# A Multiobjective Evolutionary Algorithm For The Task Based Sailor Assignment Problem

Dipankar Dasgupta, Fernando Nino*, Deon Garrett, Koyel Chaudhuri,
Soujanya Medapati, Aishwarya Kaushal
Dept. of Computer Science
University of Memphis
Memphis, TN 38152
{dasgupta,luisnino,jdgarrtt,kchdhuri,ssmdpati,kaushal1}@memphis.edu

James Simien
Research Analyst
Navy Personnel Research, Studies, and Technology
Millington, TN 38055
james.simien@navy.mil

## ABSTRACT

This paper investigates a multiobjective formulation of the United States Navy's Task based Sailor Assignment Problem and examines the performance of a multiobjective evolutionary algorithm (MOEA), called NSGA-II, on large instances of this problem. Our previous work [3, 5, 4], consider the sailor assignment problem (SAP) as a static assignment, while the present work assumes it as a time dependent multitask SAP, making it a more complex problem, in fact, an NP-complete problem. Experimental results show that the presented genetic-based solution is appropriate for this problem.

## Categories and Subject Descriptors

J.0 [**Computer Applications**]: General Paper

## General Terms

Algorithms and Experimentation

## Keywords

Sailor Assignment problem, Multiobjective evolutionary algorithms, Hybrid metaheuristics, Task Based Sailor Assignment Problem and NSGA-II

*Also Associate Professor, Department of Computer Science, National University of Colombia, Bogota, Colombia


*GECCO'09,* July 8–12, 2009, Montréal Québec, Canada.


## 1. INTRODUCTION

According to the United States Navy's personnel policies, roughly every three years sailors serving on active duty are reassigned to a different job. As a result, at any given time there exists a sizable population of sailors to be reassigned to available jobs.The Navy's goal is to identify sailor and job matches that maximize some overall criteria of satisfiability of sailors and commanders and is referred to as the Sailor Assignment Problem (SAP).

In [4], [5], evolutionary and hybrid algorithms were tested on single and multiobjective versions of the SAP. In [3] a hybrid genetic algorithm featuring an efficient SAP solver, the Kuhn-Munkres algorithm [6, 1, 8], was used to further improve performance. The Kuhn-Munkres algorithm solves linear assignment problems in $O(n^3)$ time and it was chosen due to the fact that the single objective versions of SAP with small modifications are linear assignment problems. This approach, while yielding very good solutions, suffers from very long run times as the problem size increases.

In this work, multiobjective 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 a sailor, a 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 that shift. Accordingly, a sailor can be assigned to different tasks in each time shift.

Particularly, the Nondominated Sorting Genetic Algorithm (NSGA-II) was run to obtain multiple diverse solutions to TSAP in a single run of the algorithm. Section 1.1 briefly presents the Generalized Assignment Problem, which is closely related to TSAP. Section 2 then describes the Task based Sailor Assignment problem in detail, a variation of the Sailor Assignment Problem. Also a brief overview of related work is presented there. Implementation of NSGA-II to the TSAP is described in Section 3 and then Section 4 describes the experiments and results produced.

## 1.1 General Assignment Problem

The General Assignment Problem (GAP) is a generalization of the assignment problem that was originally introduced as the problem of scheduling parallel machines with costs [12]. GAP is as follows: Given a set of bins and a set of items that have a different size and value for each bin, pack a maximum-valued subset of items into the bins. GAP can be also stated in a job assignment scenario as follows: a number of agents $n$ and a number of jobs $m$ to be performed by the agents are given. Any agent can perform any job but each agent has a budget and the sum of resources required for jobs assigned to it (sailor) 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 do not exceed their budget and 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 job $j$, then GAP can be expressed as the following integer linear program:

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

subject to the constraints:

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

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

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

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

# 2. TASK BASED SAILOR ASSIGNMENT PROBLEM

TSAP can be seen as a "fine-grained" version of SAP. 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 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, same task may not continue next day or 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, clearly same sailor cannot perform more than one task in the same time shift. Thus 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 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 that are required to complete in that shift. Accordingly, a sailor can be assigned different tasks in different shifts.

Similar to SAP, here also sailors are examined for a variety of qualifications and constraints to determine if each sailor is a valid match for one of 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, 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 set of assignments must satisfy Naval regulations and must do so at a reasonable cost to the Navy. In this work, we will assume that sailors may be assigned only those jobs for which the sailor applied and ranked.

