A decomposition solution approach to the troops-to-tasks assignment in military peacekeeping operations

Journal Title
XX(X):1–11

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Nadia Chaudry[1], Ingunn Vermedal[1], Kjetil Fagerholt[1], Maria Fleischer Fauske[2] and Magnus Stålhane[1]

Abstract

This paper considers the Peacekeeping Troops-to-Tasks Problem (PTTP). The PTTP deals with assigning battlegroup resources to a set of tasks associated with a given peacekeeping mission. The tasks may be spread across several locations, and have requirements regarding the time at which they can be handled, and the skills and skill-levels needed to complete them. There is also a utility value related to each completed task that reflects its importance. The resources are bound by a hierarchy of command, limiting their movement in relation to one another. The aim is to decide which tasks to complete, when, and by whom. We present a mathematical compact model for the PTTP, which includes a number of complicating real-life factors. Due to the complexity of the compact model, it is difficult to solve large instances using a commercial solver. Therefore, we also propose a decomposition-based solution approach, with a decomposed model where possible travel routes for the resources are generated a priori. The computational study shows that the decomposed model has better performance than the compact model, and that it can be used as a good starting point for developing a useful decision support tool for military peacekeeping operations planning.

Keywords

Scheduling, military applications, optimization, mixed integer programming

Introduction

The purpose of peacekeeping missions is to maintain and defend stability and security. The United Nations (UN) currently have 14 international peacekeeping missions ongoing, served by more than 110 000 military, police, and civilian staff. Of these 14 missions, Norway is active in five: in Liberia, South-Sudan, Mali, Haiti, and a joint operation in Lebanon, Israel, Syria, and Egypt[1]. In addition, The North Atlantic Treaty Organization (NATO), of which Norway is a founding member, have several active peacekeeping missions on the African continent and in Eastern Europe[2].

The most critical resource in peacekeeping operations is human capital. Deciding how to best utilize this resource in military operations planning is often referred to as a troops-to-tasks analysis. In such an analysis, military staff investigate who should do what, where, when, and how in operations[3]. Typically, a troops-to-task problem is solved manually. Indeed, the troops-to-tasks analysis can benefit from using optimization techniques. Optimization could provide valuable decision support for operations, ensuring the optimality and feasibility of a planning schedule, in a reasonable amount of time. It could also be a tool for testing different alternatives by varying input data and conditions to see how such variations affect the operation plan[4]. At worst, it could provide a solution that operation planners can use as a starting point.

The specific problem scenario considered in this paper is defined as the Peacekeeping Troops-to-Tasks Problem (PTTP). To address this problem, we study a fictious military battlegroup that has to handle different tasks at various locations, during a fixed time period where there is no incentive to end ahead of schedule. Tasks have multiple conditions relating to the time at which they can be handled, and the amount, and type of skills needed to complete them. There is also a utility value related to each completed task that reflects its importance. As in real life, the battlegroup in our scenario consists of units with differing qualities and abilities that make them suited to handle certain types of tasks. They are bound by a hierarchy of command, limiting their movement in relation to one another. In the PTTP, the aim is to decide which tasks to complete, when, and by whom.

Research on the application of optimization in general military operation planning is scarce. This is despite the Norwegian Armed Forces and international organizations like NATO being interested in such an approach. The Norwegian Defence Research Establishment (FFI) has conducted some research on troops-to-tasks analysis of high intensity operations[3,4], but not of low intensity operations of which peacekeeping falls under.

The PTTP is modelled as an extension of the well studied Resource Constrained Project Scheduling Problem (RCPSP), where a set of tasks have to be completed by a set of resources. The objective of the RCPSP is to find a schedule that minimizes the duration of the

[1]Norwegian University of Science and Technology (NTNU), Norway
[2]Norwegian Defence Research Establishment (FFI), Norway

Corresponding author:
Maria Fleischer Fauske, FFI Postboks 25, 2027 Kjeller, Norway
Email: maria.fauske@ffi.no

project while observing precedence and resource constraints. Tasks require a certain amount of capacity from resource performing it, and resources have to be assigned to tasks such that the capacity requirements from the tasks do not exceed the resource capacity. Each task can only start once, and some tasks have precedence over others. The RCPSP is known to be NP-hard[5], making all of its extensions NP-hard as well. Well-known extensions of the RCPSP address more challenging problems, and include the multi-mode and multi-skill extensions. In the Multi-Mode Resource Constrained Project Scheduling Problem (MMRCPSP), a mode represents a feasible combination of a project duration and resource requests that allow the underlying tasks to be accomplished[6]. The Multi-Skill Resource Constrained Project Scheduling Problem (MSRCPSP) considers resources with multiple skills. In this extension, with what skills a resource contributes to which tasks must be determined[7].

In this paper, the PTTP is modeled as an extension of the MSRCPSP. It is the first version of the RCPSP where the aim is to select which tasks to complete (or leave undone) based on the utility value of tasks, while considering resources with multiple skills of different levels and capacities, whose movements between locations are bound by military hierarchy and rest periods. Others have studied most of these aspects to some degree, but an extension with a combination of these factors is lacking. Particularly, fundamental aspects of the PTTP, such as the selection of tasks and the hierarchy of resources, are notably absent in previous research.

The main contributions of this paper are a new comprehensive mathematical compact model for the PTTP, and a solution approach based on decomposing the resources' travel routes with the other decisions to be made. The latter results in a new decomposed model with travel routes generated a priori. Furthermore, we show through a computational study that the decomposed model performs significantly better than the compact model on a number of randomly generated realistic test instances, and that it can provide a good starting point for developing a useful decision support tool for military peacekeeping operations planning.

The remainder of the paper is organized as follows: We first present an overview of relevant literature on variants and extensions of the RCPSP, before giving a detailed description of the PTTP studied in this paper. Further, we present a compact mathematical model of the problem, and how this model can be decomposed using Dantzig-Wolfe decomposition. Finally, we compare the performance of the two models in a computational study, and then give some concluding remarks regarding the contents of the paper.

Literature review

In this section we discuss the variants and extensions of the RCPSP, by going through different attributes of the problem, and discuss how different papers add different characteristics to each attribute. We begin by discussing different objective functions, before addressing task and resource characteristics. We further discuss how time is handled in different models and what solution methods have been applied.

The objective of a basic RCPSP is often time-based, and in most cases looks to minimize the makespan of a project, while the PTTP seeks to maximize the value of completing a subset of the available tasks. The value of a task can be interpreted in several different ways. By giving each task a weight and a quality measure of how well it is completed, Pollack-Johnson and Liberatore[8] seek to maximize the total quality of completed tasks. Tavana et al.[9] also assign a quality measure to tasks, but in addition to maximizing the total quality they look to simultaneously minimize cost and the makespan. In the case of the PTTP, value differs from quality as it is described in Tavana et al.[9], because it is a function of tasks *and* skill level, while quality is only dependent on tasks. This is also the case with the papers that seek to minimize cost or maximize NPV, as neither are dependent on skill level[10–14].

For scheduling problems, it is common for some tasks to have to be carried out in a certain order. Precedence, the term used to define an order of tasks, is therefore usually present in RCPSP problems, in most cases as a finish-to-start precedence relation[12,14–20]. Other precedence relations are start-to-finish, start-to-start, and finish-to-finish relations, which Tavana et al.[9] also address. Fauske[4] considers both finish-to-start and start-to-start precedence. Release times and deadlines for tasks is another common extension of the RCPSP, which affect when a task can be performed. Usually, a task has to start after its release time and be completed before its deadline. Alternatively, the model can penalize solutions where deadlines are exceeded[3,6]. In the PTTP we consider only finish-to-start precedence, and include release and completion times.

