Solving Battalion Rescheduling Problem Using Multi-objective Genetic Algorithms

Irfan Younas[1], Farzad Kamrani[2], Farshad Moradi[2], Rassul Ayani[1], Johan Schubert[2], and Anne Håkansson[1]

[1] KTH Royal Institute of Technology, Stockholm, Sweden
{irfany,ayani,annehak}@kth.se

[2] Swedish Defence Research Agency, Stockholm, Sweden
kamrani@kth.se, {farshad.moradi,johan.schubert}@foi.se

Abstract. In this paper, we consider the problem of rescheduling human resources in a battalion where new activities are assigned to the battalion by higher headquarters, requiring modification of an existing original schedule. The problem is modeled as a multi-criteria optimization problem with three objectives: (i) maximizing the number of tasks that are performed, (ii) minimizing the number of high-priority tasks that are missed, and (iii) minimizing the differences between the original schedule and the modified one. In order to solve the optimization model, we adopt Non-dominated Sorting Genetic Algorithm-II (NSGA-II). The accuracy of NSGA-II in this context is verified by considering a small-sized problem where it is easy to verify solutions. Furthermore, we consider a realistic problem instance for a battalion with 400 agents and 66 tasks in the initial schedule. We present the computational results of rescheduling when unpredictable activities emerge.

Keywords: Battalion rescheduling, Multi-objective optimization, Genetic algorithms.

1 Introduction

Similar to development in many countries, the Swedish Armed Forces (SAF) has started undergoing a major process of change over the past few years. These changes embrace transformation from an invasion defense to a mission oriented defense. One major consequence of this transformation is that the SAF has become drastically smaller and vulnerable to personnel shortage. The strictly hierarchical structure and closed nature of military organizations do not allow recruitment of new (temporary) personnel with right competencies if an urgent need arises and new tasks are assigned to the organization. Thus, military units are periodically forced to handle an essentially larger number of tasks with the same manpower resources, which requires an efficient utilization of these resources and calls for more flexible and effective planning of personnel, and scheduling of tasks.

A battalion is a military unit with 400 to 1200 soldiers, which is considered to be the smallest unit capable of independent operations. In the SAF, the battalion commander and staff (from hereon referred to as the commander) are responsible for planning and scheduling different activities of the battalion personnel such as education, training, exercises, tasks and missions. All activities are pre-planned and scheduled on a yearly basis by the commander in a way that all tasks and missions are performed and at the same time the personnel attend necessary educational and training sessions and exercises to achieve the required competencies for performing those tasks or missions. Considering that activities require different number of personnel with different types and levels of competencies, scheduling all activities per se is a difficult and complex problem. Consequently, this problem becomes even more complex and challenging since the set of workforce competencies is not static and changes over time as the personnel attend military education, training or exercises. A typical scenario is where personnel perform different types of tasks and later take part in different educational activities to acquire the right competencies needed for more complex tasks and missions scheduled to be performed after the courses. As a result, the schedule for this kind of scenario is sensitive to changes and potentially vulnerable to disturbances.

Nevertheless, in reality unpredictable activities emerge periodically, as a result of unforeseen events and are assigned to the battalion by higher headquarters. The commander then may be forced to do some rescheduling by removing personnel from scheduled tasks, such as educational activities in order to be able to deal with imminent tasks. Assigning personnel to these tasks compromises the commitment of the battalion to perform important and prioritized tasks or missions, since some personnel may miss educational activities and do not have the required competencies for forthcoming tasks and missions.

While doing the rescheduling the commander faces conflicting objectives such as maximizing the total number of performed tasks, prioritizing more critical tasks and missions, and preserving the original schedule to the largest extent. These objectives partly reflect the interests of different stakeholders. For instance, while the commander of the battalion wants to perform as many tasks as possible, and prioritizes the critical tasks, the most important criterion for the personnel is that their schedules are changed as little as possible.

