UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Meelis Tapo

Optimization of Military Convoy Routing

Bachelor's Thesis (9 ECTS)

Supervisors: Bahman Ghandchi, MSc
Dr. Dirk Oliver Theis

Tartu 2017

Optimization of Military Convoy Routing

Abstract:

Convoy movement problem is a mathematical optimization problem which tries to find optimal routing and scheduling solution for concurrent military convoy movements. In this thesis several optimization methodologies are designed and tested to find best suited algorithm for solving practical convoy routing instances in Estonia. Encouraging results are obtained by using a mixed integer programming model together with simple heuristics, by creating an exact branch-and-bound methodology and by developing fixed-order based routing approach. Bachelor's thesis also provides a complementary application to compare qualities of designed methods, to present calculated routes and schedules and to display convoy movement animations on the map of Estonia. Thesis illustrates that methods of mathematical optimization can be used to solve real-world instances of convoy movement problem fast and with quality results and hence improve decision-making in operational convoy planning practice.

Keywords:

Convoy movement problem, mathematical optimization, routing, scheduling, mixed integer programming, branch-and-bound method, heuristics, movement animation

CERCS: P170 Computer science, numerical analysis, systems, control

Militaarsete rännakukolonnide marsruutimise optimeerimine

Lühikokkuvõte:

Motoriseeritud rännakukolonnide optimeerimine on matemaatilise optimeerimise probleem, milles püütakse leida optimaalset marsruutislahendust ja vastavat ajakava samaaegsetelt liikuvatele rännakukolonnidele. Käesolevas töös luuakse valik erinevatel optimeerimistehnikatel põhinevaid meetodeid, mida testides püütakse leida parimat Eesti oludele vastavat rännakukolonnide marsruutimise optimeerimismeetodit. Häid tulemusi saavutati kasutades osalise täisarvulise planeerimise mudelit koos heuristiliste täiendustega, rakendades jaga-ja-piira tehnikal põhinevat täpset algoritmi, kui ka kasutades fikseeritud järjestusega marsruutimislahendust. Lisaks töötati bakalaureuse töö koostamise käigus välja optimeerimismeetodeid kasutav rakendus, mille abil on võimalk võrrelda erinevate meetodite käitumist ja omadusi, esitada kalkulatsioonide tulemusena leitud teekondi ja ajagraafikuid ning animeerida Eesti kaardil rännakukolonnide liikumist. Töö tulemusena võib väita, et matemaatilise optimeerimise meetodid on sobivad päriseluliste rännakukolonnide optimeerimisprobleemide kiireks ja kvaliteetseks lahendamiseks ja et neid meetodeid kasutades on võimalik parandada rännakukolonnide kavandamisel tehtavate planeerimisotsuste kvaliteeti.

Võtmesõnad:

Rännakukolonnide liikumise probleem, matemaatiline optimeerimine, marsruutimine, kalenderplaneerimine, osaline täisarvuline planeerimine, hargne-ja-piira meetod, heuristilised meetodid, liikumise animeerimine

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Table of Contents

Introduction

Planning and scheduling concurrent movements in time and space is a difficult task, which depends upon multitude of different factors. Accuracy of spatio-temporal movement planning is particularly critical in the field of military transportation operations. This applied research thesis examines the topic of military convoy movement from the viewpoint of mathematical optimization. The purpose of the thesis is to identify and design a suitable algorithm for concurrent convoy movement optimization and to create a prototype for an application that aids convoy planners in scheduling and visualizing those movements. Subject of the thesis was proposed by Estonian National Defence College in order to create the necessary basis for potential planning tool that could be used to aid decision-making of military transport operations in Estonia.

Military defence forces often have to move large number of personnel and equipment from one location to another. Depending on the size and scale of the mission, either a partial or total mobilization of forces is required. In process of mobilization each military unit moves as a convoy. Military land convoy is a group of motor vehicles traveling together for mutual support and protection. During tactical march convoy moves from its origin to its destination, while preserving its combat readiness and ability to perform its assigned orders [1].

Convoy's march is conducted in a prearranged formation and follows strict discipline. Open column formation is most often used, as it offers moving vehicles more flexibility and is less vulnerable to attacks. The enemy cannot bring effective fire to bear on a large number of vehicles separated by wide intervals. With gaps of 50 to 100 m between vehicles in a same unit and gaps of 200 to 300 m between different subunits military convoys usually reach lengths of several kilometres and it might take them hours to pass through a given junction on the road [2].

Speed of convoy is usually considerably slower than regular traffic. It depends upon the weather and road conditions, but it is generally around 50 km/h. Slower speed helps to maintain column formation and improves convoy's combat readiness, also some of the special equipment might not even be able to reach higher speeds. While peace-time convoys often pre-arrange halts on the route, during emergency situations halting is not recommended, convoys continue to move until they reach their destinations. The crossing of two convoys along the same road is termed *conflict* and is strictly forbidden be it war-time or peace-time [3].

Efficiency of motorized marches depend directly upon its planning quality. Creating detailed march order is complex task that requires lot of preceding research. This research can be roughly divided into research of the adversary and research of the environment. Commanding officer must evaluate collected intelligence to assess enemy's location, its capabilities and possible plans. Route of the convoy should be planned according to the countering threat. For example if there is a threat of ambush then narrow forest roads should be avoided or if enemy has strong air attack capabilities then crossing vast open landscapes should be limited etc. General recommendation is that tactical marches in conflict situations should be conducted with as much cover and concealment as possible. If possible convoys are sent out in the dark or in bad weather conditions. Environmental analysis can be divided into three important factors: landscape, population and weather. Landscape analyses involves surveying feasible routes and destination area as well as detecting possible obstacles and bottlenecks along the way. Also good observation areas, fire support locations and other key areas are marked. When it comes to local human activity, it is important to detect possible threats of aversion and to do everything possible to limit the distress caused by military operations. That includes studying local traffic patterns to avoid possible traffic congestions. Weather

also plays important role in convoy movement planning. It can drastically change terrain's trafficability and hinder enemy's counteractions [4].

After research has been done routing and scheduling can start. First the main route of movement is selected and then its alternatives corresponding to different scenarios are agreed upon. Furthermore often units are allocated into different marching groups and parallel routes are planned to further reduce different risks. When arranging convoy's movement schedule commanding officer has to consider when is the first possible time that the convoy can departure from its starting point, when will it need to arrive at its destination and whether any stops and checkpoints need to be planned along the route. If all this has been successfully processed then the responsible officer can start composing the march order, which additionally to the route and scheduling information also needs to contain precise description of the convoy's formation, orders for supporting units and a communications plan. Better understanding of the tasks at hand can be absorbed from convoy commander's checklist shown in the Appendix 1 [1].

As this short overview shows planning convoy movement is a diverse and challenging task. But the situation becomes much more complex when multiple convoys have to be moved simultaneously. During conflict situations this is actually very common, because all battle preparations need to be carried out in a compact and swift fashion.

Currently Estonian Defence Forces use traditional methods to plan convoy movement. For example landscape analysis is done using transparent map overlays. By stacking overlays with different map information on top of each other work of geographic information system is imitated and basic map algebraic operations are done. At the same time convoy's formation and scheduling is assembled using Microsoft Excel. Unfortunately current solution only allows constructing schedules for one convoy at a time, which is not ideal for more complex situations. Those deficiencies in planning methods cause imprecision, meaning that when it comes to decision-making officers are left without reliable support mechanisms. All the responsibility of generating accurate path variations and choosing the most valuable amongst those options stays with the responsible officer.

In order to make fewer mistakes in the planning phase and allow officers to concentrate on most critical parts of the planning process it is beneficial to automatize those subtasks that are well-defined and bounded. Plenty of such tasks can be found in the field of landscape analyses. But current thesis focuses on movement analyses.

This thesis aims to demonstrate that techniques of mathematical optimization are well suited for concurrent movement modelling and scheduling. Author believes that compared to traditional methods using mathematical optimization can make planning process faster and improve the quality of decision making. Thesis compares different methods used for convoy movement optimization and aims to find the most suitable for the case of Estonia.

The rest of the thesis is organized as follows. First chapter gives overview of the convoy movement problem, introduces the topic of optimization and gives academic background of related works. Second chapter describes different optimization methods designed in this thesis for movement optimization. Third chapter examines the designed software application and gives overview of its features. Fourth chapter analyses test cases and presents main results of the work and sixth chapter opens the discussion for further developments. Thesis is concluded with a summary.

1 Background

This chapter introduces relevant aspects and methods associated with the convoy movement problem and gives overview of current state of scientific research in this field.

1.1 Convoy movement problem

Formally, the convoy movement problem can be defined as follows. Given a set of military convoys, origin and destination pair associated with each convoy determine the optimal routes and schedules for all convoys such that the sum of the arrival times of the convoys at their respective destinations is minimized, subject to the conditions that no convoy stops en-route, minimum headway time is maintained between two convoys, and no two convoys cross or overtake each other along the roads [5].

Usually convoy movement problem is associated with transportation network represented by a graph, where intersections are represented by set of nodes and roads are represented as edges. Each convoy in the problem can be characterized with array of parameters, most regularly used of them are: length, speed, origin, destination, earliest departure time of the head of the convoy from its departure point and latest arrival time of the tail of the convoy at its destination. The completion time of a particular convoy's route is then the convoy's earliest start time plus its initial delay plus the time it takes for the convoy to traverse its path plus the time it takes for the convoy to pass through a node. It is sometimes termed as convoy's time window [6].

Many problem features listed in aforementioned definition and others associated with the convoy movement problem are often debated. Following analysis is based on overviews given by Kumar and Narendran [3] and Gopalan [7].

Length of the convoy is one of the most characteristic features of the convoys. But often zero length convoys, also referred to as particle convoys, are used in formulations instead. Modelling convoys as particles represents a good approximation for problem instances where edge lengths are large compared to convoy lengths. On the other hand this simplification removes some of the problem's complexity. If convoys have non-zero length then best routing may also want to minimize the number of nodes on the path as this can be logistically undesirable. Every time a convoy passes through a node it is occupying multiple edges at the same time.

Speed of the convoys is another contested features. Speed depends on multitude of variables, such as surface quality, gradient, curvature, road capacity etc. Furthermore each node usually represents an intersection thus in real-world passing through a node often means that convoys have to first slow down and then accelerate. This means that speed is constantly changing. Still it is widespread that in many formulations speed is used as a constant. In fairness modelling speeds for different sections in the network is a daunting task, especially in military situations. But if variable speeds are used then lower and upper bounds are usually specified for the travelling speeds of the convoys.

In some formulations convoys are not permitted to travel on the same road in the opposite directions at the same time. This constraint is referred as *blocking*.

There are differences to formulations based on whether war–time or peace-time scenario is used. In peace-time scenarios convoys are usually allowed to stop along the route while in war-like setups they generally continue until destination. There are even different idling formulations, some allow idling only at nodes and not on edges, while others allow idling only at certain nodes. In most formulations idling is allowed at starting nodes. In peace-time

scenarios convoys often have equal priorities while in emergency situations convoys may have predefined or varying priorities.

Different convoys may have different speeds, but overtaking is generally not permitted even if trailing convoy can move faster than a leading convoy. Also minimum headway between two convoys moving in the same direction is often brought into the model to avoid congestion and possible accidents.

Some formulations don't include ready times and deadlines, meaning that all convoys are ready to move instantly and there is no time limit set for their arrivals.

Additional variables may be used to represent node or edge capacities. Meaning that only certain number of convoys or vehicles can idle at given node or move on a given edge.

Routing criteria can also differ by formulation. Most often used routing objective is to minimize total flow time. This means that arrival times of convoys at their respective destinations are summarized and minimized. Travel span is also used frequently, meaning that total travel time of last convoy is minimized. Using this objective encourages convoys to take longer routes i.e. detours rather than idle at starting node in case of conflicts. Other relevant objectives include minimizing the number of edges used to transit, minimizing the cost of transportation, minimizing civilian traffic disruptions and minimizing the delay associated with each convoy at their respective departure points. Multi-objective formulations are also regularly used, in that case primary objective is optimized first and secondary objective used only if necessary.