In a standard RCPSP problem, tasks are assumed to be located at a single location or site, while in the PTTP we consider multiple locations. Of the papers reviewed in this chapter, only Laurent et al.[20] and Fauske[4] extend the RCPSP to include multiple locations. In the paper by Laurent et al.[20], rather than tasks being fixed to given locations, a subset of resources are tied to certain locations. A task can therefore be completed at various locations. Comparatively, the paper by Fauske[4] and the PTTP assume that tasks are fixed to given locations and that resources are free to move between locations. Strict hierarchy associated with military personnel does, however, directly affect the mobility of resources and hence adds an additional layer of complexity to the PTTP.

Even though the time spent on, and the costs associated with, the transport of resources is a relevant factor, particularly when considering multi-project scheduling, they tend to be neglected by most research[21]. H'Mida and Lopez[22] argue that the coordination of transportation cannot be neglected, as it affects the optimal solution. Recent research on multi-site RCPSPs that do consider travel time, includes papers by Laurent et al.[20], H'Mida and Lopez[22], Adhau et al.[23], and Fauske[4].

A task can be divisible, meaning that multiple resources or extra capacity can be assigned to a task to reduce its duration. For multi-mode models, the processing time for a task in a particular mode is dependent on the resources assigned to the task in that mode[6]. Therefore, all MMRCPSP models include the extension of divisible tasks. Al-Anzi et al.[24], on the other hand, relate the processing time of a task to the skill

level of the assigned resources in their MSRCPSP model. In the PTTP we include both divisible and non-divisble tasks.

Resources in the standard RCPSP are of one type, renewable, can operate at full capacity at all times, and may divide their individual capacity among several tasks simultaneously. To limit the resources to one task at a time, Bianco et al.[25] study dedicated resources with an availability limit for the number of tasks a resource can undertake simultaneously. Similar to Fauske[4] we characterize tasks as exclusive or non-exclusive, where exclusiveness implies that resources can only process that task at a certain point in time, and restrict the assignment of resources to tasks by considering the military hierarchy which forces all resources in a group to travel as a collective unit.

Heimerl and Kolisch[11], Wang and Zheng[26], Al-Anzi et al.[24], Myszkowski et al.[27], Bellenguez and Néron[28], and Fauske[4] all consider non-interchangeable resources that may possess different sets of skills or skill levels. For a resource to be assigned to a task, they must have a skill that matches the task's requirements. Myszkowski et al.[27] and Bellenguez and Néron[28] model tasks to require a minimum skill level to be completed, while Al-Anzi et al.[24] let higher skill levels lead to faster completion times of tasks. Wang and Zheng[26] and Heimerl and Kolisch[11] discuss how to best utilize a workforce with different skills and performance levels to meet requirements and minimize costs. Both Myszkowski et al.[27] and Heimerl and Kolisch[11] let the skill level of the workers affect the cost, and prioritize to meet the requirement of the lowest possible skill level to sufficiently complete a task. In the PTTP we require a given skill level to complete a task, but only the resource capacity, not the skill-level affects the completion time of a task.

Some models might need to include time periods within the project horizon where no activity can occur, or one or more resources cannot be utilized, for example during breaks. Drexl[29] introduce extensions to the standard RCPSP to model issues like labour time regulations. One such extension is to let resources be partly renewable, limiting them to be assigned only a given number of times in a series of time periods, hence forcing each resource to have individual forbidden periods that are not predefined. In the PTTP, time windows ensure that regular tasks are never done at night, however, for tasks that take several days, rest is enforced after the task is completed, and before a new one begins.

Because the RCPSP is NP-hard, it is generally difficult to solve real-sized problem instances. For this reason, the majority of the literature studied here examines various heuristics to find practical and efficient solution methods that are capable of handling realistic instances. Fauske[3], Tavana et al.[9], Afruzi et al.[18], Van Peteghem and Vanhoucke[16], Afshar and Majlesi[15], and Debels and Vanhoucke[30] all test different versions of evolutionary algorithms. Their results are exclusively positive, suggesting that evolutionary algorithms are efficient in solving the RCPSP and its extensions.

Another approach to solving nontrivial RCPSPs, is a decomposition approach. A problem can be decomposed in different manners. Debels and Vanhoucke[30], Zamani[31], Sprecher[32] and Rihm and Trautmann[33] all decompose their projects into sub-parts and iteratively concatenate the solutions. The decomposition approaches in these papers are shown to outperform other exact methods.

In this paper, we solve the PTTP using a decomposition approach. Although such an approach is promising, there is relatively limited research on the application of it to the RCPSP, compared to for example metaheuristic approaches[32,33]. Especially regarding military operation planning, a decomposition approach is a novel solution method.

Problem description

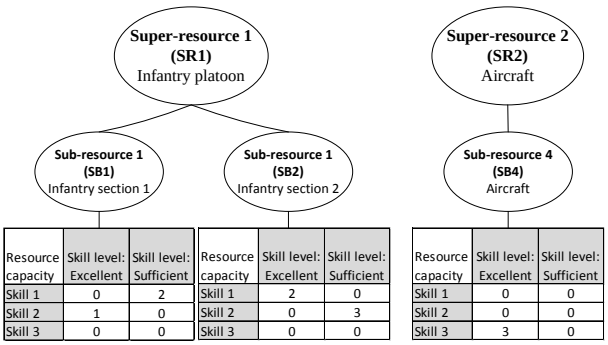
The decisions to be made in the PTTP concern which tasks the resources are to perform at any given time. Resources consist of army resources and supporting airborne resources. Examples of tasks are given in Table 1. A task's importance and a resource's ability to execute the task, both affect the value generated by completing the task. The objective of the problem is to maximize the total realized value, given the resources and time available.

Table 1. Examples of military peacekeeping tasks.

Operational planning and management
Camp security
Quick reaction force
Medical preparedness and sick quarters
Headquarters management
Air defence alert
Search and seizure
Check point
Observation point
Social patrol
Road reconnaissance
Escort of VIP
Intelligence, surveillance, and reconnaissance
Humanitarian support

The resources in the PTTP are assigned to a given area at all times. The assigned area consists of several locations, and at each of these locations, there are different tasks to be completed. One of these locations is a base camp, where the resources spend their nights and free time. During the day, they can be assigned to tasks, either at base camp or at other locations. Because the locations are geographically dispersed, travel times between them has to be considered. Locations can be visited multiple times a day.

Army resources in the PTTP are ordered in a two-level hierarchy, where each level 1 resource is a super-resource (SR) consisting of one or more level 2 sub-resources (SB). A super-resource cannot be spread over several locations, meaning that its sub-resources have to travel as a collective unit. If a super-resource is assigned to a location, it is because the sub-resources subject to it are assigned to tasks there, and the super-resource cannot leave before all of its sub-resources have completed their tasks in that location. Hierarchy therefore restricts movement between locations for the army resources. The army resources are supported by airborne resources consisting of aircraft and helicopters. The supporting resources are not bound by the same hierarchy system as the army resources, and can therefore travel independently. In modelling terms this means that a single



Resource capacity	Skill level: Excellent	Skill level: Sufficient
Skill 1	0	2
Skill 2	1	0
Skill 3	0	0

Resource capacity	Skill level: Excellent	Skill level: Sufficient
Skill 1	2	0
Skill 2	0	3
Skill 3	0	0

Resource capacity	Skill level: Excellent	Skill level: Sufficient
Skill 1	0	0
Skill 2	0	0
Skill 3	3	0

Figure 1. Example of two hierarchies and their sub-resources' skill capacities.

aircraft is both a super-resource and a sub-resource. Figure 1 presents examples of hierarchies for both an army resource and a supporting resource, as well as skill capacities of their respective sub-resources.