In this paper, we propose a novel model for rescheduling different activities of the battalions personnel. We formulate the rescheduling problem as a multi-objective optimization problem, in which we consider three objectives: (i) maximizing the total number of performed tasks, (ii) minimizing the number of high priority tasks that are missed, and (iii) minimizing the differences between the initial schedule and updated schedule. A multi-objective mathematical model, with three conflicting objectives and a set of constraints is built. In multi-objective optimization problems with conflicting objectives, the goal is to find a set of Pareto-optimal solutions. Multi Objective Evolutionary Algorithms (MOEAs) are well-known metaheuristics for sampling intractably large and highly complex search spaces. MOEAs search more than one solutions in parallel and are suitable for finding the Pareto-optimal set for multi-objective optimization problems. MOEAs maintain population of non-dominated set of individuals and comparison of two individual solutions is based on Pareto-dominance. A non-dominated solution is one in which it is impossible to improve an objective without worsening at least one other objective. In the last fifteen years, several multi-objective evolutionary algorithms have been proposed. In order to solve our proposed multi-objective problem, we adopt Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [2], which is a well-known evolutionary algorithm. To test the accuracy of NSGA-II for our problem, we consider a small problem. We further conduct experiments on a large scale problem and present computational results.

The rest of the paper is organized as follows: in Section 2, related work is briefly reviewed. In Section 3, a mathematical multi-objective model for battalion rescheduling is presented. In Section 4, the multi-objective algorithm is described. Section 5 presents the experiments, including problem instances, parametrization and discussion of results. Finally in Section 5, the conclusion and summary of the paper are outlined.

2 Related Work

To the best of our knowledge, existing work does not address multi-objective optimization rescheduling problems similar to the battalion rescheduling model discussed in this paper. Even though we find some related work on scheduling and rescheduling problems, it is different in several aspects. Clark and Walker [3] present some models and computational results for nurse scheduling and rescheduling considering nurses' preferences. Due to changes in nursing cover requirements or unavailability of nurses to work their assigned shifts, rescheduling is needed to fill the gaps in the initial schedule. The overall goal is to reschedule nurses while considering their preferences in a manner that disrupts as little as possible the existing schedule. In nurse rescheduling, the aim is to minimize changes to the initial schedule as well as minimizing the total cost.

Moz and Pato have proposed different techniques for solving nurse rescheduling problems with the objective to minimize the changes in the original schedule. Moz and Pato [4] use a genetic algorithm to solve the problem and introduce a second objective into the fitness function. The second objective is to minimize the overtime, which they describe as minimizing the difference between the number of scheduled duties and the number of performed duties.

Maenhout and Vanhoucke [5] propose an evolutionary metaheuristic to solve nurse rerostering problem. Personnel roster determines the line-of-work for each person [5]. Dynamic nature of operating environment can cause unexpected events, which in turn lead to infeasibilities and schedule disruptions. In order to cope with this situation rescheduling is necessary. Computational experiments are performed on a well designed data set and the results of the proposed method are compared with other already existing methods in literature. Chicano et al. [6] focus on the Software Project Scheduling (SPS) [7]. Resources with a set of skills are assigned to the tasks, and the objectives of the problem are to minimize the time and cost of the project. The authors compare performance of different multi-objective evolutionary algorithms in solving the proposed SPS model. Hao and

Lin [8] study evolutionary algorithms to solve multi-objective model of the job shop rescheduling problem. The model considers n jobs to be performed on m machines and the objectives of the problem are to minimize the tardiness and makespan. The rescheduling is needed in case of new job arrivals and machine breakdowns.

The nurse rescheduling problem has some similarity with our problem in one of the objectives, which is to minimize schedule disruptions. However, the battalion rescheduling problem is different in several ways. Firstly, in our model, tasks require different number of personnel with different types and levels of competencies. Secondly, new tasks with different set of requirements can be assigned to the battalion during any time of the year. Thirdly, the set of workforce competencies is not static and changes over time as the personnel attend military education, training or exercises. These characteristics make the battalion rescheduling problem more complex and challenging than the nurse rescheduling problem. Job shop rescheduling problem is also similar to our problem in some aspects but dynamic nature of the set of workforce competencies in our problem differentiates it from the job shop rescheduling problem.

