NTNU
Norwegian University of
Science and Technology

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

Nadia Chaudry
Ingunn Vermedal

Industrial Economics and Technology Management

Submission date:	June 2018
Supervisor:	Kjetil Fagerholt, IØT
Co-supervisor:	Magnus Stålhane, IØT
	Maria Fauske, FFI

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Problem Description

The purpose of this thesis is to study the troops-to-tasks problem for military peacekeeping operations. Two exact mathematical models are formulated with the intent to find the optimal schedule for a military battlegroup during a peacekeeping operation. Furthermore, two different heuristic solution methods are analyzed to determine whether the model can provide good solutions for realistic-sized test instances within a reasonable amount of time, and hence present itself as a valuable decision support tool for operation planners.

Preface

This master thesis concludes our Master of Science at the Norwegian University of Science and Technology (NTNU) during the spring of 2018. Our field of specialization is Applied Economics and Operation Research at the Department of Industrial Economics and Technology Management. The problem studied in this thesis is at the behest of the Norwegian Defense Research Establishment (FFI), and examines how an exact and heuristic optimization solution approach can assist in the planning of military peacekeeping operations. This thesis is a continuation of our project report written during the fall of 2017.

We would like to give a special thanks to our supervisors Professor Kjetil Fagerholt and associate Professor Magnus Stålhane at the Department of Industrial Economics and Technology Management, and Maria Fleischer Fauske at FFI, for their thorough and valuable guidance throughout the semester.

Nadia Chaudry and Ingunn Vermedal,
Trondheim, June 19 2018

Abstract

Military peacekeeping operations are becoming increasingly complex, while at the same time facing stricter budgetary restrictions. It is therefore imperative, now more than ever, to have good operation plans that utilize resources efficiently. Currently, the assignment of human resources to military operation tasks is carried out manually. A decision support tool has the potential to greatly improve this process, and simultaneously meet the complexity and financial challenges that exist.

The assignment of human resources to tasks in military peacekeeping operations is defined as a Peacekeeping-Troops-To-Tasks Problem (PTTP). The purpose of a troops-to-tasks analysis in a military peacekeeping operation is to determine which troops are to be assigned to which tasks, at any given time, during a certain planning period. The objective is to find an operation schedule that maximizes the total utility value of completed tasks. What distinguishes the PTTP in this thesis from previous research, is 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, unavailable time periods for resources, and a longer planning horizon. Two exact models are formulated to solve the PTTP: One compact model and one decomposed model. Three travel route generation methods are proposed for the decomposed model, one exact, and two heuristic methods, of which the exact method generates all feasible routes, and the heuristic methods look to only generate the most promising routes.

Computational studies show that even though the decomposed model outperforms the compact model, the PTTP is extensive and complicated to solve when using an exact solution method. The exact decomposed model handles multiple locations well, but high numbers of resources and tasks make it difficult to calculate good solutions in a reasonable amount of time. The heuristic solution methods provide better results for larger instances in the given run time, suggesting that the decomposed model using a heuristic method to generate travel routes has the potential to become a useful decision support tool for military operation planning.

Sammendrag

Militære fredsbevarende operasjoner blir stadig mer kompliserte, samtidig som de årlige budsjettene kuttes. Dermed er det stadig viktigere å utvikle gode operasjonsplaner som sørger for å utnytte ressurser på en effektiv måte. I dag foregår tildelingen av fredsbevarende oppgaver blant militære ressurser manuelt. Et beslutningsverktøy vil kunne forbedre en slik prosess, og samtidig løse de kompliserende og finansielle utfordringene som finnes.

Fordelingen av oppgaver blant militære ressurser i en fredsbevarende operasjon er definert som et Peacekeeping-Troop-To-Task Problem (PTTP). En slik analyse skal bestemme hvilke ressurser som skal få tildelt hvilke oppgaver til enhver tid under en operasjon. Målet er å finne oppgavefordelingen som maksimerer verdien som skapes av å utføre oppgaver. Det som skiller PTTP i denne avhandlingen fra tidligere forskning er hvor mange forskjellige måter oppgaver relaterer til hverandre på, som for eksempel koblede oppgaver og direkte-start-oppgaver, introduksjonen av flere oppgavetyper som lange oppgaver, frioppgaver og sikkerhetsoppgaver, muligheten for en oppgave til å ha flere tidsvinduer ved bruk av dupliserte oppgaver, tidsperioder hvor det er forbudt å gjøre oppgaver, og at den tar for seg lengre tidsperioder. For å løse PTTP er det utviklet to eksakte modeller: En kompakt modell og en dekomponert modell. Videre er tre rutegenereringsmetoder formulert for den dekomponerte modellen: Én eksakt metode og to heuristiske metoder. Den eksakte metoden generer alle lovlige ruter, mens de heuristiske metodene oppsøker kun de mest lovende rutene.

Testresultatene viser at selv om den dekomponerte modellen yter bedre enn den kompakte modellen, så er PTTP et omfattende problem å løse med en eksakt løsningsmetode. Den eksakte dekomponerte modellen klarer å håndtere flere lokasjoner på en god måte, men et større antall ressurser og oppgaver gjør det vanskelig å finne gode løsninger innenfor en akseptabel kjøretid. De heuristiske løsningsmetodene finner bedre løsninger innenfor den gitte kjøretiden. Basert på disse resultatene så er en dekomponert modell som bruker en heuristisk rutegenereringsmetode et lovende beslutningsverktøy for planleggingen av militære operasjoner.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

DM	Decomposed model
DTCTP	Discrete Time/Cost Trade-off Problem
FFI	Norwegian Defence Research Establishment (Forsvarets Forskningsinstitutt)
H1	Heuristic method 1
H2	Heuristic method 2
LP	Linear programming
MMRCPSP	Multi-Mode Resource Constrained Project Scheduling Problem
MSRCPSP	Multi-Skill Resource Constrained Project Scheduling Problem
NATO	The North Atlantic Treaty Organization
NPV	Net present value
CM	Compact model
PSP	Project Scheduling Problem
PTTP	Peacekeeping Troops-to-Tasks problem
RCPSP	Resource Constrained Project Scheduling Problem
SB	Sub-resource
SR	Super-resource
UN	United Nations

Chapter 1

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 (Section for Security Policy and North America, 2017). 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 (NATO, 2016).

Over the last few decades the purpose of peacekeeping missions has remained unchanged, but the dynamics have shifted. From being about serving as a neutral buffer between consenting parties, peacekeeping has evolved into becoming a much more complex task where political, economic, and social change all have to be managed simultaneously, and under increasingly difficult circumstances (Eide, 2001). In fact, the UN regard peacekeeping as one of their most complex operational tasks, as exemplified by their following statement (United Nations, 2017b):

> "*UN Peacekeepers were now increasingly asked to undertake a wide variety of complex tasks, from helping to build sustainable institutions of governance, to human rights monitoring, to security sector reform, to the disarmament, demobilization and reintegration of former combatants.*"

In addition to peacekeeping operations becoming more complex, annual peacekeeping budgets are declining. Between the fiscal year 2016–2017 and 2017–2018, the UN Peacekeeping budget decreased by 7.5% to USD 6.8 billion. This budget funds 13 of the 14 current UN Peacekeeping operations (United Nations, 2017a). The complexity and financial challenges peacekeeping operations face, underline why it is increasingly important to have good operation plans that utilize resources in an as effective manner as possible.

Figure 1.1: Soldiers planning during a NATO operation. Photo: The Norwegian Armed Forces.

Figure 1.2: Soldier on patrol during a UN operation in Mali. Photo: The Norwegian Armed Forces.

The most critical resource in peacekeeping operations is human capital. Deciding how to best utilize this resource in military operation 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 (Fauske, 2017). Typically, a troops-to-task problem is solved manually. This thesis is motivated by a request from the Norwegian Defense Research Establishment (FFI) who believe that 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 (Fauske, 2015). At worst, it could provide a solution that operation planners can use as a starting point.

The specific problem scenario considered in this thesis is defined as the Peacekeeping Troops-to-Tasks Problem (PTTP), and is outlined in collaboration with FFI. It consists of a 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. A battlegroup 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 who.

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. FFI has conducted some research on troops-to-tasks analysis of high intensity operations, 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). It is the first version where the aim is to select which tasks to complete or leave undone based on the utility value of tasks,

Figure 1.3: Transportation exercise of an injured soldier in Afghanistan. Photo: The Norwegian Armed Forces.

Figure 1.4: Soldiers instructing Peshmerga soldiers in Kurdistan. Photo: The Norwegian Armed Forces.

while considering resources with multiple skills of different levels and capacities who's 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.

This thesis is a continuation of the work by Chaudry and Vermedal (2017) who model a simple version of the PTTP. Their proposed model can only solve small problem instances for a planning period of only one day, and lacks certain important aspects of the problem. Therefore, it is not useful for solving realistic-sized problems. In this thesis, we look to extend their model to consider a longer planning period, rest periods, and additional task types and characteristics. The extended model is referred to as a compact model. Furthermore, we present a decomposed solution approach which first generates travel routes for the different units in a battlegroup, before finding the best routes, and the set of tasks to complete, in an optimization model. We propose three methods to generate travel routes: One exact method, and two heuristic methods. The objective of the compact and decomposed model is to maximize the total realized value by completing tasks. Both models can act as a decision support tool on multiple strategic levels because their flexibility makes them applicable to large operations, sub-operations, and daily operations at a camp.

The structure of this thesis is as followed: A literature study is presented in Chapter 2 to provide an outline of previous research on problems that are comparable to the PTTP. A detailed description of the problem studied in this thesis follows in Chapter 3. Chapter 4 presents a mathematical model of the PTTP, and Chapter 5 presents the reformulated, decomposed model, and describes the three methods used to generate travel routes. Input data used to test the models, and the generation of test instances, are described in Chapter 6. A computational study is presented in Chapter 7. Chapter 8 offers concluding remarks on the findings of this thesis, and lastly, Chapter 9 discusses future research.

Chapter 2

Literature Review

This chapter is intended to provide an overview of current literature relevant to the PTTP. Apart from the the paper by Fauske (2015), there is, to the best of our knowledge, very limited research on applying optimization to military operation planning. The focus of this literature review is therefore the standard RCPSP, of which the PTTP is an extension. The standard RCPSP and its most common extensions are described in the first part of this chapter in Section 2.1. A description of our research strategy for relevant literature follows in Section 2.2. Sections 2.3–2.6 present the resulting literature and is structured according to key aspects of the RCPSP. These aspects consist of a variety of objective functions, resource- and activity characteristics, temporal constraints, and solution methods. Lastly, Section 2.8, discusses how the models presented in this report relate to existing literature, and describes our contribution to the field of optimization of military operation planning.

2.1 The standard RCPSP and its most common extensions

A formulation of the standard RCPSP is given by equations (2.1)–(2.5). A project consists of a set $\mathcal{P}$ of tasks i that have to be completed by a set $\mathcal{K}$ of resources k. The objective is to find a schedule that minimizes the duration of the project while observing precedence and resource constraints. Tasks require a certain amount of capacity Q_{ik} from resource k, and resources have to be assigned to tasks such that the capacity requirements from the tasks do not exceed the resource capacity C_k. Each task can only start once, and their duration is given by the parameter U_i. Some tasks have precedence over others and are defined by the set of precedence relations in set $\mathcal{F}$. Time is represented as a set $\mathcal{T}$ of discrete time points. The binary decision variable x_{it} indicates whether task i starts at time t.

Objective function for the standard RCPSP:

$$\min \ z = \sum_{i\in\mathcal{P}}\sum_{t\in\mathcal{T}} t x_{it} \tag{2.1}$$

Subject to:

$$\sum_{t\in\mathcal{T}} x_{it} = 1 \qquad i \in \mathcal{P} \tag{2.2}$$

$$\sum_{t\in\mathcal{T}} t x_{it} + U_i \leq \sum_{t\in\mathcal{T}} t x_{jt} \qquad (i,j) \in \mathcal{F} \tag{2.3}$$

$$\sum_{i\in\mathcal{P}, t\leq\tau\leq t+U_i} Q_{ik} x_{it} \leq C_k \qquad \tau \in \mathcal{T}, k \in \mathcal{K} \tag{2.4}$$

$$x_{it} \in \{0,1\} \qquad i \in \mathcal{P}, t \in \mathcal{T} \tag{2.5}$$

Equation (2.1) minimizes the duration of the entire project, and constraints (2.2) ensure that each task starts exactly one time. Constraints (2.3) ensure precedence is upheld by forcing the start time for project j to be later than the addition of the start time and duration time for project i, if i has precedence over j. Constraints (2.4) makes sure the combined capacity requirements Q_{ik} for all tasks assigned to resource k at the same time do not exceed the resources capacity C_k. Constraints (2.5) enforce binary values for variable x_{it}.

The RCPSP is known to be NP-hard, 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 (Hartmann and Briskorn, 2010). 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 (Almeida et al., 2016). In this report, the PTTP is modeled as an extension of the MSRCPSP, because there are several parallels between the two problems considering resource characteristics. It is also possible to model the PTTP as a MMRCPSP. However, this is deemed highly impractical because the pre-processing of data required to create modes would take a considerable amount of time.

2.2 Literature search strategy

Our search strategy for relevant literature is influenced by Hartmann and Briskorn's (2010) survey article on the variants and extensions of the RCPSP problem. The survey article provides a thorough overview of the key features of this type of problem, and has therefore shaped the structure of this review. To find subsequent relevant literature, searces for RCPSP related terms, such as *RCPSP + MMRCPSP* have been conducted on Google Scholar. All the terms used in conjunction with *RCPSP* in the search are presented in the second row of Table 2.1. The first row of the table is a description of the related groups of search words. The most relevant articles resulting from the searches are accounted for in this review.

Table 2.1: Search words used in conjunction with the term *RCPSP* for literature review.

General terms	**Objective function**	**Activity characteristics**	**Resource characteristics**	**Temporal constraints**
MMRCPSP MSRCPSP Exact method Solution method Decomposition	Objective fnc Tradeoff Quality	Activity crashing Precedence Location Multi-site Value	Hierarchy Capacity Skill, Multi skill Renewable	Release, Deadline Night, Day Traveltime Transfertime Discrete Continous Setup time

2.3 Objective function

The objective of a basic RCPSP is often time-based, and in most cases looks to minimize the makespan of a project, which is defined as the total time that elapses from the beginning to the end of a project. Minimizing the makespan is exemplified by the objective function (2.1) in Section 2.1, and is also applied in the paper by Fauske (2015). Other time-based objectives are related to the lateness, tardiness, and earliness of activities. These characteristics can either be regarded separately as absolute, or combined as a weighted sum. There are also objective variants based on minimizing the cost related to renewable and nonrenewable resources whilst observing a project deadline. Tavana et al. (2014) and Pollack-Johnson and Liberatore (2006) present further extensions that introduce quality to the objective function. In the paper by Tavana et al. (2014), quality, cost and time are considered in a multi-objective, multi-mode model.

2.4 Activity characteristics

Activity characteristics are characteristics that are expressed explicitly as a function of the activity they describe. The characteristics discussed here are precedence, location, divisibility, and value. The term *activity* and *task* are assumed synonymous in this report.

For scheduling problem, it is common for some activities to have to be carried out in a certain order. Precedence, the term used to define a order of activities, is therefore usually present in RCPSP problems, in most cases as a finish-to-start precedence relation (Waligóra, 2014; Afshar-Nadjafi and Majlesi, 2014; Van Peteghem and Vanhoucke, 2010; Ranjbar et al., 2009; Afruzi et al., 2014; Gacias et al., 2010; Moukrim et al., 2015; Laurent et al., 2017). Other precedence relations are start-to-finish, start-to-start, and finish-to-finish relations, which Tavana et al. (2014) also address. Fauske (2015) considers both finish-to-start and start-to-start precedence.

In a standard RCPSP problem, tasks are assumed to be located at a single location or site. There are extensions to the problem that allow for multiple sites. Fauske (2015) considers multiple locations that each include tasks that all have to be completed. Laurent et al. (2017) propose a multi-site RCPSP model where the site on which a task is completed is not fixed, but to be determined by the model. Gacias et al. (2010) and Afshar-Nadjafi and Majlesi (2014) do not consider multiple sites in their models, but do include setup times which can be thought of as the traveling time between tasks at different locations. Hence setup time between tasks can be used to model multiple locations in a RCPSP.

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 (Hartmann and Briskorn, 2010). Therefore, all MMRCPSP models include the extension of divisible tasks (see Table 2.2). Al-Anzi et al. (2010), on the other hand, relate the processing time of a task to the skill level of the assigned resources in their MSRCPSP model. Independent of multi-mode or -skill, Waligóra (2014) presents different classes of processing rate functions, and applies an activity processing rate that is a continuous, increasing function of the amount of continuous resource capacity assigned to an activity.

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 (2006) seek to maximize the total quality of completed tasks. Tavana et al. (2014) also assign a quality measure to tasks, but in addition to maximizing the total quality they look to simultaneously minimize cost and the makespan. Waligóra (2014), Schutt et al. (2012), and Liu and Wang (2011) all associate monetary value to each task and seek to maximize either NPV or profit.

2.5 Resource characteristics

Resources in the standard RCPSP are of one type, most often manpower, renewable, can operate at full capacity at all times, and may divide their individual capacity among several tasks simultaneously. Hartmann and Briskorn (2010) point out that resources can also be non-renewable, such as a budget, in multi-mode problems.

To limit the resources to one task at a time, Bianco et al. (1998) study dedicated resources with an availability limit for the number of tasks a resource can undertake simultaneously. Fauske (2015) handles availability by characterizing tasks as exclusive or non-exclusive. Exclusiveness implies that resources can only process that task at a certain point in time. Furthermore, Fauske (2015) restricts 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 (2010), Wang and Zheng (2017), Al-Anzi et al. (2010), Myszkowski et al. (2015), Bellenguez and Néron (2005), and Fauske (2015) all consider noninterchangeable 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. (2015) and Bellenguez and Néron (2005) model tasks to require a minimum skill level to be completed, while Al-Anzi et al. (2010) let higher skill levels lead to faster completion times of tasks. Wang and Zheng (2017) and Heimerl and Kolisch (2010) discuss how to best utilize a workforce with different skills and performance levels to meet requirements and minimize costs. Both Myszkowski et al. (2015) and Heimerl and Kolisch (2010) 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.

2.6 Temporal constraints

In the basic formulation of the RCPSP presented in Section 2.1, time is discretized, meaning that tasks can only have start and end times at discrete points in time. Time can also be modelled as continuous with the use of time variables, enabling activities to start at any time. In their brief study, Kopanos et al. (2014) observe that discrete time models perform better than continuous time models when the number of tasks increases because the number of variables do not increase as fast.

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. Time-switch constraints can limit work to be done in chosen time windows by dividing the planning horizon into cycles of work and rest time windows (Hartmann and Briskorn, 2010). Drexl et al. (2000) 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.

Release times and deadlines for tasks are common extensions to add to the RCPSP. Usually, a task has to start after its release time and be complete before its deadline. Alternatively, the model can penalize solutions where deadlines are exceeded (Hartmann and Briskorn, 2010). Batptiste et al. (1999) finds that release times and deadlines can be useful when modeling problems without preemption, where tasks can be put on hold, or as an alternative to a strict precedence between tasks, instead using release times and deadlines to give an outline of the order of activities.

Even though the time spent 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 (Krüger and Scholl, 2010). H'Mida and Lopez (2013) 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. (2017), H'Mida and Lopez (2013), Adhau et al. (2013), and Fauske (2015).

2.7 Solution methods

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 (2017), Tavana et al. (2014), Afruzi et al. (2014), Van Peteghem and Vanhoucke (2010), Afshar-Nadjafi and Majlesi (2014), and Debels and Vanhoucke (2007) 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 (2007), Zamani (2011), Sprecher (2002) and Rihm and Trautmann (2014) all decompose their projects into subparts and iteratively concatenate the solutions. Trautmann and Schwindt (2005) decompose a short-term batch production problem into batching and scheduling, where batching converts the primary requirements for products into individual batches, and scheduling allocates the batches to limited resources. Tian et al. (2014) decompose a project into a series of time windows. The decomposition approaches in these papers prove to outperform exact methods.

2.8 Our contribution

The standard RCPSP is a well known and studied problem. However, it is in many ways overly simplistic to be of practical use. For this reason, researchers have added various extensions to make the model more applicable to real life problems. Of the papers reviewed here, a few tailored their models to specific applications. Pollack-Johnson and Liberatore (2006) for example, relate quality to project scheduling in the construction industry, Gacias et al. (2010) present different methods for solving parallel machine scheduling problems including setup times, Al-Anzi et al. (2010) consider staffing of software engineers with different skill levels, and Fauske (2015) models a high intensity military operation. Most of the remaining papers address different RCPSP extensions, but to more generic project scheduling applications.

The problem studied in this report is a specialized case of the general RCPSP, specifically applicable to military operation planning. For this reason, it includes a distinctive combination of aspects that give rise to a high degree of complexity (Fauske, 2015). These aspects include resource hierarchy, multiple locations, resource skills and capacity, and in the case of a military peacekeeping operation, a distinctive objective function. Table 2.2 provides a comparative overview of the PTTP and the most relevant literature reviewed in this chapter. Aspects consistent with our problem are highlighted in green. Based on this comparison and online searches, we do not believe that there exists RCPSP optimization models with similar aspects as the ones presented in this thesis.

The most obvious difference between our models and existing models are their objective functions, which look to maximizes the total value of completed tasks. Value differs from quality as it is described in Tavana et al. (2014), because value is a function of tasks and skill level, and quality is only dependent on tasks. This is also the case with the papers that seek to minimize cost or maximize NPV; cost and profit are not defined to be dependent on skill level (Santos and Tereso, 2011; Heimerl and Kolisch, 2010; Waligóra, 2014; Schutt et al., 2012; Gacias et al., 2010). The quality quantification described by Pollack-Johnson and Liberatore (2006) is comparable to value as it is defined in our models. It is worth noting though, that because they intend all tasks to be completed, the overall intention of their model is to decide what skill level to complete tasks at, subject to time and cost. The goal with our models is to provide information on which subset of tasks to complete at what time, with which resources and what skill capacity.

Multiple locations is another aspect of the RCPSP there is relatively limited research on, that we include in our models. Of the papers reviewed in this chapter, only Laurent et al. (2017) and Fauske (2015) extend the RCPSP to include multiple locations. In the paper by Laurent et al. (2017), 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 (2015) and our problem 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 our problem.

Compared to the model outlined in the project report by Chaudry and Vermedal (2017), the models studied in this thesis regard a longer project planning period and are more encompassing, including additional task characteristic, and rest period extensions. Task characteristics are extended upon because military operation tasks are rarely self-contained, and are often connected to other tasks in more ways then can be described by precedence alone. The additional characteristics regard tasks that have to be completed right after each other, or pairs of tasks where if one is completed, then the other has to be completed also. Apart from Liu and Wang (2011), who include interdependent pairs of tasks, these distinctive connections or relationships and extensions are not considered by the other papers reviewed in this chapter. Furthermore we extend our models to consider predefined and non predefined rest periods similar to Drexl et al. (2000), but subjective to standard practices in military operations.

One of our two models takes a decomposition solution approach to solve the PTTP. Although a decomposition approach is promising, there is relatively limited research on the application of it to the RCPSP, compared to for example metaheuristic approaches (Sprecher, 2002; Rihm and Trautmann, 2014). Especially regarding military operation planning, a decomposition approach is a novel solution method.

Table 2.2: Comparison of the PTTP and existing literature.

Paper	**Objective function**	**Divisible tasks**	**Time**	**Precedence**	**Multi-mode/skill**	**Multiple locations**	**Solution method**
Standard RCPSP	min makespan	no	discrete	yes	n/a	no	n/a
Our problem, PTTP	**max value**	**yes**	**continuous**	**yes**	**skill**	**yes**	**decomposition and branch & bound**
Fauske (2015)	min makespan	no	discrete	yes	skill	yes	branch & cut
Fauske (2017)	min makespan	no	discrete	yes	skill	yes	genetic algorithm
Pollack-Johnson and Liberatore (2006)	max quality	yes	discrete	no	mode	no	goal programming
Ranjbar et al. (2009)	min makespan	yes	discrete	yes	mode	no	scatter search and path relinking
Tavana et al. (2014)	multiobjective	yes	discrete	yes	mode	no	evolutionary algorithm
Liu and Wang (2011)	max NPV	no	discrete	yes	n/a	no	tabu search algorithm
Schutt et al. (2012)	max NPV	no	continuous	yes	n/a	no	lazy clause generation
Waligóra (2014)	max NPV	yes	continuous	yes	n/a	no	tabu search algorithm
Afruzi et al. (2014)	multiobjective	yes	continuous	yes	mode	no	evolutionary algorithm
Gacias et al. (2010)	min makespan	yes	continuous	yes	mode	no	branch & bound and climbing discrepancy search
Van Peteghem and Vanhoucke (2010)	min makespan	yes	continuous	yes	mode	no	genetic algorithm
Moukrim et al. (2015)	min makespan	yes	continuous	yes	mode	no	branch & bound
Afshar-Nadjafi and Majlesi (2014)	min makespan	yes	discrete	yes	no	no	genetic algorithm
Laurent et al. (2017)	min makespan	no	discrete	yes	n/a	yes	metaheuristic (approx. methods)
Bianco et al. (1998)	min makespan	yes	continuous	yes	mode	no	heuristics
Wang and Zheng (2017)	min makespan	yes	continuous	yes	mode	no	knowledge-guided fruit-fly algorithm
Heimerl and Kolisch (2010)	min cost	no	discrete	no	skill	no	generic MIP
Al-Anzi et al. (2010)	min makespan	yes	discrete	yes	skill	no	LP aprroximation
Myszkowski et al. (2015)	min cost and makespan	no	continuous	yes	skill	no	ant colony heuristic
Bellenguez and Néron (2005)	min makespan	no	continuous	yes	skill	no	branch & bound and heuristics
Sprecher (2002)	min makespan	no	discrete	yes	n/a	no	decomposition
Zamani (2011)	min makespan	no	discrete	yes	n/a	no	decomppsition and genetic algorithm

Chapter 3

Problem Description

The PTTP presented in this report is a variant of the RCPSP. The decisions to be made in the planning of a peacekeeping operation concern which tasks resources are to perform at any given time. Resources consist of army resources and supporting airborne resources. Tasks can for example be *road patrol*, *securing an area*, or of the kind given in Table 3.1. 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 a troops-to-tasks analysis in a peacekeeping operation will often be to maximize the total realized value, given the resources and time available.

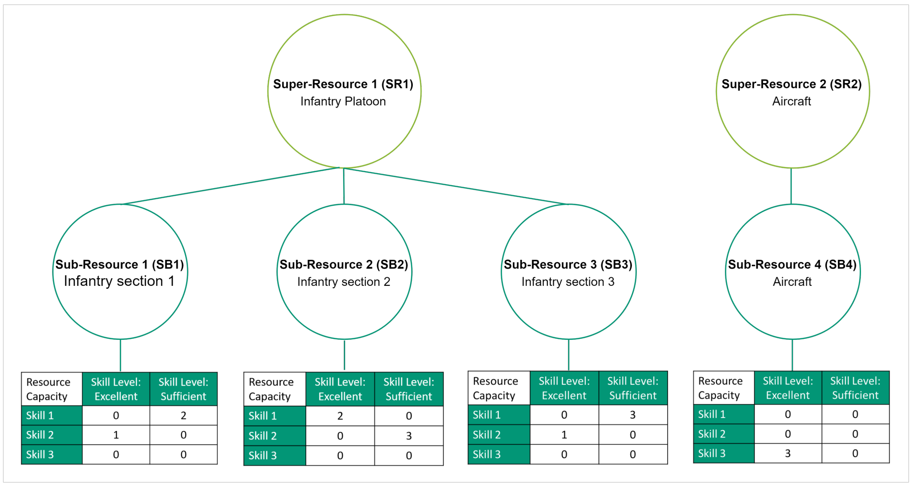
Figure 3.1: Securing a building during a training session in Lithuania.
Photo: The Norwegian Armed Forces.

Figure 3.2: Soldier searching an area for landmines during an operation in Iraq.
Photo: The Norwegian Armed Forces.

In our problem, an active army battlegroup is assigned to a given area at all times. An area consists of several locations, and at each of these locations, there are different tasks to be completed. One of these locations is base camp, where the army troops 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 time between them has to be considered. Locations can be visited multiple times a day.

The army resources in this problem are the different units in a battlegroup. The units 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 army resources. Army resources can be 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 aircraft is both a super-resource and a sub-resource.

Sub-resources have sets of skills, and for each skill, each 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 may be of different levels, such as sufficient or excellent, depending on the amount of training and the type of equipment the resources have, regarding a particular skill. All resources are renewable, meaning they do not lose capacity or skill by completing a task. Figure 3.3 presents examples of hierarchies for both an army resource and a supporting resource, as well as the skill capacities of their respective sub-resources.



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	3
Skill 2	1	0
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 3.3: Example of two hierarchies and their sub-resources' skill capacities.

A peacekeeping operation consists of a set of unique tasks of varying importance, and hence value. It is generally not possible to complete all tasks because of the limited amount of time and resources. Resources are to be assigned to these tasks. 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.

Table 3.1: 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

A task has to be completed in a continuous, uninterrupted fashion by the same resources, meaning that once a task has started, it is not possible to stop and then start it again, i.e. preemption is not permitted. The time window in which a task can be completed is dependant on the task. Moreover, some tasks have multiple time windows, meaning that they have several possible start and end times. For example, a task can have a time window between 08:00–10:00 on day 1, and another between 12:00–16:00 on day 3. However, because preemption is not permitted, the task can only be completed during one of them.

Most tasks have to be completed within regular daytime working hours because of the tasks' nature. In military peacekeeping operations, a regular working day is 16 hours long. 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 grey area of the Venn diagram in Figure 3.4 illustrates the exclusiveness and divisibility characteristics a single task can have.

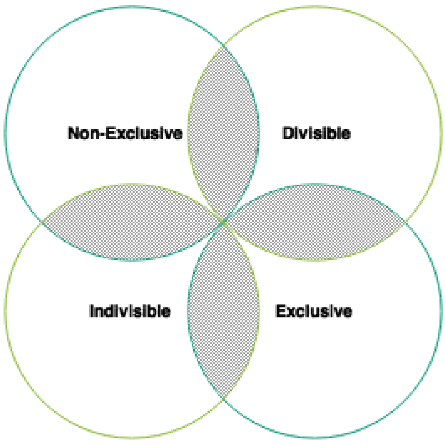


Figure 3.4: Task classification of exclusiveness and divisibility.