Sub-resources possess sets of skills, and for each skill, a sub-resource has a given capacity. *Demining*, *surveillance*, *neutralizing targets*, and *transporting people* are examples of skills a sub-resource can have. Sub-resources that possess the exact same combination of skills and skill capacities, are said to be of the same type. Super-resources consisting of the same type of sub-resources are also said to be of the same type. Skills are assumed to be one of two levels – sufficient or excellent. All resources are renewable, meaning they do not lose capacity or skill by completing a task.

In the PTTP there are unique tasks of varying importance, and hence value. It is generally not possible to complete all tasks in a peacekeeping operation, because of the limited amount of time and resources. Some tasks require other tasks to be completed before they can be undertaken. Other tasks, such as *training local forces*, consist of several sub-tasks, in this case different training sessions, and can thus be completed in parts at different occasions. With such tasks, all sub-tasks have to be completed if one of them is undertaken, by the same army resources.

A task has to be completed in a continuous, uninterrupted fashion by the same resources, i.e. preemption is not permitted. Some tasks have multiple time windows, meaning that they have several possible start and end times. However, because preemption is not permitted, the task can only be completed during one of them. Tasks may be mandatory, meaning that they have to be undertaken during their defined time-window.

In the PTTP, we assume that most tasks have to be completed within regular daytime working hours. Some tasks, such as *humanitarian support* at a local village, can span over multiple days. During the completion of these long tasks, resources do not need to travel back to base camp at night. These lengthy tasks do, however, require the assigned resources to take time off at base camp directly after completion, so that they can have some restitution. During this time, resources cannot be assigned to other tasks.

Tasks require certain skills of certain capacities to be completed. These requirements can be met by one or more resources. For example, if task A requires a capacity of 3 of skill type s, then a resource with a capacity equal to or higher than 3 of skill s will meet the requirement to complete the task. Alternatively, multiple resources with a combined capacity of 3 or higher of skill s can complete the same task. Some tasks, like *searching an area for landmines*, can be completed in a shorter time if extra capacity is assigned to them. These tasks are referred to as divisible tasks. Completion time of indivisible tasks, such as *escorting a VIP* or *monitoring a building*, must be done for a fixed period of time, and can not be shortened regardless of the capacity assigned to them. Resources' skill levels affect how well a task is done and hence the value of completing the task, but not the time it takes to complete it.

A resource's ability to work on several tasks simultaneously depends on the exclusiveness of a task. Exclusiveness entails that resources completing an exclusive task cannot undertake other tasks at the same time. Non-exclusiveness means that resources can undertake more than one non-exclusive task simultaneously, given that the resource has the skill and capacity required, and that the tasks are in the same location. For example, a resource can use its capacity of 1 on two different tasks simultaneously, each one requiring a capacity of 1, if both tasks are non-exclusive and at the same location.

The objective of the PTTP is to realize as much value as possible, given the resources and time available. The effective outcome of solving the problem is deciding which resources are assigned to which tasks at what time.

Mathematical model (CM)

In the following we propose a Mixed Integer Programming (MIP) model for the PTTP, which we refer to as the $Compact\ Model$ (CM). We define first the notation before presenting the mathematical model.

Notation

Sets

$\mathcal{G}$ Set of super-resources

$\mathcal{G}^{\mathcal{ARMY}}$ Subset of army super-resources not including supporting super-resources

$\mathcal{K}_g$ Set of sub-resources that belong to super-resource g

$\mathcal{K}$ Union of all K_g sets

$\mathcal{I}$ Set of locations including dummy locations 1 and I

$\mathcal{P}$ Set of all tasks

$\mathcal{P}^{\mathcal{E}}$ Subset of exclusive tasks

$\mathcal{P}^{\mathcal{DIV}}$ Subset of divisible tasks

$\mathcal{P}_i^{\mathcal{LOC}}$ Subset of tasks that are located at location i

$\mathcal{P}^{\mathcal{LONG}}$ Subset of long tasks, including rest tasks

$\mathcal{P}^{\mathcal{REST}}$ Subset of rest tasks

$\mathcal{P}^{\mathcal{SLEEP}}$ Subset of sleep tasks

$\mathcal{P}_d^{\mathcal{DUP}}$ Subset of tasks that are duplicates of task d

$\mathcal{D}$ Set of unique tasks, each representing a group of duplicate tasks

$\mathcal{S}$ Set of skills

$\mathcal{L}$ Set of skill levels

$\mathcal{N}_i$ Set of possible visits to location i

Parameters

T_{gij}^{TRAVEL} Travel time for super-resource g between

locations i and j

T_p^{TASK} Standard time it takes to complete task p

T_p^{MIN} Minimum percentage of standard time it can take to complete task p

T_p^{RL} Release time for task p

T_p^{DDL} Deadline for task p

C_{ps}^{REQ} Capacity requirement for task p of skill s

C_p^{MAX} Maximum excess capacity task p can utilize as a percentage of C_{ps}^{REQ}

C_{ksl}^{RES} Capacity of resource k of skill s at skill level l

H_{ps} 1 if task p requires skill s, 0 otherwise

V_{pl} Value of completing task p with skill level l

$F_{dd'}^{PREC}$ 1 if group of duplicates d has precedence over group d', 0 otherwise

$F_{dd'}^{CON}$ 1 if resources assigned to a task in groups d must also be assigned to
a task in group d', 0 otherwise

$F_{dd'}^{DIR}$ 1 if a task in group d' is to start directly after the end time of a task
in group d, 0 otherwise

$\overline{R}$ Maximum number of resources that can be assigned to any task

$\overline{T}$ End time of operation

M Big M

Variables

x_{kp} 1 if resource k is assigned to task p, 0 otherwise

q_p 1 if task p is completed, 0 otherwise

u_g 1 if super-resource g is assigned to security, 0 otherwise

w_{kpsl} The capacity of skill s, at skill level l, resource k contributes to meet the capacity requirement of task p

e_{pl} Portion of task p completed at skill level l

t_p^{START} Time task p starts

t_p^{END} Time task p finishes

a_{gim} Arrival time of super-resource g at location i for the m^{th} time

b_{gim} Departing time of super-resource g from location i for the m^{th} time

y_{gim}^{LOC} 1 if super-resource g visits location i for the m^{th} time, 0 otherwise

y_{gimjn}^{TRAVEL} 1 if super-resource g travels directly between its m^{th} visit at location i and n^{th} visit at location j, 0 otherwise

$o_{kpp'}$ 1 if resource k is occupied with long task p' and therefore not required to
complete sleep-task p, 0 otherwise

$\delta_{pp'}$ 1 if task p is completed before task p', 0 otherwise

γ_{kpm} 1 if resource k completes task p on the m^{th} visit of the task's location, 0 otherwise

θ_{gimp} 1 if super-resource g travels to location i for the mth visit before sleep task p, 0 otherwise

Model

Objective function

$$\max \; z = \sum_{p\in\mathcal{P}} \sum_{l\in\mathcal{L}} V_{pl} e_{pl} \tag{1}$$

The objective function (1) maximizes the total value achieved by completing tasks. The given value of each task being completed at a certain skill level is multiplied by the proportion of the task being done at that skill level. For unprocessed tasks, the proportions will be zero, ensuring that no value is added for these tasks. The objective function (1) is maximized subject to the following set of constraints.