Convoy movement problem belongs to the class of NP-hard problems, meaning that it is at least as hard as any NP-problem. A problem is assigned to the NP (nondeterministic polynomial time) complexity class if it is solvable in polynomial time by a nondeterministic Turing machine. This means that the time required to solve the convoy movement problem using any currently known algorithm increases very quickly as the size of the problem grows, hence the problem is not amenable to the determination of optimal solutions for large problem instances [3]. Chardaire and McKeown proved that convoy movement problem is also NP-complete, by showing that the disjoint connecting path problem, which is known to be NP-complete can be polynomially reduced to the decision version of convoy movement problem [8]. This means that convoy movement problem is both NP and NP-hard and that if a deterministic polynomial time algorithm can be found to solve the problem, then every other NP problem can also be polynomially solved.

Problem analogues to convoy movement problem include routing of automated guided vehicles, movement of luggage from different flights along a common automated transportation system to various pick-up points, strategic routing of hazardous materials, packet routing in a telecommunications network and scheduling of trains on single line track.

1.2 Optimization techniques

Mathematical optimization is a branch of applied mathematics that in the broadest sense looks for best solution with regard to some criterion from some set of available alternatives.

There a two main approaches to solving an optimization problem. The first one is to formulate the problems as mathematical models and then solve them to optimality using exact algorithms or commercial optimization packages. The second one is simply to generate good solutions of the problems using metaheuristics [7]. Although exact methods theoretically guarantee finding an optimal solution, in practice they only work in cases where optimization problem requires effort that grows polynomially in regards to the problem size. When optimization problem is NP-hard it might require exponential effort instead. In that case

even medium sized problems may become unsolvable. Thus it may be wise to look for heuristic solution to the problem. Heuristics don't guarantee solution's optimality, but can give a quality approximate solutions with a reasonable effort. They often show good performance for many NP-complete problems and can therefore have practical relevance. Heuristic methods are usually problem-specific as they exploit properties of the problem [8].

A mathematical programming model is a mathematical representation of the actual situation that may be used to make better decisions or simply to understand the actual situation better [7]. All mathematical programming models involve optimization as the goal is to either minimize or maximize some objective function subject to models boundary conditions and constraints.

Most used method of mathematical programming is linear programing. Linear programming (LP) is a technique for the optimization of a linear objective function, subject to linear equality and linear inequality constraints. The canonical form of LP (Figure 1), where c and x are n-dimensional real vectors, A is an $m \times n$ matrix with real entries and b is an m-dimensional real vector consists of: a) a linear objective function, b) linear inequalities and c) non-negative decision variables:

a)	$\min c^T x,$	$\max c^T x$
b)	subject to $Ax \geq b,$	subject to $Ax \leq b$
c)	$x_i \geq 0.$	$x_i \geq 0$

Figure 1. Canonical form of linear programming – minimization (left), maximization (right).

The objective function and all constraints of LP model are linear and only constraints of the form ≤, ≥, or = are allowed. All decision variables are continuous. The solution to a LP is an assignment to the variables that satisfies all the constraints while minimizing (or maximizing) the objective function [8].

If LP model is well-defined then most of the commercial solvers can easily solve them. Solvers usually consist of a number of optimization algorithms collected together as a package of computer routines. Those routines are then run sequentially and efficiently until problem is solved. But LP problems can also be solved by exact methods. Exact methods are suited for combinatorial optimization models. If the number of possible solutions to an optimization problem is finite and they all can be enumerated then the problem is called a combinatorial problem. General method for solving LP models with such exact method is called the simplex algorithm. Another widely used method is the interior point algorithm. The simplex algorithm searches for the optimal solution along the corner points of the solution space, whereas the interior point algorithm looks for the optimum through the interior of the feasible region [7].

Integer linear programming (ILP) is subclass of LP where all decision variables are integers. Generally, there are three types of ILP. First is called pure ILP where all variables must be integers. Second is called mixed integer linear programming (MILP) where only some of the variables must be integers. Third is called binary integer linear programming where all the variables must be binary, either 0 or 1. Their respective canonical forms are shown on Figure 2.

While all LP problems are polynomially solvable, this does not apply to ILP problems, which are NP-hard. Nevertheless there still exists wide range of different methods and techniques that are used to challenge ILP models.

$$\begin{array}{lll} \min c^T x, & \min c^T x + d^T y & \min c^T x, \\ \text{subject to } Ax \geq b, & \text{subject to } Ax + Bx \geq b & \text{subject to } Ax \geq b, \\ x_i \in \mathbb{N}_0 & x_i \in \mathbb{N}_0,\ y_j \geq 0 & x_i \in \{0,1\} \end{array}$$

Figure 2. Canonical forms of integer linear programming, mixed integer linear programming and binary integer linear programming, all presented with minimization objective.

If integer constraints of the problem are dropped then ILP model is turned into LP model. This process is called relaxation. And after relaxation methods for continuous problems can be used again. However, usually the optimal solution of a relaxed problem is not integral and is, thus, not a feasible solution for the original problem. But relaxation at least can give a first bound on the solution. Starting from the continuous optimum, so-called “cutting planes" (linear inequalities not contained in the original formulation) derived from the integrality conditions are added iteratively to modify the solution space in a manner that will eventually render the optimum extreme point satisfying the integer requirements [7].

Unlike simplex method for LP problems a good exact algorithm for very wide class of ILP problems has not been developed. Different algorithm suit for different problems. One of the most used exact methods for ILP models is called branch-and-bound algorithm. The idea of the branch-and-bound algorithm is to perform the enumeration efficiently so that not all combinations of decision variables must be examined. Branch-and-bound algorithm sets up lower and upper bounds for the optimal solution. Usually the first upper bound of the ILP is its linear relaxation and the lower bound is set to -∞. The branching strategy repetitively decreases the upper bound and increases the lower bound and fathoms suboptimal solutions. Branching and bounding continuous until the search tree is exhausted and optimal solution is reached [8]. Other important exact ILP methodologies are cutting plane method and dynamic programming.

Heuristic algorithms provide a suboptimal solution, but without a guarantee on its quality. Although the running time is not guaranteed to be polynomial, empirical evidence suggests that some of these algorithms find a good solution fast. Heuristics can be classified as either constructive (greedy) heuristics or as local search heuristics. Search heuristics are preferred for solving combinatorial optimization problems, because greedy methods are usually inefficient when the number of possible solutions is vast. Search heuristic, are based on the concept of exploring the vicinity of the current solution. Neighbouring solutions are generated by a move-generation mechanism. If the generated neighbour has a better objective value, it becomes a new current solution, or otherwise the current solution is retained. The process is iterated until there is no possibility of improvement in the neighbouring solution. The problem with local search methods is that they might terminate at local optimum rather than global. To avoid this problem number of conceptual metalevel strategies have been developed. These strategies are called metaheuristics. Most notable metaheuristics include genetic algorithms, greedy random adaptive search procedure, problem-space search, neural networks, simulated annealing, tabu search, threshold algorithms and their hybrids [7]. Approximation algorithms are heuristic optimization methods together with a bound on the degree of sub-optimality.

1.3 Related work

There is no efficient exact algorithm for solving a general case of the convoy movement problem, but several different mathematical optimization methodologies have been used to tackle the problem. Because of the NP-hardness of the problem most of the work has focused on mathematical programming based methods, often relaxing some of the strategic constraints to solve the model. But methodologies based on heuristic methods have also been used to get approximate solutions. None of the models available in the literature model all the constraints of the convoy movement problem while solving optimally [3].

Bovet *et al* [6] were the first to investigate the topic in 1991. They called it a convoy scheduling model and examined how a set of convoys could use the same road as part of their route to their destinations. The movements were subject to two main constraints: each convoy had to leave its departure point during an imposed time interval and convoys were not allowed to pass or cross each other along the road. They proposed two different formulations: one based on mixed-integer programming and another one based on graphs. They proved that this specific traffic problem is NP-complete and designed an iterative Tabu search procedure to solve the problem.

Lee *et al* [11] described three approaches to solve the general version of convoy movement problem by using mix of branch-and-bound genetic algorithm. First a branch-and-bound algorithm was used for solving a basic version of the problem with delays. Secondly a hybrid approach based on genetic algorithms and branch-and-bound was used, where genetic algorithm was used to compute the delays and branch-and-bound algorithm was used to compute paths. Thirdly a pure genetic algorithm based approach to compute the delays as well as paths associated with convoys was explored. Encouraging result were obtained, however their approaches did not guarantee that paths generated for each convoy were simple.

Chardaire *et al* [8] developed an integer programming model for a simplified version of the convoy movement problem, as they dropped the crossing and the overtaking constraints. They solved the model by using Lagrangian relaxation. Kumar and Narendran [12] used similar formulation and further improved the Lagrangian relaxation procedure for finding lower bounds for the convoy movement problem. In their model no time limit was enforced on convoys.

Tuson and Harrison [13] reformulated the problem by in introducing convoy ordering. After setting a fixed order on set of convoys each convoy was routed in turn, thus greatly decreasing original assignment's complexity. They used modified Dijkstra's algorithm with random search technique on a real-world like instances and showed that this straightforward reformulation yields surprisingly good results. They observed that the NP-hardness could be only a worst case measure of the problem's time complexity and real-world problems may not necessarily be hard. They also showed that the delay search had a positive and significant effect on solution quality. Additionally they reviewed previous work by Lee *et al* [11] and Chardaire *et al* [8] from their perspective and concluded that that practical instances of the convoy movement problem may be solved effectively through the application of Lagrangian relaxation.

Robinson and Leis [14] used genetic algorithms combined with a discrete event simulation to the problem of convoy scheduling. They showed that this approach can automatically remove conflicts from a convoy schedule thereby providing to the human operator the ability to search for better solutions after an initial conflict free schedule is obtained. They also demonstrated that it is feasible to find a conflict free schedule for realistic problems in a few minutes on a common workstation or laptop.

Kumar and Narendran [5] tested different heuristics for solving the dynamic version of the convoy movement problem where the network changes over time and convoys have changing priorities. They developed different dispatching rules to generate conflict-free routes and found significant differences in the heuristics if road network size and density changed.

Gopalan and Narayanaswamy [15] studied the on-line convoy movement problem with zero length particle convoys and extended convoys, where the demand for each convoy increases dynamically in time. They assumed that vertices have infinite capacity and proved that even when convoys are allowed to occupy any vertex simultaneously, the complexity of the problem does not change and still remains NP-Complete. They proposed approximation algorithms for solving the problem.

Kumar and Narendran [3] showed that integer programming model of convoy movement problem that includes all the constraints from the original formulation is not amenable to the determination of optimal solutions for large problem instances. They argued that this encourages the development of solution methodologies based on heuristics or meta-heuristics.

Gopalan [7] provides polynomial-time algorithm for the single criterion two-convoy movement problem. The performance of a simple prioritization–idling approximation algorithm is also analysed for the K-convoy movement problem. And 2-convoy routing with general bi-criterion objective functions is considered and demonstrated that this problem is equivalent to the constrained shortest path problem. However, the lexicographic optimization of bi-criterion objectives, for two convoy movement problems is shown to be easy for additive objective functions, but hard (even without the blocking restriction) for non-additive primary criteria like make span.

Sadeghnejad-Barkousaraie *et al* [16] examined convoy movement problem in peace-time situations from the civilian perspective seeking to minimize civilian traffic disruption. They developed an exact hybrid algorithm that combines the k-shortest path algorithm, finding a minimum weighted *k*-clique in a *k*-partite graph and branch-and bound strategy. Through this coupling scheme, they were able to exactly solve large instances of the convoy movement problem without relaxing many of its complicating constraints, although they used discrete time intervals for idling convoys at the origin.

2 Methods

This chapter describes optimization methods created in this thesis. A range of different methods were designed with underlying motivation to find method best suited for practical convoy planning in Estonia.

2.1 Mixed integer programming methods

Mixed integer programming algorithms created in this thesis were designed based on traditional mathematical model of convoy movement problem.