3 Problem Formulation and Model Description

We model the battalion rescheduling problem as a multi-objective optimization problem. Battalion personnel are modeled as the set of agents $A = \{a_i \mid i = 1, \ldots, m\}$, where m is the number of personnel in the battalion (agents). Assuming that q different types of competencies $C = \{c_1, \ldots, c_q\}$ are relevant for the function of the battalion, each agent $a_i \in A$ has a set of numerical attributes $c_i = \{c_{il} \mid l = 1, \ldots, q\}$, where $c_{il} \in \{0, \ldots, 4\}$ specifies to what extent agent a_i is equipped with competency c_l. The competencies of all agents can conveniently be denoted by matrix $C = [c_{il}]_{\{m \times q\}}$.

All activities (tasks, missions, military education, training, exercises) are modeled as the set $T = \{t_j \mid j = 1, \ldots, n\}$, where n is the number of activities. To each task $t_j \in T$, a 4−tuple $(size_j, start_j, duration_j, priority_j)$ is associated, where $size_j$ is the number of agents required to perform the task, $start_j$ is the start date of the task, $duration_j$ is the duration of the task in days, and $priority_j$ is the priority of the task with the binary values *low* and *high*.

Each task t_j consists of a set of positions. We enumerate and index all positions in a sequence from the task with the smallest index to the largest and use the notation $\overline{j} = \sum_{j'=1}^{j} size_{j'}$ for the sum of the number of positions in all tasks t_1, to t_j. Using this notation task t_j consists of positions $\{p_k \mid j = \overline{(j-1)}+1, \ldots, \overline{j}\}$. The set of all position is denoted by $P = \{p_k \mid k = 1, \ldots, \overline{n}\}$. We also define the function $tasknumber : \{1, \ldots, \overline{n}\} \curvearrowright \{1, \ldots, n\}$, which gives the index of task t_j that p_k belongs to (i.e. , $tasknumber(k) = j$).

Each position p_k has competency requirements $w_k = \{w_{lk} \mid l = 1, \ldots, q\}$, where w_{lk}, defines the minimum level of competency type c_l required by position p_k, meaning that agent a_i may be assigned to position p_k only if $\forall l, c_{il} \geq w_{lk}$. The competencies required by all positions can conveniently be denoted by matrix

$W = [w_{lk}]_{\{q \times \overline{n}\}}$. After completing some tasks, one or several competencies of agents may increase one level. These tasks model military education courses, training and exercises. In other words, courses are modeled as ordinary tasks and are distinguishes from them only by being associated with an update function ($update : w_l \curvearrowright w'_l, l = 1, \ldots, q$), which updates competencies of all agents a_i participating in the course. In this study, the update function is defined as $w'_l = w_l + 1$ for some specified values of l, depending on the course.

If all resources (agents) required by a task are available, the task will be assigned to agents, otherwise the task remains unassigned. An agent can be assigned to at most one task at any time.

A schedule is defined by matrix $X = [x_{ik}]_{m \times \overline{n}}$, where $x_{ik} \in \{0, 1\}$ and $x_{ik} = 1$ if position p_k is assigned to agent a_i, and $x_{ik} = 0$ otherwise.

A schedule X is feasible subject to the following constraints:

$$\forall k, \sum_{i=1}^{m} x_{ik} \in \{0, 1\}, \tag{1}$$

$$\forall j, \sum_{k=\overline{j-1}+1}^{\overline{j}} x_{ik}, \in \{0, size_j\} \tag{2}$$

$$\forall k, x_{ik} = 1 \Rightarrow \forall l,\ c_{il} \geq w_{lk}, \tag{3}$$

$$\begin{aligned} &\forall i, x_{ik} = 1 \wedge x_{ik'} = 1 \Rightarrow \\ &[start_j, start_j + duration_j) \cap [start_{j'}, start_{j'} + duration_{j'}) = \phi, \\ &where\ j = tasknumber(k)\ and\ j' = tasknumber(k'). \end{aligned} \tag{4}$$

Constraint 1 implies that only one agent is assigned to a position, constraint 2 ensures that tasks are either assigned to all required agents or not performed at all, constraint 3 states that all agents should have all qualifications required by the task and constraint 4 asserts that agents are assigned to only one task at any time.