Super-resource network constraints

$$a_{gim} - \overline{T}(1-\gamma_{kpm}) \le t_p^{START}, \quad g\in\mathcal{G}, i\in\mathcal{I}, k\in\mathcal{K}_g, m\in\mathcal{N}_i, p\in\mathcal{P}_i^{\mathcal{LOC}} \tag{2}$$

$$b_{gim} + \overline{T}(1-\gamma_{kpm}) \ge t_p^{END}, \quad g\in\mathcal{G}, i\in\mathcal{I}, k\in\mathcal{K}_g, m\in\mathcal{N}_i, p\in\mathcal{P}_i^{\mathcal{LOC}} \tag{3}$$

$$b_{gim} \ge t_p^{END} - \overline{T}(\theta_{gimp} + \sum_{k\in\mathcal{K}_g} \sum_{d'\in\mathcal{D}, p'\in\mathcal{P}_{d'}} \sum_{d^*\in\mathcal{D}, p^*\in\mathcal{P}_{d^*}\cap\mathcal{P}^{\mathcal{LONG}}} F_{d'd^*}^{DIR} F_{d'd^*}^{CON} o_{kpp'}), \quad g\in\mathcal{G}^{\mathcal{ARMY}}, i\in\mathcal{I}, m\in\mathcal{N}_i, p\in\mathcal{P}^{\mathcal{SLEEP}} \tag{4}$$

$$b_{gim} \le t_p^{START} + \overline{T}(1-\theta_{gimp} + \sum_{k\in\mathcal{K}_g} \sum_{\substack{d'\in\mathcal{D},\\ p'\in\mathcal{P}_{d'}}} \sum_{\substack{d^*\in\mathcal{D},\\ p^*\in\mathcal{P}_{d^*}\cap\mathcal{P}^{\mathcal{LONG}}}} F_{d'd^*}^{DIR} F_{d'd^*}^{CON} o_{kpp'}), \quad g\in\mathcal{G}^{\mathcal{ARMY}}, i\in\mathcal{I}, m\in\mathcal{N}_i, p\in\mathcal{P}^{\mathcal{SLEEP}} \tag{5}$$

Constraints (2) and (3) handle the hierarchy requirements associated with super- and sub-resources. They state that for a task to be completed by a certain sub-resource, the super-resource that this sub-resource is a part of has to arrive at the task's location before the task begins, and can only leave after the task is completed. Constraints (4) and (5) ensure that no resources leave a location during sleep periods, unless they travel from a long task ending during the night, to the location of a connected rest task with a direct start condition. $d*$ in (4) and (5) represent duplicate groups of only long tasks.

$$b_{gim} + T_{gij}^{TRAVEL} \le a_{gjn} + M(1 - y_{gimjn}^{TRAVEL}), \quad g\in\mathcal{G}, i,j\in\mathcal{I}, m\in\mathcal{N}_i, n\in\mathcal{N}_j, i\ne j \tag{6}$$

$$\sum_{i\in\mathcal{I}} \sum_{m\in\mathcal{N}_i} \sum_{j\in\mathcal{I}} \sum_{n\in\mathcal{N}_j} T_{gij}^{TRAVEL} y_{gimjn}^{TRAVEL} + \sum_{i\in\mathcal{I}} \sum_{m\in\mathcal{N}_i} (b_{gim} - a_{gim}) = \overline{T}, \quad g\in\mathcal{G} \tag{7}$$

Constraints (6) specify that if a super-resource travels directly between two locations, then it can only arrive at a location a time at least equal to the travel time T_{gij}^{TRAVEL}

after it has left from its previous location. Big M for constraints (6) equals the sum of the end time $\overline{T}$ and the highest travel time for any super-resource between any two locations, i.e. the highest T_{gij}^{TRAVEL} value. Constraints (7) ensure that, at all times, a super-resource is either traveling or at a location.

$$a_{gim} \leq \overline{T} y_{gim}^{LOC}, \quad g \in \mathcal{G}, i \in I, m \in \mathcal{N}_i \tag{8}$$

$$b_{gim} \leq \overline{T} y_{gim}^{LOC}, \quad g \in \mathcal{G}, i \in I, m \in \mathcal{N}_i \tag{9}$$

$$y_{gim}^{LOC} \leq \sum_{k \in \mathcal{K}_g} \sum_{p \in \mathcal{P}_i^{\mathcal{LOC}}} \gamma_{kpm}, \quad g \in \mathcal{G}, i \in \mathcal{I} \backslash (1, I), m \in \mathcal{N}_i \tag{10}$$

$$x_{kp} \leq \sum_{m \in \mathcal{N}_i} y_{gim}^{LOC}, \quad g \in \mathcal{G}, i \in \mathcal{I}, k \in \mathcal{K}_g, p \in \mathcal{P}_i^{\mathcal{LOC}} \tag{11}$$

$$\sum_{m \in \mathcal{N}_i} \gamma_{kpm} = x_{kp}, \quad i \in \mathcal{I}, k \in \mathcal{K}, p \in \mathcal{P}_i^{\mathcal{LOC}} \tag{12}$$

Constraints (63) and (64) assert that a super-resource cannot arrive or leave a location it is not visiting. Constraints (10) limit unnecessary travel by stating that if a super-resource visits a location for the m^{th} time, then a sub-resource belonging to that super-resource has to be assigned to a task in that location on that particular visit. Exceptions are the start and end locations, as there are no tasks at these locations. Constraints (11) state that if a sub-resource is assigned to a task, then it has to visit the task's location at least once. Constraints (12) couple the variables γ and x.

$$\sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{N}_i} y_{gimjn}^{TRAVEL} = y_{gjn}^{LOC}, \quad g \in \mathcal{G}, j \in \mathcal{I} \backslash 1, n \in \mathcal{N}_j, i \neq j \tag{13}$$

$$\sum_{j \in \mathcal{I}} \sum_{n \in \mathcal{N}_j} y_{gimjn}^{TRAVEL} = y_{gim}^{LOC}, \quad g \in \mathcal{G}, i \in \mathcal{I} \backslash I, m \in \mathcal{N}_i, i \neq j \tag{14}$$

Constraints (13) and (14) represent route continuity between locations for the super-resources.

$$y_{gim}^{LOC} \geq y_{gi(m+1)}^{LOC}, \quad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i \backslash |\mathcal{N}_i| \tag{15}$$

To avoid symmetric solutions, constraints (15) state that a super-resource cannot visit a location for the $(m+1)^{th}$ time unless it has visited the location an m number of times already.

Start and end locations

$$a_{g11} = 0, \quad g \in \mathcal{G} \tag{16}$$

$$a_{gI1} \geq a_{gim}, \quad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i \tag{17}$$

Constraints (16) and (17) handle start and end locations for all super-resources. Constraints (16) specify that all super-resources start at location 1 at time zero, while (17) state that all resources ultimately must arrive at location I. Constraints (16) and (17) also ensure that no super-resources can travel to the start location or from the end location.