Following assumptions were used in the model:

- The transportation network is represented by an undirected graph $G = (N, E)$, where N is a set of nodes or intersections and E is a set of edges or roads.
- The cost denotes the number of time units the convoy requires to traverse the underlying road segment represented by the edge. A cost of 0 means that the convoy is unable to travel along the underlying road segment.
- Each convoy needs to be sent from a specific origin to a specific destination.
- Each convoy has an earliest departure time from its origin (ready time) and a latest arrival time to its destination (finishing time).
- Each convoy has a specified physical length.
- Roads of two convoys are not allowed to cross at the same time.
- Convoys are not allowed to overtake each other on the same road.
- Two convoys cannot face each other when traversing in opposite direction of the same road.
- Convoys can only wait at their origin, waiting is not permitted elsewhere
- Each moving convoy needs to be separated from others at least by the distance of the specified headway.
- The speed of each convoy on each road of the transportation network is constant and may vary from convoy to convoy.

A list of indices, parameters, and decision variables of the model as well as model's objective function are presented in Table 1. Constraints of the model grouped in Table 2.

Objective function of the model (the routing criteria) is to minimize convoys' total flow time. This is calculated by summarizing all the arrival times of the convoys at their respective destinations.

Constraints of the model capture restrictions on the values that a set of variables may take. Constraint 1 is a flow conservation constraint that assures continuity in the sequence of edges a convoy traverses. Constraint 2 enforces that each convoy can leave a node at most once. Constraint 3 guarantees that each convoy can enter a node at most once. Constraints 4 and 5 ensure that the arrival time of the head of the convoy to a node v from edge (u, v) is the arrival time to node u, plus the time needed for the convoy to traverse edge (u, v). Constraint 6 makes sure that once convoy has started it does not stop along its route until it has reached its destination. Constraints 7 and 8 ensure that convoy i starts its trip after its earliest ready time and completes its arrival to its destination before its due date. Constraint 9 assures that headway is maintained between two convoys which pass through the same node, such that two convoys do not occupy the same node at the same time. Constraints 10-14 impose the overtaking and blocking restrictions, while constraint 15 assures that two convoys starting from the same node don't traverse the same outgoing edge at the same time.

Table 1. Indices, parameters, decision variables and objective function of the MILP model

Indices	
i, j	Convoys $\in \{1, 2, \ldots, c\}$
u, v	Graph nodes $\in N = \{1, 2, \ldots, m\}$
Parameters	
c	Number of convoys (origin-destination pairs)
m	Number of nodes on the network (intersections)
M	Large integer used to limit the value of a set of variables based on the value of a binary variable, here $M = 10m$
h	Headway between two convoys passing the same node
$E_{u,v}$	Length of the edge (from node u to node v)
L_i	Length of a convoy i
S_i	Speed of a convoy i
O_i	Node of origin for convoy i
D_i	Node of destination for convoy i
R_i	Earliest ready time (departure time from its origin) of convoy i
F_i	Latest finishing time (arrival time to its destination) of convoy i
Decision variables	
T_u^i	Continuous variables for $\forall i, \forall u$ denoting time when convoy i arrives at node u
$X_{u,v}^i$	Binary variables for $\forall i, \forall (u,v) \in E$ $\begin{cases} 1, & \text{if convoy i will traverse edge } (u,v) \in E \\ 0, & \text{otherwise} \end{cases}$
$Y_{i,j}^u$	Binary variables for $\forall u, \forall (i,j)$ $\begin{cases} 1, & \text{if convoy } i \text{ will pass node u before convoy } j \\ 0, & \text{otherwise} \end{cases}$
Objective function	
$\min \sum_i^c T_{D_i}^i \quad \forall i$	

Table 2. Constraints of the MILP model

1	$\sum_{u}^{m} X_{u,v}^{i} - \sum_{u}^{m} X_{v,u}^{i} = \begin{cases} 1, & v = D_i \\ -1, & v = O_i \\ 0, & otherwise \end{cases}$	$\forall v, \forall i$
2	$\sum_{u}^{m} X_{u,v}^{i} \leq 1$	$\forall i, \forall v$
3	$\sum_{v}^{m} X_{u,v}^{i} \leq 1$	$\forall i, \forall u$
4	$T_u^i + \frac{E_{u,v}}{S_i} + M\left(1 - X_{u,v}^{i}\right) \geq T_v^i$	$\forall i, \forall (u,v) \in E$
5	$T_u^i + \frac{E_{u,v}}{S_i} - M\left(1 - X_{u,v}^{i}\right) \leq T_v^i$	$\forall i, \forall (u,v) \in E$
6	$M \sum_{v}^{m} X_{u,v}^{i} \geq T_u^i$	$\forall i, \forall u$, u$\neq O_i$
7	$T_{O_i}^i \geq R_i$	$\forall i$
8	$T_{D_i}^i + \frac{L_i}{S_i} \leq F_i + M(1 - X_{u,D_i}^{i})$	$\forall i, \forall u$
9	$T_v^i + \frac{L_i}{S_i} + h - M\left(2 - Y_{i,j}^{v} - X_{u,v}^{i}\right) \leq T_v^j$	$\forall i, \forall j, \forall (u,v) \in E$
10	$\sum_{u}^{m} X_{u,v}^{i} + \sum_{u}^{m} X_{u,v}^{j} \geq 2(Y_{i,j}^{v} + Y_{j,i}^{v})$	$\forall v \neq O_i, \forall v \neq O_j,$ $\forall i, \forall j, i \neq j$
11	$\sum_{u}^{m} X_{u,v}^{i} + \sum_{u}^{m} X_{u,v}^{j} \leq Y_{i,j}^{v} + Y_{j,i}^{v} + 1$	$\forall v \neq O_i, \forall v \neq O_j,$ $\forall i, \forall j, i \neq j$
12	$1 + \sum_{u}^{m} X_{u,v}^{i} + \sum_{u}^{m} X_{u,v}^{j} \geq 2(Y_{i,j}^{v} + Y_{j,i}^{v})$	$\forall v = O_i \; or \; \forall v = O_j,$ $\forall i, \forall j, i \neq j$
13	$M\left(2 - X_{u,v}^{j} - X_{v,u}^{j}\right) \geq Y_{i,j}^{v} + Y_{j,i}^{v} - 1$	$\forall (u,v) \in E, \forall (v,u) \in E,$ $\forall i, \forall j, i \neq j$
14	$\sum_{u}^{m} X_{u,v}^{i} + \sum_{u}^{m} X_{u,v}^{j} \leq Y_{i,j}^{v} + Y_{j,i}^{v}$	$\forall v = O_i \; or \; \forall v = O_j,$ $\forall i, \forall j, i \neq j$
15	$Y_{i,j}^{v} + Y_{j,i}^{v} \leq 1$	$\forall v = O_i \; and \; \forall v = O_j,$ $\forall i, \forall j, i \neq j$

Given general mathematical model fed into a solver represents first method, henceforth this method will be called MILP-1.

Hybrid approaches MILP-2 and MILP-3 were developed based on the same model. Those methods combined mathematical model with simple heuristic to reduce the search space of the original model. This is achieved by using Yen's *k*-shortest path algorithm. Yen's algorithm computes *k* (user-specified number) shortest loopless paths between given pair of nodes. It finds the best route according to shortest path algorithm and then proceeds to find *k*-1 deviations of the best path. Methods MILP-2 and MILP-3 use this algorithm as follows. First *k*-shortest paths are calculated between all origin and destination pairs. Then all the nodes from those paths are saved into a new set and this set is used as a mask to remove all other nodes from the graph. This heuristic bounds optimization calculations only to relevant set of nodes and reduces the computational size of the problem. If large graph is used then achieved reduction in graph size can have significant effect on solvers runtime. This new graph modification heuristic and model's original formulation define method MILP-2.

Method MILP-3 goes further by making modifications to the mathematical formulation (Table 3). New variables are added to the model for each pre-calculated path and convoys' movement is bounded to those paths. In a sense MILP-3 is not optimizing over set of nodes rather it is optimizing over set of paths. New constraint 16 enforces that only one path out of *k* generated paths is used and constraint 17 ensures that all pair of nodes on that path are selected as part of the route.

Table 3. Additional model components in MILP-3

Indices		
q	Path enumeration $\in \{1,2, \ldots, k\}$	
n, $n+1$	Path nodes $\in P = \{1, 2, \ldots, \text{length}(P)\}$	
Parameters		
k	Number of paths generated for each convoy	
P_q^i	Path q of convoy i	
$P_{q,n}^i$	Node n of path q of convoy i	
Decision variables		
Z_q^i	Binary variables for $\forall i \forall q \begin{cases} 1, & \text{if convoy i will traverse path } P_q^i \\ 0, & \text{otherwise} \end{cases}$	
Constraints		
16	$\sum_i^c Z_q^i = 1$	$\forall i, \forall q$
17	$Z_q^i \leq X^i_{P_{q,n}^i, P_{q,n+1}^i}$	$\forall i, \forall q \forall n$

2.2 Branch-and-bound methods

Exact methodologies guarantee results as good as mathematical modelling, but are more explicit and don't depend on solvers. In this section two methods are designed based on exact algorithm presented by Sadeghnejad-Barkousaraie *et al* [16] (hereafter original algorithm). One of those methods remains exact, but other one is turned into a heuristic to make it faster. At the core of both methods is branch-and bound algorithm, thus those methods will be called BB-1 and BB-2 in this thesis.

Original method uses discrete time intervals to represent waiting times at origin points, but for BB-1 and BB-2 precise earliest compatible departure times are calculated for convoys that need to wait at start. There are more subtle differences in the methodologies, but those will be discussed during the detailed description of the method.

The strength of chosen approach lays in the problem decomposition, which divides the problem into smaller sub-problems, which can henceforth be solved more efficiently.

As a first step of the method *k*-shortest paths are calculated for each convoy using Yen's algorithm. Next those feasible paths are fed into depth first branch-and-bound algorithm to find compatible set of convoy routes.

At the core of this process is algorithm that checks path compatibility. Two paths are considered compatible if their combination does not violate overtaking, crossing and blocking constraints, and if minimum headway between them is maintained. A feasible solution to the proposed convoy movement problem is a set of paths, one for each convoy, which are pairwise compatible

Compatible set is found as a minimum weighted *k*-clique in a *k*-partite graph $G = (V, E)$, where V is a set of vertices and E set of edges. Each vertex represents a path and the collection of feasible paths for each convoy constitutes one partite of a graph. As a result *k*-partite (*k* is the number of convoys) graph is set up. There is an edge between each pair of vertices, from different partites, if their respective paths are compatible (Figure 3). Path lengths are used as weights and their sum in corresponding *k*-clique is minimized. Solving the *k*-clique problem is NP-hard, so it is important to keep the number of vertices in the graph as small as possible. This can be done efficiently by using branch-and-bound method.

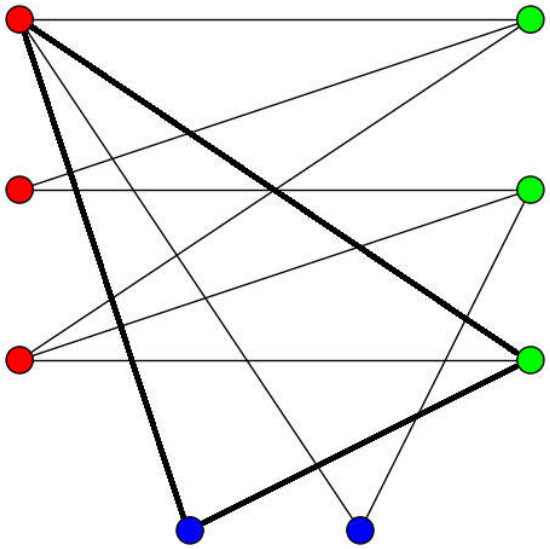


Figure 3. Example of *k*-partite graph, where $k = 3$. Colours represent partites, vertices paths, edges path compatibility and a *k*-clique (in bold) a pairwise compatible set

The input of the branch-and-bound algorithm is a list of all feasible generated paths (vertices) for each convoy and lower and upper bound. All feasible paths is updated on each branch. When a path is selected for convoy i at branching, all other paths of convoy i are deleted. Also deleted are paths of convoy j ($j \neq i$), which are not compatible with the selected path for convoy i. This operation results in list of all compatible paths. In original algorithm if at least one path was present for each unselected convoy then algorithm could continue, if not then new set of paths needed to be generated for conflicting convoys. In BB-1 and BB-2 methods all conflicts are resolved immediately. For each conflicting path new departure time is calculated (except when new departure time violates the latest arrival time restriction, in that case old conflicting path is removed and nothing new is added to the set of compatible paths). This approach ensures that best possible options are present at each branching.