To ensure safety and order at base camp, *camp security* is a mandatory task, and is the only task that does not require a certain skill from assigned resources. It is also the only task that requires one entire super-resource to complete it, rather than one or more sub-resources. The super-resource assigned to camp security has to be an army resource and there must be someone assigned to the task at all times. Camp security is an exclusive task, hence the sub-resources belonging to the assigned super-resource cannot be assigned to any other tasks. A super-resource is assigned to security for a continuous period lasting a given number of days or weeks, before switching with another super-resource.

As mentioned previously, the value of completing a task depends on the task itself, and the skill level of the resource or resources undertaking the task. Important tasks or a higher skill level, obtain a higher value compared to less important tasks or a lower skill level. The objective of solving this scheduling problem is to realize as much value as possible, given the resources and time available. The effective outcome of solving this problem is deciding which resources are assigned to which tasks at what time.

Chapter 4

Mathematical Model

This chapter presents a compact mathematical formulation of the PTTP described in Chapter 3. Section 4.1 gives an account of the modelling assumptions, and Section 4.2 presents all sets, parameters, and variables used in the model. A step-by-step description of the mathematical formulation follows in Section 4.3.

4.1 Modelling assumptions

This section provides an overview of the assumptions made for the mathematical model presented in Section 4.3. The assumptions are largely based on the nature of peacekeeping operations and made with the intent to formulate a linear model.

Deterministic model

It is assumed that the PTTP is a deterministic problem. In reality, the duration of a task, travel time of a super-resource, and resource capacity, all have an element of uncertainty related to them. Travel times, for example, are in our model assumed to be constant no matter the conditions between locations, but in reality travel times can be affected by road conditions, weather, and human and mechanical errors etc. Furthermore, sub-resources' capacity might vary, based on their mood or health, and the availability of resources could be uncertain, as damage to personnel and equipment is likely to occur. These factors, in addition to other uncertainties, can affect the duration of a task. Nevertheless, given the disciplined nature of military operations, these uncertain elements are ignored in our model because they are assumed to be minuscule and have a negligible effect on the final outcome of the operations.

Problem structure

The PTTP can be interpreted as a two-tier problem with movement of super-resources between locations on one level and task assignment to resources on the other. The *movement of super-resources between locations* can be regarded as a network flow problem, where the task scheduling at each location restricts the time windows for travel. *Task assignment to resources* addresses the matching of task requirement with resource capacity and skill. The mathematical model in Section 4.3 is structured according to this two-tier interpretation of the PTTP.

Task value

Tasks can either be completed or not completed, they cannot be partly completed. In our model, completed tasks generate a certain value depending on the skill level they are completed at. For some tasks, such as *training local troops*, there might be a large difference in the value generated with a high skill level compared to a low skill level, while for other tasks, such as *setting up medical quarters*, the value would be almost the same. Task value is not contingent on cost or any other quantifiable measurements, but based solely on judgment. An important assumption for this model is therefore that this judgment is sound.

All tasks require certain skill capacities. Our model awards solutions where, for each completed task, the summation of these capacities are at a higher skill level. For tasks that require more than one skill, we assume that it is the aggregate of capacities at different skill levels that is decisive for task value. For example, if a task provides a value of three if completed at an excellent skill level, and a value of two if completed at sufficient level, then completing it 50% excellently and 50% sufficiently will result in a value of 2.5. The alternative to calculating task value this way, would be to determine value for each skill, at each skill level, for each task. Given that value is a difficult measurement to quantify, having to determine a larger set of values would introduce a larger amount of uncertainty to the model. Therefore, the assumption of task value depending on the aggregate of capacities at different skill levels is considered a reasonable simplification.

Divisible tasks and processing time

The processing time of a task is assumed constant if the task is indivisible, and negatively correlated to its assigned capacity if it is divisible. This is due to the varying nature of different tasks. A task such as *search and seizure* might for example only take half the normal processing time if it is assigned twice as much resource capacity, whilst *escort of VIP* will take a fixed amount of time no matter how much extra capacity is assigned to it.

The processing time for divisible tasks is assumed to be a linear function of assigned capacity as illustrated in Figure 4.1. To ensure that the model does not assign an exceeding amount of capacity to divisible tasks such that the tasks' processing times become minuscule, we assume that there is a lower limit to the time it takes to complete a task. This limit and its binding capacity is given for each divisible task. For indivisible tasks, processing time is set to the maximum processing time. Consequently, indivisible tasks will never be assigned extra capacity. See Appendix A for a more detailed account of the linear interpretations of processing time. The skill level a task is completed at is assumed to have no effect on the processing time.

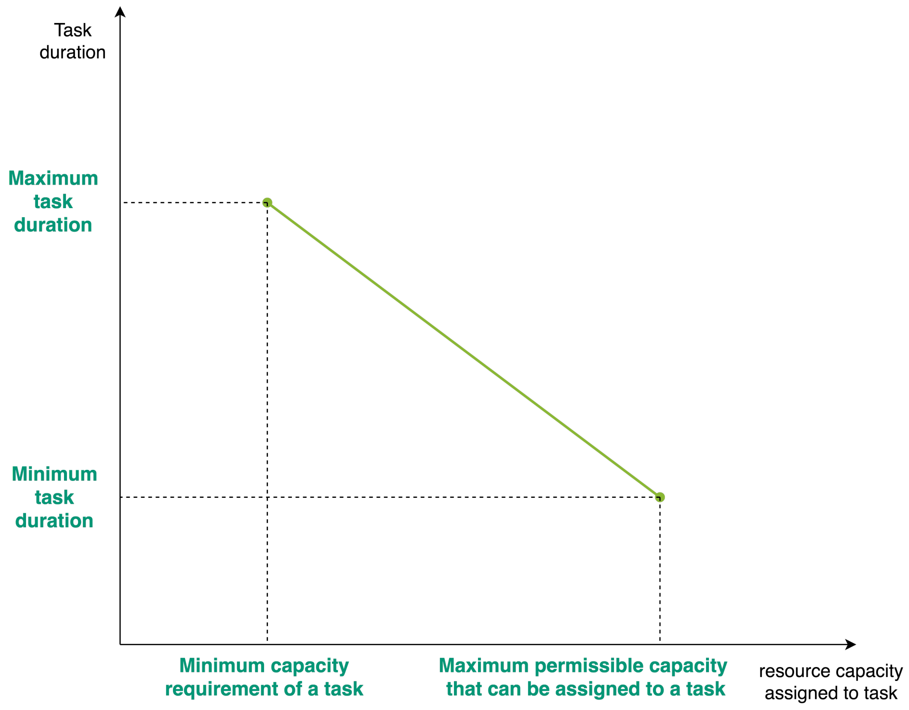


Figure 4.1: Illustration of the duration of divisible tasks.

Precedence

Although there exists four types of precedence, we only consider finish-to-start precedence in our model. A finish-to-start precedence means that one task cannot start until another task is finished. It is not limited to pairs of tasks; a single task might have to wait until several other preceding tasks are finished, or several tasks might have to wait until the same preceding task is finished. For example, it is not possible to undertake any task at a certain location until the task of *securing location* is complete. Chains of finish-to-start precedence relations are also possible, for example, if task A precedes B, and task B precedes task C. Henceforward, the term precedence implies finish-to-start precedence, unless stated otherwise.

Connected tasks

In an operation, some tasks consist of multiple subtasks, where each subtask is modelled as a normal task. Subtasks related to the same parent task are referred to as connected tasks, as illustrated in Figure 4.2. If one subtask is completed, then all its connected tasks have to be completed. Connected tasks can be undertaken simultaneously or have exclusive or precedence restrictions, and can have equal or different time windows. However, they are assumed to require skills and capacities such that they can be completed by the same resources. We further assume that connected tasks require that the same army resources complete them, but that this does not apply to supporting resources. The idea behind modelling parent and subtasks in this manner instead of modelling the parent task as one single large task, is to allow for distinct breaks in the completion of the parent task. Examples of connected task, are multiple *training local forces* tasks, all parts of one training program. All or no parts of the program have to be completed, and some of the training exercises may require others to be completed beforehand, creating precedence relations.

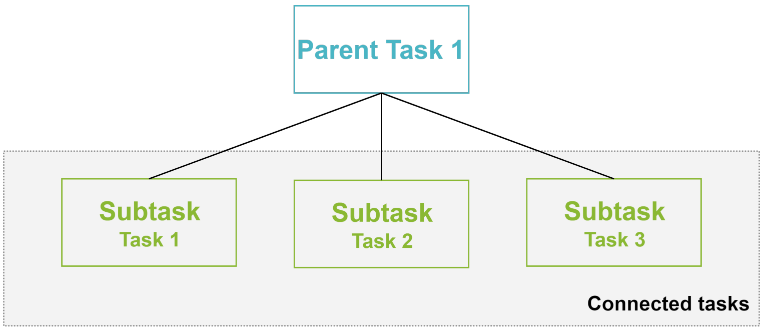


Figure 4.2: Illustration of parent, sub-, and connected tasks.

There are cases with one-way connections as well. This occurs when all resources assigned to a task must also be assigned to a second task, but may be joined by other resources. Then the first task is connected to the second, but not the other way around. An example of a one-way connection is provided in the next section under "Long and rest tasks".

Long and rest tasks

Tasks such as *training local forces*, are an example of tasks that are both connected and long. It usually spans over multiple days at a location some distance away from base camp, making it convenient to stay over night. The time off resources are entitled to after completing long tasks are modelled as tasks themselves, and referred to as rest tasks. The relation between long and rest tasks is further modelled by defining the rest tasks as exclusive tasks with zero value, and subject to precedence from a long task with a one-way connection to the rest task. As an entire super-resource will be away from camp when a long task is undertaken, even if only by a few of its sub-resources, it is assumed that all sub-resources are entitled to the same amount of time off afterwards. Supporting resources are assumed to not require time off after completing a lengthy task.

It is further assumed that if several super-resources work on the same long task, then their travel time to and from the long task's location and base camp is equal. Finally, we assume that army resources have to travel to base camp and complete a rest task immediately after completing a long task. In the mathematical model and for programming purposes, rest tasks are also defined as long tasks. In all other parts of this thesis, however, long task and rest task are treated as separate terms.

Direct start

One task having to be completed immediately after another is referred to as a direct start. A rest task, for example, has to start directly after a long task. As such, the tasks completed first will inherently have precedence over the tasks that starts directly after them. Not all task pairs that have a direct start have to be connected. The first task can be completed without the second one having to be completed. Even if both are completed, the resources assigned to the first task do not have to be assigned to the second one. In the event where both tasks must be completed if the first task is undertaken, but not necessary by the same resources, the total value of both tasks must be given to the second one, and the first task has to have zero value.

Task pairs that have a direct start can have different locations. This would mean that the same army resources can not be assigned to both tasks because their travel time between locations would violate the requirement of a direct start. If the pair happen to be connected, meaning that both tasks have to be completed, and by the same resources, then the travel time between the different task locations is taken into consideration. The second task then has to start a time period equivalent to the travel time between the locations, after the first task.

Security task

The model assumes that all army resources can undertake the camp security task regardless of their size, skill or skill capacity. In reality, this would mean that a super-resource consisting of very few sub-resources can complete the task as well as a larger super-resource with many sub-resources. Note that the number of sub-resources does not necessarily correspond with the size of the unit, nor with the capacities of the resources. It is also assumed that the planning period is of a length appropriate for one single super-resource have to be assigned to the security task for the entire planning period. The super-resource cannot be assigned to any other tasks during the planning period, even if the other tasks are located at base camp.

Sleep task

Army resources require a period of continuous free time every day. This free time is modelled as obligatory sleep tasks with no value. The time period of the sleep tasks are assumed to be concurrent for all army resources. A single day therefore includes a continuous work period and a continuous sleep period that are identical for all army resources. Free time for supporting resources is not considered in this model because the operation staff for the aircraft and helicopters rotate their shifts so that they get the required rest periods without the machinery becoming unavailable. All tasks that have a duration time that make them lengthy enough to include one or more nights, are assumed to incorporate sleep time in their duration. Therefore, resources undertaking long tasks are exempt from sleep tasks during the completion time of the long task.

Traveling at night

Traveling at night, during sleep periods, is generally not permitted. Some long tasks lasting into the night may, however, finish in the middle of a sleep period. Due to the assumption that sleep is incorporated in the duration times of these tasks, it is deemed acceptable for resources to travel at night after completing a long task, to the location of its connected task, such as a rest task at base camp.

Long tasks can have a start time that is during the night. Resources assigned to such tasks can arrive at the task location at night, but they have to leave the previous location before the sleep period starts. Thus, a resource cannot undertake two tasks that are not connected and in separate locations during one night, as this would make it more complicated to ensure regular sleep is included in all long tasks.

Duplicate tasks and multiple time windows

In a realistic operation, some tasks might be possible to complete in different time windows. This is modelled with the use of task duplicates. For example, a tasks such as *escorting a VIP* might have to adhere to different requirements. The requirement could be to retrieve the VIP after 08:00 and arrive at the destination before 16:00, but due to security restrictions no travel can happen between 10:00 and 12:00. This is one task, with two possible time windows for completion: 08:00–10:00 and 12:00–16:00. It is modelled as two tasks, one task with time window 08:00–10:00 and a second task with time window 12:00–16:00. These tasks are referred to as duplicates. Combined, they form a group of duplicates, as they are part of the same unique task. Only one of the duplicates can be completed.

All tasks in a group of duplicates have the exact same properties and requirements for skills and capacities. They may, however, have distinct values if some time windows are deemed more favorable than others. In the example with the VIP, it may be desirable to complete the task as early as possible, making the duplicate task with the time window of 08:00–10:00 more valuable. This may be the case for tasks with only one continuous time window as well, and duplicates can therefore be used to assign different value for different completion times. Furthermore, duplicates can be used for tasks that are possible to undertake at several alternative locations, such as a training exercise.

Start and end locations

In reality, resources might have different start locations or be occupied with a lengthy task, making them unavailable for a certain time period when the planning period starts. We however assume that all resources start at base camp at time zero, and terminate at base camp at the end time of the planning period. We model this by introducing a dummy start and end location, and respectively forcing the travel times from and to them, to be equivalent to the travel times from and to base camp. The travel time between the dummy start and end locations, and base camp is set to zero. The dummy start and end locations do not have any tasks associated with them. From hereon out, the term locations will refer to physical locations, not including the dummy start and end location, unless otherwise specified.

Upper limits for the number of visits and resources per task

A super-resource can visit the same location more than once. Several visits to the same location might in some cases be unwanted, either due to conditions in the practical execution of the operation, or for modeling purposes. The possibility to impose an upper limit on the number of visits any super-resource can have to the same location is therefore included in the model. Multiple visits to one location is modeled as duplicate locations, as illustrated in Figure 4.3. The figure shows a possible travel route for a super-resource, including several visits to one location. In this example, a super-resource can visit a location a maximum of three times. The limit applies per super-resource, meaning that the travel route of super-resource SR2 is not affected by how many times a particular location has been visited by SR1. An upper limit is also assigned to the number of resources that can work on the same task because it is assumed impractical and unrealistic for a large number of resources to work on the same task. This does not apply to rest and sleep tasks.

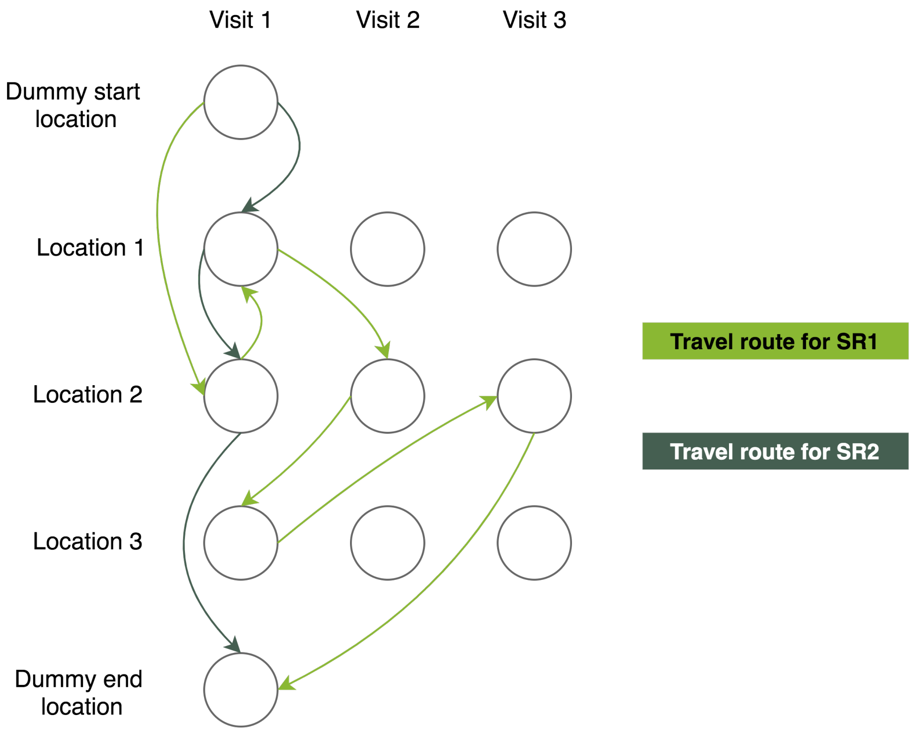


Figure 4.3: Illustration of two super-resources making multiple location visits.

Overview of task characteristics

Table 4.1 presents the various characteristics a task can have, and how those characteristics relate to one another. A bullet point means that the given combination is possible. For example, an exclusive task can be non-divisible or divisible. Combinations that are not possible are marked with an "x". A rest task cannot, for example, be divisible. The cells highlighted in green are combinations of characteristics that are binding. For example, if a task has a direct start after another task, then the other task naturally has to have precedence over the given task.

Table 4.1: Possible combinations of task characteristics.

then the task can be/have / If a task is/has	Located at base camp	Long	Rest	Sleep	Duplicates	Exclusive	Non-exclusive	Divisible	Non-divisible	Precedence over other task	Other task has precedence	Connected to other task	Direct start after other task	Value
Located at base camp	-	●	●	●	●	●	●	●	●	●	●	●	●	●
Long	●	-	●	x	●	●	●	●	●	●	●	●	●	●
Rest	●	●	-	x	●	●	x	x	●	●	●	●	●	x
Sleep	●	x	x	-	x	●	x	x	●	x	x	x	x	x
Duplicates	●	●	●	x	-	●	●	●	●	●	●	●	●	●
Exclusive	●	●	●	●	●	-	x	●	●	●	●	●	●	●
Non-exclusive	●	●	x	x	●	x	-	●	●	●	●	●	●	●
Divisible	●	●	x	x	●	●	●	-	x	●	●	●	●	●
Non-divisible	●	●	●	●	●	●	●	x	-	●	●	●	●	●
Precedence over other task	●	●	●	x	●	●	●	●	●	-	●	●	●	●
Other task has precedence	●	●	●	x	●	●	●	●	●	●	-	●	●	●
Connected to other task	●	●	●	x	●	●	●	●	●	●	●	-	●	●
Direct start after other task	●	●	●	x	●	●	●	●	●	●	●	●	-	●
Value	●	●	●	x	●	●	●	●	●	●	●	●	●	-

4.2 Definitions

Indices

g	Super-resource
k	Sub-resource
p	Task
i, j	Location
s	Skill
l	Skill level
m, n	Visit number of a location
d	Group of duplicate tasks

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

4.3 Optimization model

4.3.1 Objective function

$$\max \quad z = \sum_{p \in \mathcal{P}} \sum_{l \in \mathcal{L}} V_{pl} e_{pl} \tag{4.1}$$

The objective function (4.1) maximizes the total value achieved by completing tasks in the military peacekeeping operation. 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 uncompleted tasks, the proportions will be zero, ensuring that no value is added for these tasks.

4.3.2 Constraints and requirements

Super-resource network constraints

$$a_{gim} - \overline{T}(1 - \gamma_{kpm}) \leq t_p^{START} \qquad g \in \mathcal{G}, i \in \mathcal{I}, k \in \mathcal{K}_g, m \in \mathcal{N}_i, p \in \mathcal{P}_i^{\mathcal{LOC}} \tag{4.2}$$

$$b_{gim} + \overline{T}(1 - \gamma_{kpm}) \geq t_p^{END} \qquad g \in \mathcal{G}, i \in \mathcal{I}, k \in \mathcal{K}_g, m \in \mathcal{N}_i, p \in \mathcal{P}_i^{\mathcal{LOC}} \tag{4.3}$$

$$b_{gim} \geq t_p^{END} - \overline{T}(\theta_{gimp} + \sum_{k \in \mathcal{K}_g} \sum_{\substack{d' \in \mathcal{D}, \\ p' \in \mathcal{P}_{d'}}} \sum_{\substack{d^* \in \mathcal{D}, \\ p^* \in \mathcal{P}_{d^*} \cap \mathcal{P}^{\mathcal{LONG}}}} F_{d'd^*}^{DIR} F_{d'd^*}^{CON} o_{kpp'})$$
$$g \in \mathcal{G}^{\mathcal{ARMY}}, i \in \mathcal{I}, m \in \mathcal{N}_i, p \in \mathcal{P}^{\mathcal{SLEEP}} \tag{4.4}$$

$$b_{gim} \leq t_p^{START} + \overline{T}(1 - \theta_{gimp} + \sum_{k \in \mathcal{K}_g} \sum_{\substack{d' \in \mathcal{D}, \\ p' \in \mathcal{P}_{d'}}} \sum_{\substack{d^* \in \mathcal{D}, \\ p^* \in \mathcal{P}_{d^*} \cap \mathcal{P}^{\mathcal{LONG}}}} F_{d'd^*}^{DIR} F_{d'd^*}^{CON} o_{kpp'})$$
$$g \in \mathcal{G}^{\mathcal{ARMY}}, i \in \mathcal{I}, m \in \mathcal{N}_i, p \in \mathcal{P}^{\mathcal{SLEEP}} \tag{4.5}$$

Constraints (4.2) and (4.3) 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 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 (4.4) and (4.5) ensure 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 (4.4) and (4.5) represent duplicate groups of only long tasks.

$$b_{gim} + T^{TRAVEL}_{gij} \leq a_{gjn} + M(1 - y^{TRAVEL}_{gimjn}) \qquad g \in \mathcal{G}, i, j \in \mathcal{I}, m \in \mathcal{N}_i, n \in \mathcal{N}_j, i \neq j \tag{4.6}$$

$$\sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{N}_i} \sum_{j \in \mathcal{I}} \sum_{n \in \mathcal{N}_j} T^{TRAVEL}_{gij} y^{TRAVEL}_{gimjn} + \sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{N}_i} (b_{gim} - a_{gim}) = \overline{T} \qquad g \in \mathcal{G} \tag{4.7}$$

Constraints (4.6) specify that if a super-resource travels directly between two locations, then it can only arrive at a location a period at least equal to the travel time T^{TRAVEL}_{gij} after it has left from its previous location. Big M for constraints (4.6) 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^{TRAVEL}_{gij} value. Constraints (4.7) ensure that, at all times, a super-resource is either traveling or at a location.

$$a_{gim} \leq \overline{T} y^{LOC}_{gim} \qquad g \in \mathcal{G}, i \in I, m \in \mathcal{N}_i \tag{4.8}$$

$$b_{gim} \leq \overline{T} y^{LOC}_{gim} \qquad g \in \mathcal{G}, i \in I, m \in \mathcal{N}_i \tag{4.9}$$

$$y^{LOC}_{gim} \leq \sum_{k \in \mathcal{K}_g} \sum_{p \in \mathcal{P}^{\mathcal{LOC}}_i} \gamma_{kpm} \qquad g \in \mathcal{G}, i \in \mathcal{I} \backslash (1, |I|), m \in \mathcal{N}_i \tag{4.10}$$

$$x_{kp} \leq \sum_{m \in \mathcal{N}_i} y^{LOC}_{gim} \qquad g \in \mathcal{G}, i \in \mathcal{I}, k \in \mathcal{K}_g, p \in \mathcal{P}^{\mathcal{LOC}}_i \tag{4.11}$$

$$\sum_{m \in \mathcal{N}_i} \gamma_{kpm} = x_{kp} \qquad i \in \mathcal{I}, k \in \mathcal{K}, p \in \mathcal{P}^{\mathcal{LOC}}_i \tag{4.12}$$

Constraints (4.8) and (4.9) assert that a super-resource cannot arrive or leave a location it is not visiting. Constraints (4.10) 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 (4.11) state that if a sub-resource is assigned to a task, then it has to visit the task's location at least once. Constraints (4.12) couple the variables γ and x.

$$\sum_{i\in\mathcal{I}}\sum_{m\in\mathcal{N}_i} y^{TRAVEL}_{gimjn} = y^{LOC}_{gjn} \qquad g\in\mathcal{G}, j\in\mathcal{I}\backslash 1, n\in\mathcal{N}_j, i\neq j \tag{4.13}$$

$$\sum_{j\in\mathcal{I}}\sum_{n\in\mathcal{N}_j} y^{TRAVEL}_{gimjn} = y^{LOC}_{gim} \qquad g\in\mathcal{G}, i\in\mathcal{I}\backslash|I|, m\in\mathcal{N}_i, i\neq j \tag{4.14}$$

Constraints (4.13) and (4.14) represent route continuity between locations for the super-resources.

$$y^{LOC}_{gim} \geq y^{LOC}_{gi(m+1)} \qquad g\in\mathcal{G}, i\in\mathcal{I}, m\in\mathcal{N}_i\backslash|\mathcal{N}_i| \tag{4.15}$$

To avoid symmetric solutions, constraints (4.15) 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 location

$$a_{g11} = 0 \qquad g\in\mathcal{G} \tag{4.16}$$

$$a_{g|I|1} \geq a_{gim} \qquad g\in\mathcal{G}, i\in\mathcal{I}, m\in\mathcal{N}_i \tag{4.17}$$

Constraints (4.16) and (4.17) handle start and end locations for all super-resources. Constraints (4.16) specify that all super-resources start at location 1 at time zero, and (4.17) state that all resources ultimately must arrive at location $|I|$. Constraints (4.16) and (4.17) also ensure no super-resources can travel to the start location or from the end location.

Resource capacity constraints

$$\sum_{s\in\mathcal{S}} C^{REQ}_{ps} e_{pl} = \sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}} w_{kpsl} \qquad p\in\mathcal{P}, l\in\mathcal{L} \tag{4.18}$$

$$\sum_{k\in\mathcal{K}}\sum_{l\in\mathcal{L}} w_{kpsl} = C^{REQ}_{ps} q_p \qquad p\in\mathcal{P}, s\in\mathcal{S} \tag{4.19}$$

$$w_{kpsl} \leq C^{RES}_{ksl} x_{kp} \qquad k \in \mathcal{K}, p \in \mathcal{P}, s \in \mathcal{S}, l \in \mathcal{L} \tag{4.20}$$

Constraints (4.18) provide the proportion of a task that is carried out with each skill level. Constraints (4.19) 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 (4.20) state that a resource's contribution to a task cannot be more than its own capacity.

$$x_{kp} \leq \sum_{s \in \mathcal{S}} \sum_{l \in \mathcal{L}} w_{kpsl} \qquad k \in \mathcal{K}, p \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{DIV}} \tag{4.21}$$

$$x_{kp} \leq \sum_{s \in \mathcal{S}} \sum_{l \in \mathcal{L}} C^{RES}_{ksl} C^{REQ}_{ps} \qquad k \in \mathcal{K}, p \in \mathcal{P}^{\mathcal{DIV}} \tag{4.22}$$

Constraints (4.21) and (4.22) forbid the assignment of resources to tasks which they cannot undertake. Constraints (4.21) 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 (4.19) and (4.20) do not provide any restrictions to the skill requirement for resource k to be assigned to tasks p. Constraints (4.22) 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

$$\sum_{k \in \mathcal{K}} x_{kp} \geq q_p \qquad p \in \mathcal{P} \tag{4.23}$$

$$\sum_{k \in \mathcal{K}} x_{kp} \leq \overline{R} q_p \qquad p \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{SLEEP}} \tag{4.24}$$

$$\sum_{p \in \mathcal{P}^{\mathcal{DUP}}_d} q_p \leq 1 \qquad d \in \mathcal{D} \tag{4.25}$$

Constraints (4.23) and (4.24) 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 (4.24) also limit the number of resources that can work on a single task. Sleep tasks are not subject to this limit. Constraints (4.25) 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.

$$\sum_{g \in \mathcal{G}^{\mathcal{ARMY}}} u_g = 1 \tag{4.26}$$

$$x_{kp} \leq 1 - u_g \qquad g \in \mathcal{G}^{\mathcal{ARMY}}, k \in \mathcal{K}_g, p \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{SLEEP}} \tag{4.27}$$

Constraint (4.26) ensures that one army super-resource is assigned to the security post, and constraints (4.27) 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

$$t_p^{END} - \overline{T}(2 - (x_{kp} + x_{kp'})) \leq t_{p'}^{START} + \overline{T}(1 - \delta_{pp'}) \\ k \in \mathcal{K}, p \in \mathcal{P}, p' \in \mathcal{P}^{\mathcal{E}}, p \neq p' \tag{4.28}$$

$$t_{p'}^{END} - \overline{T}(2 - (x_{kp} + x_{kp'})) \leq t_p^{START} + \overline{T}\delta_{pp'} \\ k \in \mathcal{K}, p \in \mathcal{P}, p' \in \mathcal{P}^{\mathcal{E}}, p \neq p' \tag{4.29}$$

Constraints (4.28) and (4.29) 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

$$x_{kp} + \sum_{p' \in \mathcal{P}^{\mathcal{LONG}}} o_{kpp'} \geq 1 \qquad g \in \mathcal{G}^{\mathcal{ARMY}}, k \in \mathcal{K}_g, p \in \mathcal{P}^{\mathcal{SLEEP}} \tag{4.30}$$

Constraints (4.30) 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.

$$t_p^{START} - M(1 - o_{kpp'}) \leq t_{p'}^{END} \qquad k \in \mathcal{K}, p \in \mathcal{P}^{\mathcal{SLEEP}}, p' \in \mathcal{P}^{\mathcal{LONG}} \tag{4.31}$$

$$t_{p'}^{START} - M(1 - o_{kpp'}) \leq t_p^{END} \qquad k \in \mathcal{K}, p \in \mathcal{P}^{\mathcal{SLEEP}}, p' \in \mathcal{P}^{\mathcal{LONG}} \tag{4.32}$$

$$o_{kpp'} \leq \sum_{k' \in \mathcal{K}_g} x_{k'p'} \qquad g \in \mathcal{G}^{\mathcal{ARMY}}, k \in \mathcal{K}_g, p \in \mathcal{P}^{\mathcal{SLEEP}}, p' \in \mathcal{P}^{\mathcal{LONG}} \tag{4.33}$$

Constraints (4.31)–(4.33) 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 (4.31) and (4.32) 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 (4.33) 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 (4.31) and (4.32) equals the duration of the shortest sleep task subtracted from the end time $\overline{T}$.