Resource capacity constraints

$$\sum_{s \in \mathcal{S}} C_{ps}^{REQ} e_{pl} = \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}} w_{kpsl}, \quad p \in \mathcal{P}, l \in \mathcal{L} \tag{18}$$

$$\sum_{k \in \mathcal{K}} \sum_{l \in \mathcal{L}} w_{kpsl} = C_{ps}^{REQ} q_p, \quad p \in \mathcal{P}, s \in \mathcal{S} \tag{19}$$

$$w_{kpsl} \leq C_{ksl}^{RES} x_{kp}, \quad k \in \mathcal{K}, p \in \mathcal{P}, s \in \mathcal{S}, l \in \mathcal{L} \tag{20}$$

Constraints (18) calculate the proportion of a task that is carried out with each skill level. Constraints (19) ensure that this proportion is only positive if a task is completed. They also ensure that a task can only be completed if the task's skill requirements are met, and that the model does not award value to unnecessary work. Constraints (20) state that a resource's contribution to a task cannot be more than its own capacity.

$$x_{kp} \leq \sum_{s \in \mathcal{S}} \sum_{l \in \mathcal{L}} w_{kpsl}, \quad k \in \mathcal{K}, p \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{DIV}} \tag{21}$$

$$x_{kp} \leq \sum_{s \in \mathcal{S}} \sum_{l \in \mathcal{L}} C_{ksl}^{RES} C_{ps}^{REQ}, \quad k \in \mathcal{K}, p \in \mathcal{P}^{\mathcal{DIV}} \tag{22}$$

Constraints (21) and (22) forbid the assignment of resources to tasks which they cannot undertake. Constraints (21) state that, for indivisible tasks, x_{kp} can only be set to 1 if k contributes to the adding of value for task p, and that its total capacity contribution has to be at least 1. For divisible tasks, w_{kpsl} may be zero for resource k and hence not generate any task value, even if k is assigned to task p. This occurs if the capacity assigned to a divisible task exceeds the requirement of that task. The reason for exceeding capacity without adding value is to reduce the duration it takes to complete a task. In this case, constraints (19) and (20) do not provide any restrictions to the skill requirement for resource k to be assigned to tasks p. Constraints (22) therefore make sure resource k is not assigned to a divisible task unless it has the skills required to complete the task. Note that these constraints imply that all capacities are above 1.

Task scheduling constraints

$$\sum_{k \in \mathcal{K}} x_{kp} \geq q_p, \quad p \in \mathcal{P} \tag{23}$$

$$\sum_{k \in \mathcal{K}} x_{kp} \leq \overline{R} q_p, \quad p \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{SLEEP}} \tag{24}$$

$$\sum_{p \in \mathcal{P}_d^{\mathcal{DUP}}} q_p \leq 1, \quad d \in \mathcal{D} \tag{25}$$

Constraints (23) and (24) guarantee that a task can only be completed if one or more resources are assigned to it, and that they cannot be assigned to a task unless that task is

selected. Constraints (24) also limit the number of resources that can work on a single task. Sleep tasks are not subject to this limit. Constraints (25) make sure that tasks with multiple time windows cannot be completed more than once, meaning that only one task in a group d of duplicate tasks can be completed.

$$\sum_{g \in \mathcal{G}^{\mathcal{ARMY}}} u_g = 1 \tag{26}$$

$$x_{kp} \leq 1 - u_g, \quad g \in \mathcal{G}^{\mathcal{ARMY}}, k \in \mathcal{K}_g, p \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{SLEEP}} \tag{27}$$

Constraint (26) ensures that one army super-resource is assigned to the mandatory security post, while constraints (27) make sure that sub-resources belonging to that super-resource are not assigned to any tasks during the planning period (with the exception of sleep tasks).

Exclusive task constraints

$$t_p^{END} - \overline{T}(2 - (x_{kp} + x_{kp'})) \leq t_{p'}^{START} + \overline{T}(1 - \delta_{pp'}), \quad k \in \mathcal{K}, p \in \mathcal{P}, p' \in \mathcal{P}^{\mathcal{E}}, p \neq p' \tag{28}$$

$$t_{p'}^{END} - \overline{T}(2 - (x_{kp} + x_{kp'})) \leq t_p^{START} + \overline{T}\delta_{pp'}, \quad k \in \mathcal{K}, p \in \mathcal{P}, p' \in \mathcal{P}^{\mathcal{E}}, p \neq p' \tag{29}$$

Constraints (28) and (29) deal with the exclusiveness of certain tasks, forcing all tasks handled by resource k to either end before an exclusive task handled by the same resource k, starts, or start after the exclusive task is completed. The binary variable δ ensures that the same task is not affected by both constraints.

Sleep and long task constraints

$$x_{kp} + \sum_{p' \in \mathcal{P}^{\mathcal{LONG}}} o_{kpp'} \geq 1, \quad g \in \mathcal{G}^{\mathcal{ARMY}}, k \in \mathcal{K}_g, p \in \mathcal{P}^{\mathcal{SLEEP}} \tag{30}$$

Constraints (30) require resources to be assigned to all sleep tasks, unless the resource or other resources belonging to the same super-resource are occupied with a long task at the time. This only applies to army resources.

$$t_p^{START} - M(1 - o_{kpp'}) \leq t_{p'}^{END}, \quad k \in \mathcal{K}, p \in \mathcal{P}^{\mathcal{SLEEP}}, p' \in \mathcal{P}^{\mathcal{LONG}} \tag{31}$$

$$t_{p'}^{START} - M(1 - o_{kpp'}) \leq t_p^{END}, \quad k \in \mathcal{K}, p \in \mathcal{P}^{\mathcal{SLEEP}}, p' \in \mathcal{P}^{\mathcal{LONG}} \tag{32}$$

$$o_{kpp'} \leq \sum_{k' \in \mathcal{K}_g} x_{k'p'}, \quad g \in \mathcal{G}^{\mathcal{ARMY}}, k \in \mathcal{K}_g, p \in \mathcal{P}^{\mathcal{SLEEP}}, p' \in \mathcal{P}^{\mathcal{LONG}} \tag{33}$$

Constraints (31)–(33) make sure resources are allowed to undertake long tasks or rest tasks during sleep periods. This is only the case for sleep tasks p overlapping with long or rest tasks p', and only for army resources k belonging to super-resource g where one or more resources k' are assigned to long or rest task p'. Constraints (31) and (32) allow $o_{kpp'}$ to be 1 for sleep task p only when there is a long task or rest task p' starting before the end, and ending after the start of sleep task p. Constraints (33) force $o_{kpp'}$ to zero for sub-resource k if none of the sub-resources belonging to its super-resource are assigned to the long or rest task p'. Thus, the variable $o_{kpp'}$ is zero unless requirements in all three equations are met. Big M in constraints (31) and (32) equals the duration of the shortest sleep task subtracted from the end time $\overline{T}$.

Precedence constraints

$$\sum_{p' \in \mathcal{P}_{d'}^{\mathcal{DUP}}} q_{p'} \leq \sum_{p \in \mathcal{P}_d^{\mathcal{DUP}}} q_p, \quad d, d' \in \mathcal{D}, d \neq d', F_{dd'}^{PREC} = 1 \tag{34}$$

$$t_{p'}^{START} + \overline{T}(1 - q_{p'}) \geq t_p^{END}, \quad d, d' \in \mathcal{D}, p \in \mathcal{P}_d^{\mathcal{DUP}}, p' \in \mathcal{P}_{d'}^{\mathcal{DUP}}, d \neq d', F_{dd'}^{PREC} = 1 \tag{35}$$

Constraints (34) state that a task or one of its duplicates cannot be completed unless a task in the group(s) of duplicate tasks with precedence over it are completed, and constraints (35) set the start time for task p' after the end time of task p. Constraints (35) are not to be binding unless task p' is completed, hence the term $\overline{T}(1 - q_{p'})$.