Given agents A, initial tasks T^0, positions P^0 with requirements W^0 and an initial schedule $X^0 = [x^0_{ik}]_{m \times \overline{n^0}}$, which assigns tasks to agents, a set of new tasks T^1 are arrived at a given time ($rescheduling\_time$). The battalion rescheduling problem is defined as finding a new assigning schedule $X^1 = [x^1_{ik}]_{m \times \overline{n^1}}$ for all tasks, which have not already started such that some optimization criteria are fulfilled, subject to constraints 1 to 4. Tasks that have not started at $rescheduling\_time$ are specified by $T^1 \cup T^0 \setminus \{t_j \mid j = 1, \ldots, n, \wedge start_j < rescheduling\_time\}$.

In this paper, we focus on three criteria, τ, μ and δ, defined as:

$$\tau = |\{t_j \in T^0 \cup T^1 \mid \exists x_{ik} \in X^1\ s.t.\ tasknumber(k) = j \wedge x_{ik} = 1\}| \tag{5}$$

$$\mu = |\{t_j \mid t_j \notin \tau \wedge priority_j = high\}| \tag{6}$$

$$\delta = \sum_{i=1}^{m} \sum_{k=1}^{\overline{n^1}} (x_{ik}^1 - x_{ik}^0) + \sum_{k=1}^{\overline{n^0}} x_{ik}^1 . \tag{7}$$

The value τ is the number of tasks performed, μ is the number of high-priority tasks that are not performed, and δ is the difference between the original schedule the new schedule expressed by using the number of agents that have their schedule changed. Maximizing τ and minimizing μ and δ is desirable.

4 Multi-objective Algorithm

The rescheduling problem formulated in the previous section is a combinatorial optimization problem, where the search space becomes intractable even for moderate sized instances. Genetic Algorithms (GAs) [9][10] are widely used meta-heuristics for sampling intractably large and highly complex search spaces. The GAs have frequently been used for solving many scheduling and optimization problems [12][11][4][5]. In this paper, we adopt Non-dominated Sorting Genetic Algorithm- II (NSGA-II), which is a well-known genetic algorithm to solve multi-objective optimization problems. We modify the implementation of NSGA-II provided by jMetal framework [2] to solve our problem. In the next subsection, we explain the steps and parametrization of the algorithm for solving the proposed multi-objective optimization problem.

4.1 NSGA-II Algorithm

In order to design a GA for a particular optimization problem, first step is to devise a suitable representation scheme. In this paper, we choose a scheme in which a schedule (chromosome in GAs literature) is represented by a n-dimensional vector of subsets of tasks. In the considered scheme, n-dimensional means that we have n tasks which need to be performed by n groups of agents and each group j consists of $size_j$ agents. An example chromosome shown in Figure 1 consists of 4 tasks requiring 3, 2, 4 and 5 agents respectively. Assume that there are 5 agents available. It means that $m = 5$, $n = 4$, $size_1 = 3$, $size_2 = 2$, $size_3 = 4$, and $size_4 = 5$. We also assume that task 1 and 2 are overlapping, so an agent assigned to one of the task cannot be assigned to another.

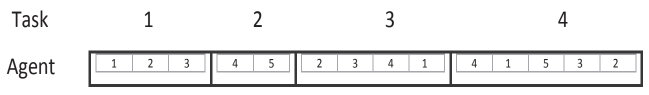

Fig. 1. Chromosome representation of a candidate solution

The steps of NSGA-II are summarized as:

- Generate initial population P_0 which consists of N non-duplicate candidate solutions (schedules). Each task is assigned only to those personnel, which are capable of performing that task.
- Evaluate each candidate solution according to the given fitness functions (5), (6) and (7).
- For $t = 0$ to $M - 1$ (M is the maximum number of generations) do:
  - Generate N non-duplicate offspring (population Q_t) using the following three procedures.

    Selection: Binary tournament selection is used to select parent solutions for reproduction. In binary tournament selection, two individuals are chosen uniformly at random from given pool P_t and the individual with higher fitness is allocated for reproduction trial. Two binary tournaments are conducted to select two parent solutions.

    Crossover: One-point crossover is used to combine two parents to produce an offspring. The crossover point p is chosen uniformly at random such that $0 < p < n$. The first p assignments are taken from parent A and the remaining $n - p$ are copied from parent B such that the constraints are not violated. The duplicate assignments within same task are avoided and no agent is assigned to more than one task at the same time. The corresponding genes where assignments violate any of the constraints are marked with * which need to be repaired. In the repair procedure each of the marked gene is assigned an agent from the set of feasible (capable) agents for that particular gene.

