Methodology

JDMS

Journal of Defense Modeling and Simulation: Applications, Methodology, Technology
1–8

DOI: 10.1177/1548512917711310
journals.sagepub.com/home/dms
SAGE

Using a genetic algorithm to solve the troops-to-tasks problem in military operations planning

Maria Fleischer Fauske

Abstract

The troops-to-tasks analysis in military operational planning is the process where the military staff investigates who should do what, where, and when in the operation. In this paper, we describe a genetic algorithm for solving troops-to-tasks problems, which are typically solved manually. The study was motivated by a request from Norwegian military staff, who acknowledged the potential for solving the troops-to-tasks analysis more effectively by using optimization techniques. Also, NATO's operational planning tool, TOPFAS, lacks an optimization module for the troops-to-tasks analysis. The troops-to-tasks problem generalizes the well-known resource-constrained project scheduling problem, and thus it is very difficult to solve. As the troops-to-tasks problem is particularly complex, the main purpose of our study was to develop an algorithm capable of solving real-sized problem instances. We developed a genetic algorithm with new features, which were crucial to finding good solutions. We tested the algorithm on two different data sets representing high-intensity military operations. We compared the performance of the algorithm to that of a mixed integer linear program solved by CPLEX. In contrast to CPLEX, the algorithm found feasible solutions within an acceptable time frame for all instances.

Keywords

Scheduling, genetic algorithms, military applications

1. Introduction

The troops-to-tasks analysis in military operational planning is the process where the military staff investigates who should do what, where, and when in the operation. A large set of tasks must be performed, often with limited resources. This analysis is done manually by the military staff. In the study we describe in this paper, our goal was to develop a tool to automate the troops-to-tasks analysis. The study was motivated by a request from Norwegian military staff, who acknowledged the potential for solving the troops-to-tasks analysis more effectively by using optimization techniques. NATO's operational planning tool, TOPFAS, lacks an optimization module for the troops-to-tasks analysis.[1] In addition to being an essential part of operational planning, troops-to-tasks analysis is also an important part of long-term defense planning, where the purpose is to investigate how a force structure should look in the future.[2] In this paper, we describe a genetic algorithm (GA) that can be used to solve real-life troops-to-tasks problems.

The troops-to-tasks analysis is a project scheduling problem closely related to the resource-constrained project scheduling problem (RCPSP).[3–6] There are many types of operations with different objectives, ranging from peace support operations where the objective might be to do as many activities as possible within a given time frame, to high-intensity operations where there is a given set of tasks, and the objective is to minimize the makespan.

There have been many efforts to incorporate methods from project scheduling and general planning into different parts of military operational planning.[7–13] The troops-to-tasks problem is particularly complex, due to aspects of

Norwegian Defence Research Establishment, Norway

Corresponding author:
Maria Fleischer Fauske, Norwegian Defence Research Establishment, P.O. Box 25, Kjeller, NO-2027, Norway.
Email: maria.fauske@ffi.no

the problem such as resource hierarchy, locations, and resources skills and capacities, as we will describe below.

The troops-to-tasks problem resembles the multi-skill project scheduling problem (MSPSP) proposed by Néron and Baptista,[14] which is an extension of the RCPSP. In the MSPSP the resources are, for example, staff members who are able to perform more than one kind of activity. That is, they possess more than one skill. The activities have different skill requirements, instead of specific resource requirements, as is the case in the RCPSP. Many skills may be necessary to process one activity. Staff members can only process one activity at a time. The model allocates resources and starting times to each activity, and the objective is to minimize the makespan. The troops-to-tasks problem is a MSPSP with additional characteristics and constraints, where the troops are like staff members with different skills. Li and Womer[15] also studied problems with multi-skilled personnel, or multi-purpose resources. They studied both minimization of the total number of resources[9] and the minimization of project costs.[16] The MSPSP is a variant of the more studied multi-mode RCPSP (MRCPSP), with a very large set of modes for each activity.