Connected tasks and direct start constraints

$$\sum_{p \in \mathcal{P}_d^{\mathcal{DUP}}} x_{kp} \leq \sum_{p' \in \mathcal{P}_{d'}^{\mathcal{DUP}}} x_{kp'}, \quad g \in \mathcal{G}^{\mathcal{ARMY}}, k \in \mathcal{K}_g, d, d' \in \mathcal{D}, d \neq d', F_{dd'}^{CON} = 1 \tag{36}$$

$$x_{kp} = x_{k'p}, \quad g \in \mathcal{G}^{\mathcal{ARMY}}, k, k' \in \mathcal{K}_g, p \in \mathcal{P}^{\mathcal{REST}} \tag{37}$$

Constraints (36) ensure that if task p is connected to task p', i.e. $F_{dd'}^{CON}$ equals 1, then any army resource k assigned to task p must also be assigned to task p'. For pairs of connected tasks where both tasks need to be completed by the same combination of resources, $F_{dd'}^{CON}$ and $F_{d'd}^{CON}$ equal 1. Constraints (37) force all or no sub-resources in a super-resource to be assigned to rest task p if it is a rest task.

$$t_p^{END} + F_{dd'}^{CON} T_{gij}^{TRAVEL} \geq t_{p'}^{START}, \quad g \in \mathcal{G}^{\mathcal{ARMY}}, i, j \in \mathcal{I}, d, d' \in \mathcal{D}, p \in (\mathcal{P}_d^{\mathcal{DUP}} \cap \mathcal{P}_i^{\mathcal{LOC}}), p' \in (\mathcal{P}_{d'}^{\mathcal{DUP}} \cap \mathcal{P}_j^{\mathcal{LOC}}), d \neq d', F_{dd'}^{DIR} = 1 \tag{38}$$

For some tasks, there is a requirement that another task starts directly after the first task is completed. Constraints (38) ensure that these requirements are fulfilled, taking into account travel time between the tasks' locations.

Time scheduling constraints

$$t_p^{START} \geq T_p^{RL} q_p, \quad p \in \mathcal{P} \tag{39}$$

$$t_p^{END} \leq T_p^{DDL} q_p, \quad p \in \mathcal{P} \tag{40}$$

If a task is realized, constraints (39) make sure task p starts after its release time R_p, and constraints (40) ensure its completion before its deadline D_p.

$$t_p^{END} \leq t_p^{START} + T_p^{TASK} q_p, \quad p \in \mathcal{P} \tag{41}$$

$$t_p^{END} \geq t_p^{START} + T_p^{MIN} T_p^{TASK} q_p, \quad p \in \mathcal{P} \tag{42}$$

$$t_p^{END} \geq t_p^{START} + T_p^{TASK} \left(\frac{C_p^{MAX} - T_p^{MIN}}{C_p^{MAX} - 1} q_p \right) + T_p^{TASK} \left(\frac{T_p^{MIN} - 1}{\sum_{s \in \mathcal{S}} C_{ps}^{REQ} (C_p^{MAX} - 1)} \cdot \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}} \sum_{l \in \mathcal{L}} C_{ksl}^{RES} H_{ps} x_{kp} \right), \quad p \in \mathcal{P}^{\mathcal{DIV}} \tag{43}$$

Constraints (41)–(43) handle task duration and ensure that tasks are completed in a continuous fashion. Constraints (41) make sure that completing a task does not take longer than necessary, while constraints (42) and (43) limit the shortest time a task can take to complete. T_p^{TASK} is the given time it takes to complete task p when minimum capacity requirements are met, and T_p^{MIN} is the minimum proportion of time it may take if the capacity requirements are exceeded. For indivisible tasks, it is not possible to shorten the duration and T_p^{MIN} equals 1. For divisible tasks the duration can be shortened by the proportion of exceeding capacity down to the minimal proportion T_p^{MIN}. Constraints (43) set the end time of task p according to this.

Variable domains

$$w_{kpsl} \geq 0, \quad k \in \mathcal{K}, p \in \mathcal{P}, s \in \mathcal{S}, l \in \mathcal{L} \tag{44}$$

$$e_{pl} \geq 0, \quad p \in \mathcal{P}, l \in \mathcal{L} \tag{45}$$

$$t_p^{START} \geq 0, \quad p \in \mathcal{P} \tag{46}$$

$$t_p^{END} \geq 0, \quad p \in \mathcal{P} \tag{47}$$

$$a_{gim} \geq 0, \quad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i \tag{48}$$

$$b_{gim} \geq 0, \quad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i \tag{49}$$

$$x_{kp} \in \{0,1\}, \quad k \in \mathcal{K}, p \in \mathcal{P} \tag{50}$$

$$q_p \in \{0,1\}, \quad p \in \mathcal{P} \tag{51}$$

$$u_g \in \{0,1\}, \quad g \in \mathcal{G} \tag{52}$$

$$y_{gim}^{LOC} \in \{0,1\}, \quad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i \tag{53}$$

$$y_{gimjn}^{TRAVEL} \in \{0,1\}, \quad g \in \mathcal{G}, i,j \in \mathcal{I}, m \in \mathcal{N}_i, n \in \mathcal{N}_j \tag{54}$$

$$o_{kpp'} \in \{0,1\}, \quad k \in \mathcal{K}, p, p' \in \mathcal{P} \tag{55}$$

$$\delta_{pp'} \in \{0,1\}, \quad p, p' \in \mathcal{P} \tag{56}$$

$$\gamma_{kpm} \in \{0,1\}, \quad i \in \mathcal{I}, k \in \mathcal{K}, m \in \mathcal{N}_i, p \in \mathcal{P}_i^{\mathcal{LOC}} \tag{57}$$

$$\theta_{gimp} \in \{0,1\}, \quad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i, p \in \mathcal{P}^{\mathcal{SLEEP}} \tag{58}$$

Decomposition solution approach (DM)

Since the CM presented in the previous section is hard to solve using commercial MIP solvers, we propose an alternative decomposed model (DM) based on Dantzig-Wolfe decomposition[34], where all feasible routes for each super-resource are generated a priori. Let $\mathcal{R}_g$ be the set of routes for super-resource g, and let the binary parameters Y_{rgim}^{LOC} and Y_{rgimjn}^{TRAVEL} be equal to 1 if super-resource g on route r visits location (i,m) and travels directly from location (i,m) to (j,n), respectively, and 0 otherwise. We also add the binary variable λ_{rg}, which takes the value of 1 if super-resource g travels route r, and 0 otherwise. Using this notation we substitute the variables y_{gim}^{LOC} and y_{gimjn}^{TRAVEL} in the CM mode as follows:

$$y_{gim}^{LOC} = \sum_{r \in \mathcal{R}_g} Y_{rgim}^{LOC} \lambda_{rg}, \tag{59}$$

$$y_{gimjn}^{TRAVEL} = \sum_{r \in \mathcal{R}_g} Y_{rgimjn}^{TRAVEL} \lambda_{rg}, \tag{60}$$