    Mutation: Each gene is selected randomly with probability P_m and it is assigned some other agent from the set of its feasible agents such that it does not violate any of the constraints.
  - Evaluate all generated offspring using the proposed fitness functions (5), (6) and (7).
  - Build the union U_t of parent P_t and offspring Q_t ($U_t = P_t \cup Q_t$).
  - Rank all the solutions in U_t according to Pareto-dominance relation. In the ranking procedure, first step is to find out all non-dominated solutions and rank them as level 1. These solutions form a Pareto-front F_1 with rank 1. Exclude these non-dominated solutions from U_t and find all non-dominated solutions from the rest of the population. That provides a Pareto-front F_2 with rank 2. Keep on doing the same procedure until you have ranked all the solutions into s Fronts.
  - In order to select N candidate solutions for next generation $(t + 1)$, get the sorted fronts ($F = \{F_i | i = 1, \ldots, s\}$) in sequence one by one until the population P_{t+1} is filled with N non-dominated solutions. While processing each front F_i, if the number of candidate solutions is less than number of remaining solutions R of population P_{t+1} ($R = N - C$, where C is current size of P_{t+1}), copy all candidate solutions from F_i to $P_t + 1$. Otherwise, sort the candidate solutions of front F_i in descending order based on crowding distance, iterate in sequence and fill the population P_{t+1} with remaining R solutions.

– Return front F_1, which is an approximated Pareto-front found by the algorithm.

5 Experiments

In this section, we present a set of experiments and their results. The aim of the experiments is to check the accuracy and quality of the results obtained by applying NSGA-II to the proposed rescheduling problem. First, we describe the problem instances, and the parametrization of the algorithm. After that we present and discuss the experimental results of NSGA-II.

We modify the implementation of the presented algorithm which is based on Java based framework jMetal [2]. The algorithm is run on a PC with an Intel Core i5 - 2.60 GHz and 4.0 GB of RAM.

5.1 Problem Instances and Data

In order to discuss the accuracy and quality of the results, we consider two instances of the proposed problem with different sizes as shown in Table 1.

Table 1. Specification of the Problem Instances with Initial Schedule

Prob#	Total Agents	Number of Courses	Number of other Tasks	Groups (# of agents assigned to the tasks)
1	50	1	5	[20, 25, 25, **45**, 40, 40]
2	400	3	63	[100, 50, 50, 40, 30, 45, 40, 35, 70, 45, 40, 100, **50**, 50, 40, 30, 45, 40, 35, 70, 45, 38, 100, 50, 50, 40, 30, 45, 40, 35, 70, 45, 40, 100, **50**, 50, 40, 30, 45, 40, 35, 70, 45, 39, 100, 50, 50, 40, 30, 45, 40, 35, 70, 45, 40, 100, **50**, 50, 40, 30, 45, 40, 35, 70, 45, 40]

Figures in bold are courses.

In all problems, we consider 10 different types of competencies. The data for these competencies is integer values between 0 and 4 inclusive and thus we have 5 levels (0, 1, 2, 3 and 4). We generate different number of profiles of agents for each defined problem. The agents belonging to the same profile have same competencies. For instance, in military troops, one type of profile can be tank drivers. Considering problem number 1 where we have total 50 agents, we generate 10 different types of agent profiles requiring 10, 10, 5, 5, 5, 4, 4, 3, 2 and 2 agents respectively. In problem number 2, we have total 400 agents, we generate 12 different types of agent profiles requiring 10, 10, 20, 20, 30, 30, 40, 40, 50, 50, 50 and 50 agents respectively. For each profile, we select 2, 3, 4 or 5 competencies randomly with a uniform distribution and assign competency levels other than 0 with a given probability distribution. The assumed probabilities for competency levels 1, 2, 3 and 4 are 50%, 25%, 15% and 10% respectively. It means that we have less number of agents, which have higher level of competencies.

The tasks may require a combination of agents from different profiles. Some tasks may even require agents with higher competencies than available. In that case agents may need special courses, which can enhance their competencies.

Initial Schedule data (task start date, duration, priority) for courses and other tasks is generated randomly. In order to verify the working accuracy of the algorithm, we manually create initial schedule for problem number 1 as shown in Table 2.