Bounding and branch fathoming is done exactly as in original method. Algorithm starts with a summation of objective values for all the shortest paths as a lower bound and with infinity as an upper bound. Lower bound is updated on each branch. The first component of the lower-bound is the summation of objective values for all convoys that have paths assigned to them. The second component is the summation of the objective values for the best possible paths for each unassigned convoy. The upper-bound is updated when a better feasible solution is found [16].

A branch is fathomed under the following conditions [16]:

- Due to upper-bound. This happens when the value of the upper-bound is better (less) or equal to the value of the lower bound. This means that no better solution is available
- Due to feasible solution. At the very bottom level of the tree, when the objective value is less than the current upper-bound, all corresponding branches at the upper level are fathomed.
- Due to infeasible solution. When there is no feasible path for at least one convoy, based on the current selection of paths. This means that this branch is not feasible.

Original algorithm runs path generation and branch-and-bound algorithms sequentially and iteratively and ensures enumeration by generating new paths until no new feasible paths can be generated. If no conflicts are present this algorithm branches on a convoy with minimum number of paths in all compatible paths. Otherwise it fathoms current branch, generates new paths for the conflicting convoy and starts new branch-and-bound algorithm with that convoy selected. However it seems that this approach might dismiss some good path collections because there is no guarantees that newly generated paths are compatible with previous collection. If new branch-and-bound algorithm starts out with such path then emerging conflict requires that one of the previously suitable convoys needs to replace its paths.

In this thesis simpler branch-and-bound approaches were designed. In BB-1 method exhaustive branching technique was used. Each branching decision in this method is done in a loop so that all possible branching orders are considered. Combined with exact conflict resolution, this guarantees this methods optimality Pseudocode of BB-1 is presented in Appendix 2.

Brute-force approach even with accompanying branch fathoming is not ideal for large-scale problems. For that reason a heuristic variant of BB-1 was created (hereafter BB-2 method) was designed. BB-2 makes branching decisions based on objective value. This means that not every option is considered at branching rather at each branch convoy with path that has earliest arrival time to its destination is selected. This approach can't guarantee optimality

like BB-1, but it should give near-optimal results much faster. Selection of branching criteria is debatable, but it seems reasonable to assume the smaller the arrival time of selected convoy is, the less likely it is to conflict with other paths along the way. Another possible choice considered was branching on a convoy, which has path with least amount of conflicts with other feasible paths. Branching on a convoy with minimum number of paths in all compatible paths list, which was used in original method, was rejected, because generally after conflict resolution all convoys have same (k) amount of compatible paths.

2.3 Fixed-order methods

In addition to previously described algorithms two more algorithms were designed to create a wider basis for comparison: predefined fixed-order method (PFO) and exhaustive fixed-order method (EFO).

Both of those methods use path compatibility check algorithm described in the previous section and route convoys in fixed order. PFO method gets its ordering from user as an input, while EFO algorithm exhaustively works through all possible permutations of orderings. Both methods consider only the shortest route between origin and destination pairs and solve conflicts by postposing the less prioritized convoys at their origins.

PFO method calculates its solution as follows. First convoy at the top of the ordering is routed and added to the solution. Then next convoy is examined. If it is compatible with first convoy then it is also routed with departure time 0 and added to the solution, if not, then earliest compatible departure time is calculated and convoy is routed based on that. For next convoy compatibility is checked with both previously routed convoys and it is added to the solution only if all possible conflicts are resolved. This formula then continues until all convoys are routed. If convoy conflicts with more than one convoy then multiple departure time candidates are calculated and latest one of those is chosen. When new departure time is calculated then it is once more checked against all previously selected paths to ensure full compatibility. If at any point during conflict settling one of the convoys violates its latest finishing time restriction then whole collection is termed infeasible. Pseudocode of PFO is presented in Appendix 3

EFO follows the same procedure, but finds solutions for all possible orderings and returns the one with smallest total flow time. Pseudocode of EFO can be seen in Appendix 4.

3 Application

This chapter describes the designed application. Full-task application was created to visualize, compare and analyse the work of previously described optimization methods. Application also features basic convoy planning functionality. Implementation of the application can be found on GitHub[1].

3.1 Data

The dataset generalized from Estonian National Topographic Database part of the open data published by Estonian Land Board was used in this project [17]. The data was used to create base layer for the front-end map application and to construct a graph to generate distance matrix necessary for the optimization calculations in the back-end.

The graph was based on Estonian road network. To avoid difficulties of naval convoying only data from mainland Estonia was included. Three highest levels from the Estonian national highway classification, main routes, support routes and side routes, formed core of the network, but main streets from the cities were also used. As a result graph with 5593 edges and 3981 nodes was constructed.

3.2 Software Architecture

The application was built using Electron[2]. Electron is a Node.js and Chromium based framework for creating native desktop applications with web technologies like JavaScript, HTML, and CSS. Choice of building offline desktop application was inspired by the needs of the potential end-user, the Defence Forces, who require its tactical software to be usable in outdoor conditions without the use of Internet.

Backend of the application was written in Python. While branch-and-bound and fixed-order methods used pure Python, the MILP methods also relied on Gurobi Python interface. Gurobi is a math programming solver with parallel algorithms designed for large-scale linear programs, quadratic programs and mixed-integer programs[3]. Gurobi is a commercial software, but in this project, Gurobi's academic licence was used. Mathematical formulation described in section 3.1 was built into a Gurobi model and then optimized, standard Python was used to process the data before and after the optimization. Most of the pre-processing done in different methods was graph based. Graphs were built and manipulated with NetworkX Python package[4].

Frontend of the application was developed using different web-technologies and written mainly in JavaScript. At first this choice may not seem intuitive, because the intention was not to build a web-application, but web-technologies offered clearly better instruments for interactive-mapping and animation designing than Python and were therefore chosen. Interface of the application was designed using Bootstrap, jQuery, JQuery UI and basic HTML and CSS.

Interactive map was created with a JavaScript library called Leaflet[5]. As the application was built for offline use no online data source could be used. Therefore offline map tiles were designed and packaged with the application. Layers for the tiles were pre-processed using

[1] https://github.com/meelistapo/convoy_planner
[2] https://electron.atom.io/
[3] http://www.gurobi.com/
[4] https://networkx.github.io/
[5] http://leafletjs.com/

ArcGis geographic information system and designed with an application called TileMill. Tiles were turned into images using utility tool called MBUtil.

Map animations were designed based on leaflet plugins Leaflet.MovingMarker.js[6] and LeafletPlayback.js[7]. First one allows to move Leaflet markers along the polyline, while second one provides the ability to replay the movement with different playback functionalities.

Communication between front-end and back-end was upheld by cross-language remote control protocol called ZeroRPC[8].

3.3 Features

This section gives an overview of the application's user interface and main functionalities.

Main component of the graphical user interface is the map. Map is where user finishes convoy creation, where convoy movements are animated and where their final paths are visualized. Other important components of the application include a fixed top navigation bar with submenus, playback section, time section, sliding side menu for convoys, sliding top menu for results and popup window for settings.

When user opens the application only map and navigation bar can be seen. Fixed bar contains "Convoys" and "Settings" submenus and time section on the far left. Map shows whole road network of Estonia and is zoomable and pannable (Figure 4).

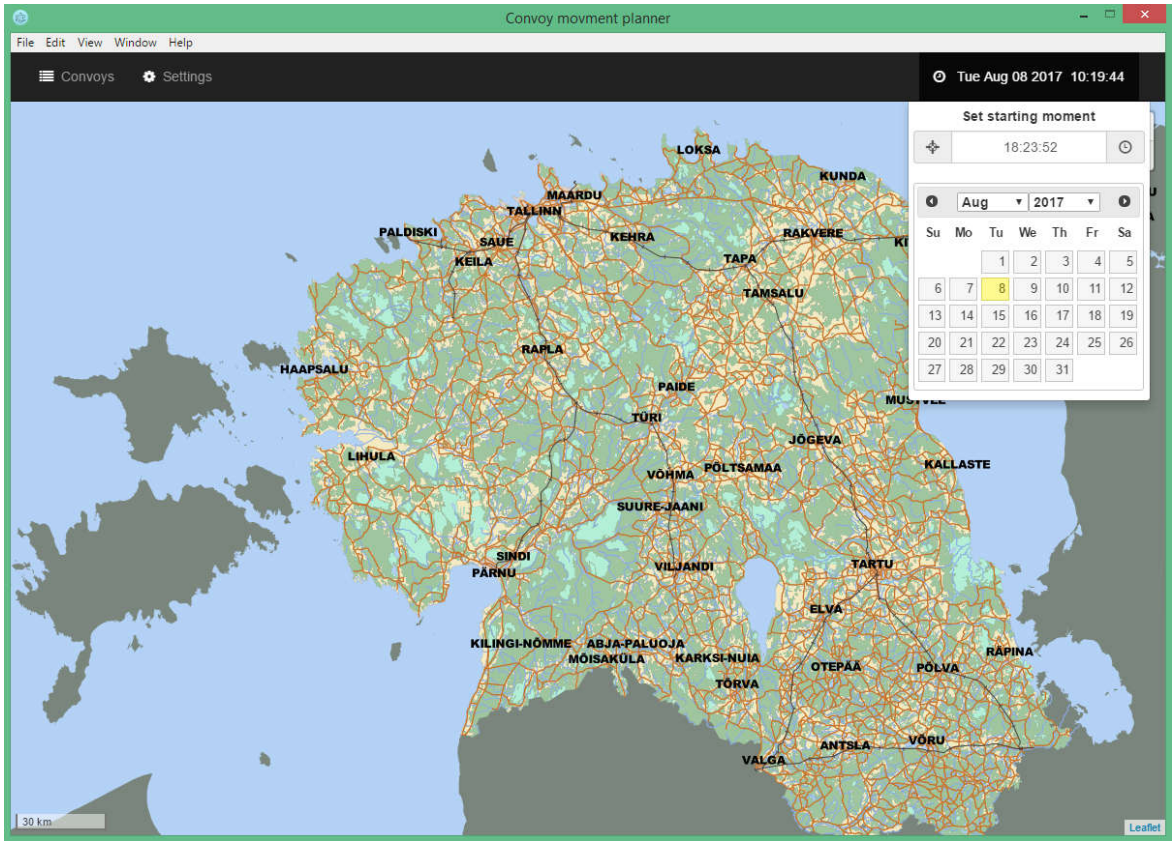


Figure 4. Opening view of the application with opened submenu of time section

[6] https://github.com/ewoken/Leaflet.MovingMarker
[7] https://github.com/hallahan/LeafletPlayback
[8] http://www.zerorpc.io/

User can start building the routing scenario by clicking on the “Convoys” submenu, which opens corresponding side menu (Figure 5). There user can click on an “Add a convoy” button to add a new convoy entry. Each entry consists of indicators for convoy ID, origin, destination, speed, length, earliest ready time and latest arrival time (due time). Last four indicators are initialized with default values, but can be modified, by using plus-minus buttons or by changing the input field. Origin and destination indicators act as icons and allow user to add respective markers to the map. When map is clicked marker is added to the nearest node on the road network and either origin or destination for particular convoy is fixed. At all times origin and destination markers can be dragged on the map to change their values.

