]. It is also worth noticing that the proposed approach can be easily extended to consider additional objectives.

Moreover, it is important to note that while in principle, these values are scaled such that the fitness values range from 0 to 1, in practice the extreme values never occur. . The reason behind this is that multiple sailors are in contention for the same tasks. Also it is assumed that each sailor can be assigned to tasks which maximize or minimize the values of each objective function. However, constraints make it very unlikely that all such requirements can be simultaneously accomplished. Thus, the actual upper and lower bounds for each objective are unknown for any problem instance of any size. As a result, it is unlikely that a single solution (assignment) reaches 1.0 value in all or in any objectives.

A crucial issue here is the assurance that all necessary constraints (discussed in the Problem Statement) need to be satisfied. Therefore, during the evolutionary optimization process, an infeasible solution is penalized for constraint violation as explained below. These necessary characteristics of the TSAP problem are taken into account by the inclusion of the following penalization functions:

- **Unassigned Tasks Penalty (UTP):** This value is used to penalize solutions which contain unassigned tasks and is defined as:

  $UTP = \frac{Numberofunassignedtasks}{Totalnumberoftasks}$

  The task has been marked as '-1' in the assignment where no sailor has been assigned. The situation is depicted in Figure 6.

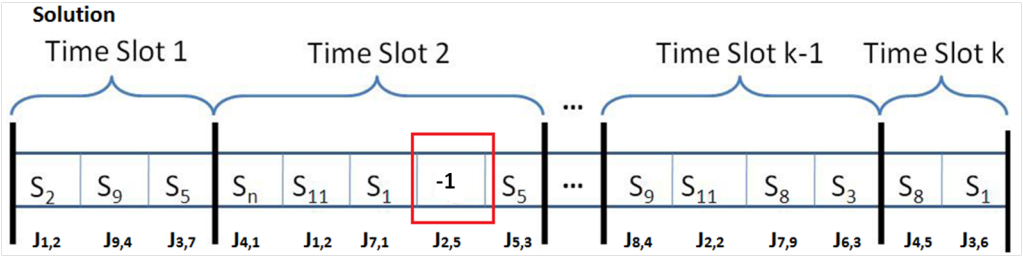


Figure 6: **Unassigned Task in the solution**

In this solution, it is clearly depicted that the 5-th instance of 2-nd task has not been assigned to any sailor.

- **Redundant Sailors Penalty (RSP):** This value penalizes solution containing same sailors repeated in a single time slot and is computed as:

  $RSP = \frac{No.of redundant sailors}{Total no. of tasks}$

  This situation might come into a single solution where one sailor may be repeated in a single time-slot which is depicted in the Figure 7.

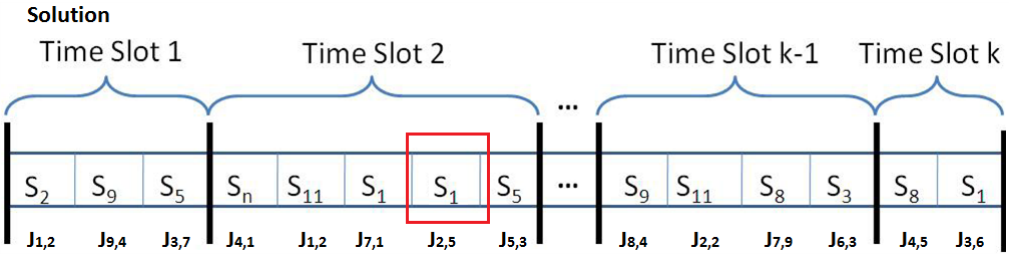


Figure 7: **Redundant Sailors in the solution**

- **Capacity Exhaustion Penalty (CEP):** CEP enforces penalty in a solution once it finds a sailor assigned to a task when his capacity has already been exhausted in that solution. It can be calculated as:

  $CEP = \frac{Capacity exhaustion}{Sum of max. resources required to do all tasks}$

The penalization value is generated from the combination of the three different penalties (UTP RSP CEP) described above. Then this penalty value is divided, and an equal portion of it is added to each objective; thus the penalization is equally distributed among all the objectives.

4.3 Genetic Algorithm and Operators

This problem, being an NP-hard problem needs to be solved by a special robust evolutionary algorithm. The Non-dominated Sorting Genetic Algorithm II (NSGA-II) has been applied to this problem, along with several problem specific operators carefully designed to suit this problem. In this work, problem specific selection, mutation and crossover operators are implemented. It is worth noting that mutation and crossover operators can produce solutions that might violate the constraints. Therefore, a repair operator is implemented to try to maintain feasible solutions. Along with them, two special problem specific operators are designed so as to enhance the solution diversity as well as solution quality. Two special operators, Local Search operator and Boundary Search operator, are applied to the solutions generated to find a better solution than the one obtained. Local Search operator is designed in two different ways: one technique shifts a specific sailor with another available and appropriate sailor, if and only if the new solution is better than the previous one with respect to objective function evaluation. In the other method, a sailor in an assignment gets swapped with another sailor in the same assignment, provided this does not violate the constraints.

The Boundary Search approach works in a distinct way. In most of the works on optimization problems, it has been noticed that the optimization solutions are highly likely to be found near the zone of boundary, or near the constraints. Keeping these works in mind, this work tries to aid genetic search with providing some solutions lying near the boundary in some special generations. This operator works by replacing an infeasible solution by the boundary solution after checking whether the fitness value is acceptable in the new solution. One simple mutation operator chooses a particular task at random, and then assigns a new qualified sailor to that task. However, after mutating a solution, the capacity of the newly assigned sailor could be exhausted. Also, a crossover operator is implemented by swapping the sailors assigned to do all

the tasks in a particular time slot in two solutions. After crossover, the redundant sailor constraint is not violated, since a whole time slot is swapped between two solutions. However, as in mutation, after performing crossover, it is likely that the offspring violates the capacity constraint. Thus, these genetic operators are described in the following sections.

4.3.1 Crossover Operator

The crossover operator picks a time slot at random from one parent and swaps it with the same time slot of another parent. This operator does not need to check for redundant sailors on the same time slot that is being swapped, as the entire time slot is swapped instead of single sailor of the time slot. However, after crossover, the capacity of some sailors may be exhausted. The crossover operator is depicted in Figure 8 and 9.

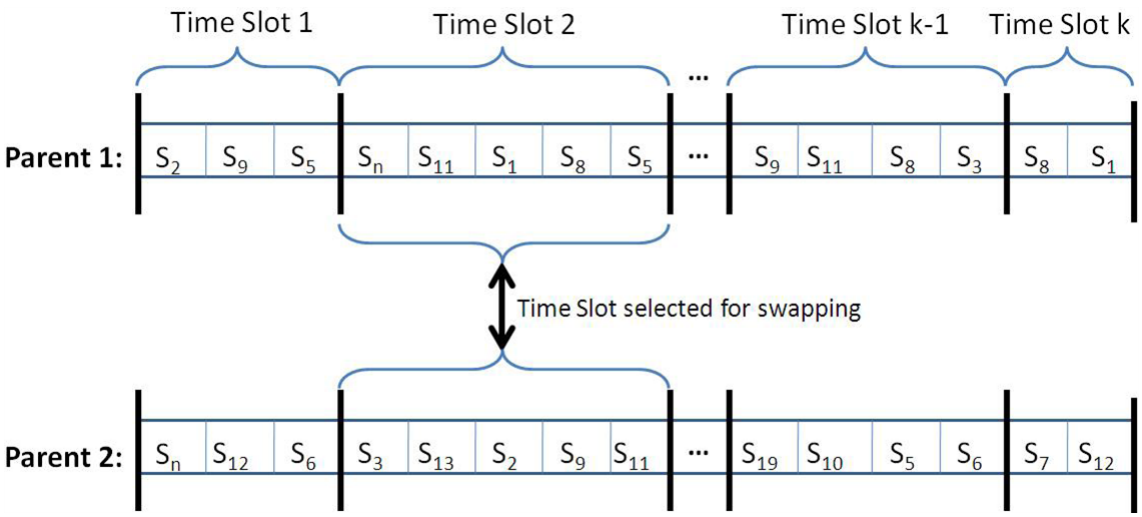


Figure 8: **TSAP Solution Before Crossover**

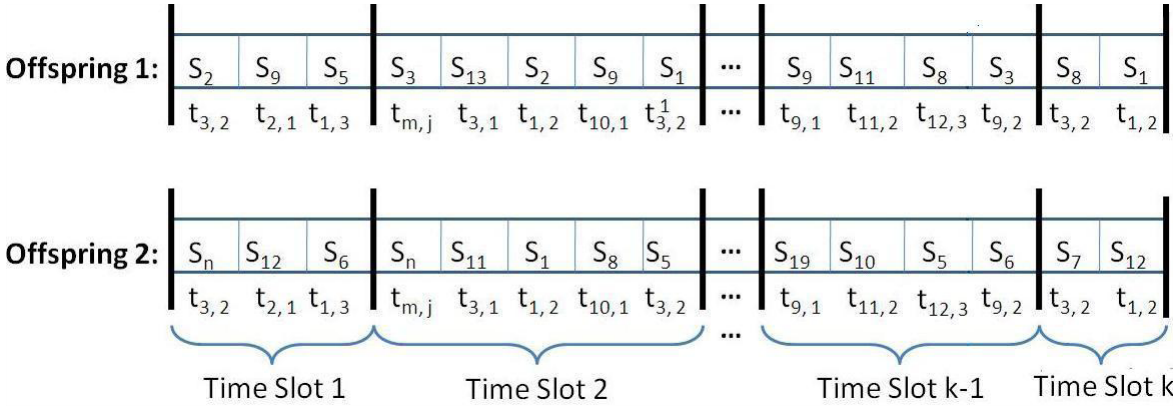


Figure 9: **TSAP Solution After Crossover**

4.3.2 Mutation Operator

The mutation operator works as follows: Initially choose one random location from the chromosome. Find out which sailor has been assigned to that task, then determine the list of possible sailors who can perform that task. Randomly pick a sailor from that list, substituting the current sailor with the new one, keeping two constraints in mind: that the new sailor is not repeated in that time slot, and that he has enough capacity to perform that particular task. This process continues until a new sailor can be assigned to that task, or all the eligible sailors are checked, in which case the sailor assigned to that task remains unchanged.

It is worth noting that given a specified mutation rate Pm, the effective mutation rate, the percentage of attempted mutations that actually effect some change in the individual, is less than Pm, since only some of the attempted mutations are completed successfully. As the proper setting of the mutation rate is a crucial aspect of any evolutionary algorithm, it is important to understand the relationship between the specified mutation rate and the effective mutation rate on the TSAP. Clearly, this relationship is intricately linked to the amount of contention for tasks, which in turn depends critically on the ratio of tasks to sailors in a particular instance of the problem. As this ratio increases, the likelihood of finding a sailor to perform a particular task that is not currently assigned to another sailor increases, thereby increasing the

percentage of mutations that can be completed successfully. Randomization of a newly created chromosome utilizes the same process as the mutation operator described above. Each task is assigned to a randomly selected qualified sailor if one is available. If no sailor is available for performing a particular task, then this randomization process regards that task as unassigned. Even in this randomized procedure, the order in which the chromosome is traversed is random in order to prevent bias toward assigning lower numbered tasks over higher numbered tasks. The mutation operator is depicted in Figure 10.

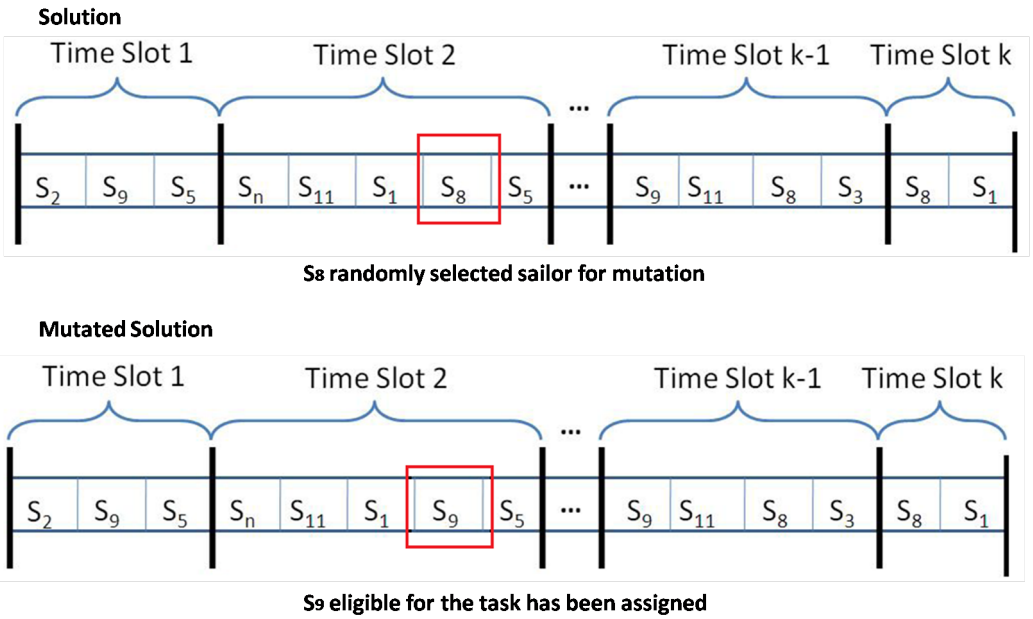


Figure 10: **Mutation Operator**

In Figure 10, it is clearly depicted how the mutation is performed in the problem of TSAP. Once the solution is produced for being processed through a mutation operator, a random sailor has been selected from the solution (S_8 in the Figure 10). Once the task is determined where the selected sailor was assigned, another eligible sailor has also been located. Before replacing the existing sailor with

the newly found eligible sailor, it is verified against all constraints, and then the existing sailor is replaced (S_9 in Figure 10).