The basic RCPSP was proven to be non-deterministic polynomial-time hard (NP hard) by Blazewics et al.,[17] which means that when problem instances grow large, there is little hope of solving them to optimality within a reasonable time. For this reason, the RCPSP and related problems are usually solved with heuristic algorithms, for example, the tabu search algorithm and GAs. We find many references to different solution methods in the literature. Kolisch and Hartmann[18] give an overview of many of them, as do Peteghem and Vanhoucke.[19]

Since the MSPSP generalizes the RCPSP, it follows that the MSPSP is also NP hard. This leads us to believe that in troops-to-tasks problems, it will also often be difficult or impossible to find solutions to larger problem instances that can be proven optimal. However, due to the constraints and the characteristics of some troops-to-tasks problems, there are real-life problem instances that are solvable to optimality within a time frame that is acceptable in practice, as shown by Fauske.[20] However, to find good solutions to any troops-to-tasks problem instance, and especially the largest and most complex ones, we are in need of an heuristic solution method. In this paper we describe a GA for solving the troops-to-tasks problem in a high-intensity operation. Such operations are offensive, with a specific use of military forces, with the purpose of changing the course of events. On the other end of the scale, low-intensity operations may involve non-military tasks, such as policing and building local institutions. The solution method can easily be adapted to different types of operations. Activity list-based GAs for solving project scheduling problems have many times been proved effective.[21,22] Since the use of the GA for project-scheduling problems is very well documented, we chose this solution method for our problem. In this paper we show how such an algorithm performs on real-sized, complex problem instances, which to our knowledge have not been solved in this manner before.

In this paper we consider high-intensity operations. Our approach could easily be modified to other types of operations, since the structure of the problem is the same for many operations. We compared our GA to a mixed linear programming model solved by CPLEX.[23] In contrast to CPLEX, the GA found feasible solutions within an acceptable time frame for every instance in our two case sets. Whenever CPLEX found the optimal solution, so did the algorithm. In several instances, the algorithm found better solutions than CPLEX. This proves that our GA is a promising approach for solving the troops-to-tasks analysis.

2. Problem description

2.1 General considerations

The problem we solve in this paper is described in detail by Fauske,[20] which also includes the mathematical formulation of the problem. The problem represents a high-intensity land operation at brigade level with supporting resources from the Air Force. A land operation is run and conducted by the Army. The implications of this are that some of the resources belong to a strict hierarchy, since in the Army a person is a part of a squad, which in turn is a part of a platoon, etc., while in the Air Force the platforms (aircraft, helicopters, etc.) do not belong to such a strict hierarchy, and thus they operate on their own. The hierarchy puts restrictions on where the different resources may be at a certain time. When planning a military operation, you would not consider all levels of the hierarchy. A commander will mainly make his plans in cooperation with the commanders of the underlying troops, that is, one level below.

The resources can process more than one activity at a time if those activities can be processed in the same location, and if none of these activities require the full attention of the resource. We model this as exclusive and non-exclusive activities. If an activity is exclusive, the resources processing it can *only* process this activity at a certain point in time. Examples of non-exclusive activities may be maintenance, preparations, or planning activities. Exclusive activities are typically assaults and attacking activities.

The resources also have a capacity for each skill. Two resources possessing the same skill might have different capacities for that skill. For example, one type of aircraft may be able to solve a task using one aircraft, while another type may need two aircraft to process the same task. A surveillance activity may, for example, require a

quantity of 0.8 of a specific surveillance skill. That means that this activity can be processed equivalently by, for example, two resources that each has a capacity of 0.4 of this skill or by one resource that has a capacity of at least 0.8 of this skill. If a resource's capacity is larger than what is required, that does not mean that it has ''left over'' capacity to use on another task at the same time. It is the exclusiveness of the activities that decides whether a resource can do more than one task at a time. A resource is situated in a location and it takes time to move between locations. Location problems have been studied in several papers,[24,25] but then mostly without the resources traveling between the locations where they are situated. Traveling time between locations may be modeled as a sort of set-up time.[25–28] We model it by saying that an activity can not be started until all resources that shall process it have had the time to travel to the activity's location.