Table 2. Initial Schedule data for problem number 1

Task#	Day of the year	Task start date	Task duration	Priority	# of agents required	Is course?
1	1	2013-1-1	50	High	20	No
2	2	2013-1-2	25	Low	25	No
3	27	2013-1-27	24	Low	25	No
4	51	2013-2-20	60	Low	45	Yes
5	111	2013-4-21	60	Low	40	No
6	171	2013-6-20	30	Low	40	No

The data for these capabilities and their weights are integer values between 0 and 4 inclusive.

5.2 Parametrization

For experiments, the initial population size is set to 50 and the algorithm terminates after $100,000$ evaluations. For problem number 2 given in Table 1, we terminate after $10,000$ evaluations. The probability for crossover is 95% and probability of mutation for each gene is 20%.

5.3 Computational Results

In this section, we present the computational results of the proposed rescheduling problem which are obtained using NSGA-II. In order to test the accuracy of the results, we consider a small problem with 1 course, 5 tasks and 50 agents (Problem number 1 in Table 1). Task 1 needs 20 agents, task 2 and 3 need 25 agents each, task 4 which is a course requires 45 agents and task 5 and 6 need to be assigned 40 agents each. The data for the initial schedule (see Table 2) is designed in such a way that at any instance of time at least 80% of the agents are busy in tasks or courses. Furthermore, the requirements for task number 6 are hard and those agents can be assigned to this task which have enhanced their competencies by taking the available course (task number 4). Given the feasible and complete initial schedule, assume that on 10th of January a new high priority task is assigned to the battalion by higher headquarters. This requires rescheduling of the assignments of agents to tasks considering given objectives and constraints. The schedule data for the new task is given in Table 3.

We see that task number 6 and 7 are in parallel to each other, it means the agents, which are assigned to one of task, cannot be assigned to the other. In the

Table 3. Schedule data for a new task introduced on 2013-01-10 for problem # 1

Task#	Day of the year	Task start date	Task duration	Priority	# of agents required	Is course?
7	172	2013-6-21	30	High	10	No

initial schedule, we already have assigned 40 agents to task number 6 and the remaining 10 agents are free. The requirements for the new task (task number 7) are set such that those 10 free agents are not qualified to perform the task. We need to reshuffle the assignments in our initial schedule. The requirements of task number 7 are such that some of the agents need to be reassigned from task number 6 to 7. According to the generated data, the other possible solution can be that we miss the last task. By executing the algorithm, we get the Pareto-optimal front as shown in Table 4. We can see that both types of solutions are found by the algorithm and there can be multiple different Pareto-optimal solutions with same objective values.

Table 4. Pareto-optimal front (for Prob# 1 with 1 new task)

Soln.#	Total number of tasks performed	Assignment Differences from initial schedule	Number of high priority missed tasks
1	6	0	1
2	6	0	1
3	7	7	0
4	7	7	0
5	7	7	0
6	7	7	0
7	7	7	0

Furthermore, to make the problem more difficult, we introduce some new tasks in parallel to the initial set of tasks. The schedule data for 6 new tasks is given below in Table 5. Task number 7 is parallel to task number 6 as before. Moreover, tasks number 8, 9, 10, 11, 12 and 5 are parallel to each other. By executing the algorithm, we get the Pareto-optimal front as shown in Table 6.

Now we consider problem 2, which is a realistic problem with 400 agents, 3 courses and 63 tasks. The requirements and schedule information (task start date, duration, priority) for all the tasks and courses are generated randomly. Given the feasible and complete initial schedule, assume that on 25th of January, 10 new tasks come in, of which 4 of them are high priority tasks. We need to

Table 5. Schedule data for 6 new tasks introduced on 2013-01-10 for problem # 1

Task#	Day of the year	Task start date	Task duration	Priority	# of agents required	Is course?
7	172	2013-6-21	30	High	10	No
8	112	2013-4-22	28	High	10	No
9	115	2013-4-25	56	High	10	No
10	120	2013-4-30	21	High	10	No
11	127	2013-5-07	35	High	10	No
12	129	2013-5-09	14	High	10	No

Table 6. Pareto-optimal front (for Prob# 1 with 6 new tasks)