Precedence constraints

$$\sum_{p' \in \mathcal{P}_{d'}^{\mathcal{DUP}}} q_{p'} \leq \sum_{p \in \mathcal{P}_d^{\mathcal{DUP}}} q_p \qquad d, d' \in \mathcal{D}, d \neq d', F_{dd'}^{PREC} = 1 \tag{4.34}$$

$$t_{p'}^{START} + \overline{T}(1 - q_{p'}) \geq t_p^{END}$$
$$d, d' \in \mathcal{D}, p \in \mathcal{P}_d^{\mathcal{DUP}}, p' \in \mathcal{P}_{d'}^{\mathcal{DUP}}, d \neq d', F_{dd'}^{PREC} = 1 \tag{4.35}$$

Constraints (4.34) 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 (4.35) set the start time for task p' after the end time of task p. Constraints (4.35) are not to be binding unless task p' is completed, hence the term $\overline{T}(1 - q_{p'})$.

Connected task and direct start constraints

$$\sum_{p \in \mathcal{P}_d^{\mathcal{DUP}}} x_{kp} \leq \sum_{p' \in \mathcal{P}_{d'}^{\mathcal{DUP}}} x_{kp'}$$
$$g \in \mathcal{G}^{\mathcal{ARMY}}, k \in \mathcal{K}_g, d, d' \in \mathcal{D}, d \neq d', F_{dd'}^{CON} = 1 \tag{4.36}$$

$$x_{kp} = x_{k'p} \qquad g \in \mathcal{G}^{\mathcal{ARMY}}, k, k' \in \mathcal{K}_g, p \in \mathcal{P}^{\mathcal{REST}} \tag{4.37}$$

Constraints (4.36) deal with connected tasks, ensuring that that if task p is connected to task p', i.e. $F^{CON}_{dd'}$ 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, $F^{CON}_{dd'}$ and $F^{CON}_{d'd}$ equal 1. When one of the connected tasks is rest task p', all resources assigned to task p must be assigned to p', but the opposite is not true, and more resources may be assigned to p' than p. Constraints (4.37) forces all or no sub-resources in a super-resource to be assigned to rest task p if it is a rest task.

$$t^{END}_p + F^{CON}_{dd'} T^{TRAVEL}_{gij} \geq t^{START}_{p'} \qquad g \in \mathcal{G}^{\mathcal{ARMY}}, i, j \in \mathcal{I}$$
$$d, d' \in \mathcal{D}, p \in (\mathcal{P}^{\mathcal{DUP}}_d \cap \mathcal{P}^{\mathcal{LOC}}_i), p' \in (\mathcal{P}^{\mathcal{DUP}}_{d'} \cap \mathcal{P}^{\mathcal{LOC}}_j), d \neq d', F^{DIR}_{dd'} = 1 \quad (4.38)$$

For some tasks, there is a requirement that another task starts directly after the first task is completed. Constraints (4.38) ensure that these requirements are fulfilled, taking into account travel time between the tasks' locations.

Time scheduling constraints

$$t^{START}_p \geq T^{RL}_p q_p \qquad p \in \mathcal{P} \quad (4.39)$$

$$t^{END}_p \leq T^{DDL}_p q_p \qquad p \in \mathcal{P} \quad (4.40)$$

If a task is realized, constraints (4.39) make sure task p starts after its release time R_p, and constraints (4.40) ensure its completion before its deadline D_p.

$$t^{END}_p \leq t^{START}_p + T^{TASK}_p q_p \qquad p \in \mathcal{P} \quad (4.41)$$

$$t^{END}_p \geq t^{START}_p + T^{MIN}_p T^{TASK}_p q_p \qquad p \in \mathcal{P} \quad (4.42)$$

$$t^{END}_p \geq t^{START}_p + T^{TASK}_p \left(\frac{C^{MAX}_p - T^{MIN}_p}{C^{MAX}_p - 1} q_p \right)$$
$$+ T^{TASK}_p \left(\frac{T^{MIN}_p - 1}{\sum_{s \in \mathcal{S}} C^{REQ}_{ps} (C^{MAX}_p - 1)} \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}} \sum_{l \in \mathcal{L}} C^{RES}_{ksl} H_{ps} x_{kp} \right) \qquad p \in \mathcal{P}^{\mathcal{DIV}}$$
$$(4.43)$$

Constraints (4.41)–(4.43) handle task duration and ensure that tasks are completed in a continuous fashion. Constraints (4.41) make sure that completing a task does not take longer than necessary, while constraints (4.42) and (4.43) 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 (4.43) set the end time of task p according to this. A deduction of constraints (4.43) is presented in Appendix A.

Non-negativity constraints

$$w_{kpsl} \geq 0 \quad k \in \mathcal{K}, p \in \mathcal{P}, s \in \mathcal{S}, l \in \mathcal{L} \tag{4.44}$$

$$e_{pl} \geq 0 \quad p \in \mathcal{P}, l \in \mathcal{L} \tag{4.45}$$

$$t_p^{START} \geq 0 \quad p \in \mathcal{P} \tag{4.46}$$

$$t_p^{END} \geq 0 \quad p \in \mathcal{P} \tag{4.47}$$

$$a_{gim} \geq 0 \quad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i \tag{4.48}$$

$$b_{gim} \geq 0 \quad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i \tag{4.49}$$

Binary requirements

$$x_{kp} \in \{0,1\} \quad k \in \mathcal{K}, p \in \mathcal{P} \tag{4.50}$$

$$q_p \in \{0,1\} \quad p \in \mathcal{P} \tag{4.51}$$

$$u_g \in \{0,1\} \quad g \in \mathcal{G} \tag{4.52}$$

$$y_{gim}^{LOC} \in \{0,1\} \quad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i \tag{4.53}$$

$$y_{gimjn}^{TRAVEL} \in \{0,1\} \quad g \in \mathcal{G}, i,j \in \mathcal{I}, m \in \mathcal{N}_i, n \in \mathcal{N}_j \tag{4.54}$$

$$o_{kpp'} \in \{0,1\} \qquad k \in \mathcal{K}, p, p' \in \mathcal{P} \tag{4.55}$$

$$\delta_{pp'} \in \{0,1\} \qquad p, p' \in \mathcal{P} \tag{4.56}$$

$$\gamma_{kpm} \in \{0,1\} \qquad i \in \mathcal{I}, k \in \mathcal{K}, m \in \mathcal{N}_i, p \in \mathcal{P}_i^{\mathcal{LOC}} \tag{4.57}$$

$$\theta_{gimp} \in \{0,1\} \qquad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i, p \in \mathcal{P}^{\mathcal{SLEEP}} \tag{4.58}$$

Constraints (4.44)–(4.49) ensure non-negativity for continuous variables. Constraints (4.50)–(4.58) enforce binary values for all binary variables.

Chapter 5

Decomposition Solution Approach

This chapter presents an alternative mathematical model to solve the PTTP, where possible travel routes for the resources are predetermined, and the optimization model selects an optimal combination of routes. Because the route generation is removed from the compact model and carried out as a separate operation, this alternative model can be regarded as a decomposition of the compact model in Chapter 4, hence the title *Decomposition Solution Approach*. Section 5.1 presents the new definitions and the alternative mathematical formulation. Section 5.2 follows, offering a description of the three methods used to generate travel routes: One exact method, and two heuristic methods.

The motivation for testing a decomposition solution approach is twofold. Firstly, because the PTTP can be interpreted as a two-tier problem with movement of super-resources between locations on one level and task assignment to resources on the other, its structure makes it well suited to be decomposed accordingly. Secondly, the results attained by Debels and Vanhoucke (2007), Zamani (2011), Sprecher (2002), Rihm and Trautmann (2014), Trautmann and Schwindt (2005) and Tian et al. (2014) carrying out a decomposition approach, are strictly positive.

5.1 Decomposed Model

The essential difference between the compact model and the decomposed model is that in the decomposed model, the travel routes for resources are not a variable, but a parameter of which one route has to be selected for each resource for the given planning period. In mathematical terms, this means that the variables y_{gim}^{LOC} and y_{gimjn}^{TRAVEL} become parameters Y_{rgim}^{LOC} and Y_{rgimjn}^{TRAVEL}, and that the variable λ_{rg} is added to constraints 4.6–4.11. Furthermore, constraints 4.13–4.17, 4.53, and 4.54 are removed, and 5.7 and 5.12 added. The model is otherwise equivalent to the compact model presented in Chapter 4. For this reason, only the constraints that differ from the compact model are presented here. The entire decomposed model is given in Appendix B.

5.1.1 Definitions

Indices

g	Super-resource
k	Sub-resource
p	Task
i, j	Location
m, n	Visit number of a location
r	Travel route

Sets

$\mathcal{G}$	Set of super-resources
$\mathcal{K}_g$	Set of sub-resources that belong to super-resource g
$\mathcal{I}$	Set of locations including dummy locations 1 and $\lvert I \rvert$
$\mathcal{P}_i^{\mathcal{LOC}}$	Subset of tasks that are located at location i
$\mathcal{N}_i$	Set of possible visits to location i
$\mathcal{R}_g$	Set of possible routes for super-resource g

Parameters

T_{gij}^{TRAVEL}	Travel time for super-resource g between locations i and j
Y_{rgimjn}^{TRAVEL}	1 if for route r, super-resource g travels directly between its m^{th} visit at location i and n^{th} visit at location j, 0 otherwise
Y_{rgim}^{LOC}	1 if for route r, super-resource g visits location i for the m^{th} time, 0 otherwise
$\overline{T}$	End time of operation
M	Big M

Variables

x_{kp}	1 if resource k is assigned to task p, 0 otherwise
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
γ_{kpm}	1 if resource k completes task p on the m^{th} visit of the task's location, 0 otherwise
λ_{rg}	1 if super-resource g travels route r

5.1.2 Constraints and requirements

$$b_{gim} + T^{TRAVEL}_{gij} \leq a_{gjn} + M(1 - \sum_{r \in \mathcal{R}_g} \lambda_{rg} Y^{TRAVEL}_{rgimjn}) \qquad g \in \mathcal{G}, i, j \in \mathcal{I}, m \in \mathcal{N}_i, n \in \mathcal{N}_j, i \neq j \quad (5.1)$$

$$\sum_{r \in \mathcal{R}_g} \sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{N}_i} \sum_{j \in \mathcal{I}} \sum_{n \in \mathcal{N}_j} \lambda_{rg} T^{TRAVEL}_{gij} Y^{TRAVEL}_{rgimjn} + \sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{N}_i} (b_{gim} - a_{gim}) = \overline{T} \qquad g \in \mathcal{G} \quad (5.2)$$

Constraints (5.1) specify that if a super-resource travels directly between two locations, then it can only arrive at a location a period at least equal to the travel time T^{TRAVEL}_{gij} after it has left from its previous location. Big M for constraints (5.1) 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^{TRAVEL}_{gij} value. Constraints (5.2) ensure that, at all times, a super-resource is either traveling or at a location.

$$a_{gim} \leq \sum_{r \in \mathcal{R}_g} \lambda_{rg} \overline{T} Y^{LOC}_{rgim} \qquad g \in \mathcal{G}, i \in I, m \in \mathcal{N}_i \quad (5.3)$$

$$b_{gim} \leq \sum_{r \in \mathcal{R}_g} \lambda_{rg} \overline{T} Y^{LOC}_{rgim} \qquad g \in \mathcal{G}, i \in I, m \in \mathcal{N}_i \quad (5.4)$$

$$\sum_{r \in \mathcal{R}_g} \lambda_{rg} Y^{LOC}_{rgim} \leq \sum_{k \in \mathcal{K}_g} \sum_{p \in \mathcal{P}^{LOC}_i} \gamma_{kpm} \qquad g \in \mathcal{G}, i \in \mathcal{I} \backslash (1, |I|), m \in \mathcal{N}_i \quad (5.5)$$

$$x_{kp} \leq \sum_{m \in \mathcal{N}_i} \sum_{r \in \mathcal{R}_g} \lambda_{rg} Y^{LOC}_{rgim} \qquad g \in \mathcal{G}, i \in \mathcal{I}, k \in \mathcal{K}_g, p \in \mathcal{P}^{LOC}_i \quad (5.6)$$

Constraints (5.3) and (5.4) assert that a super-resource cannot arrive or leave a location it is not visiting. Constraints (5.5) 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 (5.6) state that if a sub-resource is assigned to a task, then it has to visit the task's location at least once.

$$\sum_{r \in \mathcal{R}_g} \lambda_{rg} = 1 \qquad g \in \mathcal{G} \tag{5.7}$$

Constraints (5.7) state that for all super-resources, exactly one route must be selected from the given set of possible routes.

Non-negativity constraints

$$a_{gim} \geq 0 \qquad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i \tag{5.8}$$

$$b_{gim} \geq 0 \qquad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i \tag{5.9}$$

Binary requirements

$$x_{kp} \in \{0, 1\} \qquad k \in \mathcal{K}, p \in \mathcal{P} \tag{5.10}$$

$$\gamma_{kpm} \in \{0, 1\} \qquad i \in \mathcal{I}, k \in \mathcal{K}, m \in \mathcal{N}_i, p \in \mathcal{P}_i^{\mathcal{LOC}} \tag{5.11}$$

$$\lambda_{rg} \in \{0, 1\} \qquad g \in \mathcal{G}, r \in \mathcal{R}_g \tag{5.12}$$

Constraints (5.8) and (5.9) ensure non-negativity for continuous variables. Constraints (5.10)–(5.12) enforce binary values for all binary variables.

5.2 Travel Route Generation

This section describes a labeling algorithm method used to generate feasible travel routes for all super-resources in a battlegroup. The algorithm utilizes partial routes, and extends them for all possible additional locations in multiple stages. The result is a set of possible travel routes for each super-resource which determine the parameters Y^{TRAVEL}_{rgimjn} and Y^{LOC}_{rgim} described in Section 5.1. Y^{TRAVEL}_{rgimjn} divides each travel route r of super-resource g into travel distances, from visit number m at location i to visit number n at location j. Y^{LOC}_{rgim} equals 1 for all visits m to locations i included in route r for super-resource g. The optimization model presented in Section 5.1 finds the optimal combination of routes for a battlegroup, to find the best solution value.

5.2.1 Label data

For each super-resource in the given battlegroup, the labeling algorithm calculates travel routes. For every location in the travel route, a *label* keeps track of the state of parameters such as the time used up to this point. The result is multiple possible travel routes, each unique for a certain super-resource.

A label L is written as $L = (i, t, c, l, r)$, and contains the following information:

- Current location, i. This is the last location visited for a complete or incomplete travel route.
- Minimum time spent from the beginning of the planning period, t, when the super-resource is ready to leave location i.
- The least amount of time that can elapse, c, since the super-resource's last visit to base camp.
- A binary notation, l, indicating whether or not the super-resource is visiting a location where it could be assigned to a long task. If so, l is $True$, and if not, l is $False$.
- Finally, the label contains the commenced travel route, r, consisting of a list of all locations the super-resource has visited, including the current visit to i. The locations are given in chronological order of travel, and a location is included again for each new visit.

For each super-resource, the initial label is $L_0 = (1, 0, 0, False, [1])$. L_0 represents a partial travel route where location 1, the dummy start location, is visited at time zero, and the super-resource is not able to be assigned to a long task. This is referred to as an $active$ $label$, as it can, and must, be extended with additional travel. When the current location i of a label is set to the end location, the label is no longer active. This is because resources cannot travel from the dummy end location to any other location, and hence the label cannot be further extended. A label that cannot be extended is referred to as a $finalized$ $label$. All labels for a super-resource have to be finalized before the algorithm moves on to generate routes for the next super-resource.

5.2.2 Extending a travel route

Active labels must be extended. A label being extended is removed from the set of active labels and referred to as the current label. If the current label is $L_{current} = (i, t, c, l, r)$, a new active label L_{new} is created for all viable locations apart from i, where each viable location j results in a valid travel route $r(L_{new})$. The pseudocode for label extension and data configuration of new labels is presented in Algorithm 1. The criteria for valid travel routes and the function GETROUTEEXTENSION are explained in Section 5.2.3. The minimum time for the new label, $t(L_{new})$, is calculated as $t(L_{current})$ plus the minimum amount of time the super-resource g must use until it is able to travel from j, $minTime$. $minTime$ is the aggregate of the travel time T_{ij}^{TRAVEL} from i to j, the duration time for a task p, and the waiting time if p is not released when g arrives at j. Hence, $t(L_{new})$ is always equal to or greater than $t(L_{current})$. $c(L_{new})$ equals zero if location j is the base camp location, otherwise it equals $c(L_{current})$ plus $minTime$. $l(L_{new})$ is $True$ if it is possible for the super-resource to undertake a long task at location j, and $r(L_{new})$ is equal to $r(L_{current})$, but with the addition of j as the last location visited.

Algorithm 1 Pseudocode describing label generation.

```
1: procedure GENERATELABELS(super-resource $g$, set $\mathcal{P}_j$ of solvable tasks for the
   super-resource at each location, set $\mathcal{I}$ of locations, travel time parameter $T_{gij}^{TRAVEL}$)
2:     $L_0 = (1, 0, 0, False, [1])$
3:     $\mathcal{L}^A \leftarrow L_0$
4:     $\mathcal{L}^F \leftarrow \emptyset$
5:     while Set $\mathcal{L}^A$ of active labels is not empty do
6:         $L^C \leftarrow$ first element of $\mathcal{L}^A$
7:         for locations $j \mid j \neq 1$ do
8:             Empty label $L^E$ is created.
9:             if $j \neq i(L^C)$ then
10:                canVisit, minTime, canDoLong $\leftarrow$ GETROUTEEXTENSION(...)
11:                if canVisit or $j =$ end location then
12:                    $r(L^E) \leftarrow r(L^C)$
13:                    $j$ added as last element in $r(L^E)$
14:                    $i(L^E) \leftarrow j$
15:                    if $j$ is base camp then
16:                        $t(L^E) \leftarrow t(L^C) + minTime + T_{gij}^{TRAVEL}$
```

```
17:                     $c(L^E) \leftarrow 0$
18:                     $l(L^E) \leftarrow False$
19:                     $\mathcal{L}^A \leftarrow L^E$
20:                 **else if** *j is end location* **then**
21:                     $t(L^E) \leftarrow t(L^C) + T^{TRAVEL}_{gij}$
22:                     $c(L^E) \leftarrow 0$
23:                     $l(L^E) \leftarrow$ **False**
24:                     $\mathcal{L}^F \leftarrow L^E$
25:                 **else**
26:                     $t(L^E) \leftarrow t(L^C) + minTime + T^{TRAVEL}_{gij}$
27:                     $c(L^E) \leftarrow c(L^C) + minTime + T^{TRAVEL}_{gij}$
28:                     $l(L^E) \leftarrow canDoLong$
29:                     $\mathcal{L}^A \leftarrow L^E$
30:                 **end if**
31:             **end if**
32:         **end if**
33:     **end for**
34:     **end while**
35:     **return** $\mathcal{L}F$
36: **end procedure**
```

One current label may result in multiple new active labels, which can each be further extended. We refer to this method of extending travel routes as branching. An example of branching, and of extending travel routes, is given if Figure 5.1. The initial label, L_0, is marked in blue. It is the first and only active label when the branching begins, and thus removed from the set of active labels, and selected as the first current label. If we assume that location 5 is the dummy end location, a new label $L_1 = (5, 0, 0, False, [1 - 5])$ will be a possible extension, and a final label. If there were no more valid travel destinations from location 1, the travel route extension for the given super-resource would be complete, as there are no more active labels. However, traveling to base camp is always allowed, resulting in $L_2 = (2, 0, 0, False, [1 - 2])$ as an extension of L_0 and a new active label. Note that base camp is location 2. Assuming location 3 may be visited after the visit to base camp, $L_3 = (3, 13, 13, False, [1 - 2 - 3])$ becomes a new active label in the next iteration, in addition to $L_4 = (0, 24, 0, False, [1 - 2 - 5])$ as a final label, both extended from the current label L_2. The active label must again be extended, until there are no more active labels. The result in this example is the six final labels marked in light green in Figure 5.1, and thus six possible travel routes for the given super-resource.

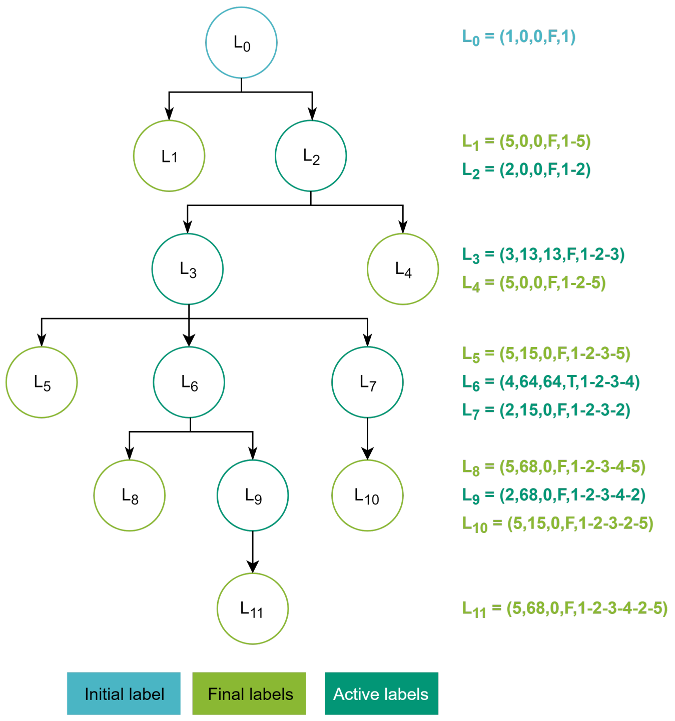


Figure 5.1: Illustration of the generation of travel routes using label branching.

5.2.3 Valid travel routes

The travel route generator does not discriminate between travel routes that can result in a high, or a low solution value. Moreover, when generating a set of travel routes for a given resource, the labeling algorithm does not consider the travel routes of other resources. It can, however, limit the generation of travel routes that are always unfeasible for a certain super-resource, regardless of which travel routes the optimization model selects for the other resources in the battlegroup. To minimize the generation of unfeasible routes, some of the constraints from the mathematical model are implemented in the travel route generating algorithms. These are described below.

- Super-resources must start at the dummy start location and end up at the dummy end location. Hence, as stated in Section 5.2.1, all travel routes begin with the start location and end with the end location.
- Super-resources must return to base camp to sleep. Therefore, resources can maximum be as many hours as the length of the work day away from base camp, unless assigned to a long task.

- Super-resources may only travel to locations where there is at least one suitable task its sub-resources are able to complete during that visit.
  - There cannot be more visits to a location than there are suitable tasks there for a certain super-resource.
  - There must exist a task at the considered location with a deadline such that the super-resource has time to travel to the location and complete the task before the deadline.
  - A super-resource cannot arrive at a location unless a task it is able to undertake at that location is either long, or has a release time early enough for the super-resource to complete it and then make it back to base camp before the working day is over.
  - Only tasks that one or more of the super-resource's sub-resources have the skills to contribute to are considered, meaning that a sub-resource must have a capacity of one or more for at least one of the skills required.
  - The tasks must be possible to complete with the combination of sub-resources in the given battlegroup. Meaning that the combination of all sub-resources must possess the required amounts of capacities of the required skills.
  - The tasks cannot be subject to precedence from, connected to, or have direct start conditions with another task, if that other task is not possible to complete with the combination of resources in the battlegroup considered.
- Super-resources cannot travel to any location unless it has time to travel to that location, complete a task and travel to the end location before the end time of the planning period.

When extending a travel route, locations recognizing these conditions are added as separate extensions, each creating a new label. Furthermore, the base location and the dummy end location are always added as two separate labels because the generator always leaves enough time to travel back to these locations. A simplified pseudocode describing how valid locations for extending a label are found, is presented in Algorithm 2. This function returns the parameters $canVisit$, $minTime$ and $canDoLong$ which are further used in Algorithm 1. These parameters respectively indicate whether a location can be visited, the minimum time the super-resource must use to get to the location and complete a task there, and whether the super-resource may undertake a long task at this location or not.

Algorithm 2 Pseudocode describing the determination of valid locations for extending a label.

```
procedure GETROUTEEXTENSION(super-resource $g$, location for consideration $j$, label $L^C$, set $\mathcal{P}_j$ of solvable tasks for $g$ at location $j$, $\overline{T}$, $T_{ij}^{TRAVEL}$, $T^{TASK}$, $T^{MIN}$, $\mathcal{P}^{\mathcal{LONG}}$, $\mathcal{P}^{\mathcal{SLEEP}}$)
    canVisit ← False
    canDoLong ← False
    minTime ← $\overline{T}$
    if number of solvable tasks at j $>$ number of visits to j then
        for task in $\mathcal{P}_j$ do
            if deadline of task $\geq$ t($L^C$) $+ T_{j,end}^{TRAVEL} + T^{TASK}T^{MIN}$ then
                if $\overline{T} -$ t($L^C$) $\geq T_{ij}^{TRAVEL} + T^{TASK}T^{MIN} + T_{j,end}^{TRAVEL}$ then
                    if l($L^C$) or task is sleep task then
                        canVisit ← True
                        minTime ← minimum of the duration time of task and the
                        duration time plus time spent waiting for the release of task
                    end if
                    if $T_{ij}^{TRAVEL} + T^{TASK}\ T^{MIN} + T_{j,END}^{TRAVEL} \leq 16 -$ c($L^C$) then
                        canVisit ← True
                        minTime ← minimum of the duration time of task and the
                        duration time plus time spent waiting for the release of task
                    end if
                    if l($L^C$) then
                        canVisit ← True
                        canDoLong ← True
                        minTime ← minimum of the duration time of task and the
                        duration time plus time spent waiting for the release of task
                    end if
                end if
            end if
        end for
    end if
    return canVisit, minTime, canDoLong
end procedure
```

5.2.4 Parameters for optimization

When all travel routes are calculated, Y_{rgimjn}^{TRAVEL} and Y_{rgim}^{LOC} are given the appropriate parameter values. All travel routes are created as discussed in Section 5.2.2, making each route unique for a given super-resource. Because only a string of visited locations is logged, visit numbers are not assigned in the route generator. In addition, tasks are not assigned to a particular visit. In other words, the travel routes may indicate a super-resource travels from location 3 to location 2 and back to location 3, but does not distinguish between completing task 1 at location 3 on the first visit and task 2 on the second visit, or the other way around. Hence, there is no symmetry in the travel routes generated. Travel routes can be the same for multiple super-resources, but because different skill levels distinguish sub-resources of the same type from each other, and several similar resources may be required to complete a single task, the exact solution method must keep all routes.

From the final labels generated, only the travel routes are of interest to set the parameters. The travel routes are segmented to provide the data required for individual travel distances and visited locations, as showed in Algorithm 3. For a problem with one super-resource with two possible travel route, from location 1 to location 3 or from location 1 via location 2 to location 3, the Y^{TRAVEL}_{rgimjn} parameter would consist of the element *(1 1 1 1 3 1) 1* for travel route 1, and *(2 1 1 1 2 1) 1* and *(2 1 2 1 3 1) 1* for travel route 2.

Algorithm 3 Pseudocode describing generation of travel route parameters.

```
1:  procedure GENERATEPARAMETERDATA( )
2:      Reads instance data from datafile
3:      for super-resource in battlegroup do
4:          travelData ← Ø
5:          visitData ← Ø
6:          final labels ← GENERATELABELS(super-resource, instance data)
7:          for label in final labels do
8:              for location in travel route of label do
9:                  travelData ← [label,super-resource,location,visit,next location,visit]
10:                 visitData ← [label,super-resource,location,visit]
11:             end for
12:         end for
13:         Saves travelData, visitData to other datafile
14:     end for
15: end procedure
```

5.3 Heuristic methods to generate travel routes

A possible measure to reduce computational time and improve the performance of the decomposition model is to limit the number of travel routes generated. A non-exact, heuristic approach is therefore introduced. Such an approach looks to remove routes that are assumed to be less likely to be part of an exact optimal solution. Theoretically, if a heuristic method is able to only remove redundant routes, the problem size will shrink and the computation time will shorten, but the solutions will be equivalent to those provided by an exact model. If the heuristic method removes routes that would otherwise be part of an exact optimal solution, then the solution will necessarily be suboptimal. Hence, the aim is to reduce the number of routes generated by only removing redundant routes.

In this section, we present two heuristic methods and their components. The first method is based on three different assumptions about the PTTP. It assumes that the least able super-resource is assigned to the security task, that the number of locations a super-resource visits during an operation is limited, and that tasks with a higher value per capacity required are more likely to be part of an optimal solution. The first method uses these assumptions to steer the travel route generator and reduce the number of routes generated. The second heuristic method looks to reduce the number of generated routes even more. Like the first method, it assumes that the least able super-resource is assigned to the security task, and that the number of locations a super-resource visits during an operation is limited. However, it differs by prioritizing total task value at a location, rather than value per capacity required. Additionally, it assumes that super-resources with equal skill sets will not travel the same routes in an optimal solution, and tasks that require more than two super-resources to complete them are not likely to be selected. An overview of the two heuristic methods' components is given in Table 5.5.

The two heuristic methods have different challenges and advantages. The first method is simpler as only the branching rate has to be decided upon. It is also not as dependent on what super-resources a test instance includes. It is probable that the branch rate will have to increase for larger instances, for the heuristic to reduce the number of routes by an amount that makes it efficient. The second method is more suitable for larger instances because one of its components, dividing routes among similar super-resources, is dependent on there being several of the same type of super-resource in an instance, which only occurs for larger instances. What could prove challenging for the second method is ensuring that all its components work in tandem and are adjusted in such a way that the heuristic reduces the number of routes enough to make the problem faster to solve, but does not remove too many favorable routes. A possible advantage of basing the reduction of routes on multiple components, is that fewer of the most favorable routes might be removed. For example, dismissing 50% of 20 routes based on one criteria, might cut off more optimal routes than cutting 50% off based on five different criteria.

Table 5.5: Overview of components that constitute heuristic methods 1 and 2.