Notice, that as a result of this, constraints (13)–(15) become redundant and can be removed from the DM. In addition, we have to add the following constraints (61) and (62) to ensure that for all super-resources, exactly one route is selected.

$$\sum_{r \in \mathcal{R}_g} \lambda_{rg} = 1, \quad g \in \mathcal{G} \tag{61}$$

$$\lambda_{rg} \in \{0,1\} \quad g \in \mathcal{G}, r \in \mathcal{R}_g \tag{62}$$

To further strengthen the linear programming relaxation of the DM model, we may calculate the earliest and latest arrival time to node (i,m), A_{rgim} and B_{rgim}, for a given route r travelled by super-resource g. Using these parameters we can bound the earliest and latest arrival time to a node (i,m) as follows:

$$\sum_{r \in \mathcal{R}_g} A_{rgim} \lambda_{rg} \leq a_{gim}, \quad g \in \mathcal{G}, i \in I, m \in \mathcal{N}_i \tag{63}$$

$$\sum_{r \in \mathcal{R}_g} B_{rgim} \lambda_{rg} \leq b_{gim}, \quad g \in \mathcal{G}, i \in I, m \in \mathcal{N}_i \tag{64}$$

To generate the set of feasible travel routes for each super-resource g, $\mathcal{R}_g$, we use a dynamic programming algorithm.

Computational study

In the following, we present a comparison of the performance of the CM and DM. For the DM, the total run time is the aggregate of the time it takes to generate travel routes, which is done in Python, and run the optimization model using the commercial MIP-solver FICO Xpress. For the CM, the total run time is only the time it takes to run the optimization model in FICO Xpress.

We have generated three sets of realistic test instances. The first set is designed to test how the models handle a varying number of resources. The second set tests for an increasing number of locations, while the third set tests for an increasing number of tasks. Testing along these dimensions makes it possible to study how the two models react to an increase in resources, tasks, and locations. Each set of instances consists of five subsets. Each of these subsets have five test instances. As a result, each set consists of 25 test instances. Since task characteristics are generated randomly for every test instance, the test instances belonging to the same subset differ even though their other parameters are equivalent.

The identification for every subset has the following logic: [set–number of super-resources–number of total tasks–number of locations]. Set 1 is labeled ”R” for resource, set 2 is labeled ”L” for location, and set 3 is labeled ”T” for tasks. The identification for a subset from set 1 with four super-resources, 30 tasks and 8 locations is for example ”R–4–30–8”.

The comparison of the performance of the decomposition-based solution approach, using the DM model, with the CM on instances from sets 1–3 is shown in Tables 2, 3, and 4, respectively. The tables report the best solutions found after one hour and after 20 minutes, the optimality gap after one hour, and the total run time of the CM and the DM for test instances 1–3, respectively. The best solution found after 20 minutes is a measurement that is included in the analysis because 20 minutes is deemed the longest acceptable run time in a real military operation planning context[4].

The results for each subset are calculated as the average over its five instances, with the exception of the optimality gap. The optimality gap is calculated using the average upper bound and average best solution. In the cases where no solution is found, the best solution after 20 minutes and the best solution found after one hour are both set to zero. These cases are marked with an asterisk (*). An alternative would be to omit these cases when calculating the average, but this would skew the results because for subsets where fewer than all instances are solved, it could lead to higher average values.

Table 2 suggests that the DM dominates the CM when increasing the number of resources (instance set 1), having equal or better best solution found, best solution found after 20 minutes, optimality gap, and run time, for all five subsets. Furthermore, for some test instances including 8, 10, and 12 super-resources, the CM did not find any solution in the given run time of one hour, while the DM found at least one feasible solution for all instances.

The results in Table 3 indicate that the DM provides better or equal results also for the instances in set 2, in a shorter amount of time than the CM. For a few test instances with 10 and 12 locations, the CM did not find any feasible solution within one hour. The DM found at least one solution for all instances, but did not find the optimal solution for all the instances in the subset including 12 locations.

Table 2. Computational results test set 1: Varying the number of resources.

Subset	Obj. value (one hour)		Obj. value (20 min)		Gap after one hour [%]		Total run time [s]	
	CM	DM	CM	DM	CM	DM	CM	DM
R-4-30-8	12.5	12.5	12.5	12.5	0.0	0.0	0.5	0.9
R-6-30-8	18.0	18.0	18.0	18.0	0.0	0.0	1.9	1.7
R-8-30-8	24.4*	41.2	24.4*	41.2	20.0*	0.0	728.3	21.2
R-10-30-8	22.0*	87.2	22.0*	84.8	74.2*	6.1	3600	2215
R-12-30-8	44.8*	74.0*	23.2*	74.0	40.0*	21.0*	2095.7	2210
Average	*24.3*	*46.6*	*20.0*	*46.1*	-	-	*1285.4*	*890*

Table 3. Computational results test set 2: Varying the number of locations.

Subset	Obj. value (one hour)		Obj. value (20 min)		Gap after one hour [%]		Total run time [s]	
	CM	DM	CM	DM	CM	DM	CM	DM
L-6-30-4	35.3	35.3	35.3	35.3	0.0	0.0	7.0	1.8
L-6-30-6	35.2	35.2	35.2	35.2	0.0	0.0	122.0	2.6
L-6-30-8	21.5	21.5	21.5	21.5	0.0	0.0	2.5	1.7
L-6-30-10	29.1*	37.4	29.1*	37.4	20.0	0.0	724.1	16.4
L-6-30-12	10.8*	43.2	10.8*	43.2	23.6*	1.6	1442.6	734
Average	*26.4*	*34.5*	*26.4*	*34.5*	-	-	*459.7*	*151*

Table 4. Computational results test set 3: Varying the number of tasks.

Subset	Obj. value (one hour)		Obj. value (20 min)		Gap after one hour [%]		Total run time [s]	
	CM	DM	CM	DM	CM	DM	CM	DM
T-6-30-8	18.0	18.0	18.0	18.0	0.0	0.0	1.9	1.7
T-6-40-8	40.4*	70.6	36.4*	70.6	45.8*	0.0	2883.8	1449
T-6-50-8	70.1*	50.5*	70.1*	50.5*	47.0*	60.0*	2883.1	2285
T-6-60-8	59.5	86.7	25.9	86.7	43.8	0.0	2883.0	1622
T-6-70-8	74.6	68.8*	73.2	68.8*	20.8	22.5*	2371.6	2286
Average	*52.5*	*58.9*	*44.7*	*58.9*	-	-	*2204.7*	*1529*

The results in Table 4 show that both models struggle to efficiently handle instances with a large number of tasks. Comparing the two models, the DM still performs better than the CM for four out of the five subsets within one hour of run time. Within a run time of 20 minutes, it provides an equal or better solution for three out of the five subsets. Furthermore, the DM does in general take a shorter amount of time to solve the instances, but the optimality gaps are significant.

Overall, the DM performs better than the CM, indicating that the proposed decomposed-based solution approach is effective. The DM does nonetheless, struggle to handle the largest instances efficiently. The number of resources and the number of tasks seem to be the most important factors influencing solvability for the DM. In particular, an increase in resources and tasks seem to greatly affect the number of routes generated, which again affects the performance of the model. As an example here, it can be mentioned that for test instances R-8-30-8, the average number of feasible routes

is 8081, while for test instances R-12-30-8 this number has grown (exponentially) to 141,107. An analysis of the run time for the DM shows that, for most subsets, the time it takes to generate routes takes less than 10% of the total run time. However, for the instances with a large number of travel routes, a considerable portion of the remaining run time is the time it takes the optimization model to read the travel route files.