4.3.3 Repair Operator

A repair operator is designed to attempt to recover from possible constraint violations after mutation and crossover in every iteration and works as follows. First, the chromosome is traversed to look for unassigned tasks and for each unassigned task a qualified sailor whose capacity has not been exhausted is assigned to this task. If no sailor is available to perform the task, it remains unassigned.

Neither mutation nor crossover operators will come with unfeasible solutions that contain redundant sailors in a single time slot. Hence, this repair operator does not have to take care of this constraint violation. However, this operator was provided to deal with this constraint violation because the hybrid approaches described in next section implement some local search operators that can produce such infeasible solutions. In such case, each time slot is checked for redundant assigned sailors in the repair operator. If a solution contains the same sailor assigned to two different tasks in the same time slot, then the list of eligible sailors for one of these tasks, chosen randomly, is traversed and a new sailor is assigned to it. If, after trying to assign a new sailor to that particular task, this constraint is still violated, the task is marked as unassigned.

In order to perform a repair mechanism on the sailor capacity constraint, a sailor capacity exhaustion is defined as the number of extra resources (hours, for example) a particular solution assigns to a particular sailor. Accordingly, for each sailor whose capacity has been exceeded, the repair operator attempts to assign a different sailor to the task he has been assigned, until the capacity is no longer

exhausted. Note that even after repair it is likely that the capacity of some sailors remains exhausted.

Since there is no known method to solve TSAP, it is difficult to evaluate the performance of the proposed approach. Therefore, in this work the GA is enhanced by including some local search operators and boundary search operators in a hybrid approach as explained next, and thus the results of several approaches on different problem instances are compared.

4.3.4 Local Search Operator

The local search operator creates vivid differences between solutions generated, by evolving through regular operators in genetic algorithm and local search operators included in it in terms of solution diversity. The basic idea behind the local search operator is to find a better solution in the local arena of a solution. It usually picks a solution from a population and tries to find out if a solution around it is better in terms of fitness values. This work comes up with two different ways to find out the better solution local to an existing solution: Shift Local Search operator and Swap Local Search operator.

In shift local search operator, a solution is to be tried to search for another solution residing locally to it in the following way:
A sailor is being randomly picked from the solution, and the task, the sailor was performing, is being shifted to another eligible available sailor, keeping all constraints satisfied. Then the generated new solution goes through the procedure of checking if it is better in terms of fitness value than the original solution. On reception of a 'yes' from the previous step, it replaces the original solution in population with the new solution generated by local search. On the way around, the original solution does not

get replaced by the new local search generated solution if it turns out to be the better one rather than the new solution with respect to fitness values.

Swap local search comes with a scintillating feature. It selects two random tasks from the current solution and gets the sailors working in those two tasks. Then it tries to swap two of them if they turn out not to be contradictory to each other. To be more specific, the swap local search operator first tries to match the sailors chosen to be swapped into two different tasks against the constraints (whether the sailors are eligible to perform the task, whether they have the capacity to work on the task, and whether they are being repeated in the time-slot into which they are to be placed). Once the sailor-task exchange operation is performed, the generated new solution undergoes the same investigation of fitness value. If the new solution generated is found to be more fit than the previous one, it replaces the previous solution in the population; otherwise the previous solution resides in its place. This operator is depicted in Figure 11

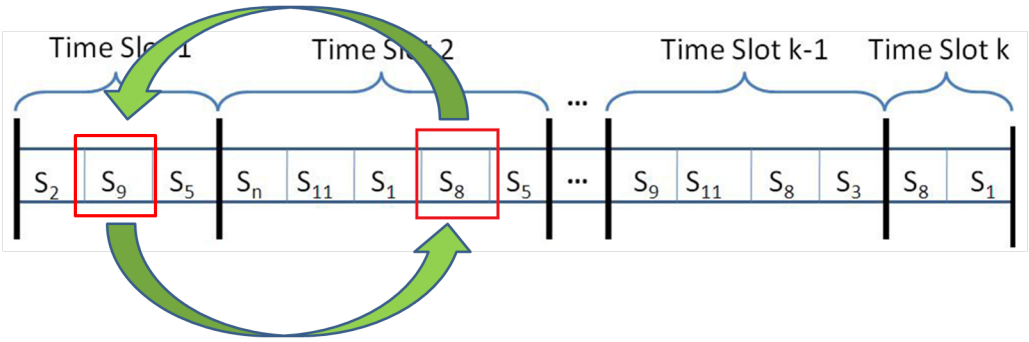


Figure 11: **Swap Local Search Operator**

Even though the local search operator contributes to reaching towards achieving better optimal solutions in the whole population, but it is found to be highly expensive in terms of computation and time. Therefore, this operator is highly unlikely to be executed in each generation of evolutionary algorithm. So, the operators are being applied onto each solution in the population in every k-th evaluation iteration

(where k can be any constant value such as 2000, 592 etc.) out of total evaluation n (where $n \geq k$).

4.3.5 Boundary Search Operator

Boundary Search operator plays a crucial role in solving TSAP using multi-objective evolutionary algorithm. As mentioned earlier, it has been noted that in multi-objective optimization problems, the optimal solutions reside around the boundary zone of the problems. To be more specific, the optimal solutions are normally found around the boundary wall of the constraints of the problem. Digging into the root of the concept, this work comes up with the innovative idea of designing another special operator that will aid genetic search by feeding solutions lying near the boundary. This operator is being designed under the impression as well as with the hope that it will help genetic search to find the optimal solution more effectively by feeding solutions near the boundary region.

Boundary search operator works as follows: Solutions in the population set sometimes might turn out to be infeasible in spite of utilizing constraint-satisfying methodologies. This operator will come into play in this scenario; it picks up the infeasible solution (i.e. the solution which has violated any one of the constraints of the problem) and tries to find the nearest boundary of it. If the solution is found to be better with respect to fitness rather than the previous original infeasible solution, then it is fed back into the population in place of the previous one.

The way boundary search operator works is as follows: Once a solution is picked up, two random positions p_1 and p_2 in that solution are chosen. Then random time-frame window size k is selected based on the positions chosen so as not to exceed the chromosome (solution) length. Thereafter, the sailors assigned in the tasks of those tome-frame window sizes have been picked up and kept inside two different buffers $_1$

and s_2 respectively. In the next step, the sailors from s_1 are being placed in the positions starting at p_2 in a reverse order and the sailors from s_2 are also being placed in the positions starting at p_1 in a reverse order. This process is depicted in the Figure 12.

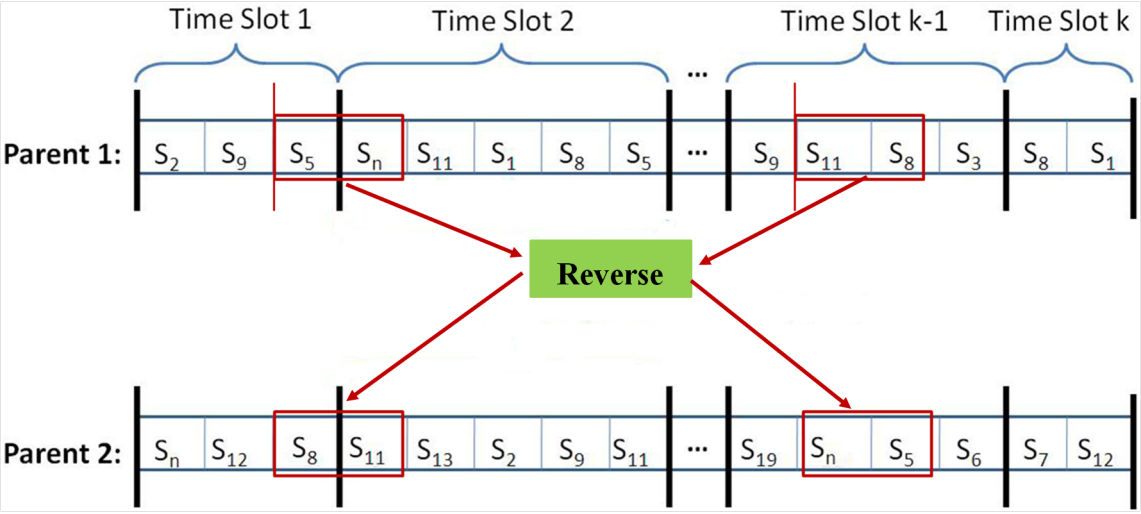


Figure 12: **Boundary Search Operator**

This operator has been found to be expensive in terms of time and computational complexity, which makes it infeasible to execute in every evaluation iteration. Based on the time it takes to execute, the algorithm has been fine-tuned to perform this boundary search operator in every p-th evaluation iteration out of total n evaluations where $p \leq n$ and $p \neq k$ mentioned in Local Search operator.

4.4 Experimental Results

In this work, a problem generator has been used for producing dummy data of sailor, task, time-slots per day, resources for sailors, etc. This parametric problem generator is capable of producing several instances of the problem with different parameters such as number of sailors and tasks, number of time-slots per day,

Table 1: **Parameters of the randomly generated problem instances used in TSAP**

Problem Instance	No. of sailors	No. of tasks	Task-Sailor ratio
Problem 1	1000	500	0.25
Problem 2	2000	1000	0.05
Problem 3	4000	2000	0.05
Problem 4	5000	200	0.01

resources for sailors, etc. During execution, it also maintains the feasibility of problem data. For example, the ratio of task and sailor should not be extreme (not very high or very low, but moderate), huge tasks should not be placed in time-slots that would exceed a sailor's availability, etc.

A large number of sample data files have been generated to execute the experiments with the multi-objective evolutionary algorithm NSGA-II. Each problem was produced maintaining a specified number of sailors, tasks and time slots. Along with it, the mean and standard deviation of a normally distributed random variable determine the number of task types each sailor can perform. In all the problems, seven days with twelve time slots per day were considered, which makes the problem harder since it increases the number of tasks for all the time slots. In **Table** 1 , the parameters of each sample problem used in this work have been depicted. The task-sailor ratio is defined here as the rate of task types a sailor can perform on average.

This work has undergone ten runs of the evolutionary algorithm NSGA-II for each problem and for three different population sizes: 100, 200 and 400 individuals where each run consists of 500,000 evaluations of the objectives. In these experiments, Binary tournament selection, where two individual solutions are chosen from the population at random and the fittest one in terms of objective values is selected, and mutation with probability (1.0/Total no. of tasks to be performed) were used. Also, all individuals in the population were chosen to be crossed, and the same amount of offspring is produced. Population of parents and offspring were combined and the best

among all of them go to the mutation process. Additionally, after performing crossover and mutation, it is highly likely for the solutions produced to become infeasible (i.e. sailors in the assignment may exhaust their capacities but still be assigned to tasks, one sailor may be repeated in a single time slot, etc.). To overcome the situation, the repair operator is executed thereafter. For doing theses experiments jMetal, an open framework for MOEAs was used.

Figure 13 depicts the change in the hypervolume in one run of NSGA-II for a 5,000 sailor instance of TSAP with 100 population size. It is quite evident form Figure 13 that after a certain number of evaluations (around 500,000) the hypervolume will remain almost constant which is a representation that the evolutionary algorithm has reached a level at which the results are very unlikely to improve.

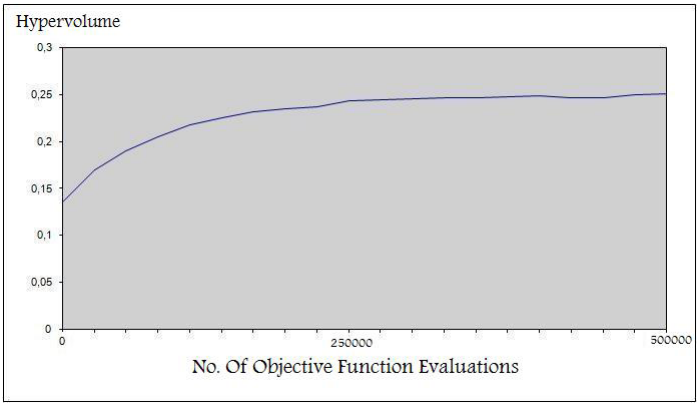


Figure 13: **Change in the hypervolume during 500,000 objective evaluations for a 5,000 sailor instance of TSAP with 100 population size**

Almost all existing performance metrics in multi-objective optimization require a set of Pareto-optimal solutions to be compared against the approximate solutions obtained. Most existing performance metrics for evaluating the distribution of solutions cannot be used in higher dimensions, because the calculation of diversity measure is not straightforward and is often computationally expensive [29]. Since Pareto-optimal solutions for considered instances of TSAP are not known, some of the

Table 2: **Mean and standard deviation of the hypervolume for the number of specified runs on each instance of TSAP**

No. of sailors	Population 100	Population 200	Population 400
1000	0.1844 ± 0.0103	0.1648 ± 0.0055	0.1529 ± 0.0040
2000	0.2223 ± 0.0015	0.1963 ± 0.0015	0.1665 ± 0.0019
4000	0.2745 ± 0.0019	0.2451 ± 0.0014	0.2096 ± 0.0023
5000	0.2565 ± 0.0029	0.2449 ± 0.0023	0.2305 ± 0.0019

Table 3: **Mean and standard deviation of the hypervolume for 10 runs of NSGA-II and the three different hybrid approaches based on local search operators (shift, swap and a combination of both) for different instances of TSAP**

No. of sailors	1000	2000	4000	5000
NSGA-II	0.1843 ± 0.0103	0.2222 ± 0.0015	0.2745 ± 0.0019	0.2565 ± 0.0029
Shift-LS	0.1864 ± 0.0106	0.2228 ± 0.0017	0.2745 ± 0.0019	0.2565 ± 0.0029
Swap-LS	0.1809 ± 0.0126	0.2229 ± 0.0014	0.2743 ± 0.0014	0.2565 ± 0.0029
Both LS	0.1906 ± 0.0056	0.2231 ± 0.0017	0.2757 ± 0.0014	0.2565 ± 0.0029

existing performance metrics cannot be used to evaluate the results of the proposed approach. Therefore, in this work, the hypervolume was used as a quality measure.