Heuristic method 1	**Heuristic method 2**
Assigning the security task	Assigning the security task
Limiting number of location visits	Limiting number of location visits
Branching strategy: task value/capacity required	Branching strategy: task value
	Dividing routes among similar super-resources
	Limiting the number of super-resources per task

5.3.1 Assigning security task

In the exact decomposed model, which super-resource is assigned to the security task, is decided upon in the optimization model. Because the security task does not have any skill or capacity requirement, but rather just demands that its assigned super-resource pays full attention to the task during the entire planning period, an obvious simplification of the PTTP is to assign the least capable super-resource to security beforehand. To decide which super-resource is the least capable, the model calculates the number of tasks each super-resource can contribute to based on skill and capacity requirements, as illustrated in Algorithm 4. Of the super-resources that can be assigned to the least amount of tasks, the one with the lowest total capacity across all of its sub-resources is assigned to security. By predetermining the assignment of the security task, one super-resource is, in effect, removed from the problem. This restricts the problem size by reducing the number of possible travel routes for the assigned super-resource to one, and limiting the possible combinations of routes for the rest of the battlegroup because fewer resources will necessarily limit the number of tasks that can be completed and routes that are then feasible.

Algorithm 4 Pseudocode describing the assignment of the security task.

```
 1: procedure ASSIGNTOSECURITY(super-resources $\mathcal{G}$, resource skills and capacities,
    tasks $\mathcal{P}$)
 2:     securityResource $\leftarrow 0$
 3:     leastNumberOfTasks $\leftarrow$ length of ($\mathcal{P}$)
 4:     lowestCapacity $\leftarrow$ largest possible amount of capacity
 5:     for superResource in $\mathcal{G}$ do
 6:         numberOfTask $\leftarrow 0$
 7:         for task in $\mathcal{P}$ do
 8:             if superResource has skill to undertake task then
 9:                 numberOfTask $\leftarrow$ numberOfTask $+ 1$
10:             end if
11:         end for
12:         if numberOfTask $<$ leastNumberOfTasks then
13:             securityResource $\leftarrow$ superResource
14:         else if numberOfTask $\leq$ leastNumberOfTasks then
15:             if capacity ofsuperResource $<$ lowestCapacity then |
16:                 securityResource $\leftarrow$ superResource
17:             end if
18:         end if
19:     end for
20:     return securityResource
21: end procedure
```

5.3.2 Limiting location visits

In reality, super-resources do not usually have time to travel to multiple locations in a single day, as most travel times and task duration times make it unpractical. This will most likely also be the case for the solutions provided by the exact decomposed model. Both heuristic methods therefore restrict the number of locations a super-resource can visit during the operation. The limit is set such that it is equivalent to a super-resource visiting only one other location apart from base camp, on average, once each day. To provide some flexibility, the limitation applies for the entire operation period, and not for each day. For example, if it leads to a better solution, a resource still has the option to visit two locations in one day, and then stay at base camp another day, without breaking the restriction. The restriction leads to one small modification in line 11 of GENERATELABELS from Algorithm 1. This modification is described in Algorithm 5, and assumes that there are seven days at locations and seven nights at base camp, in addition to traveling from the dummy start location, and to the dummy end location. The maximum number of travels is therefore 16.

Algorithm 5 Pseudocode describing how a maximum number of location visits is enforced.

```
procedure GENERATELABELS(super-resource $g$, set $\mathcal{P}_j$ of solvable tasks for the
    super-resource at each location, set $\mathcal{I}$ of locations, travel time parameter $T_{gij}^{TRAVEL}$)
...
    if (canVisit and length of($r(L^C) \leq 16$) or $j = end\ location$ then
        $r(L^E) \leftarrow r(L^C)$
        j added as last element in $r(L^E)$
        $i(L^E) \leftarrow j$
    end if
...
end procedure
```

5.3.3 Limiting super-resources per task

A single task is often completed by only a couple of super-resources in actual operations. The exact decomposed approach does not take this into account, and therefore will generate several routes which are based on tasks being possible to be completed by more than two super-resources, even if the routes are less likely to be part of a good solution. To avoid generating these routes, the second heuristic method takes into account a maximum number of super-resources that can work together on a task. This reduces the total number of generated routes, when compared to the exact decomposition method. When calculating possible tasks for a certain super-resource, the capacity requirement of each available task is measured against the combined capacity of the super-resource and a selection of one less than the given limit other super-resources. This is illustrated in Algorithm 6.

Algorithm 6 Pseudocode describing the selection of tasks being possible to complete with a limited number of super-resources.

```
1: procedure CANDOTASKINCOMBINATION(super-resource $g$, task $p$, set of all super-resources $\mathcal{G}$, $limit$ of super-resources per task, security resource, skills and capacities of resources)
2:     Creates all possible combinations of limit $-$ 1 number of super-resources from $\mathcal{G}$
3:     for combination in combinations do
4:         canDoTask $\leftarrow$ True
5:         combined skills and capacities $\leftarrow$ capacity of super-resource
6:         for SR in combination do
7:             if SR $\neq$ security resource and SR $\neq g$ then
8:                 combined skills and capacities $\leftarrow$ capacity of SR
9:             end if
10:        end for
11:        if capacity requirement of task $\geq$ combined skills and capacities then
12:            canDoTask $\leftarrow$ False
13:        end if
14:        if canDoTask then
15:            return True
16:        end if
17:    end for
18:    return False
19: end procedure
```

5.3.4 Dividing routes among similar super-resources

Battlegroups frequently consist of multiple super-resources that are of the same type. Resources of the same type possess the same skill set. These super-resources have the same possible travel routes, but we assume that in most cases, there is no need for more than one to travel the exact same route. Therefore, the second heuristic method generates routes for one of a certain type of super-resource, and then divides those routes equally among equivalent super-resources, as described in Algorithm 7. This way, the number of generated routes for two super-resources of the same type, for example, is halved. A special case is the travel route only going from the start location, to base camp and then to the end location, which all resources must have as not to produce unfeasible solutions in the optimization model. This heuristic will be more effective for larger problems, where there is a higher probability of having multiple identical super-resources.

Algorithm 7 Pseudocode describing how travel routes are divided among resources of the same type.

```
procedure GENERATEPARAMETERDATA( )
    Reads instance data from datafile
    for SRType in TypesOfResources do
        numberOfType ← length of(SRType)
        final labels ← GENERATELABELS(SRType[first element], instance data)
        for superResource in SRType do
            travelData ← Ø
            visitData ← Ø
            for first label and then every numberOfType labels in final labels do
                for location in travel route of label do
                    travelData ← [label, SR, location, visit, next location, visit]
                    visitData ← [label, SR, location, visit]
                end for
            end for
            Saves travelData, visitData to other datafile
        end for
    end for
end procedure
```

5.3.5 Branching strategy

As described in Section 5.2, the travel route generator applies a branching strategy when generating routes for super-resources. For the exact method, it has to generate all possible routes, such that the optimization model has all feasible alternatives available, and can find the optimal solution. A heuristic approach is not bound by this requirement. By only selecting the most promising choices for travel every time the model is about to add a location to a travel route being generated, the total number of routes generated could be greatly reduced. The two heuristic methods differ in their assessments of what locations should be prioritized. For the first method, the total value and total capacity required for all tasks the given resource can be assigned to at each location, are calculated. Only a predetermined percentage of the locations with the highest combined value per capacity required needs to be selected. Total task value is divided by the capacity required to prioritize locations where value is easier to realize. If only total task value was considered for this heuristic method, it is possible that many of the tasks would not be completed because resources lacked the capacity required. This increases the likelihood that routes generated based on task value alone will not be part of an optimal solution.

Because only tasks that can be completed by one or two super-resources are considered in the second heuristic, the task capacities required will more closely match the resources' abilities than for the first method. Hence resources lacking the capacity to complete tasks is not as likely for the second heuristic method as it is for the first method. The second heuristic method therefore only considers total task value when prioritizing locations to minimize the risk of removing travel routes including the most valuable tasks. As with the first method, a given percentage of the locations is branched on, and the rest of the locations are discarded. Algorithm 8 describes the branching strategy for the second heuristic, where total task value at a location is prioritized. The process is similar for the first heuristic method.

Algorithm 8 Pseudocode describing how locations are removed when using the heuristic branching strategy.

```
1: procedure GENERATEPRIORITIZEDLOCATIONS(super-resource $g$, locations $\mathcal{I}$, current label $L^C$, solvable tasks $\mathcal{P}_i$ for $g$ at each locations at $t(L^C)$, branching $rate$)
2:     possibleLocations ← 0
3:     valueAtLocationList ← ∅
4:     for i in $\mathcal{I}$ do
5:         if canVisit for i is True then
6:             possibleLocations ← possibleLocations + 1
7:             valueAtLocationList(possibleLocations) ← total value of task in $\mathcal{P}_i$
8:         end if
9:     end for
10:    bestLocationsList ← sort valueAtLocationList
11:    remove thepossibleLocations × rate last items from bestLocationsList
12:    return bestLocationsList
13: end procedure
```

Chapter 6

Instance Generator and Input Data

This chapter offers a description of the method used to generate instances that can be used to test the mathematical models presented in Chapters 4 and 5. This method will from here on be referred to as the *instance generator*. Section 6.1.1 presents the overall structure of the instance generator, and an overview of the input data required to produce a test instance using this method. Sections 6.1.2–6.1.4 provide more information about the different segments that make up the generator. Lastly, Section 6.2 describes the input data used in the generation of test instances.

6.1 Instance Generator

6.1.1 Structure of instance generator

The instance generator is required to produce a text file of the sets and parameters referred to in Sections 4.2 and 5.1.1. The sets and parameters can be segmented into three main categories, namely resources and their skills, tasks and their requirements, and locations and travel times. The structure of the instance generator is based on this segmentation.

Generating values is based on predetermined (fixed) data of resources and their skill capacity, and on randomly generated data of tasks and their requirements. Locations, travel times, and skill level are initially randomly generated, and then fixed. Fixed data is data that is equivalent for all test instances, and of which the generator can draw different amounts of on accord. For example, it is possible to draw five super-resources for one instance, and ten super-resources for another, or five locations for one instance and ten for another. They will, however, be the same five and ten super-resources and locations every time those numbers are drawn. Furthermore, regarding resources, the instance with ten super-resources will include the same resources as the instance with five resources. Adjustable parameters are the input data that can be varied for every instance, and are presented in Appendix D. The predetermined data and adjustable parameter values used to generate the random data are created in cooperation with FFI.

A visual representation of the instance generator is given in Figure 6.1. In the figure, adjustable parameters are given in blue and predetermined data is given in light green. Both the blue and light green boxes are input data for the process-boxes that they are placed above in the figure. The data in bold is data input for more than one process, and is therefore repeated in multiple boxes. The desired output is given in turquoise and linked to the mathematical models' sets and parameters in Table 6.1, and the new abbreviations used for resource data are explained in Table 6.2 in the following section.

Table 6.1: Linking the terms in flow chart to sets and parameters in mathematical model.

Output of flowchart	**Equivalent to set or parameter**
BelongToArmyResource	$\mathcal{G}^{ARMY}$
BelongToSuperResource	$\mathcal{K}_g$
BelongToTaskExclusive	$\mathcal{P}^{\mathcal{E}}$
BelongToTaskDivisible	$\mathcal{P}^{\mathcal{DIV}}$
BelongToLocation	$\mathcal{P}_i^{\mathcal{LOC}}$
BelongToTaskLong	$\mathcal{P}^{\mathcal{LONG}}$
BelongToTaskRest	$\mathcal{P}^{\mathcal{REST}}$
BelongToTaskSleep	$\mathcal{P}^{\mathcal{SLEEP}}$
BelongToTaskDuplicate	$\mathcal{P}_d^{\mathcal{DUP}}$
MaximumVisits	$\mathcal{N}_i$
Ttravel	$\mathrm{T}_{gij}^{TRAVEL}$
Ttask	T_p^{TASK}
Tmin	T_p^{MIN}
Releasetime	T_p^{RL}
Deadline	T_p^{DDL}
CapacityRequirement	C_{ps}^{REQ}
CapacityMaximum	C_p^{MAX}
CapacityResource	C_{ksl}^{RES}
TasksSkills	H_{ps}
Value	V_{pl}
Precedence	$\mathrm{F}_{dd'}^{PREC}$
Connected	$\mathrm{F}_{dd'}^{CON}$
DirectStart	$\mathrm{F}_{dd'}^{DIR}$

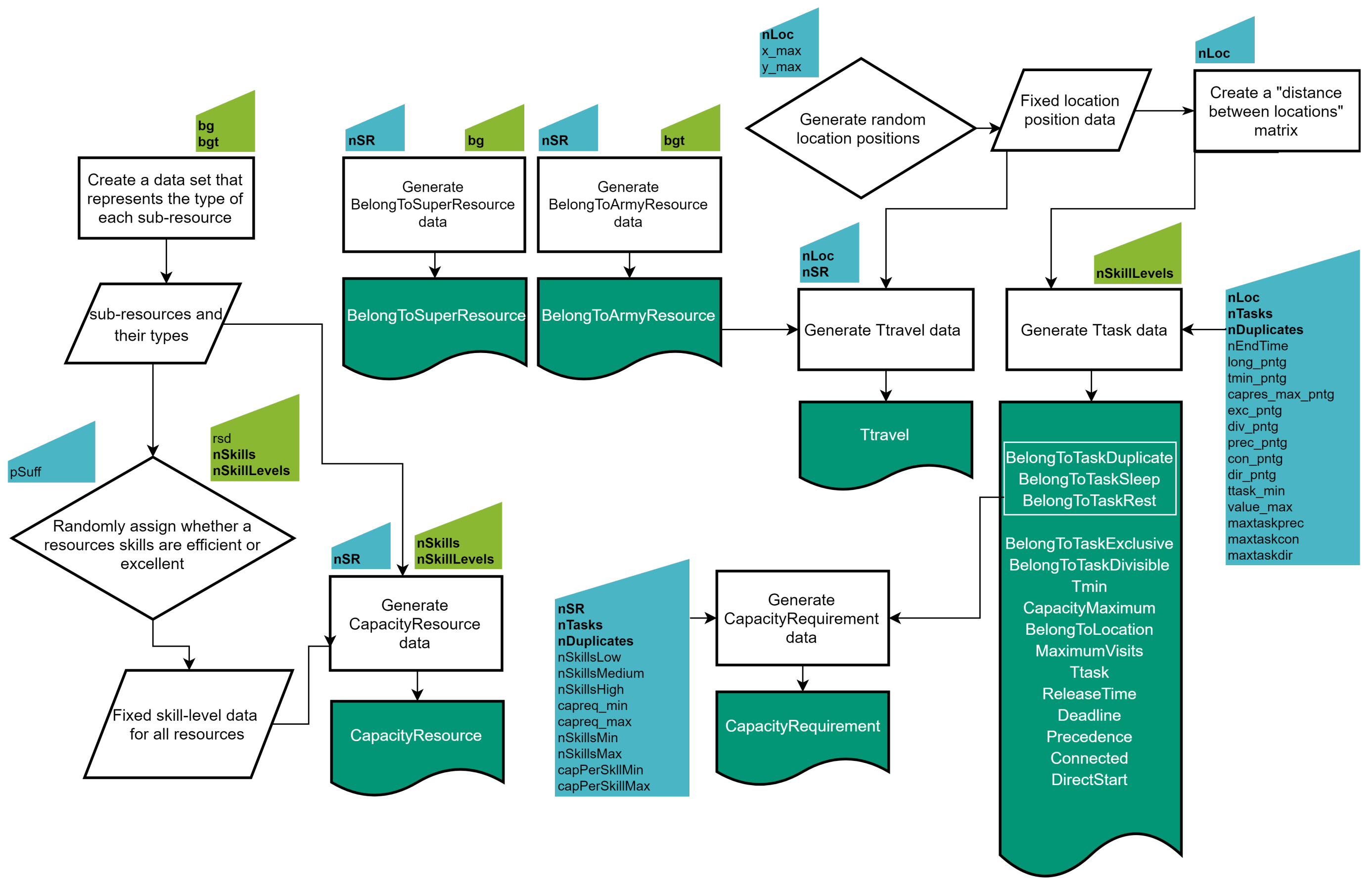


Figure 6.1: Flowchart of the processes in instance generator.

6.1.2 Resource data

An overview of the predetermined data is presented in Table 6.2. This data is a reformulation of the battlegroup description that is used for all test instances. The battlegroup description includes what super-resources and respective sub-resources it is made of, what type the sub-resources are, and what skill capacity each type has of each skill. This requires the number of skills to be predetermined as well. The battlegroup description is provided in Appendix C. An example of a battlegroup, *bg*, consisting of super-resources 1–9 and their respective sub-resources is: {1:[1], 2:[2], 3:[3,4,5], 4:[6], 5:[7,8,9], 6:[10], 7:[11], 8:[12,13,14,15], 9:[16,17,18,19,20]}. An example of the same super-resources and their sub-resource types, *bgt*, is: {1:[1], 2:[2], 3:[3,3,3], 4:[4], 5:[5,5,5], 6:[6], 7:[7], 8:[8,8,8,8], 9:[9,9,9,10,10]}. Here super-resource 9 consists of sub-resources 16, 17, 18, 19, and 20, of where 16, 17, and 18 are of type 9, and 19 and 20 are of type 10.

Table 6.2: Overview of predetermined data required to create test instances.

Predetermined data	**Description**	**Equivalent to set or parameter**
bg	Battlegroup described in terms of super-resources and their respective sub-resources	
bgt	Battlegroup described in terms of super-resources and the type of sub-resources they consist of	
rsd	Resource skill data describing the skill capacity each type of sub-resource has of each skill	
nSkills	Number of different resource skills	$\mathcal{S}$
nSkillLevels	Number of resource skill levels	$\mathcal{L}$

We use two different skill levels, where *excellent* is the high skill level, and *sufficient* is the low skill level. All resources of the same type have the same set of skills, given by the predetermined *resource skill data*. The same type of resources do not, however, need to have the same skill level for the same skill. What skill level a resource has for each of its skills, is randomly assigned to a certain extent. In most cases, army resources are trained to be very good at the skills they specialize in. This means that it is more likely for a skill level to be excellent than sufficient. This is taken into consideration in the instance generator by imposing a parameter, *pSuff*, that represents the probability of a skill level being sufficient.

6.1.3 Location and travel data

Location and travel data is based on randomly generated points on a two-dimensional coordinate system, where the only limitation is the size of the coordinate system. The size of the coordinate system is such that it is possible to travel from base camp to any other location, have time to complete a task, and travel back to base camp before the end of a workday. The travel times between locations is assumed to have a 1:1 relation with the linear distance between locations, meaning that if there is a distance of 4 between two locations, then the travel time between those locations is four hours. In the instance generator, it is assumed that supporting resources have negligible travel times, and that all army resources have equal travel times. There is no special criteria for the location of base camp. Maximum number of visits allowed for each super-resource to a certain location, is set equal to the number of tasks located at that location.

An example of a coordinate system with six locations is given in Figure 6.2, where base camp is marked with a green circle. In some cases, it might be preferred to generate a set of locations and use the same coordinates of locations for multiple test instances, rather than generate new random locations for each test instance. In this case, the locations are referred to as fixed locations.

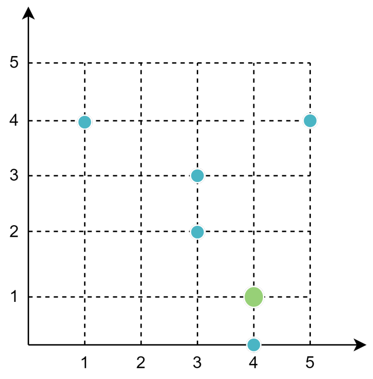


Figure 6.2: Example of six locations in a coordinate system.

6.1.4 Task data

Task data consists of all the sets and parameters that characterize tasks: what type they are, what time windows they can be completed in, how long they take to complete, their relation to other tasks, and what skills and capacity requirements they have. As such, a number of adjustable input parameters are required, as presented in Figure 6.1 and Appendix D. Apart from the relation between long and rest tasks, there is no other predetermined relations between tasks. There are, however, several links between the different characteristics of a single task as presented in Table 4.1. These links provide the framework for determining the characteristics of the randomly generated tasks.

6.1.5 Implementation

Figure 6.3 illustrates how an instance is generated given the predetermined data and fixed locations. In practical terms, the predetermined data and the fixed locations have to be manually implemented as a text file. This only needs to be done once since all instances are based on the same battlegroup data, and the locations remain the same. The adjustable parameters have to be given for each instance generated, and can be adjusted in a single file. The predetermined data and fixed locations are given in light green in the figure. The adjustable parameters are given in blue, and the desired output is given in turquoise.

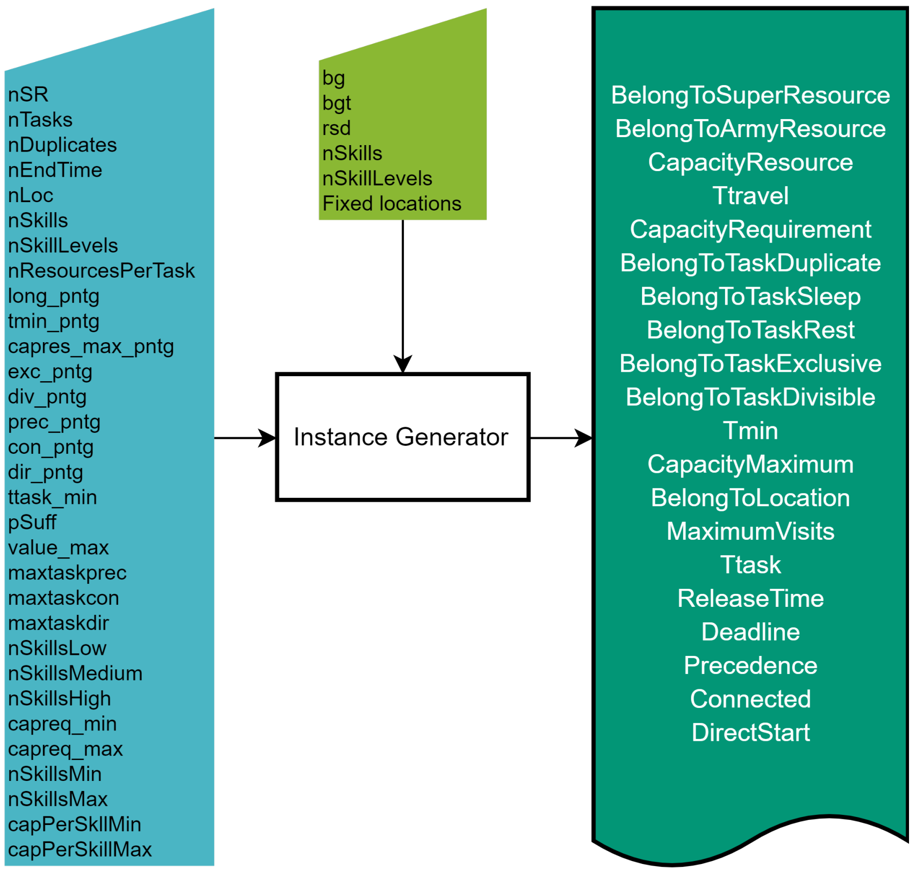


Figure 6.3: Overview of the instance generator's input and output data.

6.2 Input Data

The data implemented in this report is fictional and generated in collaboration with FFI. This section provides an overview of the parameters used to generate all test instances and a description of the test instances themselves. The input data consists of several adjustable parameters whose values can, theoretically, be varied for each instance. However, for instances to be comparable, most of these values are unaltered for all test instances used in this thesis. The values are given in Appendix D. There are four parameters that are kept alterable to test how they affect the the models performances. These are the number of super-resources, tasks, duplicates of tasks, and locations. The number of sub-resources is given by the number of super-resources, and therefore also varies, but without being an adjustable parameter. It is worth repeating that the resource and location data is fixed, whilst the task data is randomly drawn given certain constraints.

6.2.1 Time input data

Our models' configurations are based on a planning period of one week, and the time unit is set to hours. Thereby, the end time is at 168 hours. The length of the planning period affects the models' configurations because it defines what decisions the models are required to make, for them to be helpful decision support tools. If, for example, the planning period was a year, the tasks to be considered would be of a different characteristic, and there would be other criterion regarding time off, rotation, and so forth. These divergences would require a different set of constraints, and hence a different model.

A regular working day is 16 hours. The remaining 8 hours of the day are sleeping hours. All regular tasks therefore have a release time and deadline within regular working hours, as illustrated by the green zones in Figure 6.4. Exactly when the release times and deadlines are, are randomly provided for each task. The release time and deadline for a task are both during the same working day, and the time between them is at least as long as the standard time it takes to complete the task. The exceptions are long and rest tasks, which have a duration of 48 and 24 hours respectively, and are therefore not bound to start or finish within regular working hours. The release time for a long tasks can be at any time, as long as there is enough time for a rest task afterwards. In practical terms this means that the release time for a long task has to be in the interval [0, 96]. Consequently, the deadline for a long task has to be in the interval [release time of task + 48, 144], to ensure the resources have time to complete both the long task and a rest task. Rest tasks are only limited to start directly after a long task, and end before the end time of an operation.

All tasks except long and rest tasks, are assigned a standard completion time in whole hours, within a given interval. The upper limit of this interval for a task is dependent on the time it takes to travel to and from base camp and the tasks location. This is to ensure that it is possible to travel from base camp, complete a task at a location, and have enough time to travel back to base camp. The lower limit is given as two hours because there are rarely any peacekeeping tasks that take a shorter amount of time to complete. Divisible tasks can be completed in a shorter time than the standard completion time.

Travel times are directly correlated with the linear distance between locations, as explained in Section 6.1.3. Travel times from the dummy start location to all other locations, and from all locations to the dummy end location is set to be equivalent to the travel times to and from base camp, so as to model the dummy start and end location as the base camp. All travel times for aircraft are, as mentioned, assumed negligible and therefore set to zero. All army super-resources have the same traveling times because they are assumed to be operating with the same means of transport.

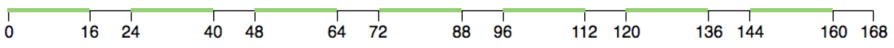


Figure 6.4: Illustration of working and sleeping hours during planning period.

6.2.2 Task input data

Tasks are randomly generated for each instance. To maintain consistency, all instances have ten tasks that are duplicates of other tasks. For example, if there are 50 unique task, a random selection of these tasks are duplicated one or more times until there are a total of 60 tasks. Tasks are assigned to locations, including base camp, at random.

A task can be long, divisible, exclusive, or a combination of these. In addition, a task can have precedence, a direct start after, or be connected to other tasks. Table 4.1 in Chapter 4 illustrates these characteristics and how they relate to each other. A task has a 10% chance of being a long task, a 50% chance of being exclusive, and a 10% chance of being divisible. A divisible task has two additional characteristics; the minimum time it takes to complete it and the maximum capacity allowed to work on it. The minimum time any task can take to complete is 50% of its standard duration time. The maximum capacity that can be assigned to a divisible task is 300% of the minimum capacity required. Hence, if a divisible task requires a total amount of eight in capacity and has a standard duration time of four hours, it could be completed in two hours if the resources assigned have a combined capacity of at least 24 of the skills required.

Furthermore, a task has a 10% chance of having precedence over other tasks, a 10% chance of being connected to other tasks, and a 10% chance of having a direct start before other tasks. This is in addition to the cases where precedence, direct start, and being connected is required because of other characteristics such as being a long or sleep task. The maximum number of tasks a single task can have precedence over is two. The same applies for direct start and being connected.

A task's value is dependent on the skill level it is completed at and the standard time it takes to complete the task. This way, the value indicates the utility of every hour spent undertaking a task with a given skill level. A random integer between one and three is first picked as a temporary parameter associated with completing the task at an excellent skill level. If this number is one, then the task's value is the same whether the task is completed at an excellent or sufficient level. If the number is two or three, then the temporary parameter for a sufficient completion is one less than that of an excellent. Once this is decided upon, the value of the task is given as the temporary value multiplied by the standard duration of the task. Sleep and rest tasks do not have any value.

A task's capacity requirement of each skill is generated at random with some given limitations. The limitations are set to enable tasks to be completed by one to three sub-resources, as the maximum number of sub-resources allowed to undertake the same task is set to three. In collaboration with FFI, The number of skills are set to 15, where skill 1 is the ability to sleep. The complete list of skills is presented in Table C.1 in Appendix C. Sleep tasks and rest tasks require a single skill that all army resources have. All other tasks require between one and five skills, and the total capacity requirement is between two and 15. Additionally, each of the required skills are to have a random capacity requirement of between one and five. To decide which skills to select, all skills are assigned a probability according to how common they are among the sub-resources: Low, medium and high. Medium probability skills are twice as likely to be selected as skills in with low probability, and high probability skills are twice as likely as skills of medium probability. Skills 2–7 are low of probability, skills 8–12 are of medium probability, and skills 13–15 are of high probability.

6.2.3 Resource input data

The relation between super-resources and sub-resources, and the number of each type of resource is based on a realistic composition of a battlegroup. There are 10 different types of super-resources and 11 types of sub-resources. As mentioned in Chapter 6, the types of sub-resources belonging to a certain type of super-resource are constant. The resource types used as input data are given in Table 6.3. One complete battlegroup of 21 super-resources and their sub-resources is given in Table C.1 in Appendix C. A top-down selection is applied to all test instances, meaning that a selection of six super-resources consists of the super-resources indexed 1–6 and the sub-resources indexed 1–10, a selection of 10 super-resources consists of the super-resource indexed 1–10 and the sub-resources indexed 1–24, and so on. This is both to limit the selection's influence on the different test instances, and to ensure a realistic composition of the smaller battlegroups. For instance, there should be a *headquarter platoon* in all instances. Also, for every fourth *infantry platoon*, there should be a new *infantry company headquarter*.

Of the 15 skills, all sub-resources have capacities as integer numbers from one to three for one to five different skills that they master. For non-mastered skills, the capacities are zero. Furthermore, army sub-resources have a capacity of skill 1, the sleeping skill, equal to the total number of sub-resources, which again is equal to the capacity required for rest tasks and sleep tasks. This way, all sleep and rest tasks may be undertaken by anything from one to all sub-resources, as each sub-resource alone is able to contribute anything from one capacity to all the capacity required. Which skills and capacities each sub-resource possesses are predetermined and equal for all sub-resources of the same type, as presented in Table C.1 in Appendix C. Which skill level a resource has of a given skill is generated at random, with a 75% probability that a certain skill is excellent. Thus, two sub-resources of the same type will always have the same capacities of the same skills, but may have different skill levels for each skill.