Conclusions

The models presented in this thesis are intended as a decision support tool for military peacekeeping operations. The problem considered is termed a Peacekeeping-Troops-To-Tasks Problem (PTTP), and it includes a number of realistic features, e.g. the combination of complex task relations, such as connected tasks and direct start requirements, the introduction of long tasks, sleep tasks, rest tasks and a security task, the possibility of multiple time windows using duplicate tasks, and unavailable time periods for resources. The mathematical models are based on the notion that the PTTP is a two-tier problem consisting of a network-flow problem for super-resources moving between locations, and a multi-skill scheduling problem for assigning tasks to sub-resources at each location.

Due to the complexity of the problem, it is difficult to solve large instances using a model based on a compact formulation of the problem. A decomposition approach is therefore introduced as an alternative solution method, where possible travel routes are generated a priori. A computational study shows that, in general, using the decomposed model formulation gives shorter computing times and provides better solutions than the compact model formulation. This is especially true as the number of resources, tasks, and locations increase, proving that a decomposed solution approach is effective. However, even the decomposed model is unable to handle the largest test instances efficiently. For these instances, heuristic solution methods are probably needed to generate feasible and good solutions in short amount of time. Anyway, we believe the models and the computational results provided here can be used as a good starting point for developing a useful decision support tool for military peacekeeping operations planning.

References

1. Section for Security Policy and North America. FNs fredsoperasjoner. https://www.regjeringen.no/no/tema/utenrikssaker/sikkerhetspolitikk/fredsbevarende_operasjoner/fn_fredsoperasjoner/id86766/, 2017. Accessed: 2017-12-03.
2. NATO. Operations and missions: past and present. https://www.nato.int/cps/en/natohq/topics_52060.htm, 2016. Accessed: 2018-08-07.
3. Fauske MF. Using a genetic algorithm to solve the troops-to-tasks problem in military operations planning. *The Journal of Defense Modeling and Simulation* 2017; 14(4): 439–446.
4. Fauske MF. Optimizing the troops-to-tasks problem in military operations planning. *Military Operations Research* 2015; 20(4): 49–57.
5. Blazewics J, Lenstra JK and Kan AHGR. Scheduling subject to resource constraints: Classification and Complexity. *Discrete Applied Mathematics* 1983; 5: 11–24.
6. Hartmann S and Briskorn D. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research* ; 207(1).
7. Almeida BF, Correia I and da Gama F. Priority-based heuristics for the multi-skill resource constrained project scheduling problem. *Expert Systems with Applications* 2016; 57: 91–103.
8. Pollack-Johnson B and Liberatore MJ. Incorporating quality considerations into project time/cost tradeoff analysis and decision making. *IEEE Transactions on Engineering Management* 2006; 53: 534–542.
9. Tavana M, Abtahi AR and Khalili-Damghani K. A new multi-objective multi-mode model for solving preemptive time–cost–quality trade-off project scheduling problems. *Expert Systems with Applications* 2014; 41(4): 1830–1846.
10. Santos M and Tereso A. On the multi-mode, multi-skill resource constrained project scheduling problem - a software application. In *Soft Computing in Industrial Applications*. Springer, pp. 239–248.
11. Heimerl C and Kolisch R. Scheduling and staffing multiple projects with a multi-skilled workforce. *OR Spectrum* 2010; 32(2): 343–368.
12. Waligóra G. Discrete-continuous project scheduling with discounted cash inflows and various payment models - a review of recent results. *Annals of Operations Research* 2014; 213(1): 319–340.
13. Schutt A, Chu G, Stuckey PJ et al. Maximising the net present value for resource-constrained project scheduling. In *International Conference on Integration of Artificial Intelligence (AI) and Operations Research (OR) Techniques in Constraint Programming (CPAIOR)*. Springer, pp. 362–378.
14. Gacias B, Artigues C and Lopez P. Parallel machine scheduling with precedence constraints and setup times. *Computers & Operations Research* 2010; 37(12): 2141–2151.
15. Afshar-Nadjafi B and Majlesi M. Resource constrained project scheduling problem with setup times after preemptive processes. *Computers & Chemical Engineering* 2014; 69: 16–25.
16. Van Peteghem V and Vanhoucke M. A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research* 2010; 201(2): 409–418.
17. Ranjbar M, De Reyck B and Kianfar F. A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. *European Journal of Operational Research* 2009; 193(1): 35–48.
18. Afruzi EN, Najafi AA, Roghanian E et al. A multi-objective imperialist competitive algorithm for solving discrete time, cost and quality trade-off problems with mode-identity and resource-constrained situations. *Computers & Operations Research* 2014; 50: 80–96.
19. Moukrim A, Quilliot A and Toussaint H. Branch and price for preemptive and non preemptive rcpsp based on interval orders on precedence graphs. In *Recent Advances in Computational Optimization*. Springer, 2015. pp. 85–106.
20. Laurent A, Deroussi L, Grangeon N et al. A new extension of the rcpsp in a multi-site context: Mathematical model and metaheuristics. *Computers & Industrial Engineering* 2017; 112: 634–644.

21. Krüger D and Scholl A. Managing and modelling general resource transfer in (multi-)project scheduling. *OR Spectrum* 2010; 32(2): 369–394.
22. H'Mida F and Lopez P. Multi-site scheduling under production and transportation constraints. *International Journal of Computer Integrated Manufacturing* 2013; 26(3): 252–266.
23. Adhau S, Mittal ML and Mittal A. A multi-agent system for decentralized multi-project scheduling with resource transfers. *International Journal of Production Economics* 2013; 146(2): 646–661.
24. Al-Anzi FS, Al-Zame K and Allahverdi A. Weighted multi-skill resources project scheduling. *Journal of Software Engineering and Applications* 2010; 3(12): 1125.
25. Bianco L, Dell'Olmo P and Speranza M. Heuristics for multimode scheduling problems with dedicated resources. *European Journal of Operational* 1998; 107(2): 260–271.
26. Wang L and Zheng Xl. A knowledge-guided multi-objective fruit fly optimization algorithm for the multi-skill resource constrained project scheduling problem. *Swarm and Evolutionary Computation* 2017; .
27. Myszkowski PB, Skowroński ME, Olech P et al. Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem. *Soft Computing* 2015; 19(12): 3599–3619.
28. Bellenguez O and Néron E. Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills. In *Practice and Theory of Automated Timetabling V*. Springer, pp. 229–243.
29. Drexl A, Nissen R, Patterson JH et al. Progen/πx–an instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions. *European Journal of Operational Research* 2000; 125(1): 59–72.
30. Debels D and Vanhoucke M. A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. *Operations Research* 2007; 55(3): 457–469.
31. Zamani R. A hybrid decomposition procedure for scheduling projects under multiple resource constraints. *Operational Research* 2011; 11(1): 93–111.
32. Sprecher A. Network decomposition techniques for resource-constrained project scheduling. *Journal of the Operational Research Society* 2002; 53(4): 405–414.
33. Rihm T and Trautmann N. A mip-based decomposition heuristic for resource-constrained project scheduling. In *14th International Conference on Project Management and Scheduling*. p. 193.
34. Dantzig GB and Wolfe P. Decomposition principle for linear programs. *Operations Research* 1960; 8(1): 101–111.