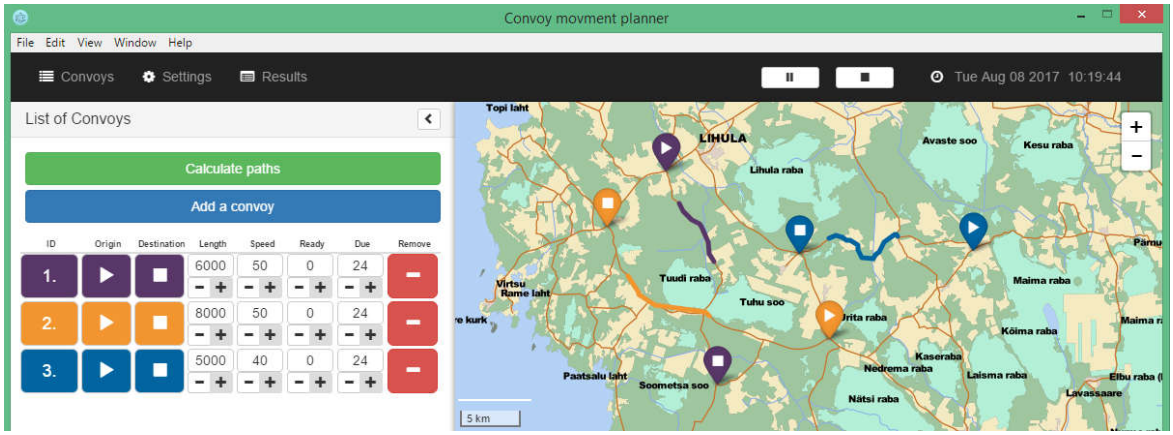


Figure 5. Convoys side menu on the left and running movement animation on the right

If there is at least one convoy entry with both origin and destination points set, then “Calculate paths” button is revealed. Clicking on this button computes the enquired paths and corresponding schedules and reveals “Results” submenu and playback buttons on the navigation bar. In addition to modifying, convoy entries can also be removed, by clicking on a “Remove” button behind each entry. If all fixed convoy entries are removed then “Results” submenu and playback options are once again concealed.

The time section on the navigation menu shows running date and time that represents the time 0 for convoy scheduling. Once “Calculate paths” button is pushed this clock is stopped and current date and time are saved as starting moment of calculations. User can also explicitly set the starting moment. If user clicks on the time section then clock is stopped and new drop-down submenu is opened (Figure 4). There user can set suitable starting date and time or return to the running clock.

Playback section is where user can start movement animations. Self-explanatory “Play”, “Pause”, “Stop” and ”Playback Speed” buttons control the flow of playback. Convoy movements are imitated as moving polylines on the map (Figure 5).

Clicking on a results submenu opens up the sliding top menu with scheduling results in tabular form. For each routed convoy start, finish, travel and pass times are shown. Travel time represents the time necessary to travel the distance of the route and pass time denotes the time required for the whole convoy to pass any given point along the route. Finish time is the sum of travel and pass times. Clicking on “Results” submenu also changes map appearance by drawing selected paths for each convoy on the map. For custom comparison convoy results can be individually turned on and off and user can zoom to any particular convoy by clicking the zoom icon (Figure 6).

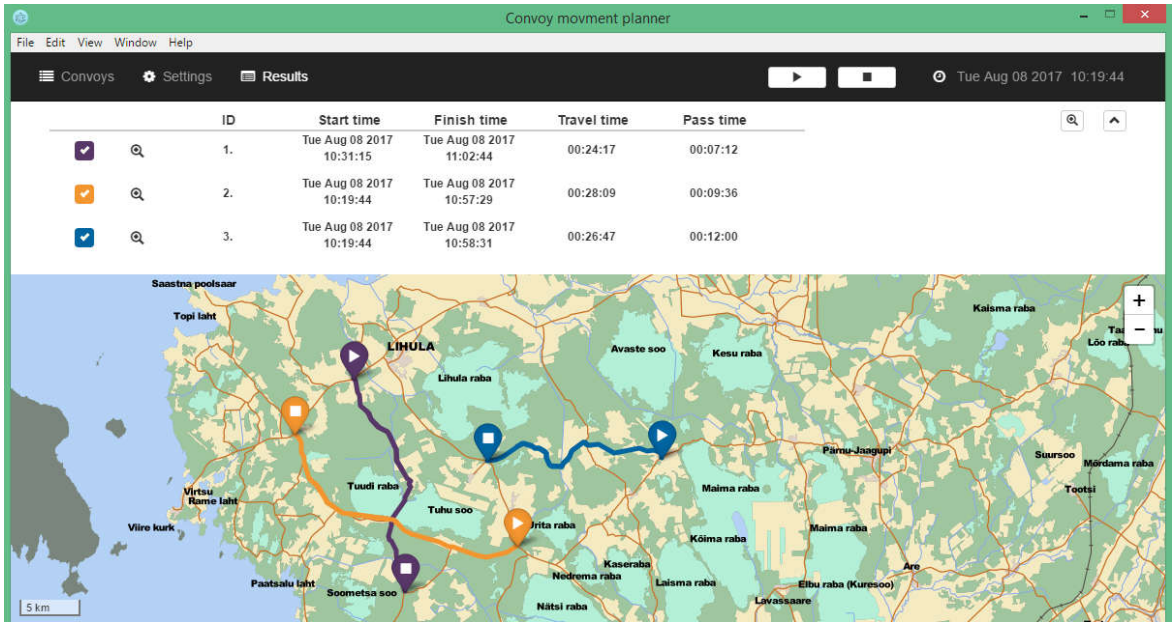


Figure 6. Results menu with scheduling results and selected paths on the map

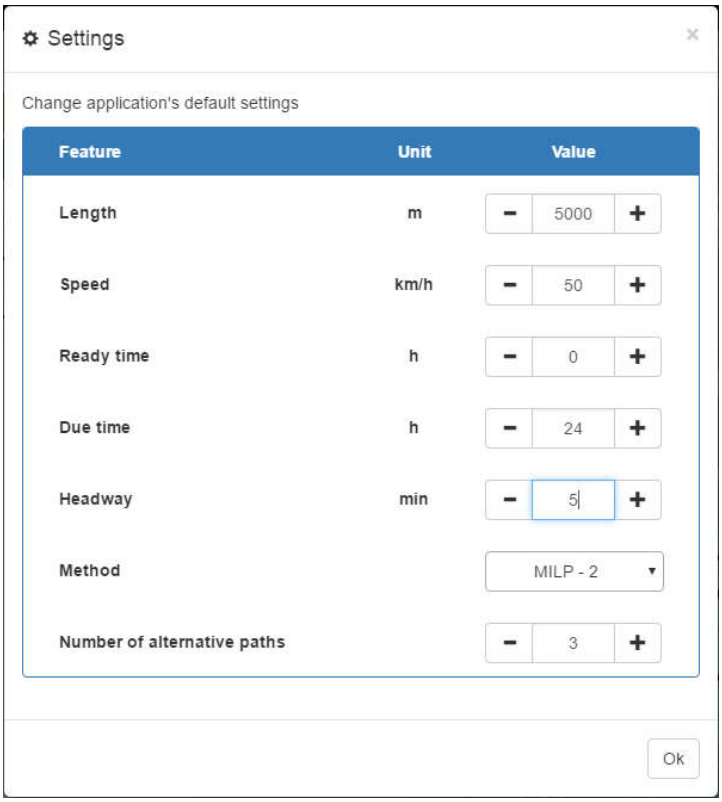


Figure 7. Settings pop-up window

Settings submenu opens in a pop-up window and allows user to change application's default settings. Those are default convoy parameters: length, speed, ready time and due time. And default scenario parameters: headway, method and number of alternative paths. Last setting, number of alternative paths, is available only if one of the MILP or BB methods is selected.

4 Results

In order to study the performance of different methods and the nature of the problem, several tests with different scenarios were performed on the problem. All computational studies presented in this section were run on an Intel Core i5-4200U 1.60 GHz processor with 8 GB of RAM with Windows 8.1 Enterprise as operating system.

Test cases are presented in appendix VI. In all scenarios all used convoys had the same length and speed properties and earliest ready time and latest arrival time constraints were not restrictive. All test cases used headway of 5 minutes between convoys. Test cases varied in following aspects: number of convoys used, number of conflicts between convoys, number of alternative paths calculated and length of paths used. Methods' runtime was restricted with 30 minute limit as this is could be a realistic upper bound on time used to perform calculations in outdoor conditions. All running times in tests were measured in real time (wall clock time). Test result are presented in Table 4.

Method MILP-1 is the only methodology here that guarantees optimal solution, it considers every possible route choice in the whole graph and every possible combination of scheduling. But as expected it fails almost all test cases. With more than 5000 nodes in the graph, MILP-1 creates multitude of variables for each node. For example for test case 1 total of 14613 variables is used when building a model with 37633 rows and 125776 columns. Optimizing over such set is where NP-hardness of the problem reveals itself. It is not straightforward to say what exact component causes the problem to be hard, but as a rule of thumb, the higher the number of variables and constraints gets, the harder the problem becomes. MILP-1 solved only test cases 1 and 2, but could not solve anything harder in allocated time. When used on smaller graphs the algorithm itself works well as can be seen from the results of method MILP-2.

MILP-2 ensures optimal solution in limited search space defined by the nodes of the *k*-shortest paths. As can be seen from the results the method runs fast when number of convoys is small and they are non-conflicting. In practice it can be used as a method of choice only for the simplest cases. For example test-case 5 took to five and half hours to solve.

Compared to previous methods MILP-3 and BB-1 are bounded to only limited set of paths for each convoy. So they can't guarantee optimality, but they produce good results for real-world test cases. Both methods consider every possible scheduling option for given paths. Result show that both methods produce same outcome in terms of objective value. When the number of paths to consider is relatively small and there are not many interactions between them then BB-1 runs substantially faster than MILP-3, but once problems get harder MILP-3 clearly outperforms exact method of BB-1. This is due to the fact that setting up the mathematical model takes significant effort, but once it is done, optimization on it is a lot faster. The efficiency of Gurobi routines is far superior to the pure branch-and-bound method. Depending on the number of conflicts, number of alternative paths used and the length of those paths, test cases indicate that this transition, when MILP-3 becomes faster than BB-1, happens when model has somewhere between 6 to 8 convoys in it. As the size of the problem grows the performance gap between the methods quickly gets wider and wider, indicating that methods clearly belong to different complexity classes. MILP-3 was able to solve all test cases, while BB-1 started to struggle with problems consisting of 10 convoys and failed to solve test case 11, where 12 convoys were used.

Table 4. Results for the test cases

	MILP-1	MILP-2	MILP-3	BB-1	BB-2	EFO	PFO
Test case 1							
Objective value (h)	0.22	0.22	0.22	0.22	0.22	0.22	0.22
Running time (s)	387.21	0.31	0.28	+	+	+	+
Test case 2							
Objective value (h)	0.63	0.63	0.63	0.63	0.63	0.63	0.63
Running time (s)	1167.34	0.99	0.54	0.03	0.03	+	+
Test case 3							
Objective value (h)	#	9.67	9.67	9.67	9.67	9.67	9.67
Running time (s)		18.77	17.09	11.81	11.79	0.14	0.14
Test case 4							
Objective value (h)	#	0.74	0.74	0.74	0.74	0.74	0.74
Running time (s)		339.62	4.47	0.27	0.13	0.05	+
Test case 5							
Objective value (h)	#	#	2.98	2.98	3.72	2.98	4.30
Running time (s)			3.03	0.78	0.17	1.33	+
Test case 6							
Objective value (h)	#	#	2.95	2.95	3.67	2.98	4.30
Running time (s)			23.43	79.53	30.47	1.33	+
Test case 7							
Objective value (h)	#	#	1.45	1.45	1.45	1.45	1.45
Running time (s)			4.73	0.50	0.08	4.47	+
Test case 8							
Objective value (h)	#	#	3.69	3.69	4.43	3.69	5.01
Running time (s)			10.38	30.16	6.31	81.73	+
Test case 9							
Objective value (h)	#	#	2.01	2.01	2.01	2.01	2.01
Running time (s)			12.09	93.99	4.29	710.56	+
Test case 10							
Objective value (h)			4.47	4.47	4.47	#	5.58
Running time (s)			21.17	1736.81	238.42		+
Test case 11							
Objective value (h)	#	#	5.86	#	#	#	7.18
Running time (s)			346.86				+

+ Method solved test case instantly, time could not be recorded

Method could not solve the test case in 30 minutes

BB-2 method is slightly faster than BB-1 method, but the loss in precision is also noticeable. The advantage of approximation reveals itself with more complicated test cases, but it is not magnitudes of faster therefore it also fails the last test case. It could be argued that alternative branching criteria could somewhat improve the result, but that still would not make this method relevant compared to others.