Table 6.3: Description of the different types of super-resources and their respective sub-resources.

Super-resource	**Index (g) / SR type**	**Sub-resources**	**Index (k)**	**SB type**
Infantry company headquarter 1	1	Infantry company headquarter 1	1	1
Headquarter platoon	2	Headquarter platoon	2	2
Infantry platoon 1	3	Infantry section 1.1	3	3
		Infantry section 1.2	4	3
		Infantry section 1.3	5	3
Reconnaissance team 1	4	Reconnaissance team 1	6	4
Medical platoon 1	5	Medical section 1.1	7	5
		Medical section 1.2	8	5
		Medical section 1.3	9	5
Sanitary helicopter	6	Sanitary helicopter	10	6
Transport helicopter	7	Transport helicopter	11	7
Mortar platoon	8	Mortar section 1.1	12	8
		Mortar section 1.2	13	8
		Mortar section 1.3	14	8
		Mortar section 1.4	15	8
Combat service support platoon	9	Combined maintenance team 1.1	16	9
		Combined maintenance team 1.2	17	9
		Combined maintenance team 1.3	18	9
		Recovery team 1.1	19	10
		Recovery team 1.2	20	10
Anti-aircraft platoon	10	Anti-aircraft 1.1	21	11
		Anti-aircraft 1.2	22	11
		Anti-aircraft 1.3	23	11
		Anti-aircraft 1.4	24	11

Chapter 7

Computational Study

In this chapter the performances of the different models and solution methods presented in Chapters 4 and 5 are tested and analyzed. The instances used to test the models are presented in Section 7.1. Section 7.2 presents the computational results and analysis of the exact models. A comparison of the exact decomposition solution approach and the two heuristic decomposition approaches follows in Sections 7.3 and 7.4.

The purpose of testing and comparing the compact model with the decomposed model, is to evaluate whether a decomposition solution approach provides better solutions within a given run time. Furthermore, we wish to evaluate the applicability and limitations of the decomposed model with respect to solving realistic instances of military peacekeeping operations. We hypothesize that the exact decomposition approach will prove to be more efficient than the compact model, but not efficient enough to handle realistic instances. We therefore also test two different heuristic decomposition approaches, and compare them to the exact decomposition approach, to evaluate whether they are more applicable to real-sized problems.

The instance generator described in Chapter 6, and travel route generator described in Chapter 5.2 is implemented in Python version 3.6.4. The model and the test instances are solved using the commercial optimization software FICO Xpress IVE 8.3, which is set to apply the branch-and-bound solution method with a depth-first search strategy. The generation of test instances and travel routes are carried out on a computer with a 2.6GHz Intel Core i5 CPU, 8GB RAM, running a 64 bit macOS High Sierra version 10.13.4. All other computational testing is performed on a computer with a 3.6 GHz Intel(R) Core(TM) i7-7700 CPU, 32 GB RAM, running a 64 bit Windows 10 operating system. The maximum run time for each test instance is for practical reasons set to one hour (3600 seconds) in FICO Xpress.

7.1 Test instances

This section describes the three test instance sets used to compare the compact and decomposed model. The instances are created using the input data presented in Section 6.2. 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, and the third set tests for an increasing number of tasks. These three sets are given by tables 7.1, 7.2, and 7.3 respectively.

As presented in the tables, each set consists of five subsets. Each of these subsets have five test instances. As a result, each set consists of 25 test instances. Because task characteristics are generated at random for every test instance, the test instances belonging to the same subset differ even though their other parameters are equivalent. Running five test instances for each subset is a corrective measure against the uncertainties and variance introduced by randomly generating task data in the test generator.

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 test instances are equivalent to their respective subsets, but are in addition numbered from 1–5. The identification for a subset with four super-resources, 30 tasks and eight locations is for example "R–4–30–8", and one of its test instances is "R–4–30–8–1".

The three sets reflect the three most prominent dimensions that determine the size of the problem the models are to solve. Testing along these dimensions makes it possible to study how the models react to an increase in resources, tasks, and locations. As a decision support tool in military peacekeeping operations, FFI are most interested in how the models perform given an increasing number of resources. Given that battlegroups are regarded as collective units in operation planning, the number of resources the models can handle is a good indication of how useful they are. If for example the models cannot handle more than five super-resources, and a battlegroup consists of 20 super-resources, then the models are not of practical use. The number of locations is also of interest because it reflects the geographical area the models can encompass. Considering tasks in a peacekeeping operation, there will often be many more than can be completed, given the limited resources and time. However, it is still of interest to test for how many tasks the models can handle, because it could distinguish different traits in the two models.

Table 7.1: Test set 1: Increasing the number of super-resources.

Subset	#Super-resources	#Sub-resources	#Total tasks	#Unique tasks	#Locations
R-4-30-8	4	6	30	20	8
R-6-30-8	6	10	30	20	8
R-8-30-8	8	15	30	20	8
R-10-30-8	10	24	30	20	8
R-12-30-8	12	30	30	20	8

Table 7.2: Test set 2: Increasing the number of locations.

Subset	#Super-resources	#Sub-resources	#Total tasks	#Unique tasks	#Locations
L-6-30-4	6	10	30	20	4
L-6-30-6	6	10	30	20	6
L-6-30-8	6	10	30	20	8
L-6-30-10	6	10	30	20	10
L-6-30-12	6	10	30	20	12

Table 7.3: Test set 3: Increasing the number of tasks.

Subset	#Super-resources	#Sub-resources	#Total tasks	#Unique tasks	#Locations
T-6-30-8	6	10	30	20	8
T-6-40-8	6	10	40	30	8
T-6-50-8	6	10	50	40	8
T-6-60-8	6	10	60	50	8
T-6-70-8	6	10	70	60	8

7.1.1 Remarks on the test instances

The input data used to generate the test instances are decided upon in cooperation with FFI, but are based on assumptions, and not on real data. This is because similar models to the ones in this thesis have not been studied before, so there has not been a need to quantify the input parameters that are required. In addition, the military operation data FFI has access to is classified, making it difficult to analyze previous operations which would provide a better estimate for the parameters. Many of the parameters are also difficult to quantify, for example skills and the value of tasks.

The consequence of the test instances being based on assumptions and parameters that are difficult to quantify, is that tasks might have characteristics that make them difficult to complete, for example time windows might be narrow, or capacity requirements might be high. This restricts the problem and reduces the solution space, and could lead to solutions including time periods where resources are idle. Results from these instances might therefore be more positive than if resources were expected to be in constant activity, as that they might be easier to calculate.

However, the results are still relevant because they can speak to how the models perform relative to each other. In addition, the models might be considered to schedule larger, more important tasks. Idle time between tasks will then be inherent in a solution. The idle time that will be inherent in these solutions can then be used to carry out routine tasks such as maintenance. Moreover, FFI has stated that in their experience, there can be a lot of idle time, also in manually scheduled operation plans.

7.2 Comparison of the exact models

This section presents the results and analysis of the application of the exact models to test sets 1–3. Tables 7.4, 7.6, and 7.8 compare the best solution found, the best solution after 20 minutes, the optimality gap, and the total run time of the compact model (CM) and the decomposed model (DM). 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 (Fauske, 2015). For the DM, the total run time is the aggregate of the time it takes to generate travel routes in Python, and run the optimization model in FICO Xpress. For the CM, the total run time is only the time it takes to run the optimization model in FICO Xpress.

The results for each subset are calculated as the average of its five instances, with the exception of the optimality gap. The optimality gap is calculated using the average upper bound and average best solution. The results for every test instance are provided in Appendix E. 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. Also note that because average values are used as a base for analyzing the results, there may be discrepancies between total run times and optimality gaps. This is especially considering subsets that have a high optimality gap, but also a total run time that is less than the maximum time permitted.

7.2.1 Increasing the number of resources

Table 7.4 suggests that the DM dominates the CM when increasing the number of resources. This is based on the DM having equal or better results regarding the best solution found, the best solution found after 20 minutes, the optimality gap, and run time, for all five subsets. Furthermore, for some test instances including eight, ten, and twelve super-resources, the CM did not find any solution in the given run time of one hour. The DM found at least one solution for all instances (see Appendix E).

Larger instances naturally have longer run times for both models. Figure 7.1 provides a breakdown of the DM's total run times. Although the overall trend is an increase in duration for both the route generation and optimization, the optimization duration seems to increase at a higher rate, as generating routes seems to take a decreasing proportion of the total run time. This suggests that an increase in the problem size due to an increase in the number of resources does not affect the route generation algorithm as much as it does the optimization model, and that for even larger instances, the time saved by decomposing the compact model, is worth the time it takes to generate routes.

Table 7.4: Comparison of test set 1 with an increase in resources.

Subset	**Best solution**		**Best solution after 20 min**		**Optimality gap after an hour**		**Total run time [s]**	
	CM	DM	CM	DM	CM	DM	CM	DM
R-4-30-8	7.3	12.5	7.3	12.5	71.4%	0.0%	721	0.9
R-6-30-8	21.5	21.5	21.5	21.5	0.0%	0.0%	6.3	1.7
R-8-30-8	20.4*	41.2	20.4*	41.2	102%*	0.0%	1485	21.2
R-10-30-8	16.4*	72.4	16.4*	68.8	548%*	37.3%	3600	2215
R-12-30-8	24.6*	54.2	24.6*	40.0	270%*	67.9%	2973	2210
Average	*18.0*	*40.4*	*18.0*	*36.8*	-	-	*1757*	*890*

Note that these results are only strictly accurate for the instances where the optimal solution was found within a total run time of one hour. For the cases where an optimal solution was not found within an hour, the percentage of time it takes to generate routes as shown in Figure 7.1, is higher than is actually the case. This is because the route generation is run to completion first, and the optimization model is cut short if it has not found an optimal solution within an hour. If given enough time to find an optimal solution the optimization model would necessarily take a larger portion of time than is currently implied, and the portion of time it takes to generate routes would be even smaller.

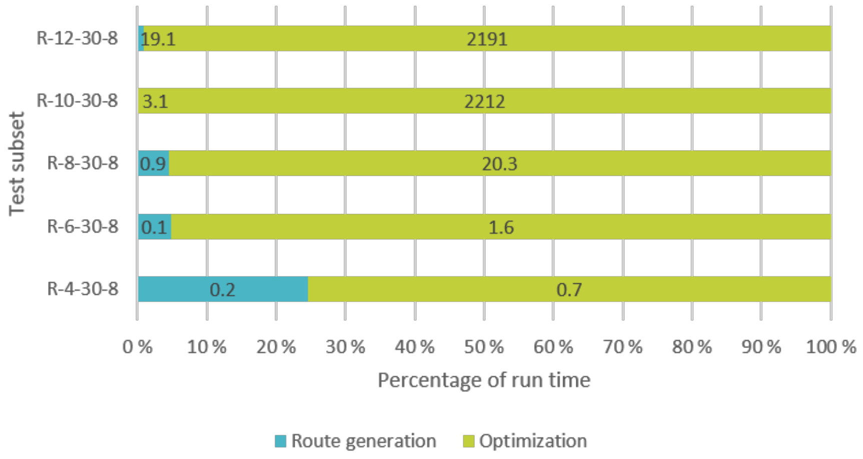


Figure 7.1: Exact decomposed models route generation and optimization run times for test set 1.

The total number of routes generated might have been expected to increase linearly with the number of resources, because each resource's set of possible routes is generated individually, independent of other resources. However, this is not the case. An increase in resources, and hence number of skills and capacity available, will in most cases lead to a higher number of tasks being possible to complete, in addition to tasks being possible to complete by several additional combinations of resources. Because of these extra possibilities, the number of routes that are generated for the DM increase in an exponential fashion, as illustrated in Figure 7.2. The extra tasks that can be completed, and the additional combinations of resources that can complete tasks, also affect the performance of the CM. The number of routes generated for a certain instance therefore indicate, to a certain extent, how well the CM will perform, even though the CM does not apply them.

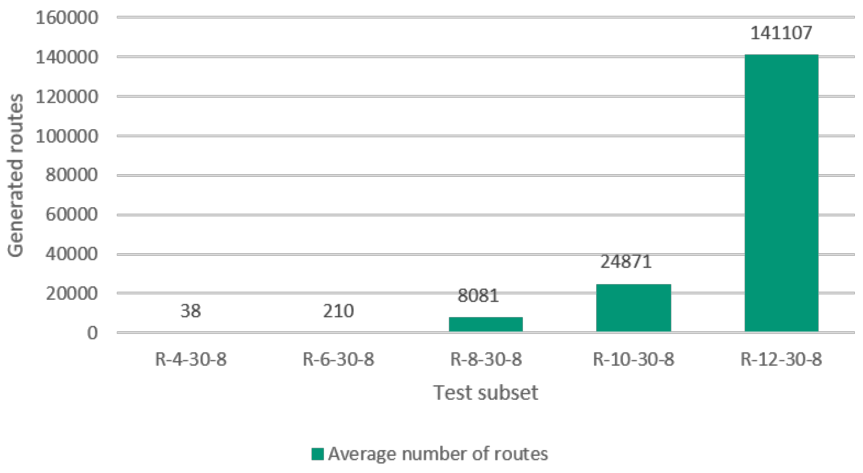


Figure 7.2: Number of routes generated by exact decomposed model for test set 1.

The results in Table 7.4 show that the CM is capable of handling most test instances up to six super-resources, and the DM up to 10 super-resources, before the optimality gap becomes significant, and the solutions suboptimal. Based exclusively on these results, the DM is a significant improvement on the CM. Table 7.5 provides additional information on the number of solutions found for each model and subset within 20 minutes and one hour. This table corroborates the advantage the DM has over the CM. The cells highlighted in green show that the DM finds an optimal solution in 76% of its cases, compared to 52% for the CM. The cells highlighted in blue show that for 92% of its cases, the DM found its best solution within 20 minutes, compared to 74% for the CM, suggesting that the DM is more likely to provide a good solution within the acceptable amount of time of 20 minutes, and hence be of more practical use. In addition, the DM finds at least one solution for all its instances within an hour. The CM fails to do so for 24% of its instances. This is especially true for larger instances, exemplified by the cells highlighted in red. Both the DM and CM have high optimality gaps for these larger instances, but the DM does at least find a feasible solution for all instances. In practical situations, and especially for larger problems, having these feasible solutions can be a useful starting point to base planning decisions on.

Table 7.5: Comparison of the number of solutions for test set 1.

Subset	Within 20 minutes						Within one hour			
	Optimal solution		**Best solution**		**At least one solution**		**Optimal solution**		**At least one solution**	
	CM	DM	CM	DM	CM	DM	CM	DM	CM	DM
R-4-30-8	4	5	5	5	5	5	4	5	5	5
R-6-30-8	5	5	5	5	5	5	5	5	5	5
R-8-30-8	3	5	4	5	4	5	3	5	4	5
R-10-30-8	0	2	2	4	2	5	0	2	2	5
R-12-30-8	1	2	3	4	3	5	1	2	3	5
Average	*2.6*	*3.8*	*3.8*	*4.6*	*3.8*	*5.0*	*2.6*	*3.8*	*3.8*	*5.0*

7.2.2 Increasing the number of locations

The results in Table 7.6 indicate that the DM provides better or equal results, in a shorter amount of time than the CM. For a few test instances regarding 10 and 12 locations, the CM did not find any solution within an hour of run time. 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 twelve locations (see Appendix E). This suggests that the CM is able to efficiently handle up to eight locations, and the DM up to 10 locations.

Table 7.6: Comparison of test set 2 with an increase in locations.

Subset	**Best solution**		**Best solution after 20 min**		**Optimality gap after an hour**		**Total run time [s]**	
	CM	DM	CM	DM	CM	DM	CM	DM
L-6-30-4	34.7	35.3	34.7	35.3	1.7%	0.0%	723	1.8
L-6-30-6	30.8	35.2	29.4	35.2	14.3%	0.0%	1445	2.6
L-6-30-8	21.5	21.5	21.5	21.5	0.0%	0.0%	6.3	1.7
L-6-30-10	25.8	37.4	25.8*	37.4	54.7%	0.0%	1445	16.4
L-6-30-12	6.4*	33.0	1.3*	23.8	643%*	39.4%	1523	734
Average	*23.8*	*25.9*	*22.5*	*30.6*	-	-	*1029*	*151*

The average run times for the DM are well within the practical limit of 20 minutes for the given subsets, but it appears to increase exponentially for large instances. In comparison, the average run times for the CM are considerably higher, and with the exception of L-6-30-8, have a more linear trend. This could suggest that for smaller instances with fewer locations, the route generator is able to restrict the scope of the problem to a large degree. It can restrict the scope of the problem because having fewer locations leads to fewer possible routes, and hence more variables in the optimization model become fixed. For the CM, there is no procedure that fixes variables to the same extent.

Similar to the case for test instance set 1, the time it takes to generate routes for test instance set 2 is an insignificant amount compared to the optimization time (see Figure F.1 in Appendix F). There is, however, less of a trend regarding the time it takes to generate routes. This would indicate that increasing the number of locations does not affect the route generation as much as increasing the number of tasks does.

Studying the average number of routes generated for each subset would, however, suggest that there is an increase in the number of possible routes with an increase in locations (see Figure F.2 in Appendix F). One cause for this is for example, if 20 tasks with the same time windows are at the same location, then a resource can only travel there once, but if the same 20 tasks are at 20 different locations, then the resource has 20 different travel options.

Because the run time and number of routes generated do not have as clear a trend with an increase in locations as with an increase in resources, it is more difficult to predict the DMs behaviour for larger instances. Nevertheless, the results in Table 7.6 strongly suggest that the DM is an improvement over the CM. Moreover, the green cells in Table 7.7 show that for 96% of its cases, the DM managed to find an optimal solution within 20 minutes, in contrast to 60% for the CM. The blue cells highlight the instances where the model did not find a better solution after 20 minutes. The CM finds its best solution for 76% of its instances within 20 minutes, the DM 96%. This means that for most of their cases, there is nothing to gain by running the model for more than 20 minutes, but the DM is more likely to provide a good solution within the acceptable amount of time. As with an increase in resources, the conclusion for test set 2 is that the DM is a better solution approach than the CM.

Table 7.7: Comparison of the number of solutions for test set 2.

	Within 20 minutes						**Within one hour**			
Subset	**Optimal solution**		**Best solution**		**At least one solution**		**Optimal solution**		**At least one solution**	
	CM	DM	CM	DM	CM	DM	CM	DM	CM	DM
L-6-30-4	4	5	5	5	5	5	4	5	5	5
L-6-30-6	3	5	4	5	5	5	4	5	5	5
L-6-30-8	3	5	4	5	5	5	5	5	5	5
L-6-30-10	3	5	4	5	4	5	3	5	5	5
L-6-30-12	2	4	2	4	3	5	3	4	4	5
Average	*3.0*	*4.8*	*3.8*	*4.8*	*4.4*	*5.0*	*3.8*	*4.8*	*4.8*	*5.0*

7.2.3 Increasing the number of tasks

The results in Table 7.8 show that both models struggle to efficiently handle instances with a large number of tasks. Comparing the two models, the DM provides an equal or better solution for four out of the five subsets within an 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. The mixed results mean that the DM does not dominate the CM for this test set, even though it is better for most subsets.

Table 7.8: Comparison of test set 3 with an increase in tasks.

Subset	**Best solution**		**Best solution after 20 min**		**Optimality gap after an hour**		**Total run time [s]**	
	CM	DM	CM	DM	CM	DM	CM	DM
T-6-30-8	21.5	21.5	21.5	21.5	0.0%	0.0%	6.3	1.7
T-6-40-8	39.6*	57.2	39.6*	57.2	97.5%*	35.1%	2882	1449
T-6-50-8	60.5*	50.5*	60.5*	50.5*	95.4%*	134.6%*	2181	2285
T-6-60-8	21.8	62.8*	21.8	61.6*	300%	50.0%*	2232	1622
T-6-70-8	54.7	60.2*	54.7	51.7*	84.9%	65.4%*	3012	2286
Average	*39.6*	*50.4*	*39.6*	*48.5*	-	-	*2063*	*1529*

There seems to be an overall increase in the time it takes to generate routes, and carry out the optimization for the DM (see Figure F.3). As with an increase of resources and locations, the time it takes to generate routes is still only a small fraction of the overall run time. However, for the subsets with 50 and 70 tasks, it takes over 90 seconds to generate travel routes, which is markedly higher than any of the subsets tested for in test set 1 and 2. This indicates that the DM is not as effective at handling an increase in tasks, as it is an increase in resources or locations.

The subsets with 50 and 70 tasks also have a considerable number of possible routes (see Figure F.4). When the number of routes is of this scale, the decomposed model is either incapable of finding any solution, or finds a solution that is far from optimal, within one hour of run time. This is because the text files that contain the travel routes are very large, and hence it takes FICO Xpress a substantial amount of time to only read the file. To test how long it could take to read a large file, instance T-6-70-8-1, comprising of 1 309 144 routes, was allowed to run till completion in FICO Xpress. It took 165 minutes to read the file, and an additional 87 minutes to solve to optimality. At 20 minutes after the file had been loaded, the optimality gap was 78%, and after one hour the gap was 15%, both consistent with the other results in the same subset. What this test proves is that the loading time of the route file to FICO Xpress is a factor that greatly affects the effectiveness of the DM for large instances.

Compared to the CM, the green cells in Table 7.9 do, however, suggest that it is more likely the DM will find an optimal solution within 20 minutes. The CM, on the other hand, finds more of its best solutions within 20 minutes, marked in blue. Furthermore, the CM does not seem to do any worse than in the other test sets when it comes to finding at least one solution within an hour. Regarding the DM, this test set is the only set where it has not found at least one solution for every instance, as marked in red. What this confirms is that while the CM does not struggle more with an increase in tasks than it does an increase in resources or locations, the DM is sensitive to the number of tasks in a problem.

Table 7.9: Comparison of the number of solutions for test set 3.

	Within 20 minutes						**Within one hour**			
Subset	**Optimal solution**		**Best solution**		**At least one solution**		**Optimal solution**		**At least one solution**	
	CM	DM	CM	DM	CM	DM	CM	DM	CM	DM
T-6-30-8	5	5	5	5	5	5	5	5	5	5
T-6-40-8	1	3	3	5	3	5	1	3	3	5
T-6-50-8	2	2	4	2	4	2	2	2	4	2
T-6-60-8	2	3	3	3	3	4	2	3	3	4
T-6-70-8	1	2	5	2	5	4	1	2	5	4
Average	*2.2*	*3.0*	*4.0*	*3.4*	*4.0*	*4.0*	*2.2*	*3.0*	*4.0*	*4.0*

7.2.4 Summary of comparison of the exact models

An analysis of the results from test set 1, where the number of resources is increased, strongly implies that the DM is superior to the CM. The CM can efficiently handle six super-resources and 10 sub-resources, whilst the DM can handle 10 super-resources and 24 sub-resources. The DM is also faster, and finds more solutions within 20 minutes, making it more practical than the CM for instances with a larger number of resources.

The results from test set 2, where the number of locations varies, also suggest that the DM performs better than the CM. The CM does not show any clear trend, but generally does not perform well for this test set. It is therefore difficult to identify how many locations it can handle efficiently, but an optimistic estimate would be eight locations. The DM can handle 10 locations well. Furthermore, the DM provides better solutions in a fraction of the time.

An increasing number of tasks proves to be a challenge for both models. Overall, the DM performs slightly better than the CM, but the results are more difficult to interpret than it is for the two first sets. What is noteworthy of the results from this test set is that while an increase in the number of tasks does not seem to affect the performance of the CM a great deal more than increasing the number of resources or locations, it does seem to greatly affect the performance of the DM. Why the DM is not as effective at handling an increase in tasks, as it is an increase in resources or locations, is most likely because more tasks increases the number of possible routes considerably. A large increase in the number of routes increases the time it takes to generate the routes, but also the time it takes for FICO Xpress to read the text file that includes the routes.

Overall, the DM performs better than the CM, indicating that a decomposed solution approach is effective. The DM does nonetheless, struggle to handle the largest instances efficiently. The number of resources and tasks seem to be the most prominent factors in 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. An increase in the number of locations does not seem to affect the DM to as great a degree. 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. For 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. One measure to reduce the time it takes to read the file and improve the performance of the DM, is to reduce the number of routes generated. The next section examines this measure.

7.3 Analysis of the first heuristic decomposition

This section presents test results and an analysis of the heuristic decomposition solution approach using the first heuristic method to generate routes. The first heuristic method is hereby referred to as H1 and is described in Section 5.3. A heuristic method does not generate all possible routes. The reason for using such an method is that a reduction in travel routes can improve the overall performance of the decomposed model. Test sets 1 and 3 presented in Tables 7.1 and 7.3 are tested on H1 and compared to the exact decomposition model discussed in Section 7.2. Further testing of test set 2 is not prioritized because the number of locations does not seem to affect the number of generated routes to the same extent as the number of resources and tasks. In addition, FFI does not consider the number of locations to be as critical as the number of resources and tasks. H1 is tested for four different rates of travel route branching: 75%, 50%, 30%, and 15%, also referred to as branching rates. With 75% branching, the heuristic is referred to as H1–75. This percentage only considers the 75% best alternative destinations each time the procedure adds a location to a travel route, thereby cutting off 25% of the possible branches at each decision node in the route generator. Figure 7.3 illustrates a branching of H1–75. Here location 6, 10, and 11 are the best 75% alternative destinations that a resource has from location 2, and hence the procedure cuts off the branch that would travel to location 5. The heuristic model tested with the other rates are referred to as H1–30, H1–30 and H1–15. The exact decomposed model is still referred to as the DM.

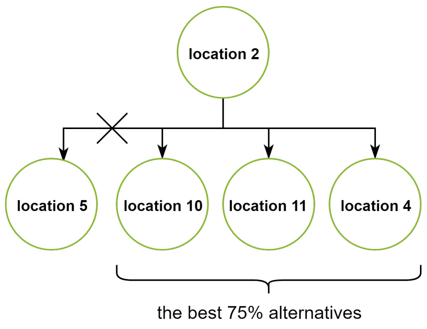


Figure 7.3: Illustration of branching with a 75% branching rate.

7.3.1 Increasing the number of resources

Figures 7.4 and 7.5 compare the solutions of the four different branching rates for the first heuristic approach, with the solution for the exact decomposition. The graphs show the heuristics' deviation from the exact method's best solution, represented by the 0% line on the horizontal axis. Results below 0% mean the heuristic solution is worse than the best solution from the exact model, indicating that it cuts off too many suitable travel routes. The comparison is carried out for the exact method's best found solution within one hour and best found solution within 20 minutes. This is because the purpose of the comparison is to judge whether or not the heuristic looks promising as an alternative in practical settings where the exact decomposition method has proven to have some limitations.

For low numbers of super-resources, where the exact decomposed model is able to find an optimal solution in a few seconds, the heuristic model is inferior. For these small instances, the solutions seem to worsen as more routes are cut. However, Figure 7.4 shows that for higher numbers of super-resources, the versions of the heuristic cutting fewest routes, H1–75 and H1–50, perform better or as well as the exact decomposition with an hour of run time. For solutions found within 20 minutes, Figure 7.5 shows that H1–50 is better at 10 super-resources, and that the heuristic is as good or better at 12 super-resources for all the different branching rates tested. A comparison of R–8–30–8 in Figures 7.4 and 7.5 proves that a heuristic approach is dominant for this subset given 20 minutes, but not so when the model is allowed to run for an hour. When run for an hour H1–30 and H1–15 underperform. The tests seem to indicate that reducing travel routes will provide better solutions where the exact methods struggle, but there is a trade-off between making the problem simple enough to provide a solution in reasonable time, and getting the best possible solution.

As the number of super-resources increases, the number of routes increases exponentially for all branching rates, as illustrated in Figure 7.6. However, there are consistently fewer travel routes for the models with higher removal rates compared to the ones with lower removal rates, for all subsets. This is because of how the travel route generator works; A higher branching rate which cuts off fewer options has a higher growth rate, with an increasing amount of super-resources, than a lower branching rate. The heuristics with a removal rate of 50% or higher have an almost minuscule number of traveling routes compared to H1–75 and H1–100, for the largest test subsets.

Comparing the number of travel routes with the run time from Figure 7.7, there seems to be some relation between number of routes and run time. There is no conclusive correlation, as H1-75 has a higher run time than the exact method for multiple test subsets, and the run time decreases from 10 to 12 super-resources even though the number of travel routes increases. Nevertheless, apart from H1–75, the graph shows that the run time decreases with a decreasing branching rate, and hence with a decreasing number of routes. An interesting observation is that even though H1–50 only has a fraction of the routes and reduces the run time by more than 50% compared to the exact model and H1–75, it provides a better solution within 20 minutes, and is within an acceptable range of the solution H1–75 finds within one hour. H1–15 and H1–30 have few routes, which result in very good run times. However, the restricted amount of routes also overly simplifies the problem, and can lead to solutions that are far from optimal, as Figure 7.4 proves.

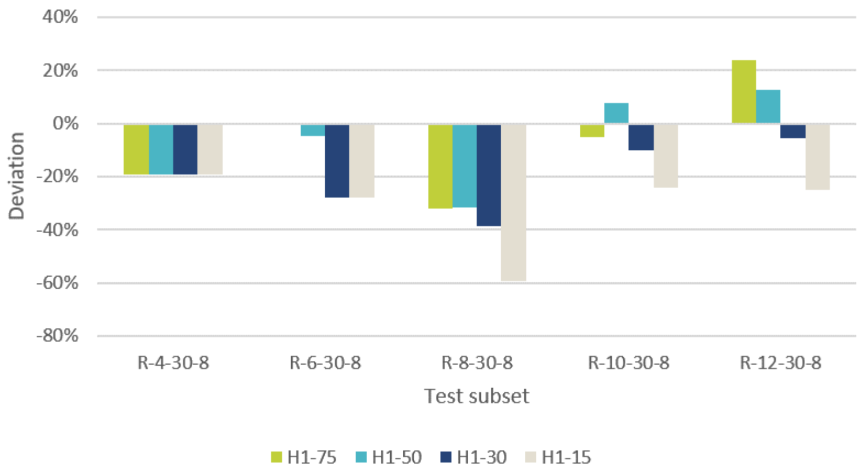


Figure 7.4: First heuristic solutions' deviation from decomposed exact method's best solutions after run time of one hour with an increase in resources.