In this work, several approaches are implemented and the results obtained are compared. Particularly, the results of standard NSGA-II are compared against the hybrid NSGA-II (NSGA-II conglomerated with special operators such as local search and boundary search). Accordingly, in each run, the final Pareto set approximation was recorded. The mean and standard deviation of the final hypervolumes over the specific number of experiments are reported in **Table** 2 and **Table** 3. It is evident from **Table** 2 that as the difficulty of the problem increases (Refer to **Table** 1), corresponding standard deviation increases. As the size of the population increases, the number of non-dominated solutions also increases, which in turn gives more precise value of the hypervolume, which can be noticed from the value of the standard deviation for the corresponding population sizes.

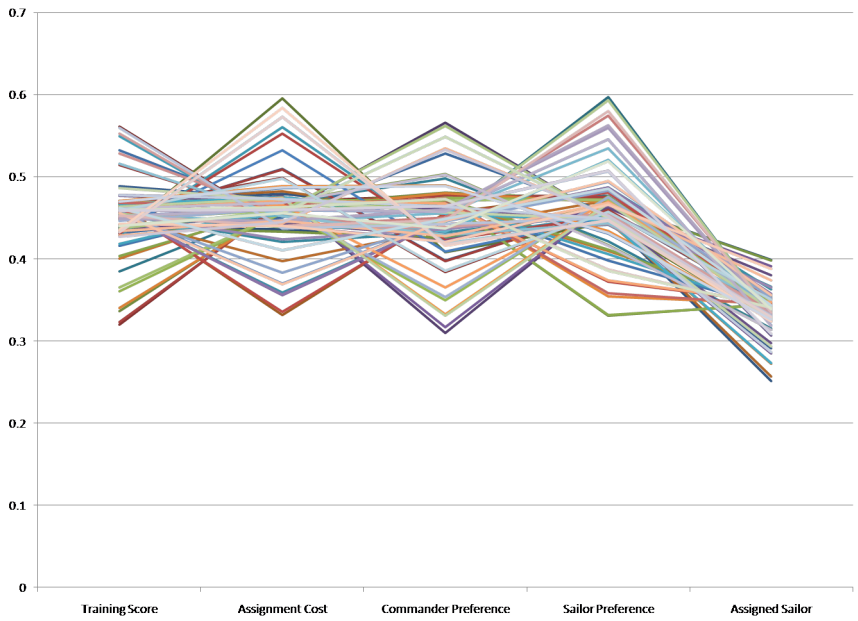


Figure 14: **Diversity of solutions found in the Pareto Front using NSGA-II on a 1,000 sailor instance of the TSAP and population size equal to 100**

Another observation is worth mentioning here; the objective plot of the solutions does not maintain their consistency in the last objective 'Assigned Sailor' with the change in problem specification. The problem designed in a way that consists of more number of tasks than the sailors actually present always turns up with the results where the last objective goes higher. This scenario is depicted in Figure 15

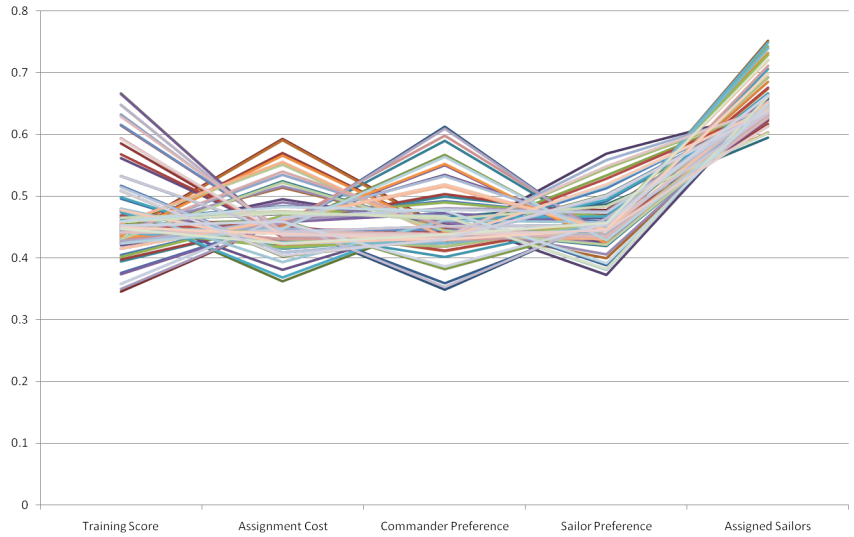


Figure 15: **Diversity of solutions found in the Pareto Front using a NSGA-II on a 1,00 sailor, 1200 tasks instance of the TSAP and population size equal to 100**

Figure 14 shows a parallel plot for sample solution by applying NSGA-II for 1,000 sailors, while Figure 16 shows the results for the same problem instance obtained by the hybrid approach which is a combination of NSGA-II and special local search operator. In these figures, each objective is plotted along the x-axis, whereas ranges of these objectives are represented by y-axis and each line represents a complete solution, i.e. a particular assignment of sailors to tasks in all time slots. In general, the hybrid approach, including local search operators, provided better diversity in all the objectives as observed in sample solutions shown in Figure 16. Specifically, shift local search operator is performed in every 10,000th evaluation whereas swap local search operator is executed in every 15,000th evaluation.

Same as above, Figure 17 shows a parallel plot for a sample solution by applying NSGA-II with local search operators and boundary search operator. This boundary search operator works on the infeasible solution and tries to feed the feasible solutions residing near the boundary of the problem in every iteration. From Figure 17

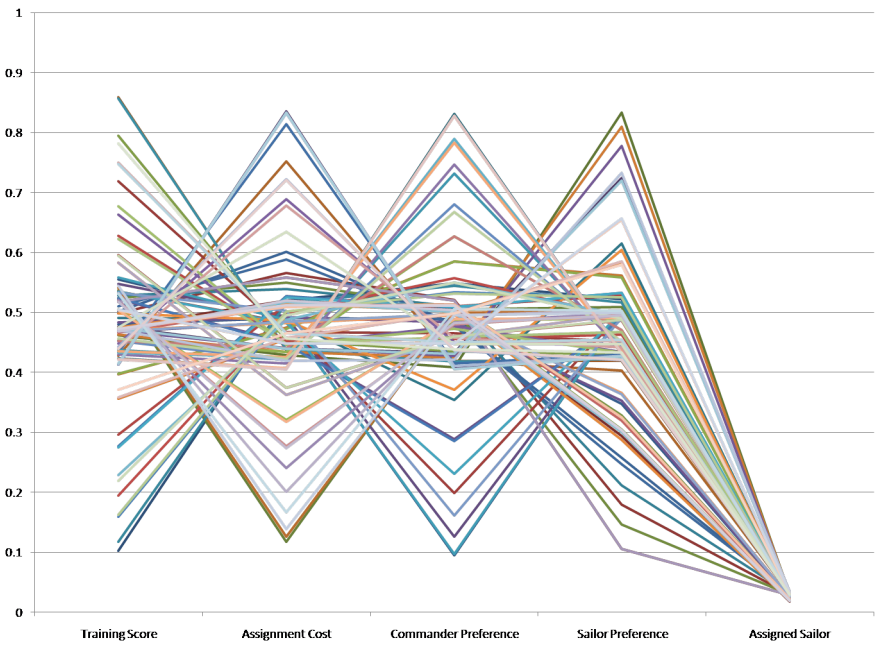


Figure 16: **Diversity of solutions found in the Pareto Front using a hybrid approach alternating both local search operators on a 1,000 sailor instance of the TSAP and population size equal to 100**

, it can be easily observed that the diversity of each objective is enhanced with the application of special operators. Moreover, the diversity of solutions in the last objective of assigned sailors has vividly increased using this approach. This infeasible boundary search operator is highly expensive in terms of computational complexity as well as time, which prompts non-execution of the same in every iteration of the problem. Therefore, this operator is performed in every k-th evaluation depending upon the maximum number of evaluations where k contains a considerable high value.

It is quite evident from all pareto front depicted here, that the last objective does not maintain any consistency in diversity. The diversity of the last objective depends entirely on the task-sailor ratio. Table 3 summarizes the results of comparing 17 runs of NSGA-II to three different hybrid approaches, based on local search operators (shift, swap, a combination of both and boundary search) for the same problems defined in Table 1 and population size 100. It is observed that the hybrid approach that alternates both local search operators and the boundary search operator

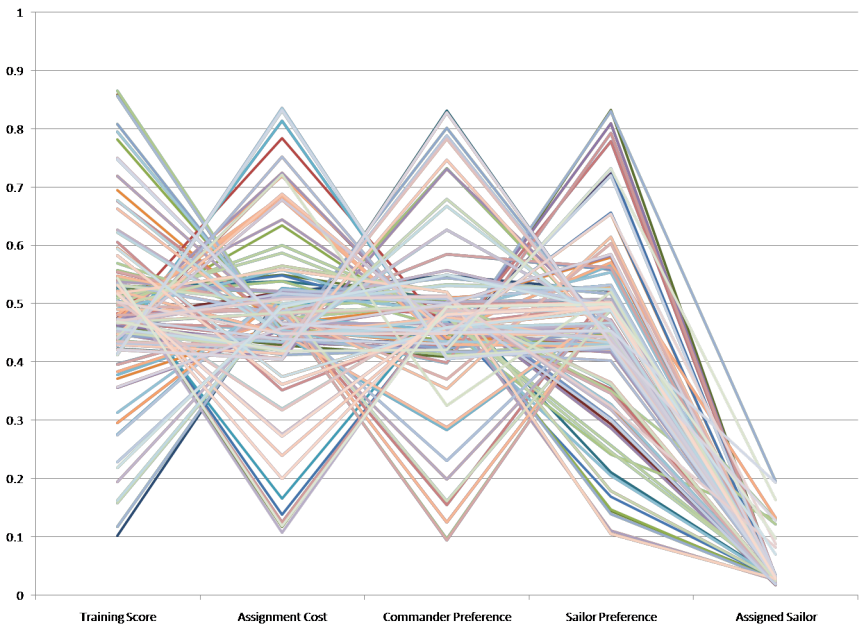


Figure 17: **Diversity of solutions found in the Pareto Front using a hybrid approach alternating both local search operators and infeasible boundary search operator on a 1,000 sailor instance of the TSAP and population size equal to 100**

provides better solutions from fitness perspective. Here, notice that the hybrid approach provides better solutions from an objective point of view, i.e. better approximations towards the optimal Pareto front.

5 Proposed Approach for Coverage and Lifetime Optimization Problem

This section is dedicated to the detailed discussion of MOEA, based on NSGA-II and SPEA-II used to solve Coverage and Lifetime Optimization Problem (CLOP). An example of wireless sensor network monitoring a special terrain is depicted in the Figure 18.

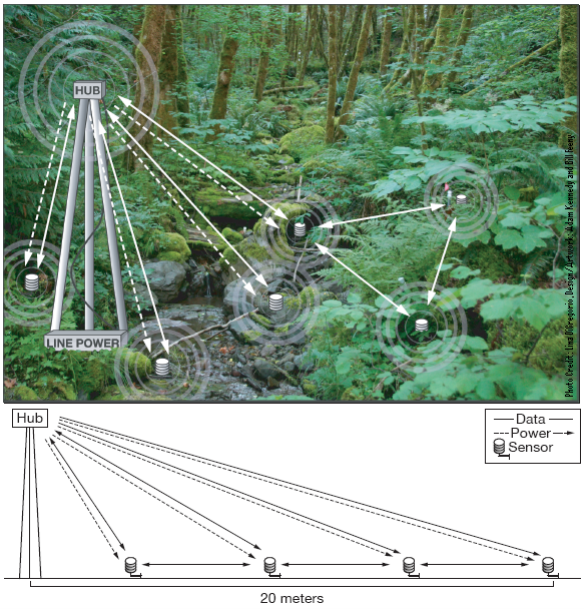


Figure 18: **Wireless Sensor Network Monitoring Real Life situation**

5.1 Chromosome Representation

In this problem, an analogous methodology like the one mentioned in the above problem has been followed to represent the entire solution. A chromosome is made up of a set of real values, each representing the location of sensor nodes over the hostile terrain. Location of sensor nodes refer to the x and y coordinates of each of the

sensor node. Each of the sensor nodes is given a unique real number. The length of the chromosome is equal to the $2*$(total number of sensor nodes). In this work, we assume that the payload of the aircraft carrying sensor nodes to deploy is limited thereby making the number of available sensor nodes constant.

The chromosome representation is used as follows: The terrain on which the sensor nodes are to be deployed consists of fixed parametric parameters based on which the real numbers for sensor nodes coordinates are randomly generated. The sensor nodes coordinates are placed side by side such as X_iY_i to form the entire chromosome. The chromosome representation is depicted in Figure 19 below.