There may be precedence relations between some of the activities. If the operation is divided into phases, as is common, we can use precedence relations to model this. We make a certain activity mark the end of a phase, and we set all activities belonging to that phase as predecessors to that activity.

Some of the activities may have specific release dates (first allowed starting time) or deadlines (last allowed starting time). Several papers consider release dates and deadlines.[29–33]

This leads us to defining the following problem: a set $\mathcal{N}$ of activities, numbered 1 to N, is to be scheduled. Activity N is a dummy end activity that represents the end of the project. There is a set $\mathcal{L}$ of locations, and each activity i must be processed in a location $l_i \in \mathcal{L}$. There exists a set $\mathcal{R}$ of renewable resources. Each resource r belongs to some resource unit u. Resources belonging to the same unit must always be in the same location, but they do not have to process the same tasks. There is a traveling time t_{rlm} for resource r between locations l and m. There is a set of skills $\mathcal{K}$, and each resource possesses a capacity of $c_{kr} \geq 0$ for each skill $k \in \mathcal{K}$. An activity i requires an amount of $c'_{ki} \geq 0$ of each skill $k \in \mathcal{K}$. The duration of each activity is denoted d_i. Every activity has a release date e_i and a deadline f_i. An activity i has a set of predecessors $\mathcal{J}_i \subset \mathcal{N}$, and the starting time of activity i must be larger than the starting time of all predecessors of i. There is a set $\mathcal{N}_e \subseteq \mathcal{N}$ of exclusive activities. While processing an exclusive activity, the resource can not process other activities. The problem consists in allocating a number of resources to each activity i such that c'_{ki} is satisfied for all k, and finding a starting time s_i for every activity, so that the makespan $m = s_N$ is minimized.

2.2 Genetic algorithm approach

Our goal is to assign resources and starting times to activities in such a way that the total duration of the operation, that is, the makespan of the project, is minimized. To this end, we developed a GA. GAs were first introduced by Holland.[34] Inspired by evolutionary biology, GAs start with an initial population of solutions (individuals), and then seek to find better solutions by repeatedly modifying the population. At each step, the population is divided into pairs of parents to produce children through the *crossover* operator. The *mutation* operator is used to introduce variability to the population. The selection procedure evaluates the children to decide which individuals will survive in the population. Over successive generations, the population evolves toward better solutions. In the following sections we describe the different parts of our GA. We start by describing the representation of the individuals, before we reveal the details about the GA operators.

2.3 Representation of the individuals

In Hartmann,[21] different ways of representing the individuals in GAs for the RCPSP are examined. They conclude that the activity list-based GA performs best. There, individuals are represented by precedence feasible activity lists. Precedence feasible means that for each activity i in the list, every predecessor of i has to be before i in the list. Hartmann and Kolisch[22] concluded from experimental tests that activity list-based procedures outperformed other procedures. In our case, an individual consists of a precedence feasible activity list, and a corresponding list of lists, containing the resource assignment of the activities. All individuals in the population are feasible.

2.4 Initial population

The individuals in the initial population $\mathcal{POP}$ are generated through a two-phase procedure. In the first phase we generate an ordered list of activities, and in the second phase we assign resources to the activities. The activity list is generated through a procedure based on the serial schedule generation scheme (SGS), as explained by Kelley,[35] Kolisch,[36] and Hartmann,[21] which produces a precedence feasible activity list. The procedure starts by determining the set $\mathcal{S}$ of currently selected activities, which in the beginning is empty. Then the set $\mathcal{E}$ of eligible activities is determined, that is, those activities that do not have any predecessors, or whose predecessors are already in $\mathcal{S}$. From the set $\mathcal{E}$, one activity is picked randomly and put in $\mathcal{S}$. Then the set $\mathcal{E}$ is redetermined. This process is repeated until all activities are selected. The dummy end activity represents the end of the project. All other activities are set as predecessor activities of the end activity. After the generation of the activity list, resources are assigned to each of the activities. For each skill requirement of each activity, resources are drawn randomly until the skill quantity requirement is satisfied. The algorithm assumes that