EFO algorithm gives good base for comparison. As can be seen from the test cases it produces same results as MILP-3 and BB-1 when latter don't use alternative paths. But EFO algorithm is clearly slower due to the optimization techniques used in MILP-3 and BB-1. Due to its exponential time complexity brute-force algorithm quickly becomes unusable once the number of convoys used approaches 10. The worst case time complexity of branch-and-bound algorithm is also exponential, but when comparing the results of EFO with BB-1 test cases show that on average it works much more efficiently.

Having a fixed order makes solving the problem simple. But if every possible ordering is not exhaustively considered then significant loss in objective value is expected. In presented test cases average loss of precision for method PFO was around 25%. But in practice this loss may not be relevant. Often convoys have priorities and in war-time situations this actually becomes prevalent. So priorities can be seen a as a feature not as a limitation. It must also be emphasized that in all test cases random order was used when determining the ordering. Therefore the average loss in routing precision can be significantly reduced when commanding officer takes location based priority decisions on convoys with similar priorities. PFO algorithm solved all test cases in this test set almost instantly. Test case 3 indicates that the limiting factor for this method might be the speed of Yen's algorithm. The longer the paths get the longer it takes to solve the problem. Even problem with 20 convoys, each with short paths, took less to solve than the test case 3 with only few long paths (0.03 vs. 0.14 seconds).

Case can be made that adding alternative paths to the optimization pool does not improve the result all that much. All methods designed in this project that use alternate paths, rely on Yen's algorithm. This algorithm looks for the smallest possible deviation from the original path to generate next shortest paths. This means that more than often generated path is very similar to the original path. And this similarity only grows if the paths get longer. Considering that Estonia's road network is relatively tense most generated paths usually have lot of nodes. This means that when path is incompatible with another path, its next alternative routing is probably also incompatible with that path. It might take several generations until conflict is resolved. But each new generation is costly and significantly reduces algorithms running time. This effect is clearly presented in running times of test cases 5 and 6, which differ only in the number of alternative paths used. Solving conflicts by generating new starting delays rather than new paths is computationally much more efficient. It appears wise to use alternative paths generations only in very tight conditions, where multiple convoys use the same small area for navigation.

Conflict resolution has important effect on running time for all methods. Comparing results from test case 4 and test case 6 shows that difference explicitly. But it is important to note that test case 6 was specifically designed to analyse conflict resolution and therefore it might not be representative for real-world instances. Considering Estonia's tense road network it can be assumed that conflicts are not actually as frequent as dreaded. Testing on real-world use-cases would be necessary to confirm that hypothesis. But if it is valid then this would clearly endorse the use of heuristic techniques and approximation algorithms, which in such conditions get much closer to optimal solution then otherwise. For example for PFO method this would make choice of ordering less critical and facilitate near-optimal results.

5 Discussion

This thesis has presented methodological foundations and a prototype for potential convoy routing tool. This section looks at the problem from a broader perspective and debates on several arguments that could make difference when pursuing to build this prototype into an operational planning tool.

Based on discussions held with specialists from Estonian National Defence College routing 10 convoys concurrently is plausible upper bound for real-world routing demands in Estonia. Accumulated results show that most reliable method for such conditions developed in this project is method MILP-3. It solved all test cases and guarantees quality results. If only one algorithm could be chosen to build a convoy routing application then this would be the algorithm of choice. In ideal conditions a hybrid of MILP-3 and BB-1 methods would best suited to solve routing problems. In such combination small and less complex problems could be solved with BB-1, while more demanding problems could use MILP-3.

MILP-1, MILP-2, BB-2 and EFO method did not yield as good as results as previously mentioned methods. When considering building a routing tool those methods can be excluded without regrets.

Alternative choice for the application would be to use PFO method. This method is clearly fastest, most transparent and does not rely on any commercial software. Moreover the significance of the loss in its objective value is certainly debateable. Firstly real-world instances of convoy movement problem are not necessarily always NP-hard, thus emphasizing effective conflict resolution and aiming for optimal solution might be overvalued. For example changing convoy's speed by few kilometres per hour or using smaller gaps between vehicles can sometimes have far greater effect on resulting objective value then solving the problem to its full optimality from average feasible solution. Also, constructing an optimal solution with a tight schedule could have significant downfalls. If something were to go wrong with one of the convoys, movement of all other convoys could be in jeopardy. It is hard to guarantee on-time performance for one convoy, but it is even harder to control the concurrent movement of set of convoys, not even considering potential adjustments that need to be made and communicated during the execution. Maybe it is a risk best left avoided and intended diversion from the optimal solution should be endorsed. If this is the case then PFO algorithm is more than sufficient for convoy planning needs.

Considering aforementioned statements it is unfortunate that more heuristic and meta-heuristic techniques were not explored in this theses. Literature review showed many examples where such methods were tried on convoy movement problem, but with little success in terms of solutions quality. But those articles were written from the mathematical perspective. Maybe setting sights on more practical goals, especially when considering the underlying need for a practical tool like in present case, those techniques could yet reveal new discoveries. Looking back, those methods should have been examined instead of the exact methodology approaches that was studied in this work.

Prototype that was designed for this thesis is not ready to be used as a planning tool, it is useful for basic visualisation and method evaluating, but lacks several features that should be added to the final application.

Currently results from the calculations cannot be saved nor exported. To fully integrate this tool into march order preparation and distribution this is the first thing that needs to be added. It would also be useful to have an opportunity to save set of convoys as a template

to speed up the problem construction process. Furthermore integrating a convoy formation functionality would bring the whole convoy building process into the application. This functionality should allow users to pick units and standalone vehicles from predefined list to form custom convoy columns with user-defined gaps between the vehicles.

Current prototype does not show path lengths in the results. This feature clearly has significant importance. For example, when planning fuel consumption for convoys. Luckily this feature is easy to incorporate.

For more precise movement execution convoy schedules almost always include several checkpoints along the route. Projected planning tool should allow user to add checkpoints on calculated paths and incorporate corresponding arrival times at those points into the schedule.

Suggested application should allow the addition of restrictions to the map. These could represent enemy locations or damages to the road network, but in terms of routing they would mark this set of edges where convoy movement is prohibited. If live graph modification is not available then otherwise operational application could quickly become obsolete in conflict situations.

Current prototype does not have a good priority management system for PFO method. This should definitely be added if this method is to be integrated into the tool. Additionally if hybrid solution of different methods is used in the application then maybe even a two-tier system could be used that differentiates convoys with same priority and convoys with different priority. If this is the case then maybe convoys with same priority could be routed by MILP-3, while departure time of next level convoy could be determined by PFO. Futhermore maybe even dynamic priorities could be incorporated to allow recalculations of routing during convoy movement. This functionality could be used to limit the damage of suboptimal movement execution and could have significant value for all convoy planning instances.

The application could also have an option to change optimization problem's objective function to make it more versatile. In addition to predictable travel-span add-on more ambitious routing objective could set a goal of undisturbed movement for the convoys. This could give preference either to primary roads, so that convoys would not have to yield the right of way to civilian vehicles on intersections or give preference to roads with least amount of traffic or limit the number of edges used in routing to avoid as much intersections as possible.

Several features could be added considering peace-time instances. Current formulation does not allow convoys to stop along the route, but this could be necessary for different reasons in peace-time conditions. Adding this feature into the application does not necessarily require reformulation of the problem. Convoy's movement stages could be modelled as different convoys in terms of original formulation and summarized in the user interface to give the perused impression.

Another possible sought feature would be to allow opposite direction movement for convoys on the same roads at the same time. This feature was actually already designed for branch-and-bound and fixed-order methods. But author was unable to come up with the correct mathematical formulation needed for the MILP methods. Consequently this feature was excluded from the presented results as no quantitative comparisons could have been made. There is no question that the ability to switch this feature on and off in pursued planning tool would have great value and further widen the possible use cases for this application.

There are many other modifications and additions that can improve this prototype, but author remains hopeful that current solution will ignite the interest and act as a spring board for future research and developments.

Summary

The goal of this thesis was to identify and design an algorithm suitable for concurrent military convoy movement optimization in Estonia and to build complementary application to visualize the calculated routing and scheduling results.

Thesis first introduced military backround of convoy planning to emphasize its complexity and to indicate deficiencies in current operational practice. Then formal definition of convoy movement problem used in academic literature was presented and thoroughly analysed. Based on that optimization methods and techniques relevant to the problem were examined and tied together with comprehensive overview of most notable publications in the field.

To identify the most suitable methodology for the problem seven different methods were designed and implemented on Estonian road network based graph. Three of those, distinguished by different heuristic add-ons, were based on mixed integer linear programming model that was implemented using Gurobi math programming solver. Next two branch-and-bound methodology based algorithms were developed using Python. And finally fixed-order routing methods using either exhaustive approach or predefined convoy ordering were created and coded in Python.

Main accomplishment of this work is the quality of designed methods. Even though convoy movement problem has been proven to be computationally NP-hard, its practical instances are not necessarily that complex. Test cases mimicking real-world like problem instances were all successfully solved. Mixed integer programing model with a heuristic extension that bounded all movements to predefined set of paths proved to produce quality solutions with reasonable running times for all examined test cases. The exact branch-and-bound method developed returned similar results, working even faster on smaller test cases, while just starting to struggle with most complex test cases. Designed fixed-order method with predefined convoy ordering was argued to be valuable even though it produces near-optimal result only in best case scenarios. Other algorithms created were less effective but provided a good comparison for analysis.

All aforementioned methods were integrated into a convoy routing application. This offline desktop application was built using modern web technologies and the main component of its simple user interface is an interactive map of Estonia where convoys' paths can be visualized and their movement animated. Designed application's core functionalities included convoy creation and modification, changing convoys' parameters, selecting an optimization method, displaying results of routing calculations as paths on map and as schedule in a table and animating convoy movement in a playback.

Thesis showed that the use of mathematical optimization methods can solve real-world instances of convoy movement problem fast and with quality results Hence they can make convoy planning process faster and improve the quality of decisions taken by officers. Futhermore the use of interactive application can make the process more transparent and controllable. Encouraging results presented in this thesis can be used as a foundation to future developments. At the end of this thesis, author described possible requirements and work needed to be done to turn current prototype application into an operational planning tool used to aid decision-making in military transportation operations in Estonia.

References

[1] U.S. Marine Corps. Convoy Operations Handbook. Honolulu, Hawaii: University Press of the Pacific. 2005.

[2] Tõniste T. Jalaväe pataljoni motorännak. Praktilisi näpunäiteid. Sõdur, 2015, no 83(2), pp 19-25

[3] Kumar P.N.R., Narendran T.T. Convoy Movement Problem - An Optimization Perspective, Innovations in Defence Support Systems – 1, pp 79-93

[4] Pehme A. Motoriseeritud rännak - kas asi iseeneses? Kaitse kodu!, 2009, no 86(2), pp 16-32

[5] Kumar P.N.R., Narendran T.T. Heuristics for Convoy Movement Problem, Strategic Analysis, 2009 no 33(4), pp 590-606.

[6] Bovet J., Constantin C., de Werra D. A convoy scheduling problem. Discrete Applied Mathematics, 1991, no 30(1), pp 1–14.

[7] Gopalan R. Computational complexity of convoy movement planning problems. Mathematical Methods of Operations Research, 2015, no 82(1), pp 1–30.

[8] Chardaire P., McKeown G.P., Verity-Harrison S.A., Richardson S.B. Solving a time-space network formulation for the convoy movement problem. Operations Research 2005, no 53(2), pp 219–230.

[9] Ho W., Ji P. Optimization Techniques. Optimal Production Planning for PCB Assembly, pp 7- 18.

[10] Rothlauf F. Optimization methods. Design of Modern Heuristics. Berlin, Heidelberg: Natural Computing Series, Springer, 2011, pp 45-102

[11] Lee Y.N., McKeown G.P., Rayward-Smith V.J. The Convoy Movement Problem with Initial Delays. Modern Heuristic Search Methods. England: John Wiley Sons 1996, pp. 213–233.