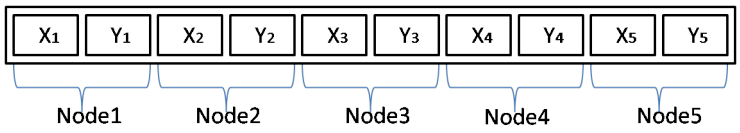


Figure 19: **Chromosome Representation for CLOP**

The hostile terrain is where sensor nodes will be deployed for monitoring, where they can monitor anything within R_{Sensor}, and where they can communicate with any other node located within R_{COMM}. The base station (HECN), with which every sensor must communicate (either directly or via hops maintaining the minimum path to reach the base station), is placed anywhere in the area of interest. This assumption has been taken to maintain the simplicity of the problem and for the convenience of computation. At the incipient stage of sensor deployment in the terrain, each sensor has the same energy in its battery for sensing and communicating the data to the base station, and in our problem, we have taken the assumption that the energy of each sensor decreases by $\frac{1}{SensorRemainingEnergy}$ unit for every data transmission through one link to next hop.

5.2 Objective and Fitness Measure

Out of many challenging objectives in wireless sensor network, the most prominent is the network coverage. Coverage is a highly crucial objective in wireless sensor network because it determines how much area from the base station, or the network comprised of the fixed number of nodes, can cover. In this work, sparse sensor node deployment is emphasized rather than dense sensor node deployment for the sake of computational simplicity. In [36], the problem has been formulated keeping dense deployment in mind which prompts the authors to express the network coverage in terms of union of the disks of radii R_{Sensor} centered at each connected sensor, normalized by the total area. My problem has been motivated by the problem in [36] but CLOP differs from their problem by dint of several aspects, deployment type, objective function expression terms, constraints maintenance etc. In this work, objective function of network coverage is expressed in terms of minimum overlap areas between sensor nodes.

Another objective function that has been considered is maximum network lifetime. Wireless sensor network suffers from the issue of limited sensor node energy resources. According to the type of sensor network, the sensor node energy content is limited, unlike the wired networks. Replacing the battery or recharging it is next to impossible, as the nodes are meant to be deployed in hostile environments. In dense deployment, failure of regular sensors may not harm the overall functioning of a wireless sensor network since neighboring sensors can take over, but in sparse node deployment case, the scenario is different. In that case, once a node's energy gets depleted, the network is considered broken down. In this problem, the network lifetime is defined as the ratio of the time to the first sensor node failure and the maximum lifetime of a sensor.

For the sake of computational simplicity it has been assumed that all sensors gather data at the same time, and once their data sensing is done, they relay it to the

HECN; this procedure is referred to as a sensing cycle. Now to get data to the base station the data might need to be relayed by several sensors, due to absence of a direct communication link between the node and the base station. Therefore, at every sensing cycle, the sensor nodes need to transmit their own data and probably the data from other sensors, acting as communication relay. So at each sensing cycle, the data from every sensor needs to be routed to the base station in a way that will maximize the remaining energy in the nodes. In order to find these routes, the outgoing edges of every node are weighted by 1/nodes remaining energy, and then the Dijkstra algorithm is used to find the route of minimum weight. Looping through the calculation until and unless the energy of at least one node is depleted enables the maximum number of sensing cycles possible in a particular wireless sensor network layout. This value is then normalized by T_{max}, the maximum number of sensing cycles possible, which is obtained when all sensors are directly connected to the base station so that none act as a communication relay. Therefore from the above mentioned discussion about the objective functions of CLOP, it turns out to be crucial to maximize the network lifetime so that the network can transmit information to the base station for a longer period of time and maximize the coverage area at the same time.

The strength and the challenge of the problem lies in the competing nature of the two objectives; maximizing network coverage and maximizing network lifetime. To achieve maximum network lifetime, the sensor should be ideally placed within one hop (where each of them can directly communicate to the base station) from the base station. For such deployment, there will be sufficiently enough overlap between sensor nodes because the one hop area around the base station is maximum of ΠR^2_{comm}. Now if we want to maximize coverage, then the sensors have to be placed far from the base station to minimize overlap between them. For such deployment, the sensors ideally will not be able to communicate directly to the base station; instead they would require one or more sensor nodes in between to act as relay nodes in order to communicate to

the base station. Now for each data transmission, from a far away sensor node, it will not only spend its own energy, but also be responsible for the energy expenditures of the relay nodes. Therefore the cost of transmission in terms of energy depletion of the network, increases when nodes are placed far away from the base station. This will in turn reduce the lifetime of the network. The competing nature of the two objectives functions motivate the design of a deployment where we can achieve fair enough coverage and yet maximize the lifetime of the network.

To be more specific, two objectives of CLOP are competing with each other. In case of the objective of network coverage, it always wants to spread out network layouts, where sensors are as far apart from each other as possible in order to minimize the overlap between sensing disks. This in turn implies a large number of relay transmissions for sensors communicating directly with the base station, thereby making it highly probable that the network will fail quickly due to the huge energy exploitation; hence the network lifetime will be poor in that case. On the contrary, in order to gain maximum lifetime, all the sensors must communicate directly to the base station, so that their energy is used only for their own data transmission, but not for transmission relaying. This implies a clustered configuration around the base station with a lot of overlap between the sensing disks, causing poor network coverage. As mentioned above, the objective functions of the problem CLOP are the following:

- Maximum Network Coverage represented in terms of minimum sensing radius overlap:
  $MaximumCoverage = \sum_{i=1}^{n}(\Pi R_{sensor,i}^2 - EllipticalCurveError_i)$
  where $EllipticalCurveError_i = \sum_{j=1,i\neq j}^{n} OverlappedAreas_{i,j}$
  where $OverlappedAreas_{i,j} = \Pi ab$
  where $a = \sqrt{b(2d-b)}$ and $b = R_{sensor} - d/2$ where d= center distance between node i and j

Finally the obtained coverage is normalized by the maximum network coverage possible which is $n\Pi R^2_{sensor}$ for n number of nodes. Hence the final Coverage = $\frac{ObtainedMaximumCoverage}{MaximumPossibleCoverage}$

- Network lifetime for a random deployment of sensor nodes is computed by the number of sensing cycles possible before the energy of any node gets depleted. One sensing cycle of the network is said to be complete if all the nodes are successful in transmitting their messages to the base station without running out of energy. Since each node spends 1=RemainingSensorEnergy to transmit data to 1 hop neighbor so at every data transmission energy gets depleted for a node. The route from a node to the base station is computed using Djikstra algorithm, which gives the path of least energy expense. Hence the number of sensing cycles depends on the deployment of sensor nodes and the order in which the nodes run out of energy. The more the number of sensing cycle the higher is the network lifetime.
  Maximum network lifetime: $\min(\frac{T_{failure,i}}{T_{max}})\forall i = 1, 2, ..., n$

This problem formulation has undergone one robust constraint, which tries to keep all sensor nodes in the network connected. Due to the nature of objectives, the layout of the sensor network always tends to spread out, thereby making it highly likely for the nodes to get disconnected from the network. This speeds up the failure of the network. To prevent this situation, this work has introduced a constraint that takes care of the network connectivity. This constraint, once it obtains a solution, tries to check for the connectivity of each node to the network, and it penalizes the solution if any node is found disconnected from the network. The penalty that is applied to the solution depends upon the number of nodes disconnected from the network, thereby assuring the connectivity persistence of the network. The penalty of a solution imposed by the constraint is decided to be the fraction of the disconnected sensor nodes over the entire network.

To maintain the simplicity of the computational complexity, this work has taken several assumptions. They are described below.

- No overlapped area will be shared by more than two sensor nodes. This scenario is depicted in Figure 20.

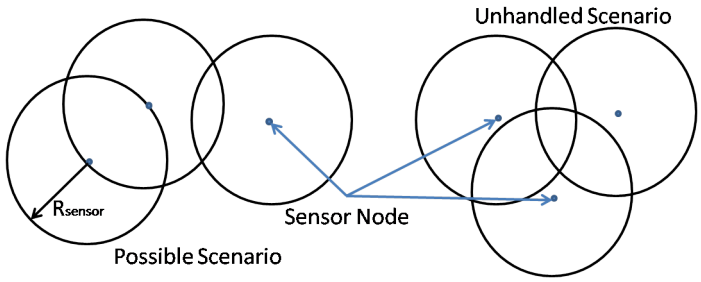


Figure 20: **Assumptions taken for CLOP**

- The sensor nodes used here are homogeneous sensors. To be more specific, each of them has a constant sensing radius (R_{sensor}) and a fixed communication radius (R_{comm}).

- Each sensor node starts with its battery energy for both sensing and communicating the data to the base station. The energy of each sensor decreases by $\frac{1}{SensorRemainingEnergy}$ unit for every data transmission through one link to the next hop.

5.3 Genetic Algorithm and Operators

The objectives of CLOP are highly non-linear, because a small deviation from the location of a sensor node can cause an abrupt change in objective values of the entire solution, thereby making the design space non-linear. This complicated nature of the problem prompts application of several robust multi-objective evolutionary algorithms such as NSGA-II and SPEA-II and this work finally comes up with the

comparative analysis of the results obtained from these MOEAs. The genetic operators used in these algorithms are all problems specifically designed to aid the MOEAs to find the optimal solutions. The crossover, mutation, and selection operators, along with the special operators local search and boundary search, have been implemented to align with the problem. These MOEAs have been proved to perform quite well with the non-linear objectives, hence these algorithms have been selected to be applied to CLOP. This MOEA is targeted to come up with a set of Pareto-optimal design layouts of the sensor node networks. This provides a set of solutions that depict the trade-off between the two objectives, or how much lifetime can be given up in order to gain some coverage (or vice versa). It helps the end-user to select from a set of optimal solutions depending upon their application requirement, i.e. their emphasis (either on network coverage or network lifetime or both), the end-user can choose the solutions from the final result set.

5.3.1 Crossover Operator

The crossover operator works in alignment with the one-point-crossover operator in terms of random crossover position selection, but the way it performs crossover is fairly different. Crossover operator once receives two parent solutions, it randomly picks two different positions of two parents to perform crossover. This operator also generates a random number which serves as the information window size. Later the information from these two windows are exchanged during crossover. To be more specific, the position p_1 has been randomly generated for $parent_1$ and position p_2 for $parent_2$. The constant window size s is another factor that is randomly generated depending upon the chromosome length. Now the node locations (x and y coordinates of the nodes) starting from the position p_1 of the window from $parent_1$ have been exchanged with the same from the window of $parent_2$ starting from

position p_2 in a reverse order. Figure 21 depicts the CLOP specific crossover which clearly presents how crossover positions have been selected and how crossover is being performed to form new offspring out of them.

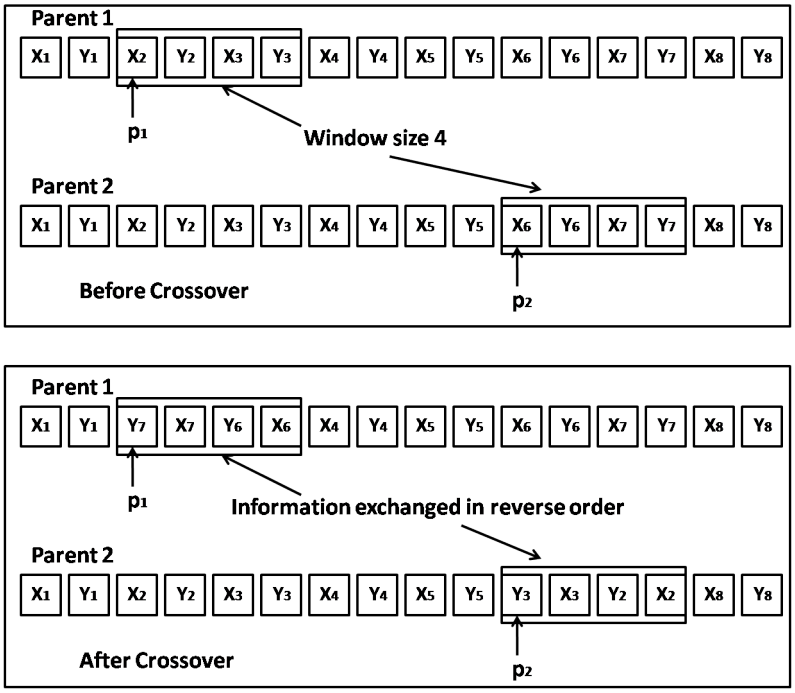


Figure 21: **CLOP Crossover Operator**

This operator performs crossover on a pair of solutions a constant number of times depending upon the chromosome length. It has been noticed that keeping the crossover rate high is highly likely to provide better solutions. Therefore motivated by the empirical evidences, this work has emphasized keeping a higher rate of crossover operation.

5.3.2 Mutation Operator

The sole objective of the mutation operator is to mutate/change a solution to obtain a new solution with twofold objectives; one is to maintain and introduce genetic

diversity from one generation of a population of chromosomes to the next generation, and another is to aid the genetic search in finding another solution that might be more fit than its root solution. It facilitates genetic search to avoid local minima by preventing the solution to be analogous to its parent solution. In this work, the mutation operator works very simply. It picks a random sensor node from the solution. Next it obtains the location (i.e. coordinates) of the sensor node selected from the solution. Afterwards it generates a random real value within the range of the terrain where the sensor nodes are going to be deployed. The generated new random real value replaces the selected sensor node coordinates, which in turn mutates the solution. Then this solution is fed back to the population. The rate of the mutation operator is carefully chosen for this problem, because an excessively low mutation rate is highly likely to lead towards genetic drift, and on the other side, extremely high mutation works poorly for this problem because it may lead to the loss of good solutions. Therefore, mutation rate in this problem has been kept proportionate to the chromosome length. The mutation is depicted in Figure 22.