the skill quantity requirement can be covered by the available resources. Based on the activity list and the resource assignment, the earliest possible starting times of each activity are determined through a scheduling procedure. The starting time of each activity will depend on the activity's precedence relations, release date, deadline, location, exclusiveness, the assigned resources' traveling times, and the resource hierarchy (since resources in the same resource unit must be in the same location). Since these calcluations have to be repeated for each individual in each generation, it inevitably has a large impact on the run-time of the algorithm.

2.5 Crossover

The crossover operator is used to add new solutions to the population. We chose to use a two-point crossover operator. The population $\mathcal{POP}$ is divided into pairs of parents (a mother and a father). For some probability p_{cross}, the pair is chosen for crossover to make two children that are added to the set of children $\mathcal{CHI}$. If the pair is not chosen, the parents are added to $\mathcal{CHI}$ unchanged. The two-point crossover procedure works as follows.

Two random integers $q1$ and $q2$ are drawn, with $1 \leq q1 < q2 \leq N$, where N is the number of activities. The first offspring is determined in the following way: the first $q1$ activities and its respective resources are taken from the mother. The next $q2 - q1$ are taken from the father. If an activity from the father has already been taken from the mother, this activity is skipped, and the next one is picked instead. Finally, the remaining activities are taken from the mother. The second offspring is determined in the same manner, but the first $q1$ activities are chosen from the father instead of the mother, the next $q2 - q1$ from the mother, and so on.

When the two-point crossover scheme is applied to precedence feasible activity lists, the offspring will also be precedence feasible. This is proven by Hartmann.[21] Both the one-point crossover, used by, for example, Elloumi and Fortemps,[37] Peteghem and Vanhoucke,[19] and Okada et al.,[38] and the two-point crossover, as described by Hartmann,[21] Alcaraz et al.,[39] and Lova et al.,[40] are commonly used in the literature. Both may perform well.

2.6 Resource mutation

As several resources may possess the skills necessary to perform a certain task, we introduce resource mutation as a way of adding variability into the population. With a probability of p_{res}, an individual in $\mathcal{CHI}$ is chosen for resource mutation. If chosen, one of the activities in the individual is drawn randomly, and new resources are assigned to this activity. Another common way to model mutation is to decide for every activity of the selected individual whether there should be a mutation. This type of mutation is done by, for example, Lova et al.[40] and Elloumi and Fortemps.[37]

2.7 Activity mutation

With a probability of p_{act}, an individual in $\mathcal{CHI}$ is chosen for activity mutation. If chosen, a random activity is picked and swapped with the activity to the left if this does not violate the precedence relations between the activities.

2.8 Location-based adaptation

In our problem, the resources travel between locations. If it is possible to do several activities in one location at the same time or in sequence before traveling to another location, this might contribute to a shorter duration of the total project. We found that the algorithm consequently performs better if we add a procedure that tries to place activities close to each other in the activity list if these activities are to be processed in the same location. To avoid convergence, we chose not to do this in every generation. Instead, we applied the location-based adaptation in every g_{adapt} generation. The location-based adaptation worsk as follows: starting with the leftmost activity in the activity list, for every activity i, activity $j \neq i$ in the list that is processed in the same location as activity i and that has no precedence relations, is moved to the spot right next to activity i in the activity list.

2.9 Fitness computation

After crossover, mutation and location-based adaptation, the earliest possible activity starting times are calculated for the individuals in $\mathcal{CHI}$. The objective of our model is to minimize the makespan of the project. Therefore, the fitness of the individual is equal to the individual's makespan. However, we add a small penalty for starting activities later than their deadlines. Instead of having hard deadline constraints, we used the deadline approach to provide diversity, that is, to avoid convergence in the population, which is important in the GA. Through experiments, we found that adding a fixed penalty of two time steps for each overdue activity was sufficient to provide good results. This means that the total fitness is the sum of the makespan and the penalties for not making the deadlines. Thus, the algorithm will try to avoid solutions that do not meet the deadlines.