[12] Kumar P.N.R., Narendran T.T. On the usage of lagrangean relaxation for the convoy movement problem. Journal of the Operational Research Society, 2011, no 62(4), pp 722–728

[13] Tuson A.L., Harrison S.A. Problem difficulty of real instances of convoy planning. Journal of the Operational Research Society 2005, no 56(7), pp 763–775.

[14] Robinson E.M, Leiss E.L. Applying Genetic Algorithms to Convoy Scheduling. Artificial Intelligence in Theory and Practice, Santiago, 2006, pp 315-323

[15] Gopalan R., Narayanaswamy N.S. Analysis of algorithms for an online version of the convoy movement problem. Journal of the Operational Research Society 2009, no 60, pp 1230–1236

[16] Sadeghnejad-Barkousaraie A., Batta R., Sudit M. Convoy Movement Problem: A Civilian Perspective. Journal of the Operational Research Society, 2017, no 68(1), pp 14-33.

[17] http://geoportaal.maaamet.ee/eng/Ordering-Data/Opendata-for-download/Generalized-Estonian-topographic-data-p554.html [Accessed 4 May 2017].

Appendix

I. Convoy commander's checklist

Mission Requirements

- Current Intelligence/Situation
- Task Vehicles: Type and Quantity
- Personnel
- Cargo by Type, Class, and Size
- Security Vehicles: Type and Quantity
- Maintenance Vehicles
- Materials Handling Equipment
- Command and Control Vehicles: Type and quantity
- Lighting/Blackout Conditions / NVGs

Reconnaissance

- Map and Photo
- Physical

Route Selection

- Road
- Bridges and Tunnels
- Grades and Curves
- Traffic Density
- Requirements for Route Preparation or Repair

Liaison and Coordinate

- Units along Route
- Units Being Moved
- Supporting Units
- Highway Control Agencies/Movement Control Centers
- Shippers / Cargo Handlers
- Engineer / Explosive ordnance disposal requirements

Convoy Organization

- Size of Serials / March Units
- Type of Column
- Operating Gaps
- Serials/March Units
- Vehicles
- Positions of Security and Supporting Units
- Positions of Control Personnel/Escorts Guides
- Organization for Command
- Vehicle Marking

Movement Plan

- Controlled Route
- Convoy Clearance / Movement Credit
- Road Movement Table
- Special Permits or Authorization
- Distance, Time, and Rate of Movement
- Trip Distance
- Required Start Time
- Column Length
- Slowest Vehicle
- Required Delivery Time
- Rate of Movement / Speed (Speedometer Multiplier)
- Maximum Catch-up Speed
- Loading
- Time and Place
- Report to
- Type / Class Cargo
- Outsize Loads
- Materials Handling Equipment Required
- Blocking, Bracing, and Cargo Restraints
- Staging
- Location

Movement Plan (Continued)

- Vehicle Checks
- Cargo Checks
- Time to Start Point
- Operator Briefing
- Start Point
- Location / Grid Coordinates
- Identification Characteristics / Alphanumeric Designators
- Checkpoints
- Guides and Markers
- Positions
- Posting and Pickup
- Halts
- Purpose
- Time Duration
- Locations
- Maintenance
- Trail
- Enroute Support
- Medical Support
- Organic Capability
- Evacuation
- Release Point
- Location/Grid Coordinates
- Identification Characteristics
- Report Requirements
- Control of Vehicles and Operators
- Unloading
- Time and Place
- Report to HHQ at Destination
- Materials Handling Equipment Required
- Backload and Turn Around

Security Enroute

- Action in Event of Attack
- Air Attack
- Artillery Attack
- Ground Attack
- Sniper
- Air Support Procedures
- Fire Support Procedures
- Use of Lights/Blackout Restrictions

Service Support

- Fuel
- Location / Times
- Types and Quantity
- Accompanying Convoy
- Messing / Rations
- Units on Route
- Prescribed Loads

Communications

- Convoy Control Net
- Serial/March Unit Commanders
- Parent Unit/Headquarters
- Alert/Broadcast Net
- Security/Tactical Nets
- Fire and Air Support Nets
- Medical Evacuation
- Visual Signals
- Sound Signals
- Interpreter Requirements

Convoy Commander's After-Action Report

II. BB-1 method's pseudocode

Input: Set of convoys, number of alternative paths (*K*)
Output: Set of pairwise compatible paths (*BestCollection*)

```
for Convoy in Convoys:
    AllFeasiblePaths <- GenerateShortestPaths(Convoy, K);
LB = -∞;    # lower-bound
UB = ∞;     # upper-bound
Collection = {};
Level = 1;
N = length(Convoys);
BestCollection, UB = Branch(AllFeasiblePaths, Collection, LB, UB, Level, N)
    if Level == N:
        Collection <- First path in AllCompatiblePaths for N;
            Update UB;
            Fathom by 2;
    else:
        for Convoy in Convoys:
            for Path in AllFeasiblePaths[Convoy]:
                Collection <- Path;
                AllCompatiblePaths = ReplaceIncompatiple(AllFeasiblePaths, Collection);
                if AllCompatiblePaths is empty for non-assigned convoy:
                    Fathom by 3;
                Update LB;
                if LB ≥ UB:
                    Fathom by 1;
                Level <- Level + 1;
                Collection, UB = Branch(AllCompatiblePaths, Collection, LB, UB, Level, N);
return BestCollection
```

III. BB-2 method's pseudocode

Input: Set of convoys, number of alternative paths (*K*)
Output: Set of pairwise compatible paths (*BestCollection*)

```
for Convoy in Convoys:
    AllFeasiblePaths <- GenerateShortestPaths(Convoy, K);
LB = -∞;    # lower-bound
UB = ∞;     # upper-bound
Collection = {};
Level = 1;
N = length(Convoys);
BestCollection, UB = Branch(AllFeasiblePaths, Collection, LB, UB, Level, N)
    if Level == N:
        Collection <- First path in AllCompatiblePaths for N;
            Update UB;
            Fathom by 2;
    else:
        SelectedConvoy = FindPathWithMinimumObjectiveValue(AllFeasiblePaths)
        for Path in AllFeasiblePaths[SelectedConvoy]:
            Collection <- Path;
            AllCompatiblePaths = ReplaceIncompatiple(AllFeasiblePaths, Collection);
            if AllCompatiblePaths is empty for non-assigned convoy:
                Fathom by 3;
            Update LB;
            if LB ≥ UB:
                Fathom by 1;
            Level <- Level + 1;
            Collection, UB = Branch(AllCompatiblePaths, Collection, LB, UB, Level, N);
return BestCollection
```

IV. PFO method's pseudocode

Input: Set of Convoys, Ordering
Output: Set of pairwise compatible paths (*Collection*)

```
for Convoy in Convoys:
    Paths <- GenerateShortestPath(Convoy);
Paths = Sort(Paths, Ordering);
Collection = {};
for Path in Paths:
    if Path has conflict with paths in Collection:
        Path = CalculateNewDepartureTime(Path, CurrentCollection);
Collection <-Path;
return Collection
```

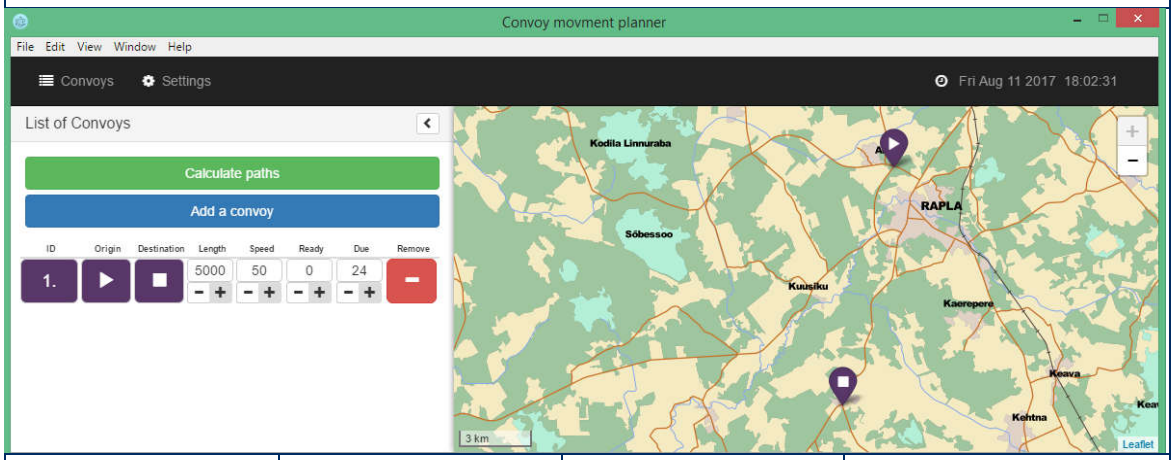
V. EFO method's pseudocode

Input: Set of convoys
Output: Set of pairwise compatible paths (*BestCollection*)

```
for Convoy in Convoys:
    Paths <- GenerateShortestPath(Convoy);
BestUpperBound = ∞;
BestCollection = {};
Orderings = FindAllPermutations(Paths);
for Ordering in Orderings:
    CurrentCollection = {};
    for Path in Ordering:
        if Path has conflict with paths in CurrentCollection:
            Path = CalculateNewDepartureTime(Path, CurrentCollection);
    CurrentCollection <-Path;
    UpperBound = SummarizeObjectiveValues(CurrentCollection);
    if UpperBound < BestUpperBound:
        BestUpperBound = UpperBound;
        BestCollection = CurrentCollection;
return BestCollection
```