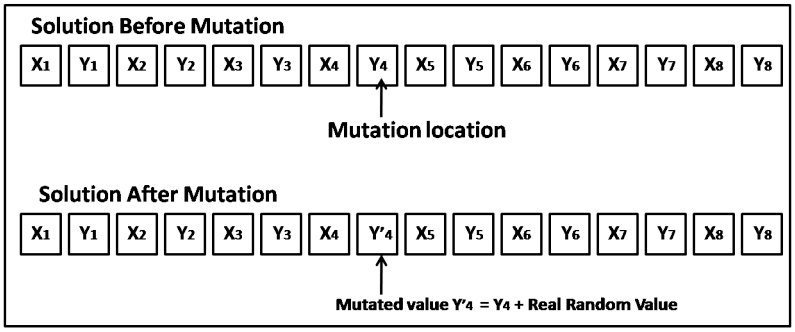


Figure 22: **CLOP Mutation Operator**

5.3.3 Local Search Operator

This special operator is used to fine-tune the results obtained from the regular genetic search flow. The local search operator depicts sere difference between

solutions generated by evolving through regular operators in genetic algorithm and local search operator included in it in terms of solution diversity as well as quality of solution. The sole purpose of this local search operator is to find better solution in the local arena of a solution. It usually picks a solution from a population and tries to find out if the solution around it is better in terms of fitness values. This work implements the problem-specific local search operator called Shift Local Search operator.

In shift local search operator, a solution is to be tried to search for another solution residing locally to it in the following way:
In this problem, a random sensor node has been selected from the solution and its coordinates are obtained. Then, this operator shifts this original solution towards the point in its top right corner. This operation is performed by adding sensor radius to its x and y coordinates. During the generation of new solution, always the newly generated sensor node coordinates have been checked as to whether they are lying inside the region or else another node has been selected. Thus once a new solution is obtained, it is checked against the original solution for fitness. If the newly searched solution turns out to be more fit than the previous one, then it is fed back to the population in place of the original solution. Otherwise, the original solution remains same.

The local search operator contributes to reaching towards better optimal solutions in the whole population, but it turns out to be highly expensive in terms of computation and time. Therefore, this operator is highly unlikely to be executed in each generation of evolutionary algorithm. So, the operators are being applied to each solution in the population in every k-th evaluation iteration (where k can be any constant value such as 2000, 592 etc.) out of total evaluation n (where $n \geq k$). The local search operator is shown in Figure 23

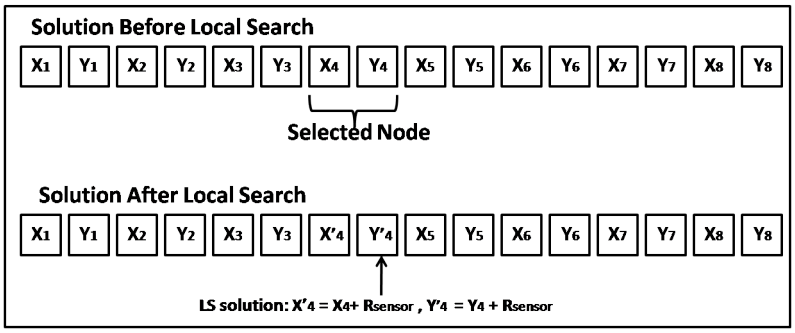


Figure 23: **CLOP Local Search Operator**

Table 4: **Parameters of the randomly generated problem instances used in CLOP**

Problem Instance	Population Size	Evaluation	No. of sensor nodes
1	300	50000	5
2	500	70000	15
3	1000	100000	20
4	5000	200000	30

5.4 Experimental Results

CLOP problem undergoes a large number of genetic algorithm generations and parameter tuning to generate different types of results. Those results are compared with each other, and the comparative study has been presented in this section. The parameters used in this problem are depicted in Table 4. These parameters have been used in NSGA-II and SPEA-II algorithms for ten runs of each problem instance. The different layouts of wireless sensor network are presented in this work, along with the comparative analysis of the fitness value results found in [36].

A specific solution in this work has been represented by dint of a straight line. Two terminals of the straight line reflect two objective values. It is to be noted that the straight line does not represent a mathematical straight line ($y = mx + c$), but rather reflects a solution having two different objective function values for better understanding of the trade-off between two objectives; network coverage and lifetime. The straight line here acts as a connector between the two objective values of a solution

for making the representation more humanly understandable. The CLOP has been run with NSGA-II and the objective plot of the same has been depicted in Figure 24.

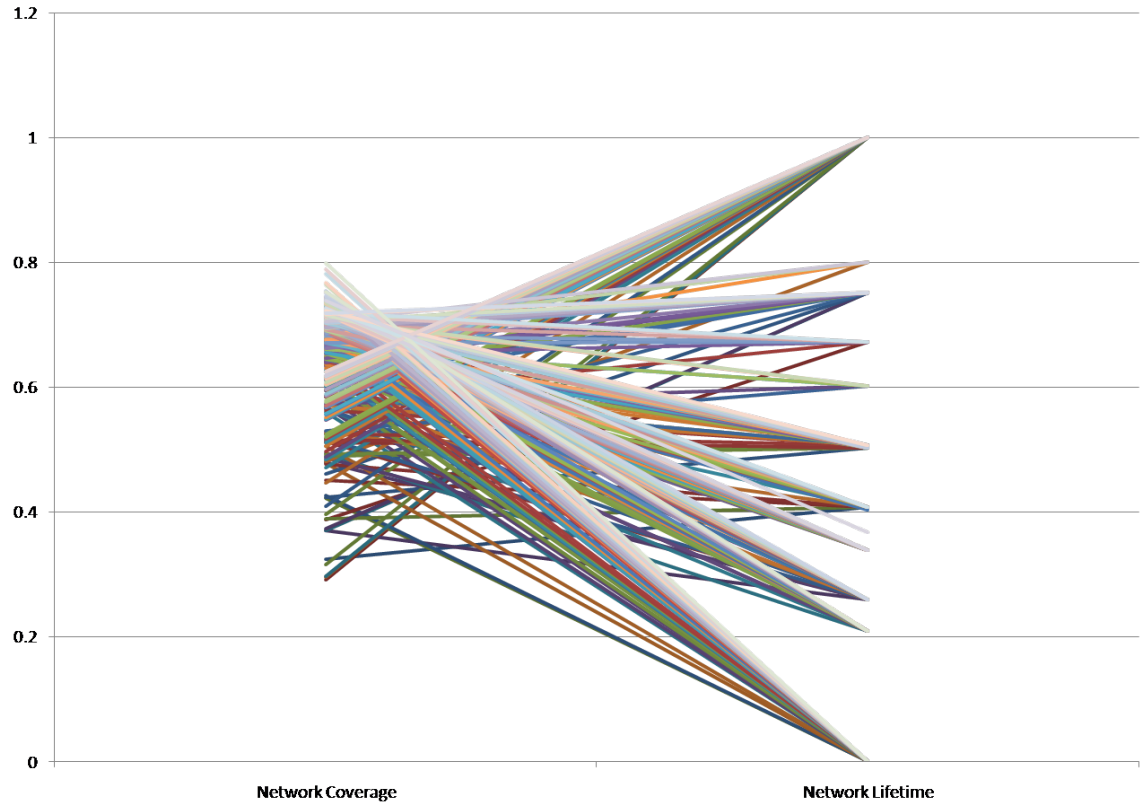


Figure 24: **Diversity of the solutions found the Pareto Front using NSGA-II on 10 sensor nodes**

From Figure 24, it is clearly evident that the solutions with lifetime near around 1 suffer from less coverage. And also, the opposite scenario is also clearly evident because for maximum coverage values near about the zone of 0.8, the lifetime of the solutions turns out to be poor, i.e. near to the zone of 0 to 0.1. This defines the trade-off between multiple objectives in multi-objective optimization problem. Only those solutions can be considered as balanced acceptable and close to optimal solutions which serve in between, or do not give up coverage for lifetime or vice versa. From Figure 24, it is clear that the solutions lying near the zone of 0.7 serve the purpose of balancing between two objectives. The solution having coverage objective

value 0.710029 and lifetime value 0.75244 turns out to be the balanced acceptable optimal solution in this experiment.

This problem CLOP has undergone the same algorithm conglomerated with special operator local search so as to provide genetic diversity in the population. The result found from ten runs in NSGA-II and shift local search operator is displayed in Figure 25.

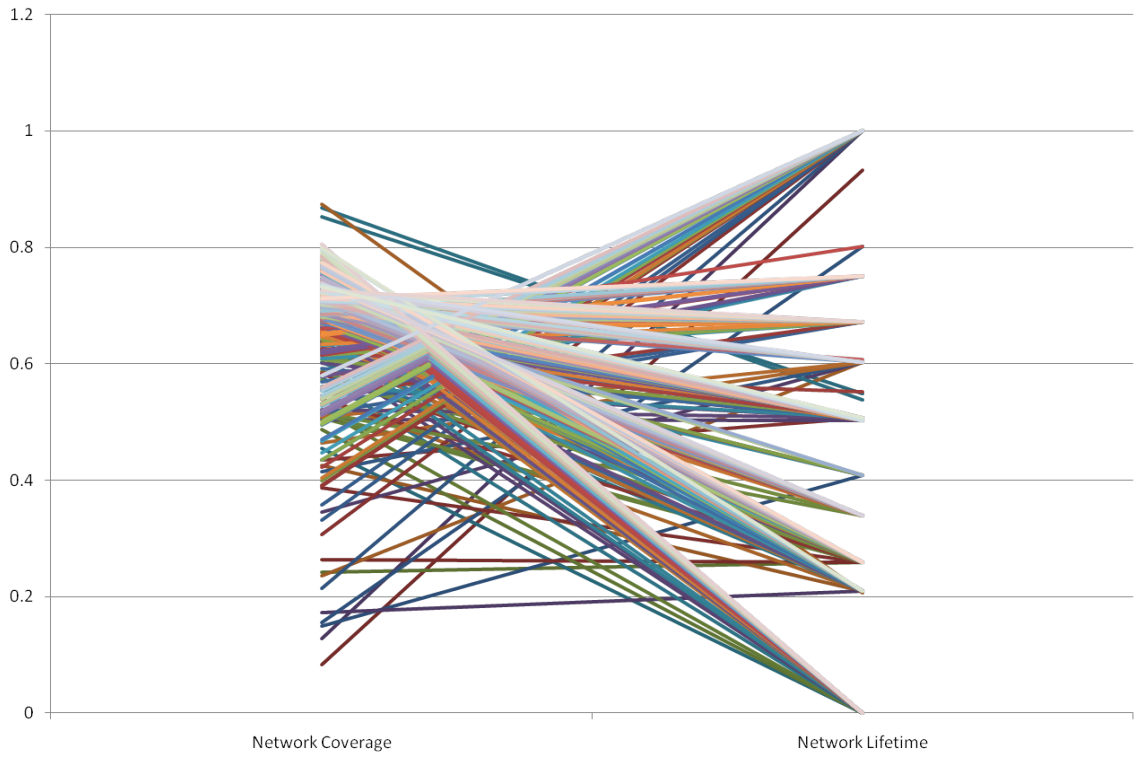


Figure 25: **Diversity of the solutions found in the Pareto Front using hybrid approach combining NSGA-II and Shift Local Search operator on 10 sensor nodes**

From Figure 25 depicted, the result can be analyzed in such as a way that the local search operator once applied to the problem, generates more diversified solutions. The reason behind is the nature of the operator; it tries to produce a new solution from an existing one by replacing one solution by another one. This changes the entire wireless network topology thereby causing an effect on objective values. In this operator since the newly generated solution is verified against the older solution with respect to fitness, always the better fit solution is replaced into the population. This operation always calls for better fit solutions to be placed in the population as well as spread the searching over the entire search space. Since this operator looks for newer solutions over the whole decision variable space, therefore it is highly likely to obtain diversified solutions which normal algorithm remains incapable of finding.

After NSGA-II, this problem has been experimented with another robust multi-objective evolutionary algorithm named SPEA2. The diversity of solutions obtained in the Pareto front has been displayed shown in Figure 26. It can be observed from the objective plot that it comes up with substantially extreme solutions with higher coverage and lower lifetime, and vice versa. The strength of this algorithm lies in the fact that producing solutions having higher coverage and higher lifetime as well. This algorithm turns out to be better performing in the problem than NSGA-II, since it shows up with the solutions having both objective values high. But it is evident from Figure 26, that this algorithm suffers from producing a significant number of balanced solutions. SPEA2 also depicts that a tiny change in the sensor node placement (solution in turn) can cause abrupt changes in objective values. Note in Figure 26 that one solution having higher coverage (lying in the zone of 0.42) has low lifetime, whereas another solution having higher coverage in the same zone (almost juxtaposed with the above-mentioned solution) has higher lifetime too.