After the command preview stage, the detailer must construct the set of assignments in accordance with the regulations and preferences of the Navy. The complexity of the detailing process limits the Navy's ability to make effective decisions. Therefore, an automated system for quickly finding good solutions to the problem is required. Formally, TSAP can be described as follows:

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 its 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 number of sailors along with fulfilling the previous objectives of SAP, namely, maximize total training score, the sailor preference score, the commander preference score, and minimize the cost.

Additionally, the following constraints are involved in TSAP:

- the sum of the resources for a sailor-task assignment over given time should not exceed its capacity
- same sailor cannot be assigned to multiple tasks in one time slot, and
- 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 following multidimensional integer linear program:

$$\text{minimize} \quad \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), ..., f_N(\mathbf{x}))$$

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

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

with $C_{ij}^{obj}$ the cost of assigning sailor $i$ to task $j$ on time slot $k$ determined by objective $obj$; and $\mathbf{F}(\mathbf{x})$ is subject to the

constraints:

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

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

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

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

The generalized assignment problem has been shown to be NP-hard. In addition, recently, it was shown that it is $(e/(e-1)-\epsilon)$ hard to approximate for every $\epsilon$ [11]. Therefore, it can be shown that TSAP is NP-complete as explained here. Since, GAP does not have any time dependency in it, therefore to reduce an instance of GAP to TSAP problem, consider a single time slot which is having entire tasks to be done by sailors in GAP. Here no sailor will be repeated in this time slot and the sum of the resources for sailor-task assignment will not exceed their capacities. Since there is only one time slot covering all the tasks, it is ensured that all the tasks are assigned to different sailors. All the constraints are then fulfilled and GAP instance is reduced to TSAP instance. Thus by reduction process, it can be inferred that the solution for TSAP and GAP will be identical.

In this paper, a MOEA is proposed to solve the TSAP, which is presented in the next section. The MOEA will produce a set of nondominated solutions that are provided to a solution analyzer to facilitate closer examination of the results by the detailor.

# 3. MULTIOBJECTIVE EA FOR TSAP

Some of the most promising approaches to many multiobjective optimization problems arise from evolutionary techniques. One of the primary benefits touted by practicioners is the ability of multiobjective evolutionary algorithms to cover the Pareto front in a single run of the algorithm. Any multiobjective EA has two basic goals: First the algorithm must push the initial population in the direction of Pareto optimal solutions. Ideally, the algorithm would terminate with a set of nondominated solutions such that no possible solution could dominate any member of the Pareto front. This set is called the true Pareto optimal set, or true Pareto front. In practice, we often wish only to find solutions that are very good and nondominated with respect to one another. This requires that the algorithm routinely improve the quality of the initial randomly generated solutions until some level of acceptability has been met.

Several MOEAs have been proposed in the literature with varying degrees of success. Recent research has prompted a flurry of activity among MOEA researchers resulting in a series of new algorithms based on Pareto dominance and incorporating features such as elitism. Of these algorithms, the Nondominated Sorting Genetic Algorithm by Deb and others (NSGA-II) [2] and the Strength Pareto Evolutionary Algorithm (SPEA2) of Zitzler et al. [10] have been widely studied and found to be effective across a range of common test functions as well as combinatorial optimization problems.

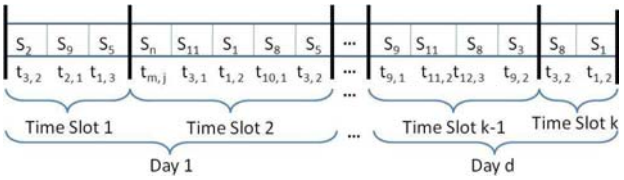



**Figure 1: Chromosome representation**

In this work, NSGA-II is used to solve TSAP as described next.

## 3.1 Chromosome Representation Scheme

One of the main decisions to be made when adapting a genetic algorithm to a particular problem is how to encode solutions to the problem in a manner amenable to genetic search. In this work, we utilize an integer encoding. 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 time 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 variable number of tasks to be done by eligible sailors. Fig. 1 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 $S_i$.

Decoding a chromosome, $C$, is simply performed by assigning sailor $S_i$ to 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 1 a random crossover or mutation could easily result in a violation of the constraints. The second and third constrains of TSAP (in Section 2) ensure that feasible solutions do not assign two different tasks to the same sailor at a particular time slot, the sailor's capacity should not be exceeded and that all the tasks are assigned a sailor to perform it. But, the proposed encoding cannot prevent from violations of the constraints to occur. Notice that it can also happen that a sailor or task is left unassigned.