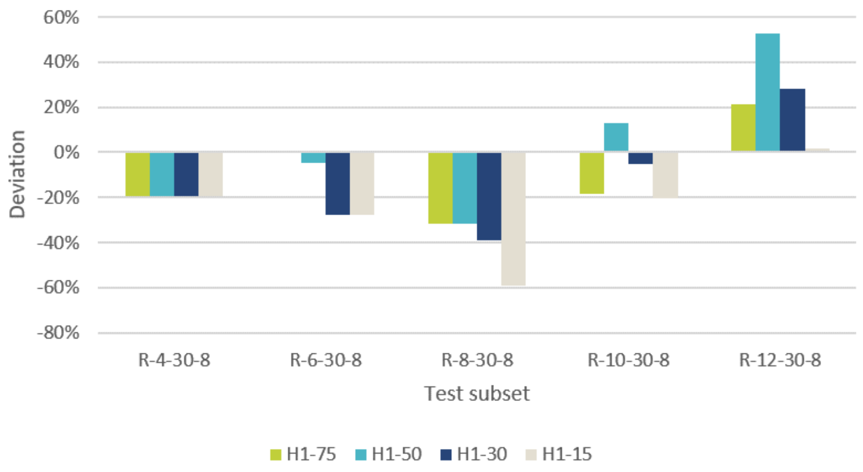


Figure 7.5: First heuristic solutions' deviation from decomposed exact method's best solutions after run time of 20 minutes with an increase in resources.

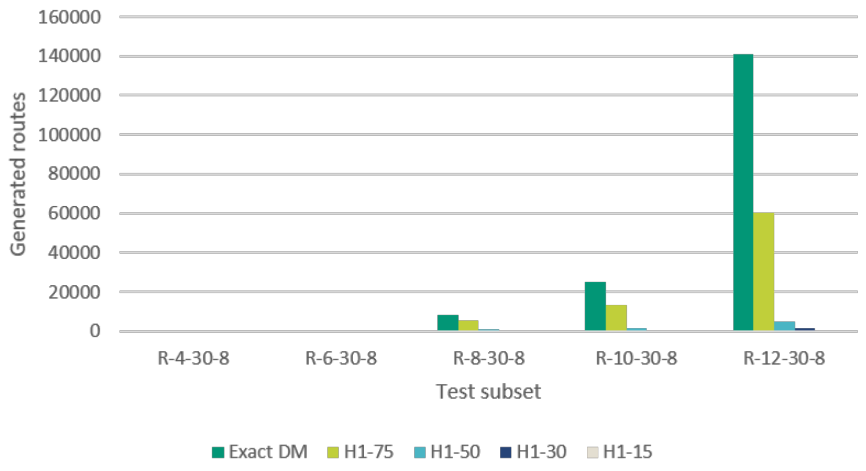


Figure 7.6: Comparison of the number of routes generated by the first heuristic method with an increase in resources.

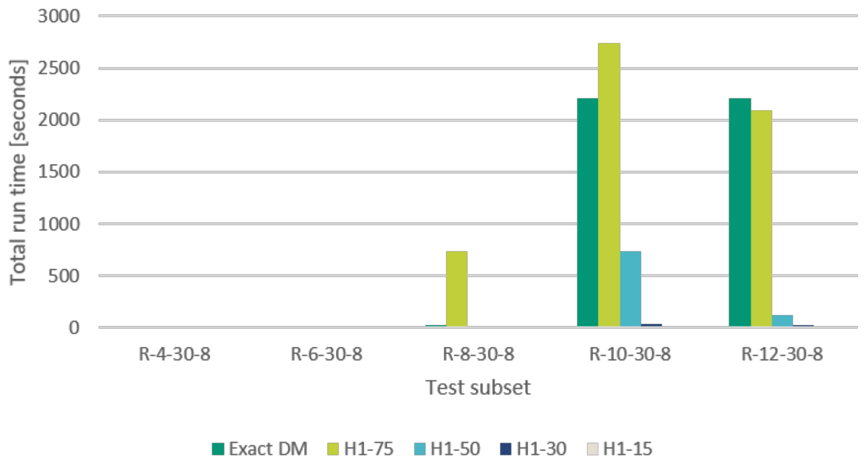


Figure 7.7: Comparison of total run times for the first heuristic approach with an increase in resources.

7.3.2 Increasing the number of tasks

The results for test set 3 are illustrated in figures given in Appendix G. They suggest that for instances including more than 40 tasks, branching rates H1–75, H1–50 and H1–30 provide better solutions than the exact model after an hour of run time. The results also show a decreasing quality in solutions as the branching rate decreases and an increasing number of routes are cut, meaning that travel routes that are part of better solutions are removed. Solutions within 20 minutes have similar results. 75% seems to be the best rate for these five subsets, but might have more trouble with larger numbers of tasks, as the other branching rates seem to be closing the gap for the one hour solution and provide better solutions within 20 minutes, for the largest subset.

Test subset T–6–50–8 is the subset where all branching rates perform better than the exact model, and represents the highest deviation between several of the rates and the exact model. This is also the subset with the highest number of routes, and where the exact model has the most trouble in finding an exact solution, based on its optimality gap of 134.6% (Table 7.8). As the number of routes correspond with the optimality gap for all subsets for the exact model, it is not surprising that it also reflects the performance of the heuristic models. The subsets with a high number of routes also have high run times, but the run times and number of routes are not proportional. Results show that H1–30 has a consistently low run time and provides almost as good solutions as H1–50 for these test subsets. The deviation in solutions between H1–30 and H1–50 might however increase for larger instances. If this is the case, then the trade-off between a short run time and good solution, which H1–30 provides, and a longer run time, but better solution, which H1–50 provides, has to be made.

7.3.3 Summary of the first heuristic decomposition

The analysis of test set 1 indicates that the first heuristic approach, given the right branching rate, provides better solutions in a reasonable amount of time than the exact decomposed model, for larger test instances with an increasing number of super-resources. For test set 1, H1–50 is the most promising, when considering both solution value and run time. H1–75 seems best overall for test set 2 with an increasing number of tasks, but both H1–50 and H1–30 are better for the largest test subset, and might be more promising for even higher numbers of tasks.

Considering all 50 test instances for each branching rate across test subsets, H1–75 reaches optimality in 78% of the cases, H1–50 reaches optimality in 90%, and H1–15 and H1–30 reach optimality for all instances. Note that the optimal value is not the same for all branching rates, and that optimal values for lower rates are in most cases lower. This means that although H1–15 and H1–30 are easier to solve, they might be oversimplified, and hence provide results of lower value.

Regarding the run time, all instances apart from two H1–75 instances, found their best solution within 20 minutes, suggesting that it is needless to run the models for more than 20 minutes. Overall, while this heuristic method does remove some favorable travel routes that are part of exact solutions, reducing the number of routes generated proves an effective method to provide solutions within a shorter computation time.

7.4 Analysis of the second heuristic decomposition

This section provides an analysis of a heuristic decomposition solution approach which uses the second heuristic method to generate routes. The second heuristic method is hereby referred to as H2. This method is described in Section 5.3 and is designed to remove more routes than H1. The reason for testing H2 is to investigate how the second heuristic approach compares to the first approach, and the exact decomposed model. In addition, testing this method can provide some insight into how the different components that make up H2 affect the solution.

This method is only applied to the three largest instances of test set 1 and 3 presented in tables 7.1 and 7.3. The smaller instances are omitted from this analysis because, as the results from Section 7.3 show, the exact decomposed method outperforms the first heuristic method for these instances, and would most likely also outperform the second method.

The second heuristic decomposed solution approach is tested for two different rates of travel route branching: 100% and 75%, referred to as H2–100 and H2–75 respectively. For all cases, the maximum number of super-resources per task is set to two. As described in Table 5.5, the second heuristic method consists of five different components, where a branching strategy based on task value is one of them. Testing a branching rate of 100% would in effect mean that only the other four components of the heuristic actively reduce the number of routes generated. This makes it possible to analyze the effect of the branching strategy, and the other four components separately. Note that because the two methods have different configurations, the same branching rate would result in a lower number of generated routes for the second method than it would the first method. Therefore, branching rates of 30% and 50% are not tested as they are expected to reduce the number of generated routes by an amount that will result in unacceptably low solution values. All test results are given in Table E.5 in Appendix E.

7.4.1 Increasing the number of resources

With an increase in the number of resources, the exact method seems to do better for the smallest instance. For larger instances, this heuristic method provides better solutions within 20 minutes. H2–75 yields better results than H2–100 for the first two instances, but H2–100 has a steady positive trend for solutions provided within 20 minutes. This suggests that H2–100 might be more promising in a practical setting, but it could be due to coincidence.

A comparison between the exact method and H2, as presented in Figure 7.8, confirms that the second heuristic method greatly reduces the number of routes generates. This is most likely the reason both H2–100 and H2–75 have shorter run times than the exact method for the largest subsets. A breakdown of how many routes the different heuristic components remove relative to the exact method is illustrated in Figure 7.9. The branching rate used for this analysis is 75%. It is clear that the other components, and not the branching strategy, remove the majority of generated routes. This is most likely because there are many routes and tasks that the other components can effectively remove, before the branching strategy is put into effect in the travel route generator. A possible conclusion from this observation is that, for an increase in resources, the other components of H2 are very efficient in reducing the number of routes, more so than the branching strategy, and still lead to acceptable solutions. A higher branching rate would lead to a higher amount of routes being removed by the branching strategy, but then probably at the cost of solution value.

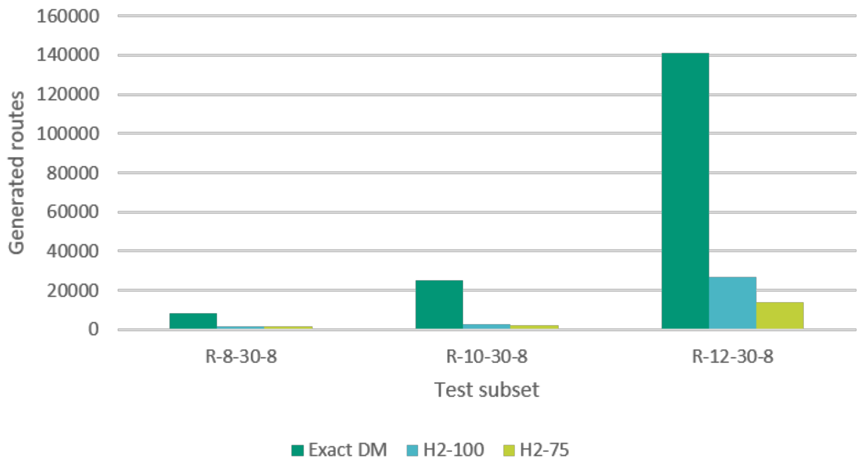


Figure 7.8: Comparison of the number of routes generated by the second heuristic method, with an increase in resources.

7.4.2 Increasing the number of tasks

With an increase in the number of tasks, the second heuristic approach provides better solutions than the DM measured at 20 minutes and an hour, even though the total run times are fairly similar. Of the two branching rates, H2–75 has better solutions for the

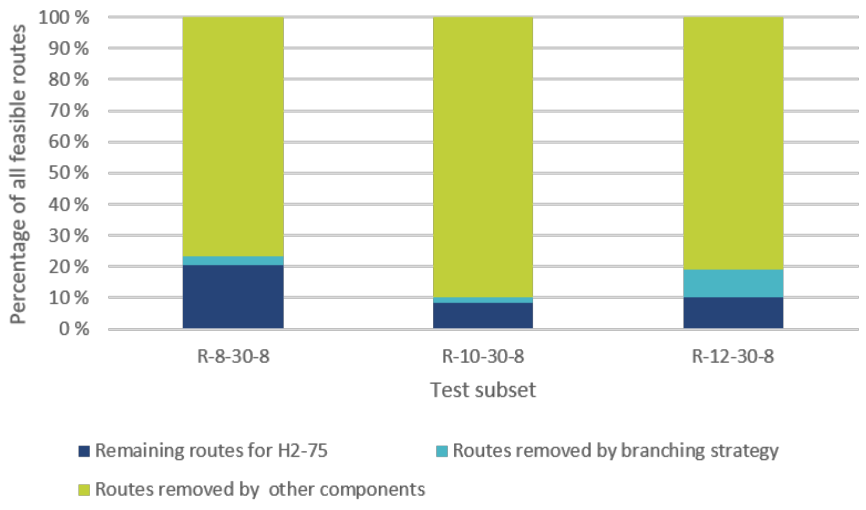


Figure 7.9: Percentage of feasible routes removed by the different heuristic components of H2, with an increase in resources.

two largest instances after an hour of run time, and for all instances with a run time of 20 minutes. H2–75's advantage over H2–100 implies that for problems with a large number of tasks, a branching strategy is effective.

Similar to the case with an increase in resources, Figure 7.10 presents a breakdown of the number of routes the different heuristic components remove relative to the exact method. A branching rate of 75% is still applied. The figure shows that the number of routes generated for subsets T–6–50–8 and T–6–70–8 are markedly reduced by introducing a branching strategy. Subset T–6–60–8 is not reduced to the same extent. An explanation for this could be that the subset already had a relatively low number number of routes before the branching strategy is put in to effect, and therefore the branching strategy has a limited number of options to cut. Another observation based on the same figure is that the other components have a diminishing effect on the number of routes that are generated, with an increase in the number of tasks. This would suggest that for problems with a large number of tasks, a branching rate is an efficient method to reduce the number of generated routes. More so than the other components of H2.

7.4.3 Comparison of the heuristic methods

Table 7.10 compares the results from H1 and H2. With a branching rate of 75%, H2 does, as expected, generate a lower number of routes than H1, for all subsets. Furthermore, H2

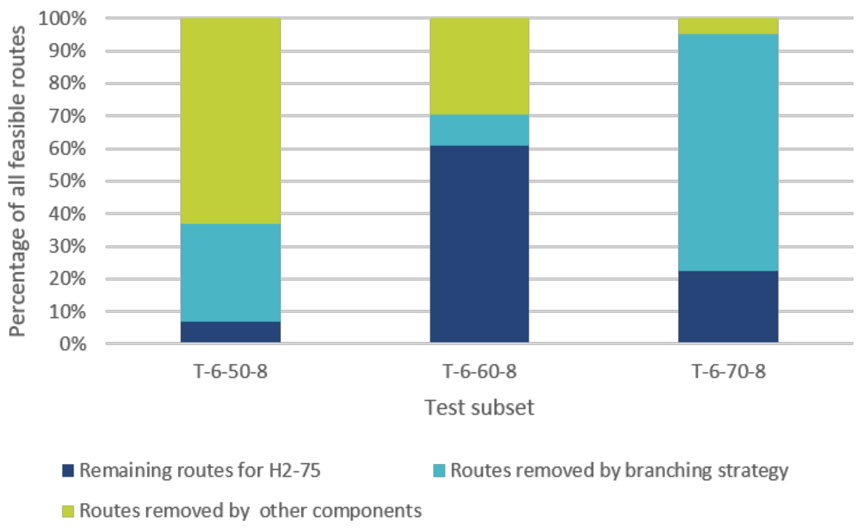


Figure 7.10: Percentage of feasible routes removed by the different heuristic components of H2, with an increase in tasks.

reduces the number of generated routes more for the subsets in test set 1 than in test set 3. This is possibly the reason that the second heuristic approach halves the run time of the first approach, on average, for subsets in test set 1. H2 also provides better solutions than H1, both within 20 minutes and within one hour. For an increase in resources, H2 therefore seems a more efficient method to reduce the number of generated routes. With an increase in tasks, H2 actually leads to a longer run time on average, but does provide better or equal solutions than H1. It is therefore difficult to distinguish which method is better for an increase in tasks.

Table 7.10: Comparing results provided by H1–75 and H2–75.

Subset	**Routes generated**		**Total run time [s]**		**Best solution within 20 min**		**Best solution after an hour**	
	H1-75	H2-75	H1-75	H2-75	H1-75	H2-75	H1-75	H2-75
R-8-30-8	5440	1665	734	726	28	27.9	28	27.9
R-10-30-8	13378	2077	2741	882	56.2	84	68.7	84.4
R-12-30-8	60176	14118	2091	1382	48.6	42.6	67.2	61.6
Average	*26331*	*5953*	*1855*	*997*	*44*	*52*	*55*	*58*
T-6-50-8	70188	33818	1649	2234	91.7	83.3	99.7	111.4
T-6-60-8	15392	14199	141	1577	85.9	76.8	85.9	86.7
T-6-70-8	106642	89762	1678	1582	53.8	71.3	85.2	94.0
Average	*64074*	*45927*	*1156*	*1797*	*77*	*77*	*90*	*97*

7.4.4 Summary of the second heuristic decomposition

A source of error in this analysis is the limited number of subsets that are tested, and therefore, absolute conclusions should not be drawn from this particular analysis. Nevertheless, it is possible to detect certain trends. Based on the results from test set 1 and 3, H2 generates fewer routes, and looks to perform better than the exact method for larger instances, especially for instances with a large number of tasks. Comparing H1 and H2 with a branching rate of 75%, H2 is superior when considering problems with a larger number of resources. For problems with many tasks, it is not possible to make any conclusions of which heuristic method performs better.

Considering the different components that make up H2, the branching strategy seems to be effective when there are a large number of tasks, and the other components seem to be very effective for a large number of resources. Why the other components are effective in reducing the number of generated routes for a larger number of resource is presumably because of the two particular components that look to limit the number of super-resources per task, and divide routes among similar super-resources. Dividing routes among similar super-resources is only effective if the battlegroup in question is big enough to includes several of the same type of super-resources. Therefore, it will only remove routes for instances that have a a high number of resources. Limiting the number of super-resources per task is a component that works by removing all tasks in an instance that cannot be completed by one or two super-resources. In theory, it removes routes for all instances tested here. However, because an increase in resources means that more tasks become possible to complete and more routes become feasible, limiting the number of resources per task becomes increasingly effective for instances with a larger number of resources.

Dividing routes among similar super-resources, and limiting the number of super-resources per task are components that are part of H2, but not H1. Of the instances tested here, only R–12–30–8 has more than one of the same type of super-resource, and hence it is the only instance where dividing routes among similar super-resources reduces the number of generated routes. The subpar solutions provided by H2–75 compared to H1–75 for this test instance, would however suggest that this component removes routes that would otherwise be part of a good solution, and is therefore not an efficient heuristic component. To verify this conclusion would require further testing of additional instances with multiple super-resources of the same type. Limiting the number of super-resources per task is a component that is more difficult to interpret, but there is nothing to suggest that it leads to diminished solutions. It does effectively reduce the number of generated routes, which most likely leads to a shorter run time. Based on this argument, it is a useful heuristic component.

Chapter 8

Concluding Remarks

The models presented in this thesis are extensions of the model presented in the project report by Chaudry and Vermedal (2017), and are intended as a decision support tool for military peacekeeping operations. The problem considered is termed a Peacekeeping-Troops-To-Tasks Problem (PTTP). What distinguishes the PTTP in this thesis from its predecessor in the project report, is 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, unavailable time periods for resources, and that it considers a longer planning period. 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 introduced as an alternative solution method, where possible travel routes are generated a priori. Furthermore, two heuristic solutions methods are proposed, both aiming to generate fewer travel routes to reduce the size of the problem and make it easier to solve. The second heuristic method is an extension of the first, and as such generates fewer travel routes. 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, both heuristic solution methods provide better solutions in the given run times of twenty minutes and of one hour.

Analysis of the largest instances tested for an increase in the number of resources and tasks shows that for the first heuristic approach, a 50% branching rate is efficient. For the second heuristic approach, the computational study indicates that a branching strategy seems especially efficient when having to consider many tasks, while limiting the numbers of super-resources per task seems more effective for instances with a large number of resources. Furthermore, there is a slight indication that dividing routes among similar super-resources removes too many favorable routes, and hence is not a preferable heuristic.

For the instances with the largest number of resources, results indicate that the second heuristic method is more promising than the first heuristic method. For the instances with the largest number of tasks, the results are inconclusive. For both methods, a higher branching rate seems necessary to solve even larger test instances, and the test results prove that this negatively affects the solution value. The heuristic methods therefore have potential for improvement.

Nevertheless, the PTTP seems to rely on the use of a heuristic solution approach in order to provide solutions for instances of realistic proportions, which often include at least 20 super-resources. Even though the solutions provided by a heuristic are most likely not the optimal solutions, FFI has stated that any solution would be of great interest as a starting point for an operation planner. It is then preferable to have a short computation time to enable re-runs where adjustments can be made to the parameter data, and different planning options can be tested and analyzed. With the right use of a heuristic decomposition approach, the computational study in this thesis indicates that the model is able to provide a solution to most problems. Furthermore, it seems sufficient to have a run time of 20 minutes, as the solutions provided rarely improve by running the model for a longer time than this. These results suggest that the model has the potential to become a useful decision support tool for military peacekeeping operation planning.

Chapter 9

Future Research

In this thesis, we seek to develop a model and solution method that solves the PTTP in an efficient manner. A decomposition solution approach combined with a heuristic method for generating routes has proved itself to be promising, solving smaller instances effectively. However, we realize that for the model to be a useful decision support tool for military operation planning, it has to be improved upon to handle larger instances.

Alternative solution methods, such as more advanced heuristics, have the potential to do just this. Research shows that heuristic solution methods such as genetic algorithms have lead to good results for the RCPSP (Tavana et al., 2014; Schutt et al., 2012; Afruzi et al., 2014; Van Peteghem and Vanhoucke, 2010; Afshar-Nadjafi and Majlesi, 2014). Heuristic solution methods combining a genetic algorithm with a decomposition approach such as the ones Debels and Vanhoucke (2007) and Zamani (2011) present, seem particularly promising, and would be an interesting continuation of the study conducted in this thesis. Regarding the generation of travel routes for a decomposed model, alternative search strategies could prove to enhance the performance of such a solution approach.

An alternative to a heuristic approach is to move more of the PTTP to the travel route generation. For example, deciding what tasks to be completed during each location visit when calculating travel routes, will make the optimization easier. On the other hand, it will also greatly increase the number of routes generated, lengthening both the time it takes to generate routes and the time it takes the optimization model to load the text files containing the generated routes. Future research on this strategy would therefore also have to consider programming the travel route generator and optimization model in a more effective manner to save computational time.

A simplification of the model is another viable method of making the model more efficient, and should be considered in future studies. An observation made during the computational study is that the model is very complex, especially regarding task characteristics and how they relate to one another. Though this makes the model more encompassing, the complexity also makes the problem more difficult to solve, and hence negatively affects how the models perform. Thus, all the complicating factors are not necessarily beneficial to model. Removing long and rest tasks from the model would, for example, simplify the model without greatly affecting the effectiveness of operation planning. This is because long and rest tasks are relatively lengthy tasks, which might be more practical to schedule manually.

Extensions of the objective function could also be of interest for future research. The computational results in Chapter 7 suggest that there are some trade-offs to consider in a PTTP. For example, maximizing value does not necessarily lead to more tasks being completed. The value of tasks makes it possible to rank their importance, but it is an imaginary parameter that is difficult to define. Consequently, maximizing value might not lead to the best practical solutions. Maximizing the number of tasks is an alternative objective, but will render task values and skill levels nonfunctional, and hence the model will lose some of its generality and complexity. It may therefore be more fitting to include both value and the number of tasks completed in the objective function. Minimizing idle time for resources during an operation might also be desirable for logistical and economical reasons, and could either be part of a multi-objective function, or a singular objective function.

Discretizing time would be another interesting direction for future research. Regarding time as discrete is a simplification, but in the case of military peacekeeping operations where tasks are lengthy, it might be a reasonable simplification. In fact, Kopanos et al. (2014) finds that time discrete models perform better than continuous models for when there are a large number of tasks to complete, which is the case for the PTTP.

Bibliography

Adhau, S., Mittal, M. L., and Mittal, A. (2013). A multi-agent system for decentralized multi-project scheduling with resource transfers. *International Journal of Production Economics*, 146(2):646–661.

Afruzi, E. N., Najafi, A. A., Roghanian, E., and Mazinani, M. (2014). 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*, 50:80–96.

Afshar-Nadjafi, B. and Majlesi, M. (2014). Resource constrained project scheduling problem with setup times after preemptive processes. *Computers & Chemical Engineering*, 69:16–25.

Al-Anzi, F. S., Al-Zame, K., and Allahverdi, A. (2010). Weighted multi-skill resources project scheduling. *Journal of Software Engineering and Applications*, 3(12):1125.

Almeida, B. F., Correia, I., and da Gama, F. (2016). Priority-based heuristics for the multi-skill resource constrained project scheduling problem. *Expert Systems with Applications*, 57:91–103.

Batptiste, P., Pape, C. L., and Nuijten, W. (1999). Satisfiability tests and time-bound adjustments for cumulative scheduling problems. *Annals of Operations Research*, 92(0):305–333.

Bellenguez, O. and Néron, E. (2005). Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills. In *Practice and Theory of Automated Timetabling V*, pages 229–243. Springer.

Bianco, L., Dell'Olmo, P., and Speranza, M. (1998). Heuristics for multimode scheduling problems with dedicated resources. *European Journal of Operational*, 107(2):260–271.

Chaudry, N. and Vermedal, I. (2017). Optimization of the troops-to-tasks assignment in military peacekeeping operations.

Debels, D. and Vanhoucke, M. (2007). A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. *Operations Research*, 55(3):457–469.

Drexl, A., Nissen, R., Patterson, J. H., and Salewski, F. (2000). Progen/πx–an instance generator for resource-constrained project scheduling problems with partially renewable resources and further extensions. *European Journal of Operational Research*, 125(1):59–72.

Eide (2001). The peacekeeping challenge. `https://www.nato.int/docu/review/2001/Peacekeeping-Challenge/Peacekeeping-past-present/EN/index.htm`. Accessed: 2018-06-07.

Fauske, M. F. (2015). Optimizing the troops-to-tasks problem in military operations planning. *Military Operations Research*, 20(4):49–57. doi: `10.5711/1082598320449`.

Fauske, M. F. (2017). Using a genetic algorithm to solve the troops-to-tasks problem in military operations planning. *The Journal of Defense Modeling and Simulation*, 14(4):439–446.

Gacias, B., Artigues, C., and Lopez, P. (2010). Parallel machine scheduling with precedence constraints and setup times. *Computers & Operations Research*, 37(12):2141–2151.

Hartmann, S. and Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of operational research*, 207(1):1–14.

Heimerl, C. and Kolisch, R. (2010). Scheduling and staffing multiple projects with a multi-skilled workforce. *OR Spectrum*, 32(2):343–368.

H'Mida, F. and Lopez, P. (2013). Multi-site scheduling under production and transportation constraints. *International Journal of Computer Integrated Manufacturing*, 26(3):252–266.

Kopanos, G. M., Kyriakidis, T. S., and Georgiadis, M. C. (2014). New continous-time and discrete-time mathematical formulations for resource-constrained project scheduling problems. *Computers & Chemical Engineering*, 68:96–106.

Krüger, D. and Scholl, A. (2010). Managing and modelling general resource transfer in (multi-)project scheduling. *OR spectrum*, 32(2):369–394.

Laurent, A., Deroussi, L., Grangeon, N., and Norre, S. (2017). A new extension of the rcpsp in a multi-site context: Mathematical model and metaheuristics. *Computers & Industrial Engineering*, 112:634–644.

Liu, S.-S. and Wang, C.-J. (2011). Optimizing project selection and scheduling problems with time-dependent resource constraints. *Automation in Construction*, 20(8):1110–1119.

Moukrim, A., Quilliot, A., and Toussaint, H. (2015). Branch and price for preemptive and non preemptive rcpsp based on interval orders on precedence graphs. In *Recent Advances in Computational Optimization*, pages 85–106. Springer.

Myszkowski, P. B., Skowroński, M. E., Olech, Å. P., and OślizÅ‚o, K. (2015). Hybrid ant colony optimization in solving multi-skill resource-constrained project scheduling problem. *Soft Computing*, 19(12):3599–3619.

NATO (2016). Operations and missions: past and present. `https://www.nato.int/cps/en/natohq/topics_52060.htm`. Accessed: 2018-08-07.

Pollack-Johnson, B. and Liberatore, M. J. (2006). Incorporating quality considerations into project time/cost tradeoff analysis and decision making. *IEEE Transactions on Engineering Management*, 53:534–542.

Ranjbar, M., De Reyck, B., and Kianfar, F. (2009). A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. *European Journal of Operational Research*, 193(1):35–48.

Rihm, T. and Trautmann, N. (2014). A mip-based decomposition heuristic for resource-constrained project scheduling. In *14th International Conference on Project Management and Scheduling*, page 193.

Santos, M. and Tereso, A. (2011). On the multi-mode, multi-skill resource constrained project scheduling problem - a software application. In *Soft Computing in Industrial Applications*, pages 239–248. Springer.

Schutt, A., Chu, G., Stuckey, P. J., and Wallace, M. G. (2012). Maximising the net present value for resource-constrained project scheduling. In *CPAIOR*, pages 362–378. Springer.

Section for Security Policy and North America (2017). FNs fredsoperasjoner. `https://www.regjeringen.no/no/tema/utenrikssaker/sikkerhetspolitikk/fredsbevarende_operasjoner/fn_fredsoperasjoner/id86766/`. Accessed: 2017-12-03.

Sprecher, A. (2002). Network decomposition techniques for resource-constrained project scheduling. *Journal of the Operational Research Society*, 53(4):405–414.

Tavana, M., Abtahi, A.-R., and Khalili-Damghani, K. (2014). A new multi-objective multi-mode model for solving preemptive time–cost–quality trade-off project scheduling problems. *Expert Systems with Applications*, 41(4):1830–1846.

The Norwegian Armed Forces. Mediearkiv. `https://mediearkiv.forsvaret.no/fotoweb/`. Accessed: 2017-12-08.

Tian, J., Liu, Z., and Yu, W. (2014). An approach with decomposition on time windows for resource-constrained project scheduling. In *Control and Decision Conference (2014 CCDC), The 26th Chinese*, pages 4897–4903. IEEE.

Trautmann, N. and Schwindt, C. (2005). A minlp/rcpsp decomposition approach for the short-term planning of batch production. In *Computer Aided Chemical Engineering*, volume 20, pages 1309–1314. Elsevier.

United Nations (2017a). How we are founded. `https://peacekeeping.un.org/en/how-we-are-funded`. Accessed: 2017-06-07.

United Nations (2017b). Our history. `https://peacekeeping.un.org/en/our-history`. Accessed: 2018-06-07.

Van Peteghem, V. and Vanhoucke, M. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research*, 201(2):409–418.

Waligóra, G. (2014). Discrete-continuous project scheduling with discounted cash inflows and various payment models - a review of recent results. *Annals of Operations Research*, 213(1):319–340.