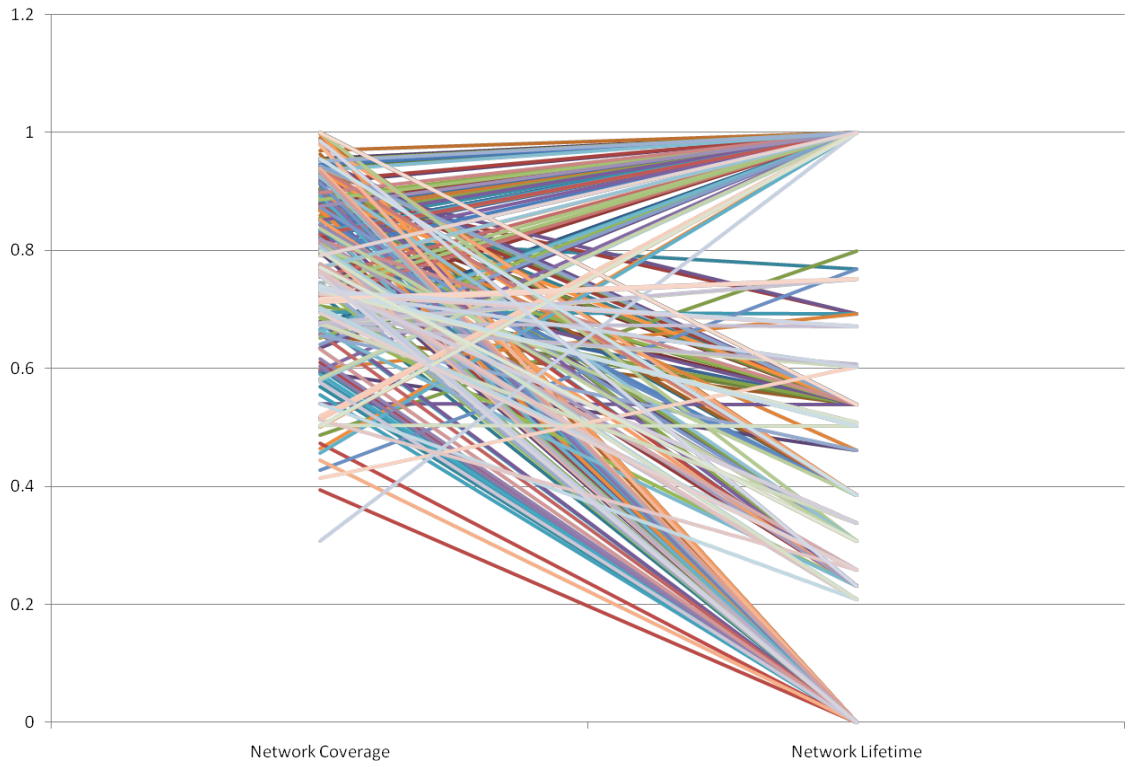


Figure 26: **Diversity of the solutions found in the Pareto Front using SPEA-II on 10 sensor nodes**

Local search operator, when applied on the SPEA-II algorithm, produces even bigger diversity in the maximum coverage, and also produces certain potential solutions where a big spread of solutions can be achieved with fair network lifetime. The objective plot of SPEA-II with local search applied can be seen in Figure 27 below. From Figure 27, , it is quite apparent that the special operator ”Local Search”, which is carefully designed for the problem, plays a crucial role in coming up with solution diversity. Compared with the results generated by regular SPEA2, hybrid SPEA2 amalgamated with ”Local Search” operator maintains diversity of solutions. In the later one, solutions with coverage lying around the zone of 0.1 are found, whereas the former algorithm starts showing up solutions with coverage lying in the zone of 0.3. This eventually comes up with the fact that the conglomeration of local search

operator with the robust MOEAs selected for this problem work sufficiently this problem, and prove to produce a broad diversity of solutions.

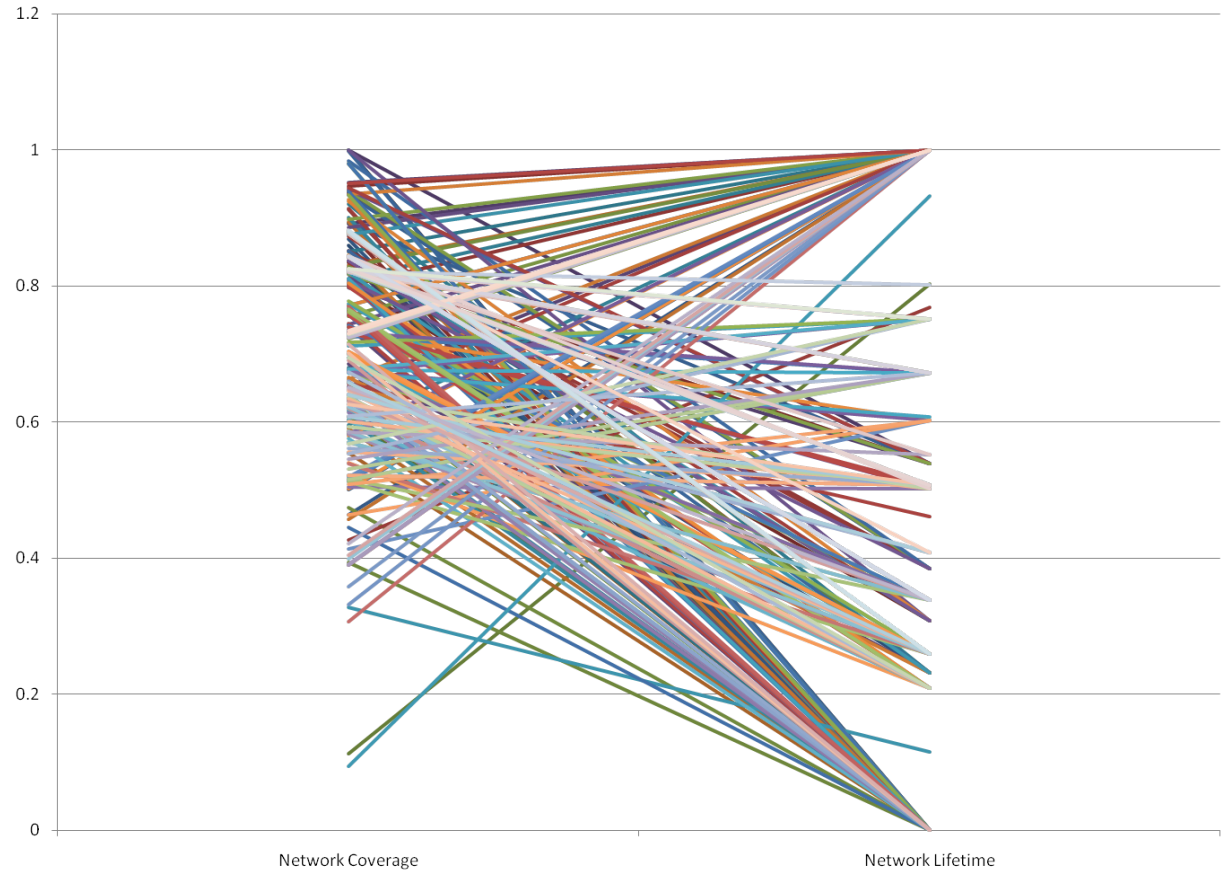
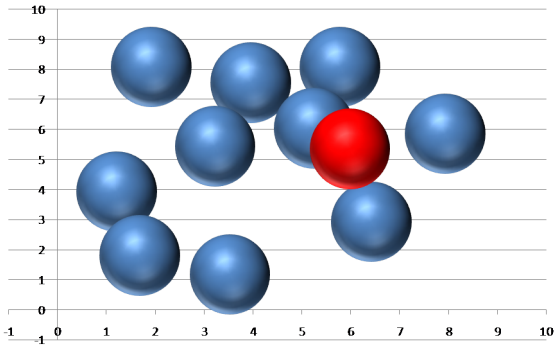


Figure 27: **Diversity of the solutions found in the Pareto Front using hybrid approach combining SPEA-II and shift Local Search Operator on 10 sensor nodes**

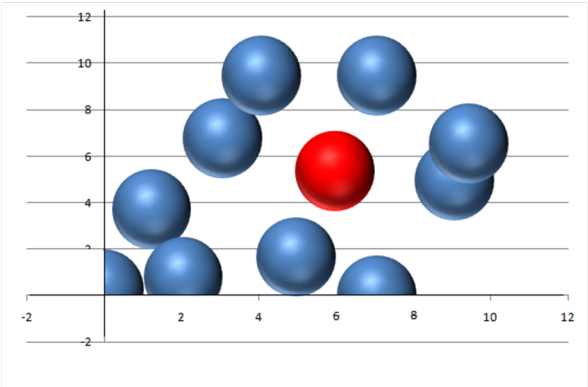
From the solution fitness point of view, application of several robust MOEAs such as NSGA-II and SPEA-II turn out to produce better results than the work from where this problem is motivated [36]. Their application of regular genetic algorithm proved to produce good enough solutions, but these algorithms come up with better solutions than those of the problem in [36]. Analytically, it can be said that the presence of problem specific operators such as 'CLOPCrossover', 'CLOPMutation', 'CLOPShiftLocalSearch' along with the potential of the selected multi-objective genetic algorithms 'NSGA-II' and 'SPEA-II', plays a crucial role of producing better solutions with acceptable fitness.

It is lucidly identified in the objective plots of two different algorithms that, the balanced solutions provide better wireless sensor network design topology than the one-sided extreme solution. To depict the result on wireless sensor network layout, Figure 28 and Figure 29 have been shown for NSGA-II and SPEA-II respectively.



WSN Layout for NSGA II where Coverage =0.710029, Lifetime= 0.751244

Figure 28: **Wireless Sensor Network node deployment topology using NSGA-II**



WSN Layout for SPEA II where Coverage =0.804846646 , Lifetime= 0.7692307

Figure 29: **Wireless Sensor Network node deployment topology using SPEA-II**

Figure 28 and Figure 29 display the deployment of sensor nodes in the hostile terrain. The red-colored node represents the base station (HECN) to whom the other nodes (blue colored) transmit data periodically. The bubbles in the figures represent the sensing zone of each sensor node and the sensor node lies in the center of the sphere. It has been observed from the deployment scenarios that some of the balanced

solutions produced using NSGA-II algorithm are found to be better than the same generated by SPEA2 algorithm. This keeps a shadow on the wireless sensor network design topology too, which is clearly visible in the figures showing deployment. SPEA-II has tried to place the sensor nodes, to some extent uniformly, around the base station, whereas NSGA-II deploys the sensor nodes in a somewhat skewed fashion around the HECN. From the results depicted above, it is clear that NSGA-II produces better, feasible, acceptable, balanced solutions, which is reflected in the deployment; the deployment shows how NSGA-II tries to spread the sensor nodes over the terrain, thereby making both coverage and lifetime balanced.

The pareto fronts obtained from the experiment with NSGA-II have been depicted in Figures 30 and 31.

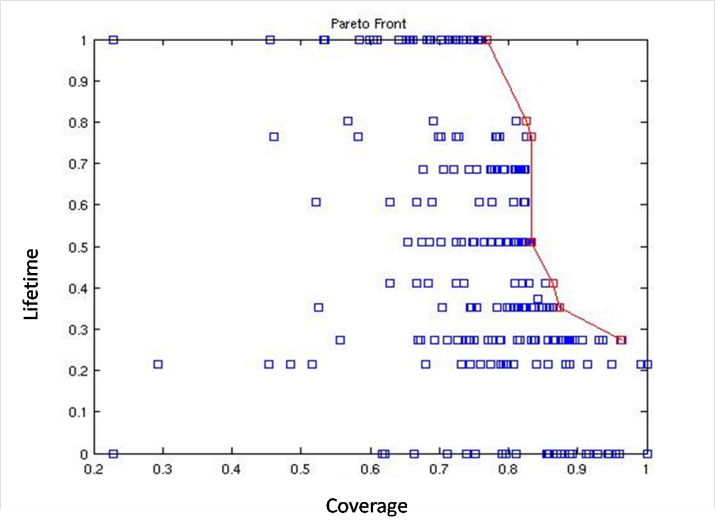


Figure 30: **Pareto Front for NSGA-II on 250 sensor nodes and 1000 evaluation**

Pareto front represents the optimal solutions in the problem. From Figure 30, , it is quite comprehensible that the objective ”coverage” has been plotted against objective ”lifetime”. Pareto front shows the tarde-off between them in this figure. The red line here reflects the Pareto front for maximizing objectives ”coverage” and ”lifetime”. Figures 30 and 31 reflect that no other solution can dominate those points that have taken part of generating Pareto front.

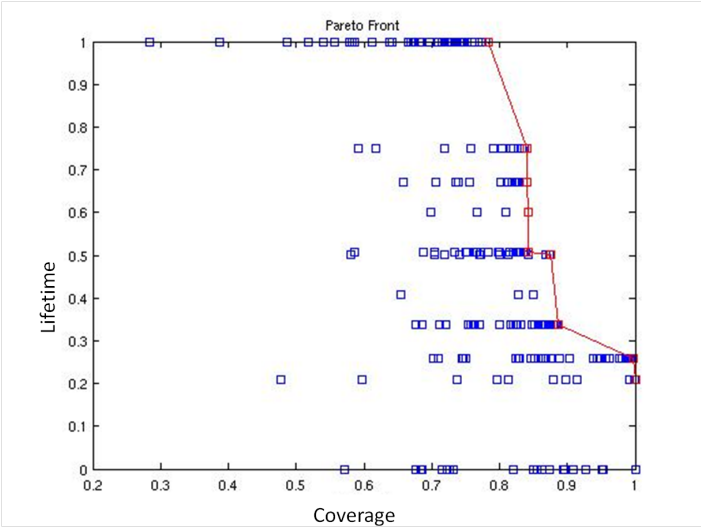


Figure 31: **Pareto Front for SPEA-II on 1000 sensor nodes and 5000 population**

5.5 Graphical User Interface Implementation

To endow the end-user with the facility to select from a set of Pareto optimal solutions, this work has implemented one graphical user interface which is tried to be kept as user friendly as possible. The homepage of the interface is displayed in Figure 32.