2.10 Selection

The last step to obtain the new generation is to select the best $|\mathcal{POP}|$ individuals from $\mathcal{POP} \cup \mathcal{CHI}$. There are several ways to do this, but in our algorithm we keep the best $|\mathcal{POP}|$ individuals and discard the rest.

2.11 Computational experiments

The purpose of the computational experiments was to investigate the performance of the algorithm on problem instances of different sizes, and to compare the performance with that of the mathematical model given by Fauske,[20] which we implemented in the commercial solver CPLEX. To this end, we generated two different realistic data sets based on the principles described below.

2.12 Data set generation

In both data sets we used the same force structure for all instances. This structure is described below. Also, all instances are based on the type of operation we described earlier in this paper. In Case set 1 we generated 20 random, but realistic, instances. These instances represent an operation with 25 activities, where the last activity is a dummy end activity. As an operation using the given force structure might include more activities, we also generated a second case set. In Case set 2 set we generated 10 instances, each representing an operation with 35 activities, where the last activity is a dummy end activity. The activity durations, activity skill requirements, the location for each activity, and precedence relations were generated randomly for every instance.

Our force structure consists of four battalions (resource units), each with three underlying companies. In addition, there are six aircraft available. We assume that supporting resources (e.g., medical, logistics, and Command and Control Information System) have dedicated tasks, and are therefore not included in the scheduling. Each company has five skills, and the types of skills they possess vary among them. Each aircraft has two different skills. There are 12 locations. All army units have the same traveling time between locations, while the aircraft have negligible traveling times. The operation has two phases in all instances in Case set 1, and three phases in all instances in Case set 2. Activity durations vary between one and three time steps, and each activity requires between one and three skills. The chance of an activity being assigned a precedence activity was 0.3. With a probability of 0.6, an activity was set to be an exclusive activity.

We implemented the GA in MATLAB, on a computer with a 2.90 GHz Intel(R) Core(TM) i7-3520M CPU, running the 64-bit Windows 7 operating system. In addition to solving each instance with the GA, we tried to solve them mathematically using CPLEX. We set the time limit in CPLEX to 1 hour, as run-times longer than this would soon become impractical in a real planning situation.

2.13 Configuration of the algorithm

Finding a good configuration of the GA means choosing good values for the parameters $|\mathcal{POP}|$, p_{cross}, p_{res}, p_{act}, and g_{adapt}. After investigating the cases we generated, we saw that the optimal parameter value configuration is somewhat case dependent, and we conclude that in a real-life planning context some testing must be done for the best configuration for the specific problem. In this paper, however, we use the same parameter configuration for all instances in a case set. Table 1 shows this configuration. We did experience that increasing the number of individuals also increased the number of optimal values found, but it also increased the run-times significantly. The choice of parameter values must balance the need for good solutions and the need for shorter run-times. We found that using 1200 individuals was a good choice for Case set 1, and 2000 was a good choice for Case set 2. We stopped the algorithm after 150 generations, because we did not see any improvements in solutions after such a number of generations.

Table 1. Best parameter value configuration for the genetic algorithm.

Parameter	Value
POP (small case set)	1200
POP (large case set)	2000
p_{cross}	1.0
p_{act}	0.6
p_{res}	0.8
g_{adapt}	3

3. Results

3.1 Case set 1

CPLEX found proven optimal values in nine of the 20 cases in this case set. In the remaining 11 cases, feasible solutions were found, but they were not proven optimal. For most cases, the GA did not find the optimal value every time we ran the case. Therefore, we ran each case 10 times, and we recorded how many times the optimal value was found. Table 2 shows the results of all of these runs. In 19 of the 20 cases, CPLEX and the GA found the same best value. In the remaining case (case 9), the GA found a better solution than CPLEX. On average, the GA found the best solution in 7.35 of 10 runs, after 5.6 minutes. This involved calculating 80, 464 individuals, over 66.7 generations, on average. This indicates that the stopping criteria of 150 generations will, on average, be quite large.