## 3.2 Objectives and Fitness Measure

As mentioned in Section 2, 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)

$$\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{t} x_{ijk}CR_{ij}}{\sum_{i=1}^{n} max_{j=1}^{m} SR_{ij}}$$

Even though some of the objectives must be maximized, the multiobjective optimization process used in this paper consists of minimizing 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.

However, it should be noticed that while in principle, these values are scaled such that the objective function values range from 0 to 1, in practice the extreme values never occur. The reason is that multiple sailors are in contention for the same tasks. Also it is assumed that each sailor can be assigned to those 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 (slate) reaches 1.0 value in all or in any objectives.

A crucial issue here is the assurance that all necessary constraints (discussed in Section 2) 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:

$$UTP = \frac{\text{No. of unassigned tasks}}{\text{Total no. of tasks}}$$

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

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

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

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

A single penalization valued was computed as the sum of $UTP$, $RSP$ and $CEP$. 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.

## 3.3 Genetic Operators

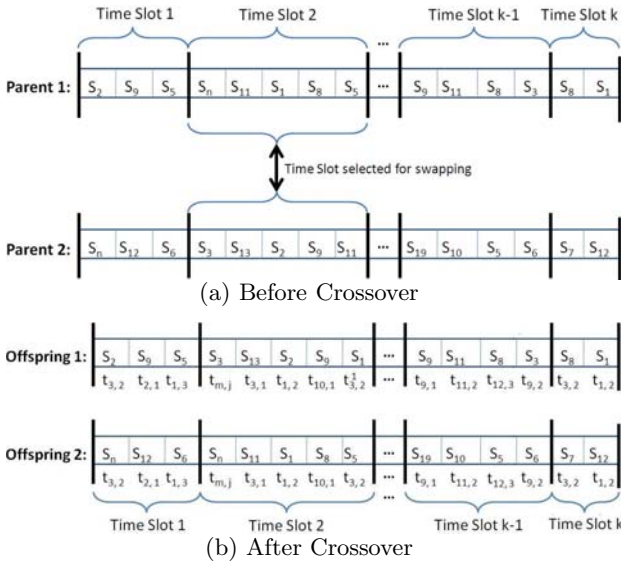
Specific selection, mutation and crossover operators were implemented. It is important to note that mutation and crossover operators can produce solutions that might violate the constraints. Therefore, a repair operator is implemented to try to maintain feasible solutions. One simple mutation operator chooses a particular task at random and then assigning 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 violate the capacity constraint. Thus, These genetic operators are described in the following sections.

### 3.3.1 Mutation

The mutation operator works as follows: Initially choose one random location from the chromosome. Find out the sailor who has been assigned to that task. Then determine the list of possible sailors who can perform that task and randomly pick a sailor out of them and substitute the current sailor with the new one keeping two constraints in mind: new sailor is not repeated in that time slot and 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 important to notice that given a specified mutation rate $P_m$, the effective mutation rate, the percentage of attempted mutations that actually effect some change in the individual, is less than $P_m$, 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 which 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 a randomly selected qualified sailor



(a) Before Crossover

(b) After Crossover

**Figure 2: The offspring produced by swapping of time slots between two parents**

if one is available. If no sailor is available for performing a particular task, then this randomization process makes 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.

### *3.3.2 Crossover*

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 Fig. 2.

### *3.3.3 Repair operator*

A repair operator is implemented 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.

It is worth noticing that 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 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 (for example, hours) 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 genetic approach is enhanced by including some local search operators in a hybrid approach as explained next and thus the results of several approaches on different problem instances are compared.

## 3.4 Hybrid approach

The proposed genetic based technique was combined with some local search operators and is as follows. Specifically, two local search operators were implemented: a sailor shift operator and a sailor swap operator. The sailor shift search operator performs a random shift of a sailor on a solution in the same way as the TSAP mutation operator. On the other hand, the sailor swap operator picks a particular sailor assigned to perform a task and tries to swap it with another sailor chosen at random, previously checking feasibility and constraint violations. This process is repeated for a prespecified number of times. Several experiments were carried out for all four versions of genetic algorithms: one for standard NSGA-II and other three for hybrid NSGA-II.