Figure 32: **HomePage of the CLOP Software**

Clicking on the arrow displayed in bottom right corner will take the end user in the page which will display the objective plots of the solutions and the wireless sensor network design/layout. This is displayed in Figure 33 .

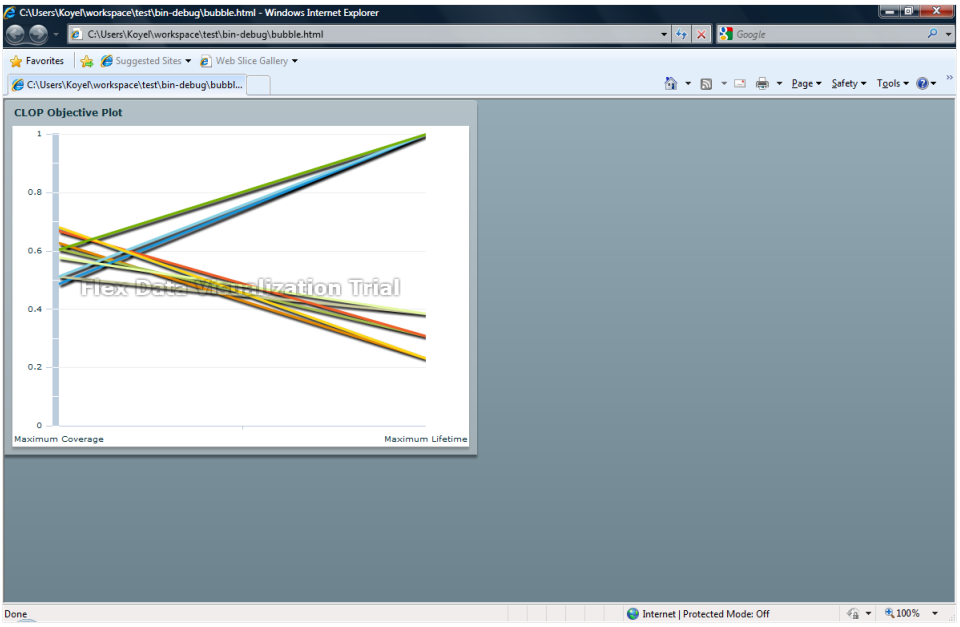


Figure 33: **Objective Plot of the solutions in the CLOP Software**

In Figure 33 , each solution is represented by the straight line shown. Each terminal of the straight line reflects objective values separately. An end-user is capable of hovering the mouse over the solution and it then displays the objective values accordingly as depicted in Figure 34.

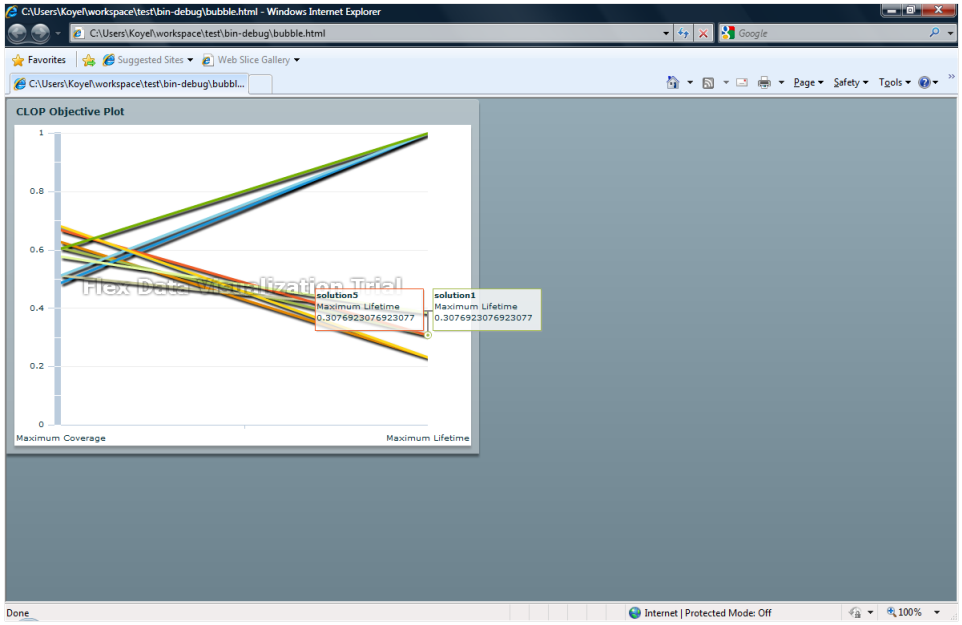


Figure 34: **Objective Values Display Plot of the solutions in the CLOP Software**

Figure 34, on clicking on a solution line, will display the corresponding wireless sensor network design topology as shown in Figures 35 and 36.

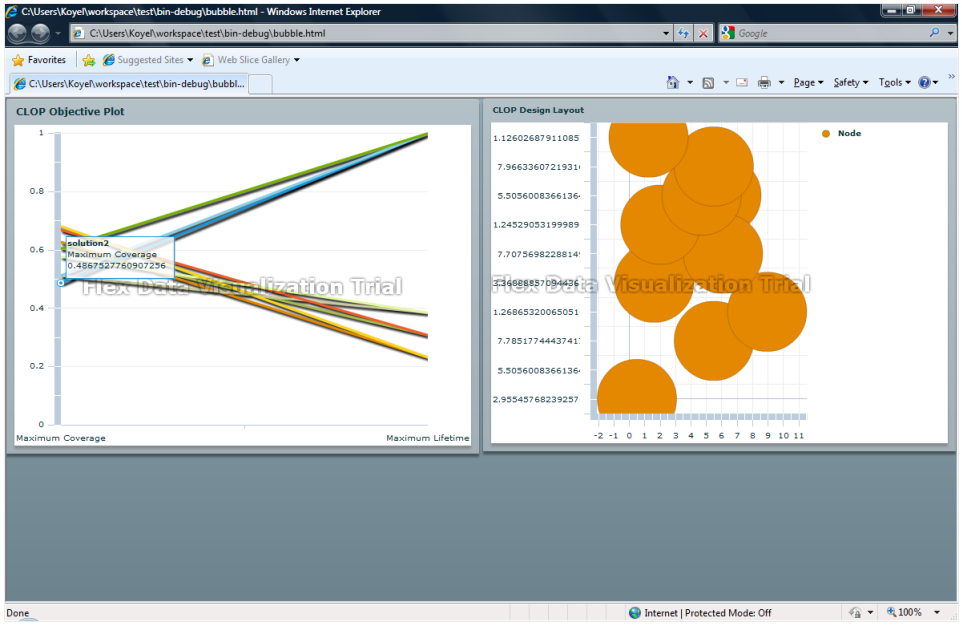


Figure 35: **Wireless Sensor Network Deployment Display of the solutions in the CLOP Software**

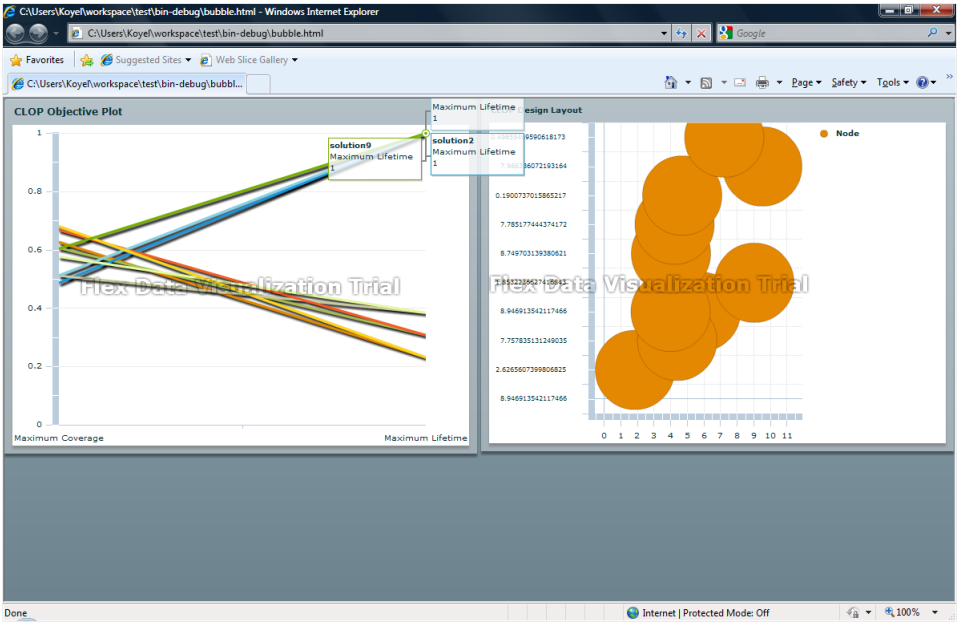


Figure 36: **Wireless Sensor Network Deployment Display of another solutions in the CLOP Software**

End-users may also view the details of each sensor node, such as their locations, sensing radius etc. by hovering the mouse over them as shown in Figure 37. Here in the figure, the top values below ”Node” represent the x coordinates, y coordinates and sensing radius (R_{sensor}) of the sensor node respectively in the deployment.

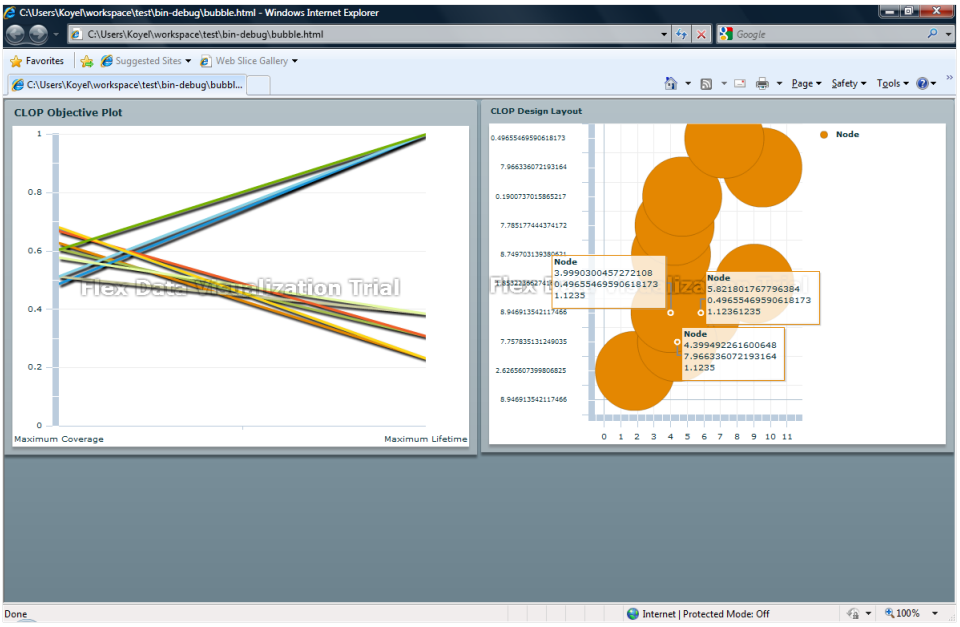


Figure 37: **Sensor Node Details Display of the solutions in the CLOP Software**

6 Conclusion and Future Work

This work is dedicated to apply multi-objective evolutionary approaches to solve variation of the US Navy’s Sailor Assignment Problem named Task based Sailor Assignment Problem (TSAP) and Wireless Sensor Network’s Coverage and Lifetime Optimization Problem (CLOP). Compared to the previous studies on Sailor Assignment Problem [20, 17, 15], in this work, investigation explores a much more complex time-dependent multitasking problem in the form of TSAP. The proposed

approach based on the multi-objective evolutionary algorithm NSGA-II has depicted a broad gamut of solutions in the interior portions of the trade-off surface. A good diversity of solutions is generated by NSGA-II along the Pareto set approximation.

In the case of TSAP, further work is needed to explore the missing objectives or any other constraints. It is also worth digging into some other special operators or other hybridization mechanisms. Moreover, implementing some domain-specific genetic operators might potentially come up with further improvements in the solutions both in design space as well as in objective space. Specifically, a crossover operator which is mainly responsible to maintain solution feasibility would reduce the need to repair individuals. Such an operator, if carefully designed, might be beneficial with respect to both in solution quality as well as computational time. The proposed approach currently encompasses five distinct objectives. Several other additional components do exist which require further investigation, such as the idea that all tasks might be associated with a priority/importance tag to fill. An appropriate algorithm should be designed in such a way that it turns out to be aligned to those solutions that apply to only the most important tasks. In the future, the algorithm should be investigated to check the performance tuning factors. The problem of TSAP needs to undergo several other state-of-the-art algorithms too, so as to evaluate the proposed approach more accurately. In this work, the degree of constraint violation was equally distributed among all the objectives; thus, extensive experimentation is required to investigate other weighting schemes of the penalty functions. Also, a more exhaustive experimentation could be performed to fine-tune the parameters of the proposed MOEA.

On the other hand, CLOP presented the application of Multi-Objective Genetic Algorithm to optimize the coverage and lifetime of the wireless sensor network. The algorithm aims at maximizing the coverage and lifetime of the network while maintaining the connectivity of the network. This work has applied the robust

multi-objective evolutionary algorithms NSGA-II and SPEA-II, and fine-tuned the results obtained by them by applying local search operator. This application clearly demonstrates that the diversity of the solution is improved in this hybrid approach.