Wang, L. and Zheng, X.-l. (2017). A knowledge-guided multi-objective fruit fly optimization algorithm for the multi-skill resource constrained project scheduling problem. *Swarm and Evolutionary Computation*. doi: `10.1016/j.swevo.2017.06.001`.

Zamani, R. (2011). A hybrid decomposition procedure for scheduling projects under multiple resource constraints. *Operational Research*, 11(1):93–111.

Appendix A

Deduction of *divisible task duration* constraints

This appendix presents a deduction of constraints 4.43 in Section 4.3, Chapter 4. These constraints limit the time it takes to complete a divisible task taking into account assigned sub-resources and capacity.

Constraints 4.43:

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

The duration of a divisible task is assumed to be linearly proportional to its assigned capacity. For a task to have a duration, i.e. be completed, its minimum capacity requirement, $\sum_{s \in \mathcal{S}} C^{REQ}$, has to be met. At this amount of assigned capacity, the task will take the standard amount of time to complete, U_p^{TASK}. This relationship is represented by the point (z_1, y_1) in Figure A.1. Point (z_2, y_2) represents the relation between the minimum duration a task can have, $H^{MIN} U_p^{TASK}$, and the maximum capacity a task can wield, $\sum_{s \in \mathcal{S}} C_p^{MAX} C_{ps}^{REQ}$. Knowing these two points, it is possible to linearly interpolate between them and deduce a linear function for the duration of a divisible task.

$$y = t_p^{END} - t_p^{START} \qquad p \in \mathcal{P}^{\mathcal{DIV}} \tag{A.2}$$

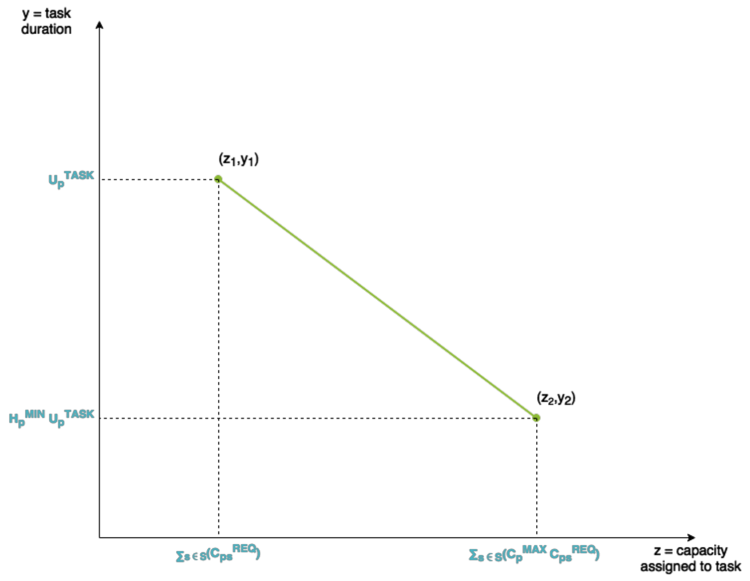


Figure A.1: Relation between divisible task duration and assigned capacity.

$$z = \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}} \sum_{l \in \mathcal{L}} C_{ksl}^{RES} H_{ps} x_{kp} \qquad p \in \mathcal{P}^{\mathcal{DIV}} \tag{A.3}$$

$$y = \alpha z + \beta \tag{A.4}$$

Equations (A.2)-(A.3) apply to every divisible task. (A.4) is a standard linear equation where α is the slope of the function and β is the intersection point of the vertical axis. The slope can be calculated using the aforementioned points, (z_1, y_1) and (z_2, y_2), as presented in (A.5). Using the slope and point (z_1, y_1), β is given in (A.6). Equation (A.1) is the result of combining equations (A.2)-(A.6). q_p is multiplied with the first terms to ensure that t_p^{END} is not forces to be positive for tasks that are not completed.

$$\begin{aligned} \alpha &= \frac{y_2 - y_1}{z_2 - z_1} \\ &= \frac{U_p^{TASK}(U_p^{MIN} - 1)}{\sum_{s \in \mathcal{S}} C_{ps}^{REQ}(C_p^{MAX} - 1)} \end{aligned} \tag{A.5}$$

$$\begin{aligned} \beta &= y_1 - \alpha z_1 \\ &= U_p^{TASK} - \frac{U_p^{TASK}(U_p^{MIN} - 1)}{\sum_{s \in \mathcal{S}} C_{ps}^{REQ}(C_p^{MAX} - 1)} \sum_{s \in \mathcal{S}} C_{ps}^{REQ} \end{aligned} \tag{A.6}$$

Appendix B

Decomposed Model of the PTTP

B.1 Definitions

Indices

g	Super-resource
k	Sub-resource
p	Task
i, j	Location
s	Skill
l	Skill level
m, n	Visit number of a location
d	Group of duplicate tasks
r	Traveling route

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 $\lvert I \rvert$
$\mathcal{P}$	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 at camp
$\mathcal{P}^{\mathcal{SLEEP}}$	Subset of sleep tasks at camp
$\mathcal{P}_d^{\mathcal{DUP}}$	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
$\mathcal{R}_g$	Set of possible routes for super-resource g

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
Y_{rgimjn}^{TRAVEL}	1 if for route r, super-resource g travels directly between its m^{th} visit at location i and n^{th} visit at location j, 0 otherwise
Y_{rgim}^{LOC}	1 if for route r, super-resource g visits location i for the m^{th} time, 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
$o_{kpp'}$	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
λ_{rg}	1 if super-resource g travels route r

B.2 Optimization model

B.2.1 Objective function

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

The objective function (B.1) maximizes the total value achieved by completing tasks in the military peacekeeping operation. 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 uncompleted tasks, the proportions will be zero, ensuring that no value is added for these tasks.

B.2.2 Constraints and requirements

Super-resource network constraints

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

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

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

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

Constraints (B.2) and (B.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 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 (B.4) and (B.5) ensure 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 long and rest task with a direct start condition. $d*$ in (B.4) and (B.5) represent duplicate groups of only long tasks.

$$b_{gim} + T_{gij}^{TRAVEL} \leq a_{gjn} + M(1 - \sum_{r \in \mathcal{R}_g} \lambda_{rg} Y_{rgimjn}^{TRAVEL})$$
$$g \in \mathcal{G}, i, j \in \mathcal{I}, m \in \mathcal{N}_i, n \in \mathcal{N}_j, i \neq j \tag{B.6}$$

$$\sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{N}_i} \sum_{j \in \mathcal{I}} \sum_{n \in \mathcal{N}_j} \sum_{r \in \mathcal{R}_g} \lambda_{rg} T_{gij}^{TRAVEL} Y_{rgimjn}^{TRAVEL} + \sum_{i \in \mathcal{I}} \sum_{m \in \mathcal{N}_i} (b_{gim} - a_{gim}) = \overline{T}$$
$$g \in \mathcal{G} \tag{B.7}$$

Constraints (B.6) specify that if a super-resource travels directly between two locations, then it can only arrive at a location a period at least equal to the travel time T_{gij}^{TRAVEL} after it has left from its previous location. Big M for constraints (B.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 (B.7) ensure that, at all times, a super-resource is either traveling or at a location.

$$a_{gim} \leq \sum_{r\in\mathcal{R}_g} \lambda_{rg}\overline{T}Y_{rgim}^{LOC} \qquad g\in\mathcal{G}, i\in I, m\in\mathcal{N}_i \tag{B.8}$$

$$b_{gim} \leq \sum_{r\in\mathcal{R}_g} \lambda_{rg}\overline{T}Y_{rgim}^{LOC} \qquad g\in\mathcal{G}, i\in I, m\in\mathcal{N}_i \tag{B.9}$$

$$\sum_{r\in\mathcal{R}_g} \lambda_{rg}Y_{rgim}^{LOC} \leq \sum_{k\in\mathcal{K}_g}\sum_{p\in\mathcal{P}_i^{\mathcal{L}OC}} \gamma_{kpm} \qquad g\in\mathcal{G}, i\in\mathcal{I}\backslash(1,|I|), m\in\mathcal{N}_i \tag{B.10}$$

$$x_{kp} \leq \sum_{m\in\mathcal{N}_i}\sum_{r\in\mathcal{R}_g} \lambda_{rg}Y_{rgim}^{LOC} \qquad g\in\mathcal{G}, i\in\mathcal{I}, k\in\mathcal{K}_g, p\in\mathcal{P}_i^{\mathcal{L}OC} \tag{B.11}$$

$$\sum_{m\in\mathcal{N}_i} \gamma_{kpm} = x_{kp} \qquad i\in\mathcal{I}, k\in\mathcal{K}, p\in\mathcal{P}_i^{\mathcal{L}OC} \tag{B.12}$$

Constraints (B.8) and (B.9) assert that a super-resource cannot arrive or leave a location it is not visiting. Constraints (B.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 (B.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 (B.12) couple the variables γ and x.

$$\sum_{r\in\mathcal{R}_g} \lambda_{rg} = 1 \qquad g\in\mathcal{G} \tag{B.13}$$

Constraints (B.13) state that for all super-resources, exactly one route must be selected for the given set of possible routes.

Resource capacity constraints

$$\sum_{s\in\mathcal{S}} C_{ps}^{REQ} e_{pl} = \sum_{k\in\mathcal{K}}\sum_{s\in\mathcal{S}} w_{kpsl} \qquad p\in\mathcal{P}, l\in\mathcal{L} \tag{B.14}$$

$$\sum_{k\in\mathcal{K}}\sum_{l\in\mathcal{L}} w_{kpsl} = C^{REQ}_{ps} q_p \qquad p \in \mathcal{P}, s \in \mathcal{S} \tag{B.15}$$

$$w_{kpsl} \leq C^{RES}_{ksl} x_{kp} \qquad k \in \mathcal{K}, p \in \mathcal{P}, s \in \mathcal{S}, l \in \mathcal{L} \tag{B.16}$$

Constraints (B.14) provide the proportion of a task that is carried out with each skill level. Constraints (B.15) 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 (B.16) state that a resource's contribution to a task cannot be more than its own capacity.

$$x_{kp} \leq \sum_{s\in\mathcal{S}}\sum_{l\in\mathcal{L}} w_{kpsl} \qquad k \in \mathcal{K}, p \in \mathcal{P}\backslash\mathcal{P}^{\mathcal{DIV}} \tag{B.17}$$

$$x_{kp} \leq \sum_{s\in\mathcal{S}}\sum_{l\in\mathcal{L}} C^{RES}_{ksl} C^{REQ}_{ps} \qquad k \in \mathcal{K}, p \in \mathcal{P}^{\mathcal{DIV}} \tag{B.18}$$

Constraints (B.17) and (B.18) forbid the assignment of resources to tasks which they cannot undertake. Constraints (B.17) 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 (B.15) and (B.16) do not provide any restrictions to the skill requirement for resource k to be assigned to tasks p. Constraints (B.18) 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

$$\sum_{k\in\mathcal{K}} x_{kp} \geq q_p \qquad p \in \mathcal{P} \tag{B.19}$$

$$\sum_{k\in\mathcal{K}} x_{kp} \leq \overline{R} q_p \qquad p \in \mathcal{P}\backslash\mathcal{P}^{\mathcal{SLEEP}} \tag{B.20}$$

$$\sum_{p\in\mathcal{P}^{\mathcal{DUP}}_d} q_p \leq 1 \qquad d \in \mathcal{D} \tag{B.21}$$

Constraints (B.19) and (B.20) 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 (B.20) also limit the number of resources that can work on a single task. Sleep tasks are not subject to this limit. Constraints (B.21) 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.

$$\sum_{g \in \mathcal{G}^{\mathcal{ARMY}}} u_g = 1 \tag{B.22}$$

$$x_{kp} \leq 1 - u_g \qquad g \in \mathcal{G}^{\mathcal{ARMY}}, k \in \mathcal{K}_g, p \in \mathcal{P} \backslash \mathcal{P}^{\mathcal{SLEEP}} \tag{B.23}$$

Constraint (B.22) ensures that one army super-resource is assigned to the security post, and constraints (B.23) 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

$$t_p^{END} - \overline{T}(2 - (x_{kp} + x_{kp'})) \leq t_{p'}^{START} + \overline{T}(1 - \delta_{pp'})$$
$$k \in \mathcal{K}, p \in \mathcal{P}, p' \in \mathcal{P}^{\mathcal{E}}, p \neq p' \tag{B.24}$$

$$t_{p'}^{END} - \overline{T}(2 - (x_{kp} + x_{kp'})) \leq t_p^{START} + \overline{T}\delta_{pp'}$$
$$k \in \mathcal{K}, p \in \mathcal{P}, p' \in \mathcal{P}^{\mathcal{E}}, p \neq p' \tag{B.25}$$

Constraints (B.24) and (B.25) 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

$$x_{kp} + \sum_{p' \in \mathcal{P}^{\mathcal{LONG}}} o_{kpp'} \geq 1 \qquad g \in \mathcal{G}^{\mathcal{ARMY}}, k \in \mathcal{K}_g, p \in \mathcal{P}^{\mathcal{SLEEP}} \tag{B.26}$$

Constraints (B.26) 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.

$$t_p^{START} - M(1 - o_{kpp'}) \leq t_{p'}^{END} \qquad k \in \mathcal{K}, p \in \mathcal{P}^{\mathcal{SLEEP}}, p' \in \mathcal{P}^{\mathcal{LONG}} \tag{B.27}$$

$$t_{p'}^{START} - M(1 - o_{kpp'}) \leq t_p^{END} \qquad k \in \mathcal{K}, p \in \mathcal{P}^{\mathcal{SLEEP}}, p' \in \mathcal{P}^{\mathcal{LONG}} \tag{B.28}$$

$$o_{kpp'} \leq \sum_{k' \in \mathcal{K}_g} x_{k'p'} \qquad g \in \mathcal{G}^{\mathcal{ARMY}}, k \in \mathcal{K}_g, p \in \mathcal{P}^{\mathcal{SLEEP}}, p' \in \mathcal{P}^{\mathcal{LONG}} \tag{B.29}$$

Constraints (B.27) – (B.29) 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 (B.27) and (B.28) 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 (B.29) 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 (B.27) and (B.28) equals the sum of the duration of the shortest sleep task and the shortest long task, subtracted from the end time $\overline{T}$.

Precedence constraints

$$\sum_{p' \in \mathcal{P}_{d'}^{\mathcal{DUP}}} q_{p'} \leq \sum_{p \in \mathcal{P}_d^{\mathcal{DUP}}} q_p \qquad d, d' \in \mathcal{D}, d \neq d', F_{dd'}^{PREC} = 1 \tag{B.30}$$

$$t_{p'}^{START} + \overline{T}(1 - q_{p'}) \geq t_p^{END}$$
$$d, d' \in \mathcal{D}, p \in \mathcal{P}_d^{\mathcal{DUP}}, p' \in \mathcal{P}_{d'}^{\mathcal{DUP}}, d \neq d', F_{dd'}^{PREC} = 1 \tag{B.31}$$

Constraints (B.30) 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 (B.31) set the start time for task p' after the end time of task p. Constraints (B.31) are not to be binding unless task p' is completed, hence the term $\overline{T}(1 - q_{p'})$.

Connected task and direct start constraints

$$\sum_{p \in \mathcal{P}_d^{\mathcal{DUP}}} x_{kp} \leq \sum_{p' \in \mathcal{P}_{d'}^{\mathcal{DUP}}} x_{kp'}$$
$$g \in \mathcal{G}^{\mathcal{ARMY}}, k \in \mathcal{K}_g, d, d' \in \mathcal{D}, d \neq d', F_{dd'}^{CON} = 1 \tag{B.32}$$

$$x_{kp} = x_{k'p} \qquad g \in \mathcal{G}^{\mathcal{ARMY}}, k, k' \in \mathcal{K}_g, p \in \mathcal{P}^{\mathcal{REST}} \tag{B.33}$$

Constraints (B.32) deal with connected tasks, ensuring that 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 $F_{d'd}^{CON}$ equals 1. When one of the connected tasks is rest task p', all resources assigned to task p must be assigned to p', but the opposite is not true, and more resources may be assigned to p' than p. Constraints (B.33) forces all or no sub-resources in a super-resource to be assigned to rest task p.

$$\begin{aligned} t_p^{END} + F_{dd'}^{CON} T_{gij}^{TRAVEL} \geq t_{p'}^{START} \qquad & g \in \mathcal{G}^{\mathcal{ARMY}}, i, j \in \mathcal{I} \\ d, d' \in \mathcal{D}, p \in (\mathcal{P}_d^{\mathcal{DUP}} \cap \mathcal{P}_i^{\mathcal{LOC}}), p' \in (\mathcal{P}_{d'}^{\mathcal{DUP}} \cap \mathcal{P}_j^{\mathcal{LOC}}), & d \neq d', F_{dd'}^{DIR} = 1 \end{aligned} \tag{B.34}$$

For some tasks, there is a requirement that another task starts directly after the first task is completed. Constraints (B.34) ensure that these requirements are fulfilled, taking into account travel time between the tasks' locations.

Time scheduling constraints

$$t_p^{START} \geq T_p^{RL} q_p \qquad p \in \mathcal{P} \tag{B.35}$$

$$t_p^{END} \leq T_p^{DDL} q_p \qquad p \in \mathcal{P} \tag{B.36}$$

If a task is realized, constraints (B.35) make sure task p starts after its release time R_p, and constraints (B.36) ensure its completion before its deadline D_p.

$$t_p^{END} \leq t_p^{START} + T_p^{TASK} q_p \qquad p \in \mathcal{P} \tag{B.37}$$

$$t_p^{END} \geq t_p^{START} + T_p^{MIN} T_p^{TASK} q_p \qquad p \in \mathcal{P} \tag{B.38}$$

$$\begin{aligned} t_p^{END} \geq t_p^{START} + T_p^{TASK} \left(\frac{C_p^{MAX} - T_p^{MIN}}{C_p^{MAX} - 1} q_p \right) \\ + T_p^{TASK} \left(\frac{T_p^{MIN} - 1}{\sum_{s \in \mathcal{S}} C_{ps}^{REQ} (C_p^{MAX} - 1)} \sum_{k \in \mathcal{K}} \sum_{s \in \mathcal{S}} \sum_{l \in \mathcal{L}} C_{ksl}^{RES} H_{ps} x_{kp} \right) \qquad p \in \mathcal{P}^{\mathcal{DIV}} \end{aligned} \tag{B.39}$$

Constraints (B.37) - (B.39) handle task duration and ensure that tasks are completed in a continuous fashion. Constraints (B.37) make sure that completing a task does not take longer than necessary, while constraints (B.38) and (B.39) 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 (B.39) set the end time of task p according to this. A deduction of constraints (B.39) is presented in Appendix A.

Non-negativity constraints

$$w_{kpsl} \geq 0 \qquad k \in \mathcal{K}, p \in \mathcal{P}, s \in \mathcal{S}, l \in \mathcal{L} \tag{B.40}$$

$$e_{pl} \geq 0 \qquad p \in \mathcal{P}, l \in \mathcal{L} \tag{B.41}$$

$$t_p^{START} \geq 0 \qquad p \in \mathcal{P} \tag{B.42}$$

$$t_p^{END} \geq 0 \qquad p \in \mathcal{P} \tag{B.43}$$

$$a_{gim} \geq 0 \qquad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i \tag{B.44}$$

$$b_{gim} \geq 0 \qquad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i \tag{B.45}$$

Binary requirements

$$x_{kp} \in \{0,1\} \qquad k \in \mathcal{K}, p \in \mathcal{P} \tag{B.46}$$

$$q_p \in \{0,1\} \qquad p \in \mathcal{P} \tag{B.47}$$

$$u_g \in \{0,1\} \qquad g \in \mathcal{G} \tag{B.48}$$

$$o_{kpp'} \in \{0,1\} \qquad k \in \mathcal{K}, p, p' \in \mathcal{P} \tag{B.49}$$

$$\delta_{pp'} \in \{0,1\} \qquad p, p' \in \mathcal{P} \tag{B.50}$$

$$\gamma_{kpm} \in \{0,1\} \quad i \in \mathcal{I}, k \in \mathcal{K}, m \in \mathcal{N}_i, p \in \mathcal{P}_i^{\mathcal{LOC}} \tag{B.51}$$

$$\theta_{gimp} \in \{0,1\} \quad g \in \mathcal{G}, i \in \mathcal{I}, m \in \mathcal{N}_i, p \in \mathcal{P}^{\mathcal{SLEEP}} \tag{B.52}$$

$$\lambda_{rg} \in \{0,1\} \quad g \in \mathcal{G}, r \in \mathcal{R}_g \tag{B.53}$$

Constraints (B.40) - (B.45) ensure non-negativity for continuous variables. Constraints (B.46) - (B.53) enforce binary values for all binary variables.

Appendix C

Battlegroup description

This appendix provides a description of the battlegroup that is used as a basis for generating test instances in this thesis, as referred to in Chapter 6. Table C.1 presents the skill capacities resources have, which is dependent on the type of resource they are. Sleep is included as a skill because it is modelled as such in the mathematical models presented in this thesis. All resources have a sleep skill. There is no value assigned to it in the table because the value is dependent on the number of resources in a test instance. Table C.2 presents the super-resources and their respective sub-resources and sub-resource types.

Table C.1: Description of skill capacities according to sub-resource type.

		Sub-resource type										
Skill		1	2	3	4	5	6	7	8	9	10	11
1	Sleep	x	x	x	x	x	x	x	x	x	x	x
2	Fly	0	0	0	0	0	3	3	0	0	0	0
3	Planning	2	1	0	0	0	0	0	0	0	0	0
4	Coordinating	2	0	0	0	0	0	0	0	0	0	0
5	Attacking	0	0	2	0	0	0	1	2	0	0	0
6	Explosives	0	0	0	2	0	0	0	0	0	0	0
7	Run headquarter	0	3	0	0	0	0	0	0	0	0	0
8	Bomb disposal	0	0	0	0	0	0	0	2	0	0	0
9	Anti-aircraft	0	0	0	0	0	0	0	0	0	0	3
10	Navigation	0	0	0	3	0	0	0	0	0	0	0
11	Demining	0	0	0	3	0	0	0	0	0	0	0
12	Sanitary	0	0	0	0	2	0	0	0	0	0	0
13	Medical	0	0	0	0	3	2	0	0	0	0	0
14	Human transport	0	0	0	0	2	3	0	0	0	1	0
15	Maintenance	0	0	1	0	0	0	0	2	0	0	0
16	Logistics	0	0	0	0	0	0	3	0	2	0	0
17	Support	0	0	0	0	0	0	0	0	3	2	0
18	Security	0	0	2	0	0	0	1	2	0	0	1
19	General	0	0	2	2	2	0	0	2	2	2	0
20	Patrol	0	0	2	1	0	0	0	0	0	0	0

Table C.2: Description of super-resources and their respective sub-resources.

Super-resource	Index (g)	Sub-resources	Index (k)	SB type
Infantry company headquarter 1	1	Infantry company headquarter 1	1	1
Headquarter platoon	2	Headquarter platoon	2	2
Infantry platoon 1	3	Infantry section 1.1	3	3
		Infantry section 1.2	4	3
		Infantry section 1.3	5	3
Reconnaissance team 1	4	Reconnaissance team 1	6	4
Medical platoon 1	5	Medical section 1.1	7	5
		Medical section 1.2	8	5
		Medical section 1.3	9	5
Sanitary helicopter	6	Sanitary helicopter	10	6
Transport helicopter	7	Transport helicopter	11	7
Mortar platoon	8	Mortar section 1.1	12	8
		Mortar section 1.2	13	8
		Mortar section 1.3	14	8
		Mortar section 1.4	15	8
Combat service support platoon	9	Combined maintenance team 1.1	16	9
		Combined maintenance team 1.2	17	9
		Combined maintenance team 1.3	18	9
		Recovery team 1.1	19	10
		Recovery team 1.2	20	10
Anti-aircraft platoon	10	Anti-aircraft 1.1	21	11
		Anti-aircraft 1.2	22	11
		Anti-aircraft 1.3	23	11
		Anti-aircraft 1.4	24	11
Infantry platoon 2	11	Infantry section 2.1	25	3
		Infantry section 2.2	26	3
		Infantry section 2.3	27	3
Medical platoon 2	12	Medical section 2.1	28	5
		Medical section 2.2	29	5
		Medical section 2.3	30	5
Infantry platoon 3	13	Infantry section 3.1	31	3
		Infantry section 3.2	32	3
		Infantry section 3.3	33	3
Reconnaissance team 2	14	Reconnaissance team 2	34	4
Infantry company headquarter 2	15	Infantry company headquarter 2	35	1
Infantry platoon 4	16	Infantry section 4.1	36	3
		Infantry section 4.2	37	3
		Infantry section 4.3	38	3
Infantry platoon 5	17	Infantry section 5.1	39	3
		Infantry section 5.2	40	3
		Infantry section 5.3	41	3
Reconnaissance team 3	18	Reconnaissance team 3	42	4
Infantry platoon 6	19	Infantry section 6.1	43	3
		Infantry section 6.2	44	3
		Infantry section 6.3	45	3
Medical platoon 3	20	Medical section 3.1	46	5
		Medical section 3.2	47	5
		Medical section 3.3	48	5
Reconnaissance team 4	21	Reconnaissance team 4	49	4

Appendix D

Adjustable parameter values for instance generator

This appendix presents the adjustable input data the instance generator in Chapter 6 requires to create test instances. The values for the parameters that are unaltered for all instances are also given. The parameters that are kept alterable are not given a value here.

Table D.1: Adjustable parameters needed to create a test instance.

Adjustable Parameter	**Description**	**Value**	**Equivalent to set or parameter**
nSR	Number of super-resources	-	$\mathcal{G}$
nDuplicates	Number of unique tasks	-	$\mathcal{D}$
nTasks	Number of total tasks including duplicates	-	$\mathcal{P}$
nLoc	Number of locations	-	$\mathcal{I}$
nEndTime	Planning time period in hours	168	$\overline{T}$
nResourcesPerTask	the maximum number of sub-resources that can be assigned to a task	3	$\overline{R}$
long_pntg	Probability of a task being long	10%	
exc_pntg	Probability of a task being exclusive	50%	
prec_pntg	Probability of a task having precedence over other tasks	10%	
con_pntg	Probability of a task being connected to other tasks	10%	
dir_pntg	Probability of a task having to having direct starts after other task	10%	

Adjustable Parameter	**Description**	**Value**	**Equivalent to set or parameter**
div_pntg	Probability of a task being divisible	10%	
tmin_pntg	The minimum time a divisible task can take as a percentage of the regular time	50%	
capres_max_pntg	The maximum capacity that can be assigned to a divisible task as a percentage of the regular capacity	3	
ttask_min	The minimum time any task can take	2	
pSuff	The probability of a resources skill being at a sufficient level	25%	
value_max	The maximum value a task can have before taking the length of the task into account	3	
maxtaskprec	The maximum nr of tasks a single task can have precedence over	2	
maxtaskcon	The maximum nr of tasks a single task can be connected to	2	
maxtaskdir	The maximum nr of tasks a single task can have a direct start with	2	
nSkillsLow	Number of skills with low occurrence	6	
nSkillsMedium	Number of skills with medium occurrence (twice as likely to be required)	5	
SkillsHigh	Number of skills with high occurrence (four times as likely to be required)	3	
capreq_min	The minimum sum of capacity a task can require	2	
capreq_max	The maximum sum of capacity a task can require	15	
nSkillsMin	The minimum number of skills a task can require	1	
nSkillsMax	The maximum number of skills a task can require	5	
capPerSkillMin	The minimum capacity a task can require of each skill	1	
capPerSkillMax	The maximum capacity a task can require of each skill	5	
routeSelection	The percentage of possible travel routes to be selected when branching	-	
SRperTask	The assumed maximum number of super-resources working together on one a task	2	

Appendix E

Test results from computational study

This appendix presents results from the testing of the models presented in Chapter 4 and 5. Tables E.1–E.3 present the results from applying the exact compact and decomposed models to test sets 1–3. Highlighted in blue are the subset results. They are calculated by using the average results of the five test instances that make up a subset. The only exception is the calculation of the optimality gap which is based on the average best bound and solution.

Table E.4 presents the results from applying the exact and first heuristic method rates of the decomposed model to test sets 1 and 3. Table E.5 presents the results from applying the second heuristic method rates to the three largest instances in test set 1 and 3. The cells highlighted in green are the different percentages of routes that the travel route generator chooses to branch on. 30%, for example, means that the generator only chooses to branch on 30% percent of the possible travel routes. The rows marked "Exact" present the exact decomposed solutions.

Table E.1: Results from the exact models for test set 1.