# 4. EXPERIMENTS AND RESULTS

Due to unavailability of real sailor data, a problem generator was developed to allow testing on variety of problem instances of different sizes and difficulties. These simulated instances contain values for all the objectives along with the resource requirements for performing tasks and capacities of the sailors. These values were generated assuming normal distributions. In previous work [4], it was found that one of the most important factors governing the difficulty of a problem instance was the contention for tasks. Given a fixed number of sailors, the more total tasks available for the sailors to choose from, the less difficult the problem is. In a real world scenario, it is unlikely that there will exist many more available tasks in all time slots than sailors to fill them.

To test the evolutionary algorithm, a set of sample problems is produced which contain a reasonable ratio of sailors to tasks. Each problem was generated according to a specified number of sailors, tasks and time slots. Also, the mean and standard deviation of a normally distributed random variable determine number of task types each sailor can perform. In all the problems, 7 days with 12 time slots per day were considered which makes the problem harder since it increases the number of tasks for all the time slots. Also, a variable expected ratio of sailors per task was used. Table 1 shows the parameters of each sample problem used in this work. The task-sailor ratio is defined here as the rate of task types a sailor can perform on average.

For each problem, 10 runs of the evolutionary algorithm were performed for three different population sizes: 100, 200

| | No. sailors | No. tasks | Task/sailor ratio |
|---|---|---|---|
| Problem 1 | 1,000 | 500 | 0.25 |
| Problem 2 | 2,000 | 1,000 | 0.05 |
| Problem 3 | 4,000 | 2,000 | 0.05 |
| Problem 4 | 5,000 | 200 | 0.01 |

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

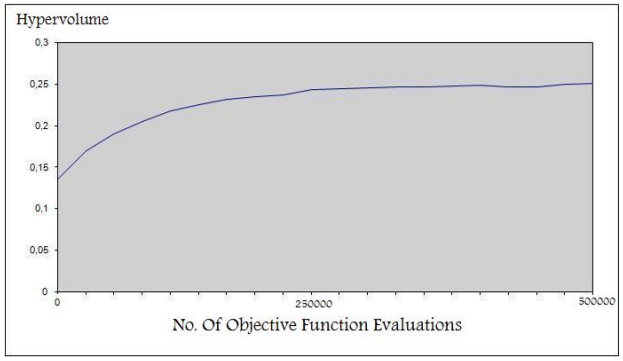



**Figure 3: Change in the hypervolume during 500,000 objective evaluations for a 5,000 sailor instance of TSAP with 100 population size.**

and 400 individuals where each run consists of 500,000 evaluations of the objectives. Binary tournament selection, where two individuals are chosen at random and the fittest one 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 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, the repair operator is executed. For doing theses experiments jMetal, an open framework for MOEAs was used [9].

Figure 3 depicts the change in the hypervolume in one run of NSGA-II for a 5,000 sailor instance of TSAP with 100 population size. Notice that after a certain number of evaluations (about 250,000 evaluations) the hypervolume will remain almost constant.

Most existing performance metrics in multiobjective optimization require to know a set of Pareto-optimal solutions, and such solutions are to compare against the approximate solutions obtained. It is also important to notice that 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 often computationally expensive [13]. Since Pareto-optimal solutions for considered instances of TSAP are not known, some of the 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 thus the results obtained are compared. Particularly, the results of standard NSGA-II were compared against the hybrid NSGA-II. 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 trials are reported in Table 2 and Table 3. It can be seen from Table 2 that as the difficulty of the problem increases (see 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. From

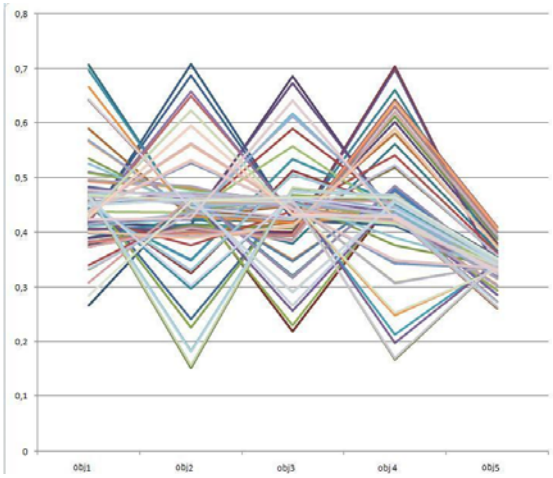



**Figure 4: Diversity of solutions found in the Pareto front using NSGA-II on a 1,000 sailor instance of the TSAP and population size equal to 100.**

Figure 4 shows a parallel plot for sample solution by applying NSGA-II for 1,000 sailors, while Figure 5 shows the results for the same problem instance obtained by the hybrid approach. 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 figures 4 and 5. Particularly, notice in Figure 5, solutions with lower values for objectives 1, 3 and 5 are found. For instance, a specific solution that assigns only 201 sailors, which corresponds to only 20.1% of the total number of sailors, is provided.