In the future, the work can be extended further to deal with the situation of the spatial error in deployment of sensors from high altitude aircrafts. In another direction, it also can include different objectives of sensor nodes such as remote surveillance of a facility. The problem CLOP can later be taken into the level where the deployed sensor nodes can communicate with a mobile base station. Further investigation can be made on number of objectives and weight can be imposed on them depending upon the requirement of the application domain. It will be a potential contribution from the future work if it remains capable of accounting for the uncertainty in the position of the sensors due to the mode of deployment.

References

[1] K. Tamura and S. Miura, "Necessary and sufficient conditions for local and global nondominated solutions in decision problems with multi-objectives," *Journal of Optimization Theory and Applications*, vol. 28, no. 4, pp. 501–523, 1979.

[2] D. Corne and J. Knowles, "Techniques for highly multiobjective optimisation: some nondominated points are better than others," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, p. 780, ACM, 2007.

[3] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms (Technical Report C3P 826)," *Pasadena, CA: Caltech Concurrent Computation Program, California Institute of Technology*, 1989.

[4] H. Feltl and G. Raidl, "An improved hybrid genetic algorithm for the generalized assignment problem," in *Proceedings of the 2004 ACM symposium on Applied computing*, pp. 990–995, ACM, 2004.

[5] C. Fleurent and J. Ferland, "Genetic hybrids for the quadratic assignment problem," *American Mathematical Society*, vol. 16, pp. 173–187, 1993.

[6] J. Knowles and D. Corne, "Towards landscape analyses to inform the design of a hybrid local search for the multiobjective quadratic assignment problem," *Soft computing systems: design, management and applications*, pp. 271–279, 2002.

[7] P. Merz and B. Freisleben, "A genetic local search approach to the quadratic assignment problem," in *Proceedings of the seventh international conference on genetic algorithms*, pp. 465–472, Citeseer, 1997.

[8] N. Krasnogor, "Towards robust memetic algorithms," *Recent advances in memetic algorithms*, pp. 185–207, 2005.

[9] P. Moscato, "Memetic algorithms: A short introduction," in *New ideas in optimization*, p. 234, McGraw-Hill Ltd., UK, 1999.

[10] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Transactions on Systems, Man, and CyberneticsPart C: Applications and Reviews*, vol. 28, no. 3, 1998.

[11] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 402–412, 2002.

[12] J. Knowles and D. Corne, "M-PAES: A memetic algorithm for multiobjective optimization," in *2000 Congress on Evolutionary Computation*, vol. 1, pp. 325–332, Citeseer, 2000.

[13] J. Knowles and D. Corne, "Memetic algorithms for multiobjective optimization: issues, methods and prospects," *Recent advances in memetic algorithms*, pp. 313–352, 2005.

[14] M. Lopez-Ibanez, L. Paquete, and T. St
"utzle, "Hybrid population-based algorithms for the bi-objective quadratic assignment problem," *Journal of Mathematical Modelling and Algorithms*, vol. 5, no. 1, pp. 111–137, 2006.

[15] D. Garrett, J. Vannucci, R. Silva, D. Dasgupta, and J. Simien, "Genetic algorithms for the sailor assignment problem," in *Proceedings of the 2005 conference on Genetic and evolutionary computation*, p. 1928, ACM, 2005.

[16] D. Gale and L. Shapley, "College admissions and the stability of marriage," *American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[17] D. Garrett, D. Dasgupta, J. Vannucci, and J. Simien, "Applying hybrid multiobjective evolutionary algorithms to the sailor assignment problem," *Advances in Evolutionary Computing for System Design*, pp. 269–301, 2007.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[19] E. Zitzler, M. Laumanns, L. Thiele, *et al.*, "SPEA2: Improving the strength Pareto evolutionary algorithm," in *Eurogen*, pp. 95–100, Citeseer, 2001.

[20] D. Dasgupta, G. Hernandez, D. Garrett, P. Vejandla, A. Kaushal, R. Yerneni, and J. Simien, "A comparison of multiobjective evolutionary algorithms with informed initialization and kuhn-munkres algorithm for the sailor assignment problem," in *Proceedings of the 2008 GECCO conference companion on Genetic and evolutionary computation*, pp. 2129–2134, ACM New York, NY, USA, 2008.

[21] A. D. Doty., "Technical report, instiIowa State University," in *http://www.public.iastate.edu/ ddoty/HungarianAlgorithm.html*.

[22] H. Kuhn, "The Hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[23] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[24] K. Ferentinos and T. Tsiligiridis, "Adaptive design optimization of wireless sensor networks using genetic algorithms," *Computer Networks*, vol. 51, no. 4, pp. 1031–1051, 2007.

[25] J. Holland, *Adaptation in natural and artificial systems*. MIT press Cambridge, MA, 1992.

[26] S. Sen, S. Narasimhan, and K. Deb, "Sensor network design of linear processes using genetic algorithms," *Computers & Chemical Engineering*, vol. 22, no. 3, pp. 385–390, 1998.

[27] D. Turgut, S. Das, R. Elmasri, and B. Turgut, "Optimizing clustering algorithm in mobile ad hoc networks using genetic algorithmic approach," *GLOBECOM-NEW YORK-*, vol. 1, pp. 62–66, 2002.

[28] G. Heyen, M. Dumont, and B. Kalitventzeff, "Computer-aided design of redundant sensor networks," *Computer Aided Chemical Engineering*, vol. 10, pp. 685–690, 2002.

[29] S. Jin, M. Zhou, and A. Wu, "Sensor network optimization using a genetic algorithm," in *Proceedings of the 7th World Multiconference on Systemics, Cybernetics and Informatics*, Citeseer, 2003.

[30] S. Aldosari and J. Moura, "Fusion in sensor networks with communication constraints," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*, p. 115, ACM, 2004.

[31] A. Bhondekar, R. Vig, M. Singla, C. Ghanshyam, and P. Kapur, "Genetic Algorithm Based Node Placement Methodology For Wireless Sensor Networks," *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2009.

[32] S. Hussain, A. Matin, and O. Islam, "Genetic algorithm for hierarchical wireless sensor networks," *Journal of Networks*, vol. 2, no. 5, p. 87, 2007.

[33] K. Ferentinos, T. Tsiligiridis, and K. Arvanitis, "Energy optimization of wirless sensor networks for environmental measurements," in *Proceedings of the International Conference on Computational Intelligence for Measurment Systems and Applicatons (CIMSA)*, pp. 250–255, 2005.

[34] S. Hussain and A. Matin, "Base station assisted hierarchical cluster-based routing," in *International Conference on Wireless and Mobile Communications, 2006. ICWMC'06*, pp. 9–9, 2006.

[35] A. Matin and S. Hussain, "Intelligent hierarchical cluster-based routing," *life*, vol. 7, p. 8.

[36] D. Jourdan and O. de Weck, "Layout optimization for a wireless sensor network using a multi-objective genetic algorithm," in *IEEE semiannual vehicular technology conference, Milan, Italy*, 2004.

[37] D. Shmoys and É. Tardos, "An approximation algorithm for the generalized assignment problem," *Mathematical Programming*, vol. 62, no. 1, pp. 461–474, 1993.

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

[39] M. Ishizuka and M. Aida, "Performance study of node placement in sensor networks," in *24th International Conference on Distributed Computing Systems Workshops, 2004. Proceedings*, pp. 598–603, 2004.

[40] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE transactions on Evolutionary Computation*, vol. 3, no. 4, p. 257, 1999.

[41] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary computation*, vol. 8, no. 2, p. 195, 2000.

[42] D. Corne, J. Knowles, and M. Oates, "The Pareto envelope-based selection algorithm for multiobjective optimization," in *Parallel Problem Solving from Nature PPSN VI*, pp. 839–848, Springer, 2000.

[43] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Parallel Problem Solving from Nature PPSN VI*, pp. 849–858, Springer, 2000.

Appendix

A General Algorithmic Steps

In this section, pseudocode of the operators designed specifically for the problems 'Task based Sailor Assignment Problem' and 'Coverage and Lifetime Optimization Problem' have been presented.

A1 Crossover Operator in TSAP

Input: Two parent solution vectors represented in terms of sailor-task assignment

Output: Two offspring solution vectors represented in terms of sailor-task assignment

Step1: Randomly select one timeslot from both the parent solution vectors. The selection depends on the probability which is inverse of the solution vector length.

Step2: Swap the entire timeslot between two parent solution vectors to generate two different new offsprings.

A2 Mutation Operator in TSAP

Input: One parent solution vector represented in terms of sailor-task assignment

Output: One mutated solution vector represented in terms of sailor-task assignment

Step1: Randomly select one sailor from one timeslot from the parent solution vector.

Step2: Obtain the task which was being assigned to the randomly selected sailor.

Step3: Acquire the list of sailors eligible to perform the task obtained in Step2.

Step4: Randomly pick any sailor from the eligible sailor list.

Step5: Assign the sailor obtained in Step4 to the task acquired in Step3 in such a way that it does not violate any constraint.

A3 Local Search Operator in TSAP

A3.1 Shift Local Search Operator in TSAP

Input: One parent solution vector represented in terms of sailor-task assignment

Output: One new solution vector represented in terms of sailor-task assignment

Step1: Randomly select one sailor from one timeslot from the parent solution vector.

Step2: Obtain the task which was being assigned to the randomly selected sailor.

Step3: Acquire the list of sailors eligible to perform the task obtained in Step2.

Step4: Randomly pick any sailor from the eligible sailor list.

Step5: Assign the sailor obtained in Step4 to the task acquired in Step3 in such a way that it does not violate any constraint.

Step6: Verify whether the newly generated solution vector is more fit than the previous solution vector in terms of objective functions.

then replace the previous solution vector with the newly generated solution vector.

else, keep the previous solution vector unaltered in the population.

A3.2 Swap Local Search Operator in TSAP

Input: One parent solution vector represented in terms of sailor-task assignment

Output: One new solution vector represented in terms of sailor-task assignment

Step1: Randomly select two different sailors S_1 and S_2 from the parent solution vector.

Step2: Obtain the tasks j_1 and j_2 respectively were being assigned to the sailors obtained in Step1.

Step3: Swap the sailors obtained in Step1 with the tasks acquired in Step2 i.e. assign sailor S_1 to task j_2 and sailor S_2 to task j_1 in such a way that no constraint gets violated.

Step4: Verify whether the newly generated solution vector is more fit than the previous solution vector in terms of objective functions.

then replace the previous solution vector with the newly generated solution vector.

else, keep the previous solution vector unaltered in the population.

A4 Boundary Search Operator in TSAP

Input: One parent solution vector represented in terms of sailor-task assignment

Output: One new solution vector represented in terms of sailor-task assignment

Step1: Randomly select two different positions p_1 and p_2 from the solution vector.

Step2: Randomly choose one window size S based on the solution vector length.

Step3: Obtain the sailors from that window starting from both positions p_1 and p_2. Let the list of sailors be S_{p1} and S_{p2} respectively.

Step4: Obtain the tasks j_{p1} and j_{p2} from that window starting from both positions p_1 and p_2. Let the list of tasks be j_{p1} and j_{p2} respectively.

Step5: Reverse the sailors obtained in Step3.

Step6: Assign sailors (reversed sailors obtained in Step5) S_{p1} to task j_{p2} and sailor S_{p2} to task j_{p1} in such a way that no constraint gets violated.

Step7: Verify whether the newly generated solution vector is more fit than the previous solution vector in terms of objective functions.

then replace the previous solution vector with the newly generated solution vector.

else, keep the previous solution vector unaltered in the population.

A5 Crossover Operator in CLOP

Input: Two parent solution vectors represented in terms of sensor nodes location coordinates.

Output: Two offspring solution vectors represented in terms of sensor nodes location coordinates.

Step1: Randomly select two different positions p_1 and p_2 from both the parent solution vectors respectively.

Step2: Randomly select one window size S based on the solution vector length.

Step3: Obtain the sensor nodes from that window starting from both positions p_1 and p_2. Let the list of sensor node coordinates be n_{p1} and n_{p2} respectively.

Step4: Swap the entire window between two parent solution vectors to generate two different new offsprings.

A6 Mutation Operator in CLOP

Input: One parent solution vector representaed in terms of sensor nodes location coordinates.

Output: One mutated solution vector representaed in terms of sensor nodes location coordinates.

Step1: Randomly select one position p_1 from the parent solution vector.

Step2: Obtain the sensor node coordinate from the position selected in Step1.

Step3: Add randomly generated real value scoped inside the hostile terrain with the sensor coordinate found in Step2.

Step4: Replace the old sensor coordinate obtained in Step2 with the newly generated real value in Step3

A7 Local Search Operator in CLOP

Input: One parent solution vector representaed in terms of sensor nodes location coordinates.

Output: One new solution vector representaed in terms of sensor nodes location coordinates.

Step1: Randomly select a sensor node from the parent solution vector.

Step2: Randomly generate probability value p.

Step3: Depending upon the value p, solutions lying around the sensor node have been selected.

if $p >= 0$ and $p < 0.25$

Add sensing radius to the x-coordinate and add sensing radius to the y-coordinate of the sensor node.

if $p >= 0.25$ and $p < 0.50$

Subtract sensing radius to the x-coordinate and add sensing radius to the y-coordinate of the sensor node.

if $p >= 0.50$ and $p < 0.75$

Subtract sensing radius to the x-coordinate and subtract sensing radius to the y-coordinate of the sensor node.

if $p >= 0.75$ and $p <= 1$

Add sensing radius to the x-coordinate and subtract sensing radius to the y-coordinate of the sensor node.

Step4: Verify whether the newly generated solution vector is more fit than the previous solution vector in terms of objective functions.

then replace the previous solution vector with the newly generated solution vector.

else, keep the previous solution vector unaltered in the population.