In a few of the cases, CPLEX outperformed the GA by finding proven optimal solutions after a short amount of time. However, the GA is a safer choice, as its performance is more stable. This is especially true if we look at the results from Case set 2.

Table 2. Results from solving Case set 1 with the genetic algorithm in MATLAB and with a mathematical model in CPLEX. Run-time values are given in minutes.

	CPLEX			Genetic algorithm				
Case	Run-time	Lower bound	Obj. value	Best fitness	# best fitness	Mean # gen.	Mean run-time	Mean # ind.
8	1.7	13	13	13	5	67.8	5.8	81,360
2	2.5	17	17	17	5	75.6	6.5	90,720
20	6.0	16	16	16	1	96.0	7.6	115,200
12	8.4	18	18	18	7	58.1	5.6	69,771
1	15.2	16	16	16	3	63.3	5.2	83,600
4	20.3	17	17	17	6	66.0	5.9	79,200
11	22.7	20	20	20	9	63.6	5.2	76,267
15	29.4	13	13	13	4	71.5	5.7	85,800
13	51.6	19	19	19	1	53.0	4.0	63,600
3	60.0	15	18	18	10	55.7	4.9	66,840
5	60.0	14	16	16	10	43.8	3.8	52,560
6	60.0	14	20	20	10	94.0	7.5	112,800
7	60.0	16	19	19	1	76.0	6.0	91,200
9	60.0	13	25	19	7	73.7	6.2	88,457
10	60.0	21	22	22	3	82.0	7.1	98,400
14	60.0	17	19	19	6	67.3	5.6	80,800
16	60.0	18	20	20	5	73.4	6.2	88,080
17	60.0	15	19	19	10	44.7	3.8	53,640
18	60.0	16	18	18	5	51.6	4.6	61,920
19	60.0	17	18	18	9	57.6	5.1	69,069
Mean	40.9				7.35	66.7	5.6	80,464

Table 3. Results from solving Case set 2 with the genetic algorithm in MATLAB and with a mathematical model in CPLEX.

CPLEX			Genetic algorithm			
Case	Lower bound	Obj. value	Fitness	# gen	Run-time	# ind
4	23	24	24	103	23.0	206,000
7	23	23	23	46	9.2	92,000
6	22	29	29	48	9.5	96,000
2	31	34	32	71	15.3	142,000
10	26	40	29	48	9.9	96,000
3	23	29	28	55	11.1	110,000
1		x	25	75	10.1	90,000
5		x	27	99	20.3	198,000
8		x	26	108	23.0	216,000
9		x	26	72	15.0	144,000

3.2 Case set 2

In a real planning situation, we assume that it would, in many cases, be impractical to run the GA many times. For Case set 1 we chose to run the GA several times to illustrate the number of times it could find the optimal solution. For Case set 2, we only run each case once, to simulate a real planning situation. The results from Case set 1 showed us that the GA will probably not find the optimal solution every time we run a case. Therefore, we do not expect to necessarily find the optimal solution when running the GA only once. Our main interest in Case set 2 is therefore to check if the GA outperforms the mathematical model implemented in CPLEX, when run only once.

The results from running Case set 2 are shown in Table 3. In this case set, there was only one case (case 7) where CPLEX found a proven optimal solution within the time limit of 1 hour. In four of the cases, CPLEX did not find feasible solutions. Of the six cases where CPLEX found feasible solutions, the GA found better solutions in three. In the other three cases, CPLEX and the GA found equally good solutions. The run-times for the GA varied between 9.2 and 23.0 minutes, which is quite acceptable in a real planning context.

The results from Case sets 1 and 2 show us that the GA we developed is suitable for solving the complex troops-to-tasks problem we defined in this paper. It finds good solutions to the problem within a time frame that is acceptable in a real planning context. However, in each planning situation, some effort must be made to find a good parameter configuration of the algorithm, which will add to the time comsumption. The main advantage of the GA compared to a mathematical approach will be the ability to find feasible solutions for larger problem instances.