Same as above, Figure 6 shows a parallel plot for a sample solution by applying NSGA-II for 5,000 sailors, while Figure 7 shows the results for the same problem obtained by the hybrid approach combining both local search operators. Specifically, shift local search operator is performed in every $10{,}000^{th}$ evaluation whereas swap local search operator is executed in every $15{,}000^{th}$ evaluation. From Figure 5 it is observed that solutions with lower values are found for all objectives. Moreover, in case of last objective (number of assigned sailors) the hybrid approach provides a solution with less number of sailors, such as 83 (0.0166) whereas NSGA-II comes with a solution with 87 sailors (0.0174). It is worth noticing that 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 and a combination of both) for same problems de-

| | Approaches | | | |
|---|---|---|---|---|
| Sailors | 1000 | 2000 | 4000 | 5000 |
| NSGA-II | .1843±.0103 | .2222±.0015 | .274480±.0019 | .2565±.0029 |
| shift LS | .1864±.0106 | .2228±.0017 | .2745±.0019 | .2559±.0028 |
| swap LS | .1809±.0126 | .2229±.0014 | .2743±.0014 | .2551±.0026 |
| both LS | .1906±.0056 | .2231±.0017 | .2757±.00014 | .2567±.0031 |

**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 (LS) operators (shift, swap and a combination of both) for different instances of TSAP**

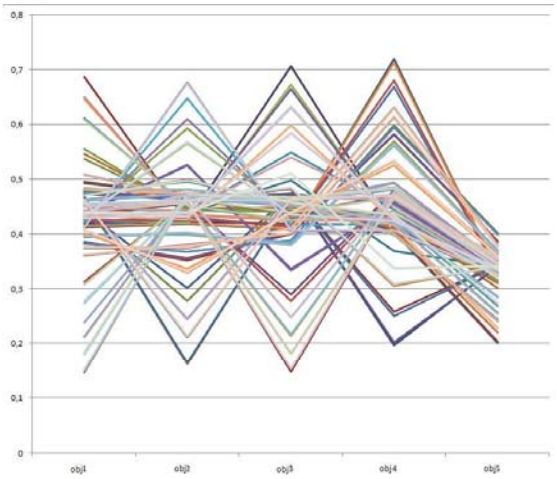



**Figure 5: Diversity of solutions found in the Pareto front using a hybrid approach alternating both local search operators on a 1,000 sailor instance of the TSAP and population size equal to 100.**

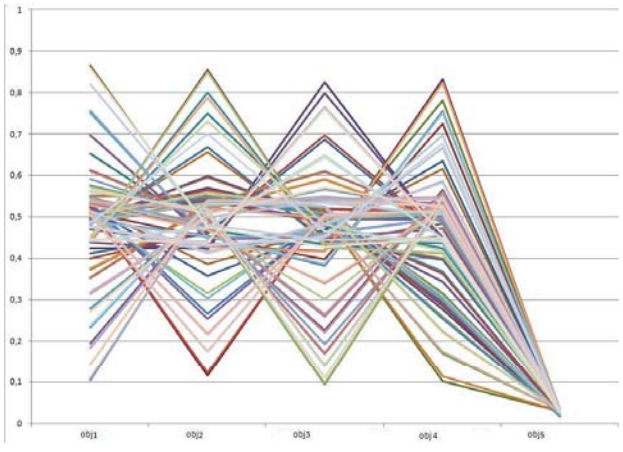



**Figure 6: Diversity of solutions found in the Pareto front using NSGA-II on a 5,000 sailor instance of the TSAP and population size equal to 100.**