	Compact model					Decomposition model							
Test instance	Total run time	LP-relaxation	Best solution	Optimality gap after one hour	Best solution after 20 min	Route generation time	Routes generated	Optimization run time	Total run time	LP-relaxation	Best solution	Optimality gap after one hour	Best solution after 20 min
R-4-30-8-1	0.7	16.0	16.0	0.0%	16.0	0.1	16	0.7	0.8	16.0	16.0	0.0%	16.0
R-4-30-8-2	3600	26.0	0.0	2600%	0.0	0.1	24	0.8	0.9	26.0	26.0	0.0%	26.0
R-4-30-8-3	0.7	0.0	0.0	0.0%	0.0	0.8	28	0.7	1.5	0.0	0.0	0.0%	0.0
R-4-30-8-4	0.7	6.0	6.0	0.0%	6.0	0.1	8	0.6	0.7	6.0	6.0	0.0%	6.0
R-4-30-8-5	0.6	14.4	14.4	0.0%	14.4	0.1	114	0.6	0.7	14.4	14.4	0.0%	14.4
R-4-30-8	720.5	12.5	7.3	0.0%	7.3	0.2	38.0	0.7	0.9	12.5	12.5	0.0%	12.5
R-6-30-8-1	3.4	25.0	25.0	0.0%	25.0	0.1	332	1.3	1.4	25.0	25.0	0.0%	25.0
R-6-30-8-2	23.0	47.0	47.0	0.0%	47.0	0.1	192	1.8	1.9	47.0	47.0	0.0%	47.0
R-6-30-8-3	1.0	0.0	0.0	0.0%	0.0	0.1	272	1.1	1.2	0.0	0.0	0.0%	0.0
R-6-30-8-4	3.2	36.0	35.6	0.0%	35.6	0.1	210	2.9	3.0	36.0	35.6	0.0%	35.6
R-6-30-8-5	1.0	0.0	0.0	0.0%	0.0	0.1	46	1.0	1.1	0.0	0.0	0.0%	0.0
R-6-30-8	6.3	21.6	21.5	0.0%	21.5	0.1	210	1.6	1.7	21.6	21.5	0.0%	21.5
R-8-30-8-1	3600	48.0	28.0	71.4%	28.0	2.8	24762	53.6	56.4	48.0	48.0	0.0%	48.0
R-8-30-8-2	1.7	0.0	0.0	0.0%	0.0	0.3	2530	3.3	3.5	0.0	0.0	0.0%	0.0
R-8-30-8-3	3600	84.0	-	-%	-	0.6	4644	29.4	29.9	84.0	84.0	0.0%	84.0
R-8-30-8-4	218	43.0	43.0	0.0%	43.0	0.5	4550	11.3	11.8	43.0	43.0	0.0%	43.0
R-8-30-8-5	4.9	31.0	31.0	0.0%	31.0	0.6	3917	3.8	4.3	31.0	31.0	0.0%	31.0
R-8-30-8	1485	41.2	20.4	19.6%	20.4	0.9	8081	20.3	21.2	41.2	41.2	0.0%	41.2
R-10-30-8-1	3600	88.5	-	-%	-%	4.4	36154	3600	3604	88.5	75.0	14.7%	75.0
R-10-30-8-2	3600	103.0	-	-%	-%	6.7	52058	173	180	73.0	73.0	0.0%	73.0
R-10-30-8-3	3600	86.0	50.0	72.0%	50.0	0.9	7436	84.3	85.2	125	86.0	0.0%	86.0
R-10-30-8-4	3600	85.0	31.8	163%	31.8	1.7	12205	3600	3602	85.0	73.8	13.6%	55.8
R-10-30-8-5	3600	169	-	-%	-%	1.8	16501	3600	3602	169	54.4	210%	54.4
R-10-30-8	3600	106	16.4	108%	16.4	3.1	24871	2212	2214.6	108	72.4	37.3%	68.8
R-12-30-8-1	3600	77.0	-	-%	-	4.0	32555	3600	3604	89.0	16.0	381%	16.0
R-12-30-8-2	3600	108	-	-%	-	40.5	295062	3600	3641	126	26.0	315%	26.0
R-12-30-8-3	464	103	103	0.0%	103	0.5	4480	13.0	13.5	103	103	0.0%	103
R-12-30-8-4	3600	13.0	6.0	117%	6.0	11.6	91132	140	151	13.0	13.0	0.0%	13.0
R-12-30-8-5	3600	154	14.0	1000%	14.0	38.7	282304	3600	3639	154	113	36.3%	42.0
R-12-30-8	2973	91.0	24.6	120%	24.6	19.1	141107	2191	2210	97.0	54.2	67.9%	40.0

Table E.2: Results from the exact models for test set 2.

	Compact model					Decomposition model							
Test instance	Total run time	LP-relaxation	Best solution	Optimality gap after one hour	Best solution after 20 min	Route generation time	Routes generated	Optimization run time	Total run time	LP-relaxation	Best solution	Optimality gap after one hour	Best solution after 20 min
L-6-30-4-1	3.7	3.0	3.0	0.0%	3.0	0.1	44	1.1	1.2	3.0	3.0	0.0%	3.0
L-6-30-4-2	3.0	32.3	32.3	0.0%	32.3	0.1	20	1.2	1.3	32.3	32.3	0.0%	32.3
L-6-30-4-3	1.1	60.0	60.0	0.0%	60.0	0.1	20	1.0	1.1	60.0	60.0	0.0%	60.0
L-6-30-4-4	3600	55.0	52.0	5.8%	52.0	0.1	48	2.8	2.9	55.0	55.0	0.0%	55.0
L-6-30-4-5	7.0	26.0	26.0	0.0%	26.0	0.1	40	2.5	2.5	26.0	26.0	0.0%	26.0
L-6-30-4	723	35.3	34.7	1.7%	34.7	0.1	34.4	1.7	1.8	35.3	35.3	0.0%	35.3
L-6-30-6-1	3600	28.0	22.0	27.3%	15.0	0.1	136	5.4	5.5	28.0	28.0	0.0%	28.0
L-6-30-6-2	5.2	68.0	68.0	0.0%	68.0	0.1	297	1.3	1.4	68.0	68.0	0.0%	68.0
L-6-30-6-3	3600	42.0	26.0	61.5%	26.0	0.1	128	2.9	3.0	42.0	42.0	0.0%	42.0
L-6-30-6-4	2.1	22.0	22.0	0.0%	22.0	0.1	28	1.1	1.1	22.0	22.0	0.0%	22.0
L-6-30-6-5	15.0	16.0	16.0	0.0%	16.0	0.1	44	1.9	1.9	16.0	16.0	0.0%	16.0
L-6-30-6	1445	35.2	30.8	14.3%	29.4	0.1	127	2.5	2.6	35.2	35.2	0.0%	35.2
L-6-30-8-1	3.4	25.0	25.0	0.0%	25.0	0.1	332	1.3	1.4	25.0	25.0	0.0%	25.0
L-6-30-8-2	23.0	47.0	47.0	0.0%	47.0	0.1	192	1.8	1.9	47.0	47.0	0.0%	47.0
L-6-30-8-3	1.0	0.0	0.0	0.0%	0.0	0.1	272	1.1	1.2	0.0	0.0	0.0%	0.0
L-6-30-8-4	3.2	36.0	35.6	0.0%	35.6	0.1	210	2.9	3.0	36.0	35.6	0.0%	35.6
L-6-30-8-5	1.0	0.0	0.0	0.0%	0.0	0.1	46	1.0	1.1	0.0	0.0	0.0%	0.0
L-6-30-8	6.3	21.6	21.5	0.0%	21.5	0.1	210	1.6	1.7	21.6	21.5	0.0%	21.5
L-6-30-10-1	1.3	27.0	27.0	0.0%	27.0	0.1	20	1.0	1.1	27.0	27.0	0.1%	27.0
L-6-30-10-2	15.2	80.0	80.0	0.0%	80.0	0.1	360	2.8	2.9	80.0	80.0	0.0%	80.0
L-6-30-10-3	3600	32.0	0.0	3156%	0.0	0.1	128	1.8	1.9	32.0	31.6	0.0%	31.6
L-6-30-10-4	3600	61.2	15.0	260%	15.0	0.1	502	74.9	75.1	61.2	41.2	0.1%	41.2
L-6-30-10-5	9.0	7.0	7.0	0.0%	7.0	0.1	46	1.1	1.1	7.0	7.0	2.1%	7.0
L-6-30-10	1445	41.4	25.8	54.7%	25.8	0.1	211	16.3	16.4	41.4	37.4	0.1%	37.4
L-6-30-12-1	410	27.0	27.0	0.0%	0.0	0.1	146	1.4	1.5	27.0	27.0	3.8%	27.0
L-6-30-12-2	1.0	0.0	0.0	0.0%	0.0	0.1	336	1.1	1.2	0.0	0.0	3.2%	0.0
L-6-30-12-3	3600	26.8	0.0	2680%	0.0	0.2	668	64.0	64.2	26.8	26.8	0.0%	26.8
L-6-30-12-4	1.5	5.0	5.0	0.0%	5.0	0.1	32	1.1	1.2	5.0	5.0	0.0%	5.0
L-6-30-12-5	3600	179	0.0	-%	0.0	1.7	15668	3600	3602	179	106	0.4%	60.0
L-6-30-12	1523	47.6	6.4	83.8%	1.0	0.4	3370	734	734	47.6	33.0	85.8%	23.8

Table E.3: Results from the exact models for test set 3.

	Compact model					Decomposition model							
Test instance	**Total run time**	**LP-relaxation**	**Best solution**	**Optimality gap after one hour**	**Best solution after 20 min**	**Route generation time**	**Routes generated**	**Optimization run time**	**Total run time**	**LP-relaxation**	**Best solution**	**Optimality gap after one hour**	**Best solution after 20 min**
T-6-30-8-1	3.4	25.0	25.0	0.0%	25.0	0.1	332	1.3	1.4	25.0	25.0	0.0%	25.0
T-6-30-8-2	23.0	47.0	47.0	0.0%	47.0	0.1	192	1.8	1.9	47.0	47.0	0.0%	47.0
T-6-30-8-3	1.0	0.0	0.0	0.0%	0.0	0.1	272	1.1	1.2	0.0	0.0	0.0%	0.0
T-6-30-8-4	3.2	36.0	35.6	0.0%	35.6	0.1	210	2.9	3.0	36.0	35.6	0.0%	35.6
T-6-30-8-5	1.0	0.0	0.0	0.0%	0.0	0.1	46	1.0	1.1	0.0	0.0	0.0%	0.0
T-6-30-8	6.3	21.6	21.5	0.0%	21.5	0.1	210	1.6	1.7	21.6	21.5	0.0%	21.5
T-6-40-8-1	7.3	102	102	0.0%	102	0.1	780	4.0	4.1	102	102	0.0%	102
T-6-40-8-2	3600	85.0	60.0	41.7%	60.0	0.1	386	25.0	25.1	85.0	80.0	0.0%	80.0
T-6-40-8-3	3600	26.0	0.0	-	0.0	0.1	396	3600	3600	26.0	20.0	0.3%	20.0
T-6-40-8-4	3600	104	36.0	133%	36.0	0.2	2026	13.5	13.7	1034	83.8	0.0%	83.8
T-6-40-8-5	3600	94.2	0.0	-%	0.0	0.1	678	3600	3600	94.2	0.0	94.2%	0.0
T-6-40-8	2882	82.2	39.6	97.4%	39.6	0.1	853	1449	1449	82.2	57.2	0.4%	57.2
T-6-50-8-1	3600	176	40.0	340%	40.0	65.3	430526	3600	3665	176	0.0%	-%	0.0
T-6-50-8-2	91.2	113	113	0.0%	113	1.0	9016	143	144	113	113	0.0%	113
T-6-50-8-3	3600	95.8	0.0	-%	0.0	167	883347	3600	3767	97.0	0.0	-%	0.0
T-6-50-8-4	14.5	155	140	0.0%	140	0.2	734	4.9	5.1	155	140	0.0%	140
T-6-50-8-5	3600	66.6	10.0	566%	10.0	244	1148580	3600	3844	66.6	0.0	-%	0.0
T-6-50-8	2181	121	60.5	95.4%	60.5	95.4	494441	2190	2285	121	50.5	1.3%	50.5
T-6-60-8-1	3600	165	4.0	3060%	4.0	1.9	15960	705	707	169	126	0.0%	126
T-6-60-8-2	3600	92.6	0.0	-%	0.0	7.2	59528	3600	3607	130	0.0	-%	0.0
T-6-60-8-3	3600	113	0.0	-%	0.0	0.5	4258	3600	113	112.5	110	0.0%	104
T-6-60-8-4	260	30.0	30.0	0.0%	30.0	2.0	15460	75.8	77.8	30.0	30.0	0.0%	30.0
T-6-60-8-5	99.6	75.0	75.0	0.0%	75.0	3.1	21182	3600	3603.1	75.0	48.0	0.6%	48.0
T-6-60-8	2232	95.0	21.8	300%	21.8	2.9	23278	2316	1622	103	62.8	0.5%	61.6
T-6-70-8-1	3600	142	76.0	87.1%	76.0	345	1309144	3600	3945	-%	0.0	-%	0.0
T-6-70-8-2	3600	109	42.7	155%	42.7	0.1	520	9.8	9.9	109	101	0.0%	101
T-6-70-8-3	3600	58.0	22.0	158%	22.0	103	576892	3600	3703	58.0	38.8	0.5%	0.0
T-6-70-8-4	3600	159	93.0	70.0%	93.0	17.7	120812	3600	3618	205	121	0.3%	117
T-6-70-8-5	657	40.0	40.0	0.0%	40.0	0.6	4104	17.1	17.6	40.0	40.0	0.0%	40.0
T-6-70-8	3012	102	54.7	84.9%	54.7	93.3	402294	2165	2259	103	60.2	65.4%	51.7

Table E.4: Results from the first heuristic approach.

Test subset	% Routes branched on	Route generation time	Routes generated	Optimization run time	Total run time	LP-relaxation	Best bound	Best solution	Optimality gap after one hour	Best solution after 20 min
	Exact	0.2	38	0.7	0.9	12.5	12.5	12.5	0.0%	12.5
	75%	0.1	31.6	0.7	0.7	10.1	10.1	10.1	0.0%	10.1
R-4-30-8	50%	0.1	19.2	0.7	0.8	10.1	10.1	10.1	0.0%	10.1
	30%	0.1	16	0.7	0.8	10.1	10.1	10.1	0.0%	10.1
	15%	0.1	16	0.7	0.7	10.1	10.1	10.1	0.0%	10.1
	Exact	0.1	210	1.6	1.7	21.6	21.5	21.5	0.0%	21.5
	75%	0.1	201	1.5	1.6	21.6	21.5	21.5	0.0%	21.5
R-6-30-8	50%	0.1	97.2	1.4	1.5	20.5	20.5	20.5	0.0%	20.5
	30%	0.1	69.6	1.4	1.5	15.6	15.5	15.5	0.0%	15.5
	15%	0.1	57.6	1.4	1.5	15.6	15.5	15.5	0.0%	15.5
	Exact	0.9	8081	20.3	21.2	41.2	41.2	41.2	0.0%	41.2
	75%	0.7	5440	734	734	41.2	41.2	28	47.1%	28
R-8-30-8	50%	0.2	878	4.8	5	29.3	28.2	28.2	0.0%	28.2
	30%	0.1	360	3.1	3.2	26.3	25.2	25.2	0.0%	25.2
	15%	0.1	166	2.3	2.4	19.7	16.8	16.8	0.0%	16.8
	Exact	3.1	24871	2212	2215	108	99.4	72.4	37.3%	68.8
	75%	2.2	13378	2739	2741	106	97.4	68.7	41.9%	56.2
R-10-30-8	50%	0.3	1619	734	734	82.9	80.3	77.9	3.1%	77.9
	30%	0.2	641	45.1	36.2	75.1	67.5	65.1	3.7%	65.1
	15%	0.1	226	13.6	13.7	68.6	54.9	54.9	0.0%	54.9
	Exact	19.1	141107	2191	2210	97	91	54.2	67.9%	40
	75%	9.7	60176	2081	2091	94.2	88.2	67.2	31.3%	48.6
R-12-30-8	50%	0.8	4868	122	122.3	75.4	61	61	0.0%	61
	30%	0.3	1516	22	22.3	59.6	51.2	51.2	0.0%	51.2
	15%	0.1	290	4.1	4.2	40.6	40.6	40.6	0.0%	40.6
	Exact	0.1	210	1.6	1.7	21.6	21.5	21.5	0.0%	21.5
	75%	0.1	201	1.5	1.6	21.6	21.5	21.5	0.0%	21.5
T-6-30-8	50%	0.1	97.2	1.4	1.5	20.5	20.5	20.5	0.0%	20.5
	30%	0.1	69.6	1.4	1.5	15.6	15.5	15.5	0.0%	15.5
	15%	0.1	56	1.4	1.5	15.6	15.5	15.5	0.0%	15.5
	Exact	0.1	853	1449	1449	82.2	77.2	57.2	35.1%	57.2
	75%	0.2	528	1450	1450	81.5	75.7	56.4	34.3%	56.4
T-6-40-8	50%	0.1	160	721.7	722	55.6	50.2	49	2.5%	49
	30%	0.1	89.6	2.9	3	54.4	48	48	0.0%	48
	15%	0.1	59.8	2.9	2.8	42.7	37.9	37.9	0.0%	37.9
	Exact	95.4	494441	2190	2285	121	118	50.5	135%	50.5
	Exact	13.1	70188	1636	1649	119	119	99.7	19.4%	91.7
T-6-50-8	50%	0.7	3942	2162	2163	98.4	97.7	72.6	34.7%	72.6
	30%	0.1	549	47.3	47.5	89.5	69.5	69.5	0.0%	69.5
	15%	0.1	97.6	6	6.1	74.2	57.2	57.2	0.0%	57.2
	Exact	2.9	23278	2316	1622	103	94.2	62.8	50.0%	61.6
	75%	2.6	15392	138	141	102	85.9	85.9	0.0%	85.9
T-6-60-8	50%	0.3	1439	40.2	40.5	80.9	67	67	0.0%	67
	30%	0.1	477	32.6	32.7	77.9	64.6	64.6	0.0%	64.6
	15%	0.1	100	514	513.9	49.7	31.5	31.5	0.0%	31.5
	Exact	93.3	402294	2165	2259	103	99.6	60.2	65.4%	51.7
	75%	21.1	106642	1657	1678	111	93.8	85.2	10.1%	53.8
T-6-70-8	50%	0.8	3909	411	411	106	80.8	80.8	0.0%	80.8
	30%	0.2	892	10.2	10.4	79.1	73	73	0.0%	73
	15%	0.1	159	7.6	7.7	68.5	59.6	59.6	0.0%	59.6

Table E.5: Results from the first heuristic approach.

Test subset	% Routes branched on	Route generation time	Routes generated	Optimization run time	Total run time	LP-relaxation	Best bound	Best solution	Optimality gap after one hour	Best solution after 20 min
R-8-30-8	Exact	0.9	8081	20.3	21.2	41.2	41.2	41.2	0.0%	41.2
	100%	0.3	1882	1442	1443	37.6	37.6	17.6	114%	17.6
	75%	0.3	1665	726	726	37.5	27.9	27.9	0.0%	27.9
R-10-30-8	Exact	3.1	24871	2212	2215	108	99.4	72.4	37.3%	68.8
	100%	0.4	2543	1459	1459	97.5	96.6	67.7	42.8%	68
	75%	0.4	2077	882	882	93.9	91.0	84.4	7.8%	84
R-12-30-8	Exact	19.1	141107	2191	2210	97	91	54.2	67.9%	40
	100%	4.0	26836	1453	1457	81.8	77.2	58	33.1%	58
	75%	2.4	14118	1379	1382	79.6	73.8	61.6	19.8%	42.6
T-6-50-8	Exact	95.4	494441	2190	2285	121	118.4	50.5	135%	50.5
	100%	35.7	181794	1891	1939	112	111	103	8.5%	71.2
	75%	6.2	33818	2227	2234	112	111	92.1	21.0%	83.26
T-6-60-8	Exact	2.9	23278	2316	1622	103	94.2	62.8	50.0%	61.6
	100%	2.5	16428	2286	2289	103	87.3	77.2	13.1%	76.4
	75%	2.0	14199	1575	1577	103	86.7	81.8	6.0%	76.8
T-6-70-8	Exact	93.3	402294	2165	2259	103	99.6	60.2	65.4%	51.7
	100%	89.6	382099	2055	2144	82.3	71.2	67.7	5.2%	53.7
	75%	16.9	89762	1565	1582	110	94.0	90.7	3.7%	71.3

Appendix F

Run times and travel routes for the exact decomposed model

This appendix includes figures illustrating the run times, and number of routes generated, by applying the exact decomposed model presented in Chapter 5 to test set 2 and 3. An interpretation of these figures is given in Section 7.2.

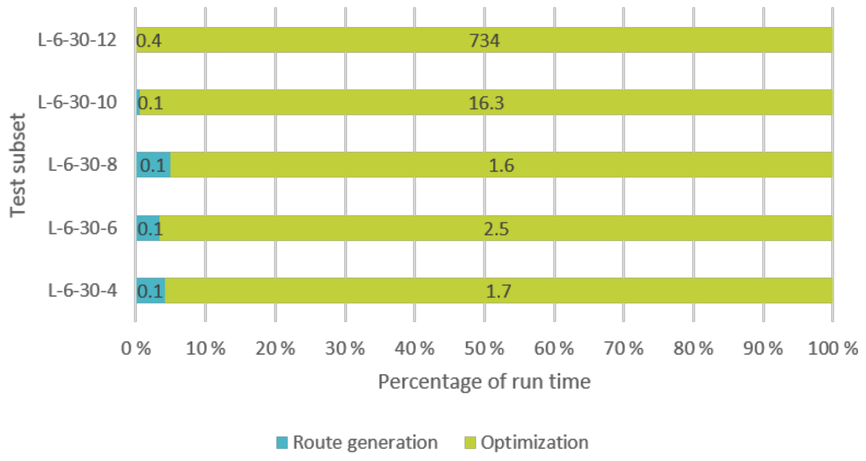


Figure F.1: The exact decomposed models' route generation and optimization run times for test set 2.

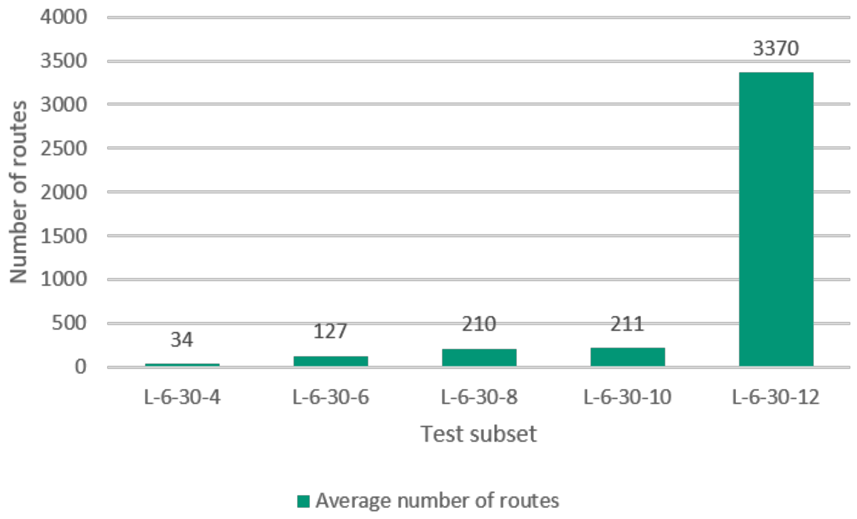


Figure F.2: Number of routes generated by the exact decomposed model for test set 2.

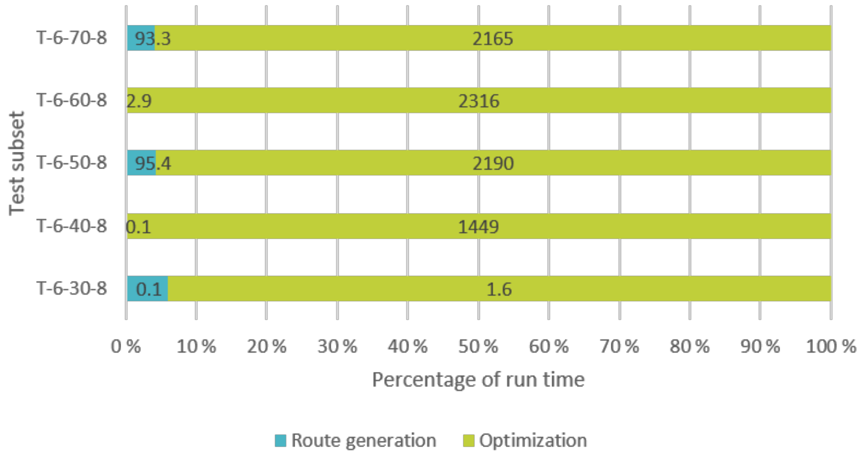


Figure F.3: The exact decomposed models' route generation and optimization run times for test set 3.

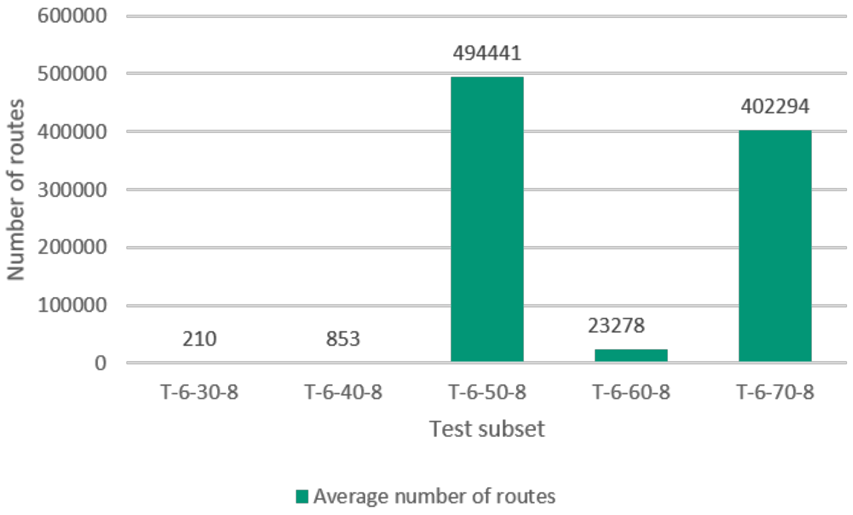


Figure F.4: Number of routes generated by the exact decomposed model for test set 3.

Appendix G

Results from test set 3 applying the first heuristic method

This appendix presents results from applying the first heuristic method to test set 3. Figure G.1 and G.2 show the heuristics' deviation from the exact method's best solution, represented by 0% line on the horizontal axis. Figures G.3 and G.4 illustrate the number of routes generates and the run times for each subset and branching rate.

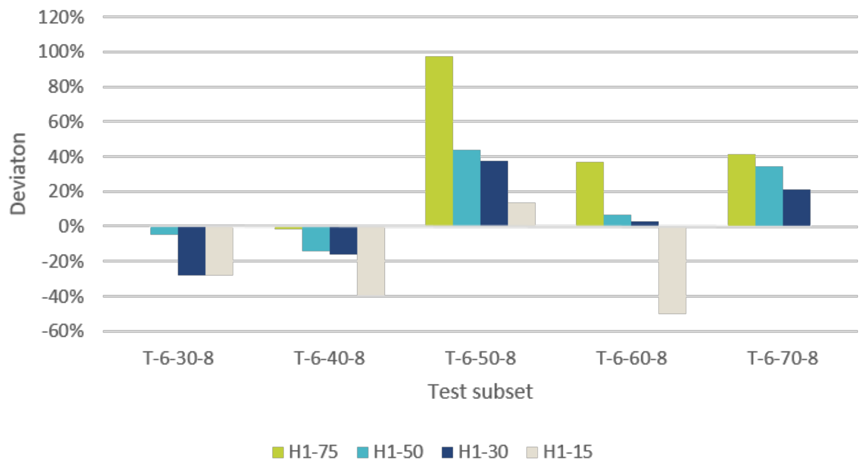


Figure G.1: The first heuristic solutions' deviation from the decomposed exact method's best solutions after run time of one hour with an increase in tasks.

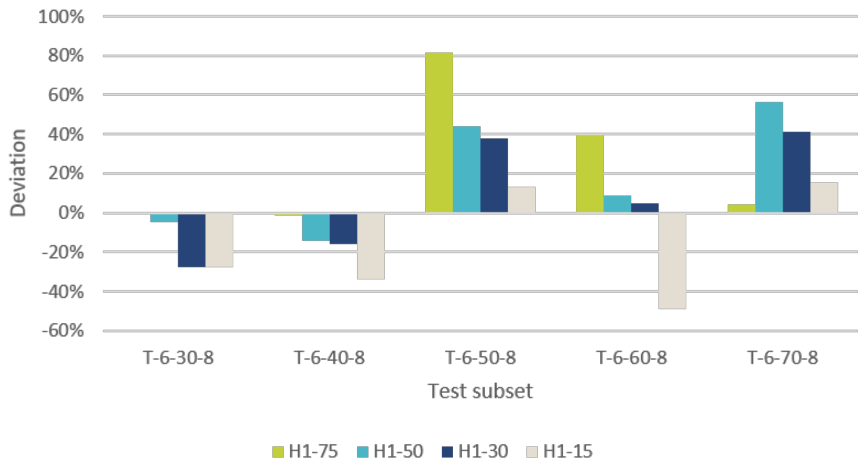


Figure G.2: The first heuristic solutions' deviation from the decomposed exact method's best solutions after run time of 20 minutes with an increase in tasks.

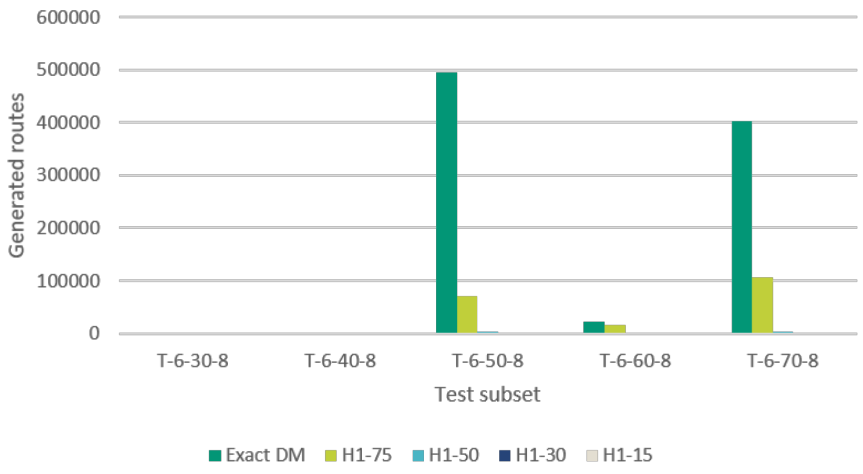


Figure G.3: Comparison of the number of routes generated by the first heuristic method with an increase in tasks.

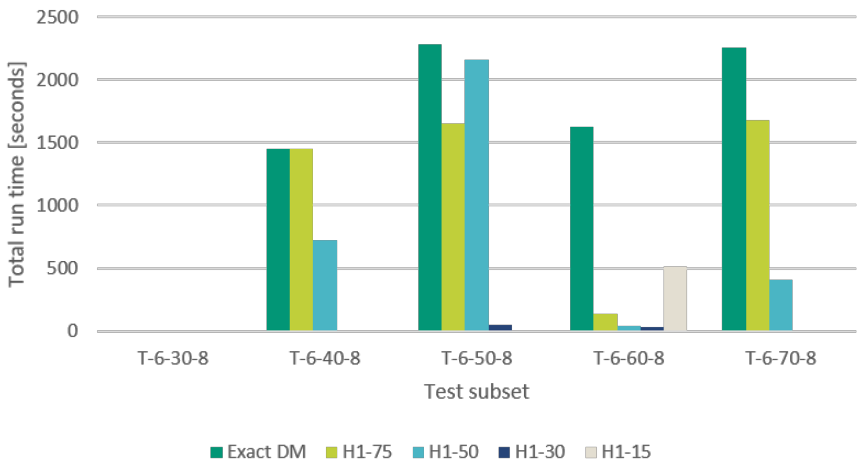


Figure G.4: Comparison of total run times for the first heuristic approach with an increase in tasks.