The performance of the GA is stable and, with only a few exceptions, it outperforms the mathematical model in the cases we developed for this paper.

4. Conclusions

In this paper, we designed a GA for solving the troops-to-tasks problem in a high-intensity military land operation. We compared the performance of the GA with that of a mixed integer linear programming model solved by a commercial solver. In contrast to CPLEX, the GA found feasible solutions within an acceptable time frame for every case in our two case sets. We introduced a location-based adaptation to the GA, which consequently made the algorithm perform better. Our study shows that the GA is suitable for solving troops-to-tasks problems in military operational planning, and in long-term defense planning. To our knowledge, there has not been any previous attempts to solve the troops-to-tasks problem by means of automatic tools. The two case sets we used in our computational experiments are available upon request to anyone who would wish to develop new algorithms for solving the problem we presented.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

1. Ménard C. Personal communication (e-mail). Contractor for NATO Communications and Information Agency, 2016.
2. Hennum AC and Glærum S. Norwegian long-term defence planning. Technical report, NATO RTO-TR-069-SAS-072, 2008.
3. Hartmann S and Briskorn D. A survey of variants and extensions of the resource-constrained project scheduling problem. *Eur J Oper Res* 2010; 207: 1–14.
4. Brucker P, Drexl A, Möhring R, et al. Resource-constrained project scheduling: Notation, classification, models, and methods. *Eur J Oper Res* 1999; 112: 3–41.
5. Özdamar L and Ulusoy G. A survey on the resource-constrained project scheduling problem. *IIE Trans* 1995; 27: 574–586.
6. Icmeli O, Erenguc SS and Zappe CJ. Project scheduling problems: a survey. *Int J Oper Prod Manag* 1993; 13: 80–91.
7. Willick K, Wesolkowski S and Mazurek M. Multiobjective evolutionary algorithm with risk minimization applied to a fleet mix problem. In: *WCCI 2010 IEEE world congress on computational intelligence (CCIB)*, Barcelona, Spain, 18–23 July 2010, pp. 1–7. Piscataway, NJ: IEEE.
8. Bui LT, Barlow M and Abbass HA. A multi-objective risk-based framework for mission capability planning. *New Math Nat Comput* 2009; 5: 459–485.
9. Li H and Womer K. A decomposition approach for shipboard manpower scheduling. *Mil Oper Res* 2009; 14: 1–24.
10. Abbas H, Bender A, Baker S, et al. Identifying the fleet-mix in a military setting. In: *second international intellingent logistics systems conference (IILS2006)*, Brisbane, 22–23 February 2006. Springer-Verlag.
11. Popken D and Cox L. A simulation-optimization approach to air warfare planning. *J Defence Model Simulat* 2004; 1: 127–140.
12. Schlabach JL, Goldberg DE and Hayes CC. FOX-GA: a genetic algorithm for generating and analyzing battlefield courses of action. *Evol Comput* 1999; 7: 45–68.
13. Wilkins DE and Desimone RV. Applying an AI planner to military operations planning. In: Zeweben M and Fox M (eds) *Intelligent scheduling*. San Mateo, CA: Morgan-Kaufmann Publishers, 1994, pp.685–708.
14. Néron E and Baptista D. Heuristics for the multi-skill project scheduling problem. In: *international symposium on combinatorial optimization (CO'2002)*, CNAM Institute, Paris, 8–10 April, 2002.
15. Li H and Womer K. Project-scheduling with multi-purpose resources: a combined milp/cp decomposition approach. *Int J Oper Quant Manag* 2006; 12: 305–325.
16. Li H and Womer K. Scheduling projects with multi-skilled personnel by a hybrid milp/cp benders decomposition algorithm. *J Schedul* 2009; 12: 281–298.
17. Blazewics J, Lenstra JK and Kan AHGR. Scheduling subject to resource constraints: classification and Complexity. *Discrete Appl Math* 1983; 5: 11–24.
18. Kolisch R and Hartmann S. Experimental investigation of heuristics for resource-constrained project scheduling: an update. *Eur J Oper Res* 2006; 174: 23–37.
19. Peteghem VV and Vanhoucke M. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *Eur J Oper Res* 2010; 201: 409–418.
20. Fauske MF. Optimizing the troops-to-tasks problem in military operations planning. *Mil Oper Res* 2015; 20: 49–57.
21. Hartmann S. *Project scheduling under limited resources*. Berlin: Springer, 1999.
22. Hartmann S and Kolisch R. Experimental evaluation of state-of-the-art heuristics for resource-constrained project scheduling problem. *Eur J Oper Res* 2000; 127: 394–407.
23. CPLEX Optimizer. Web site, https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/ (accessed 16 May 2017).