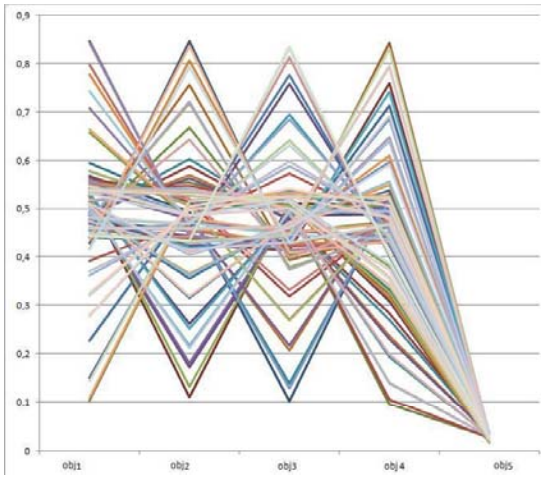



**Figure 7: Diversity of solutions found in the Pareto front using a hybrid approach alternating both local search operators on a 5,000 sailor instance of the TSAP and population size equal to 100.**

| | Population size | | |
|---|---|---|---|
| Sailors | 100 | 200 | 400 |
| 1,000 | .1844±.0103 | .1648±.0055 | .1529±.0040 |
| 2,000 | .2223±.0015 | .1963±.0015 | .1665±.0019 |
| 4,000 | .2745±.0019 | .2451±.0014 | .2096±.0023 |
| 5,000 | .2565± .0029 | .2449±.0023 | .2305±.0019 |

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

fined in Table 1 and population size 100. It is observed that the hybrid approach that alternates both local search operators provides better solutions from hypervolume perspective. Here, notice that hybrid approach provides better solutions from hypervolume point of view, i.e., better approximations towards the optimal Pareto front.

# 5. CONCLUSION

This paper has presented a multiobjective evolutionary algorithm to solve the Task based Sailor Assignment Problem. Compared to our previous studies [3, 5, 4], this work modeled a problem in a more complex fashion that is time independent multitasking problem. The proposed approach based on the multiobjective evolutionary algorithm NSGA-II has shown to provide a continuum of solutions in the interior portions of the trade-off surface and also achieved a good diversity of solutions along the Pareto set approximation. However, further work is needed to determine the effect on the performance of the algorithm.

Additional work is required to improve the MOEA by including some domain specific genetic operators which can potentially provide additional improvements. In particular, a crossover operator which explores more potential offspring while still minimizing constraint violations. Such an operator could potentially provide benefits both in solution quality as well as computation time. Another potential improvement of the algorithm could be obtained through the injection of good solutions into the initial population.

While this work currently incorporates five distinct objectives, there are several additional components which require further investigation. For instance, the fact that not all tasks are equally important to fill. An effective algorithm should bias solutions towards filling the most important tasks.

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.

# 6. ACKNOWLEDGEMENTS

This project is funded under the Scientific Services Program (TCN: 08179 Mod 1), which is sponsored by the US Research Office and is managed by Battelle under Prime Contract No: W911NF-07-D-0001. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.

# 7. REFERENCES

[1] A. D. Doty, Iowa State University. http://www.public.iastate.edu/∼ ddoty/HungarianAlgorithm.html.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[3] 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. *Proceedings of the 2008 Genetic and Evolutionary Computation* Conference (GECCO-08), ACM press, 2008.

[4] J. D. Garrett, J. Vannucci, R. Silva, D. Dasgupta, and J. Simien. Genetic algorithms for the sailor assignment problem. *Proceedings of the 2005 Genetic and Evolutionary Computation Conference (GECCO-05)*. ACM press, 2005.

[5] J. D. Garrett, J. Vannucci, R. Silva, D. Dasgupta, and J. Simien. Applying hybrid multiobjective evolutionary algorithms to the sailor assignment problem. In L. Jain, V. Palade, and D. Srinivasan, editors, *Advances in Evolutionary Computing for System Design*. Springer Verlag, 2007.

[6] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

[7] L. McCauley and S. Franklin. A large multi-agent system for navy personnel distribution. *Connection Science*, 14(4):371–385, December 2002.

[8] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 5(1):32–38, March 1957.

[9] Networking and S. Emerging Optimization (NEO), University of Malaga. JMetal: Metaheuristic algorithms java library. http://mallba10.lcc.uma.es/wiki/index.php/JMetal. Version 1.5, april 2008.

[10] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.

[11] R. Cohen, L. Katzir, and D. Raz. An Efficient Approximation for the Generalized Assignment Problem *Information Processing Letters*, 100(4):162–166, November 2006.

[12] D. B. Shmoys and Eva Tardos. An approximation algorithm for the generalized assignment problem. *Math. Program.*, 62(3):461–474, 1993.

[13] K. Deb, L. Thiele, M. Laumanns and E. Ziztler. Scalable Multi-objective optimization test problems. *In Proceedings of the Congress on Evolutionary Computation (CEC 2002), 825–830, 2002*