Soln.#	Total number of tasks performed	Assignment Differences from initial schedule	Number of high priority missed tasks
1	6	0	6
2	7	7	5
3	8	10	4
4	11	47	0
5	11	47	0
6	10	40	1
7	8	10	4
8	11	47	0
9	6	0	6
10	8	10	4
11	10	40	1
12	10	40	1

reschedule the assignments of agents to tasks in such a way that we take care of three given objectives. The schedule data for the new tasks is generated randomly and number of required agents for task number 67 to 76 are 40, 30, 50, 20, 35, 20, 25, 30, 20, and 25 respectively. By executing the algorithm, we get the Pareto-optimal front as shown in Table 7. There can be multiple solutions with same objective values but we have shown only distinct ones. The results show that performing all high priority tasks is in a sharp contrast with preserving the initial schedule and both objectives cannot simultaneously be optimized. It is up to the decision-maker to select the desired solution form the Pareto-optimal front according to her/his preferences.

Table 7. Pareto-optimal front (for Prob# 2 with 10 new tasks)

Soln.#	Total number of tasks performed	Assignment Differences from initial schedule	Number of high priority missed tasks
1	67	0	4
2	68	505	1
3	67	593	0
4	65	467	1
5	67	297	3
6	68	962	0

6 Conclusion and Summary

In this paper, we propose a novel model for rescheduling different activities of the battalions personnel. We formulate the rescheduling problem as a multi-objective optimization problem, in which we consider three objectives: (i) maximizing the total number of performed tasks, (ii) minimizing the number of high priority tasks that are missed, and (iii) minimizing the differences between the initial schedule and updated schedule.

Firstly, a multi-objective mathematical model, with three conflicting objectives and a set of constraints is built. Multi Objective Evolutionary Algorithms (MOEAs) are well-known metaheuristics for sampling intractably large and highly complex search spaces. In order to solve the optimization model, we adopt a well-known multi-objective genetic algorithm NSGA-II.

We present the computational results of the proposed rescheduling problem, which are obtained using NSGA-II. We verify the accuracy of the algorithm in this context by considering a small problem with easy to verify solutions. Furthermore, we consider a realistic problem instance for a battalion with 400 agents and 66 tasks in the initial schedule. We present the computational results of rescheduling when 10 new tasks come in. The experimental results show that NSGA-II efficiently provides the Pareto-optimal solutions for the proposed rescheduling problem. From the obtained Pareto-optimal front, the decision maker can choose one or more solutions according to her/his preferences.

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
2. Durillo, J.J., Nebro, A.J.: jMetal: A Java framework for multi-objective optimization. Advances in Engineering Software 42, 760–771 (2011)
3. Clark, A.R., Walker, H.: Nurse rescheduling with shift preferences and minimal disruption. Journal of Applied Operational Research 3(3), 148–162 (2011)
4. Moz, M., Pato, M.V.: A genetic algorithm approach to a nurse rerostering problem. Computers and Operations Research 34(3), 667–691 (2007)
5. Maenhout, B., Vanhoucke, M.: An evolutionary approach for the nurse rerostering problem. Computers and Operations Research 38(10), 1400–1411 (2011)
6. Chicano, F., Luna, F., Nebro, A.J., Alba, E.: Using multi-objective metaheuristics to solve the software project scheduling problem. In: GECCO, pp. 1915–1922. ACM (2011)
7. Alba, A., Chicano, J.F.: Software project management with GAs. Information Sciences 177, 2380–2401 (2007)
8. Hao, X., Lin, L.: Job shop rescheduling by using multi-objective genetic algorithm. In: 40th International Conference on Computers and Industrial Engineering (CIE), pp. 1–6. IEEE (2010)
9. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley, Massachusetts (1989)
10. Mitchell, M.: Introduction to genetic algorithms. MIT Press, Massachusetts (1999)
11. Younas, I., Kamrani, F., Schulte, C., Ayani, R.: Optimization of Task Assignment to Collaborating Agents. In: IEEE Symposium on Computational Intelligence in Scheduling, pp. 17–24. IEEE (2011)
12. Gonçalves, J.F., Mendes, J.J.M., Resende, M.G.C.: A genetic algorithm for the resource constrained multi-project scheduling problem. European Journal of Operational Research 189(3), 1171–1190 (2008)