24. Krüger D and Scholl A. Managing and modelling general resource transfer in (multi) project scheduling. *OR Spektrum* 2010; 32: 369–394.
25. Mika M, Waligóra G and Weglarz J. Simulated annealing and tabu search for multi-mode resource-constrained project scheduling with positive discounted cash flows and different payment models. In: Józefowska J and Weglarz J (eds) *Perspectives in modern project scheduling*. Berlin: Springer, 2006, pp.131–165.
26. Mika M, Waligóra G and Weglarz J. Tabu search for multi-mode resource-constrained project scheduling with schedule dependent setup times. *Eur J Oper Res* 2008; 187: 1238–1250.
27. Schwindt C. *Resource allocation in project management*. Berlin: Springer, 2005.
28. Neumann K, Schwindt C and Zimmermann J. *Project scheduling with time windows and scarce resources*. Berlin: Springer, 2002.
29. Drezet LE and Billaut JC. A project-scheduling problem with labour constraints and time-dependent activities requirements. *Int J Prod Econ* 2008; 112: 217–225.
30. Kis T. RCPS with variable intensity activities and feeding precedence constraints. In: Józefowska J and Weglarz J (eds) *Perspectives in modern project scheduling*. Berlin: Springer, 2006, pp.105–129.
31. Kis T. A branch-and-cut algorithm for scheduling of projects with variable intensity activities. *Math Program* 2005; 103: 515–539.
32. Klein R. Project scheduling with time-varying resource constraints. *Int J Prod Res* 2000; 38: 3937–3952.
33. Klein R and Scholl A. Scattered branch and bound – an adaptive search strategy applied to resource-constrained project scheduling. *Central Eur J Oper Res* 2000; 7: 177–201.
34. Holland JH. *Adaptation in natural and artificial systems*. Ann Arbor, MI: University of Michigan Press, 1975. Second edition, 1992.
35. Kelley JE. The critical path method: Resources planning and scheduling. In: Muth JF and Thompson GL (eds) *Industrial scheduling*. Upper Saddle River, NJ: Prentice Hall, 1963, pp.347–365.
36. Kolisch R. Serial and parallel resource constrained project scheduling methods revisited: theory and computation. *Eur J Oper Res* 1996; 90: 320–333.
37. Elloumi S and Fortemps P. A hybrid rank-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem. *Eur J Oper Res* 2010; 205: 31–41.
38. Okada I, Takahashi K, Zhang W, et al. A genetic algorithm with local search using activity list characteristics for solving resource-constrained project scheduling problem with multiple modes. *IEEJ Trans Elecctr Electron Eng* 2014; 9: 190–199.
39. Alcaraz J, Maroto C and Ruiz R. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. *J Oper Res Soc* 2003; 54: 614–626.
40. Lova A, Tormos P, Cervantes M, et al. An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *Int J Prod Econ* 2009; 117: 302–316.

Author biography

Dr. Maria Fleischer Fauske is a senior scientist at the Norwegian Defence Research Establishment (FFI). Her area of expertise is defence planning. She is currently managing the operations research competency development within FFI.