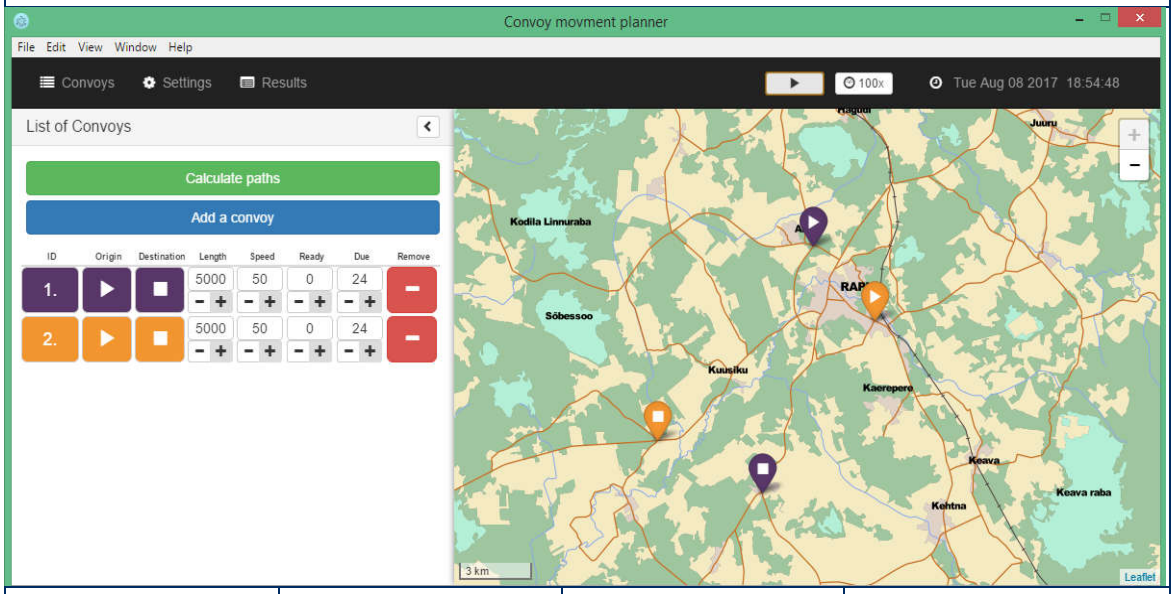
VI. Test cases

Test case 1



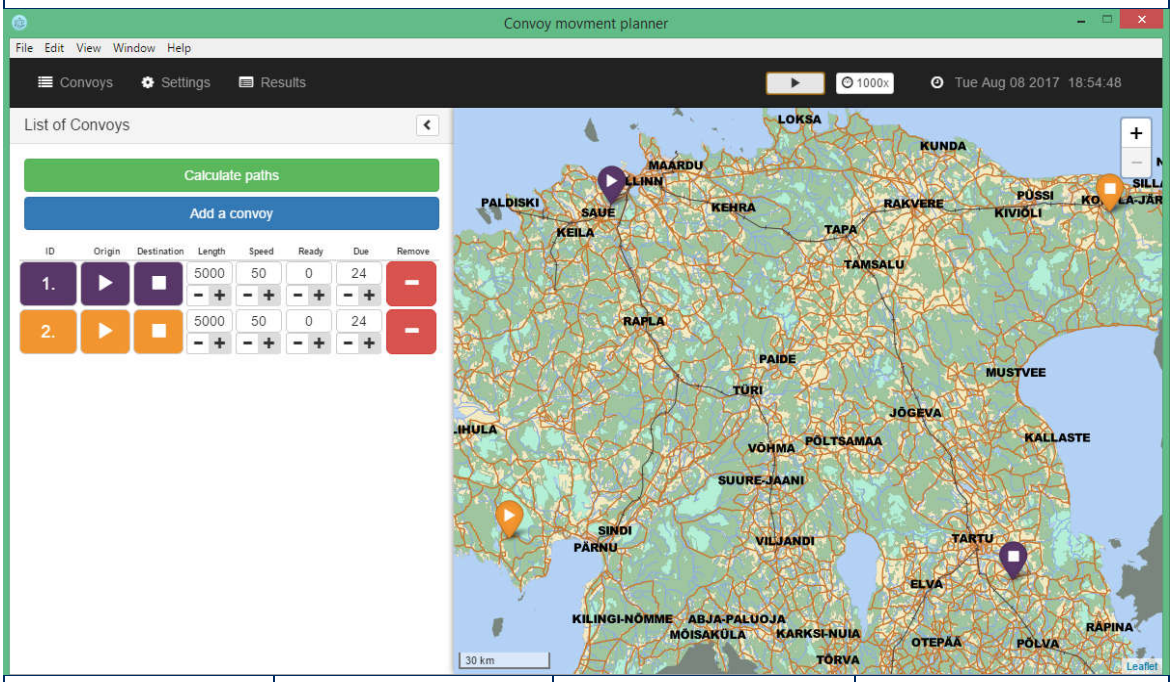
Number of convoys	Conflicts present	Number of alternative paths	Average number of nodes per optimal path
1	No	1	5

Test case 2



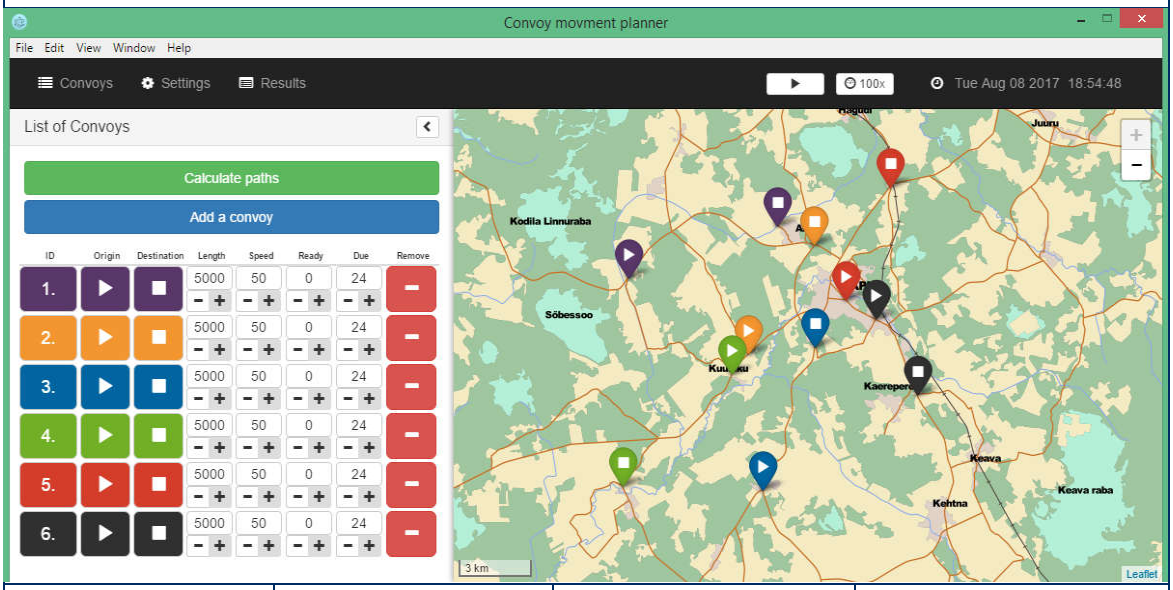
Number of convoys	Conflicts present	Number of alternative paths	Average number of nodes per optimal path
2	Yes	3	6.5

Test case 3



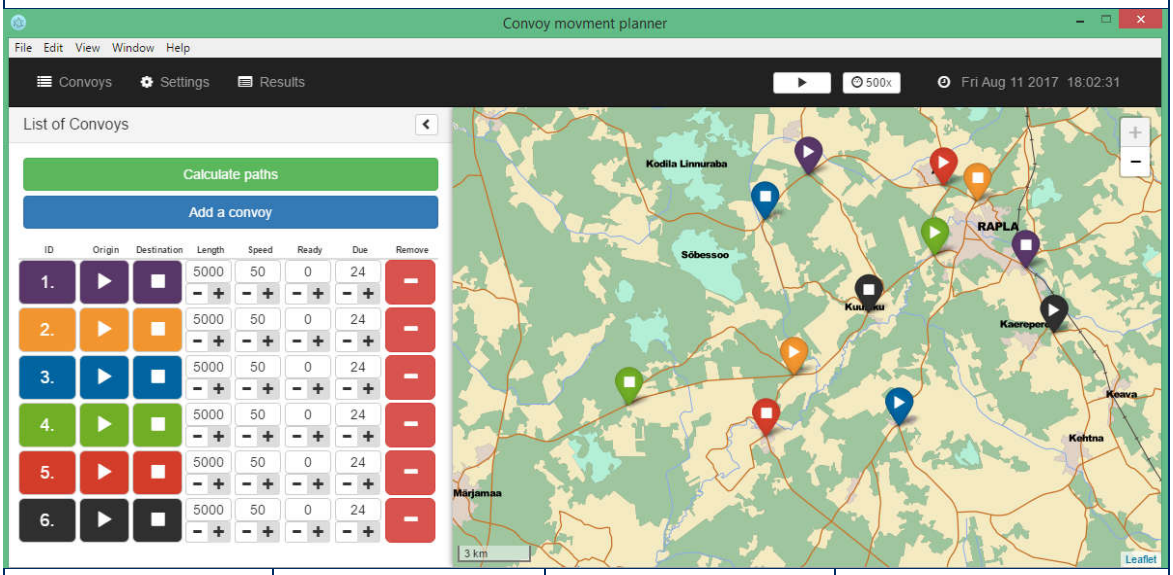
Number of convoys	Conflicts present	Number of alternative paths	Average number of nodes per optimal path
2	No	3	105.5

Test case 4



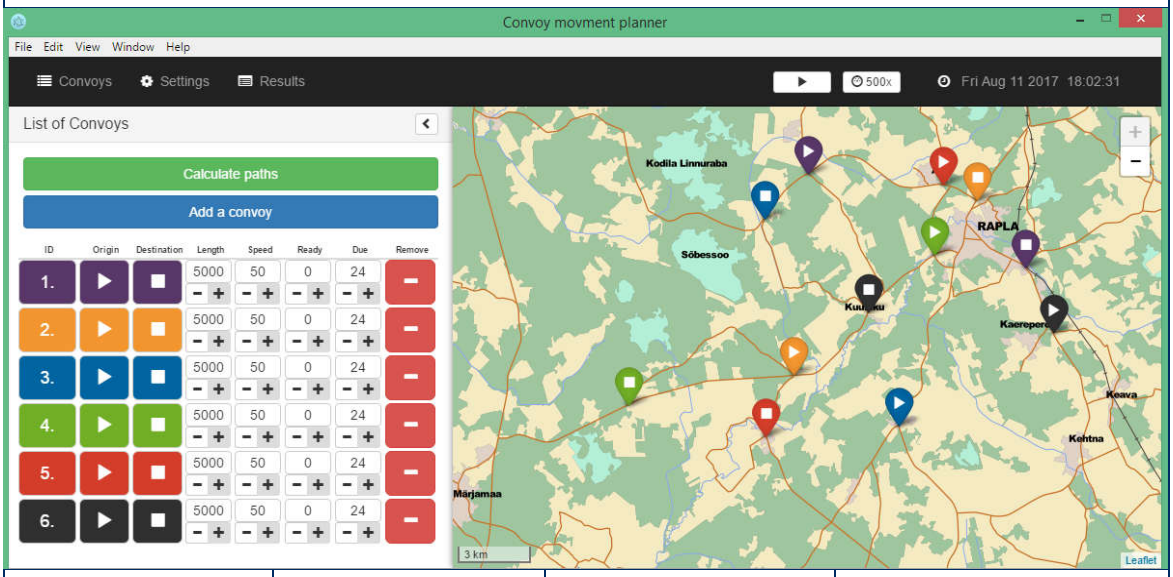
Number of convoys	Conflicts present	Number of alternative paths	Average number of nodes per optimal path
6	No	3	3.8

Test case 5



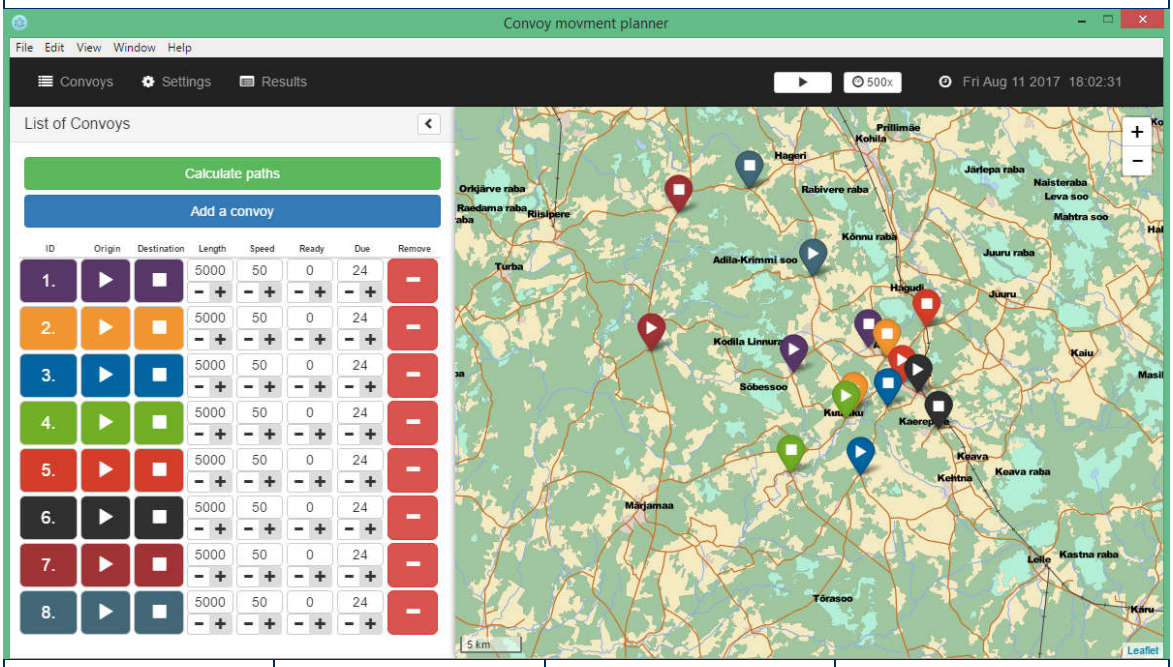
Number of convoys	Conflicts present	Number of alternative paths	Average number of nodes per optimal path
6	Yes	1	8.5

Test case 6



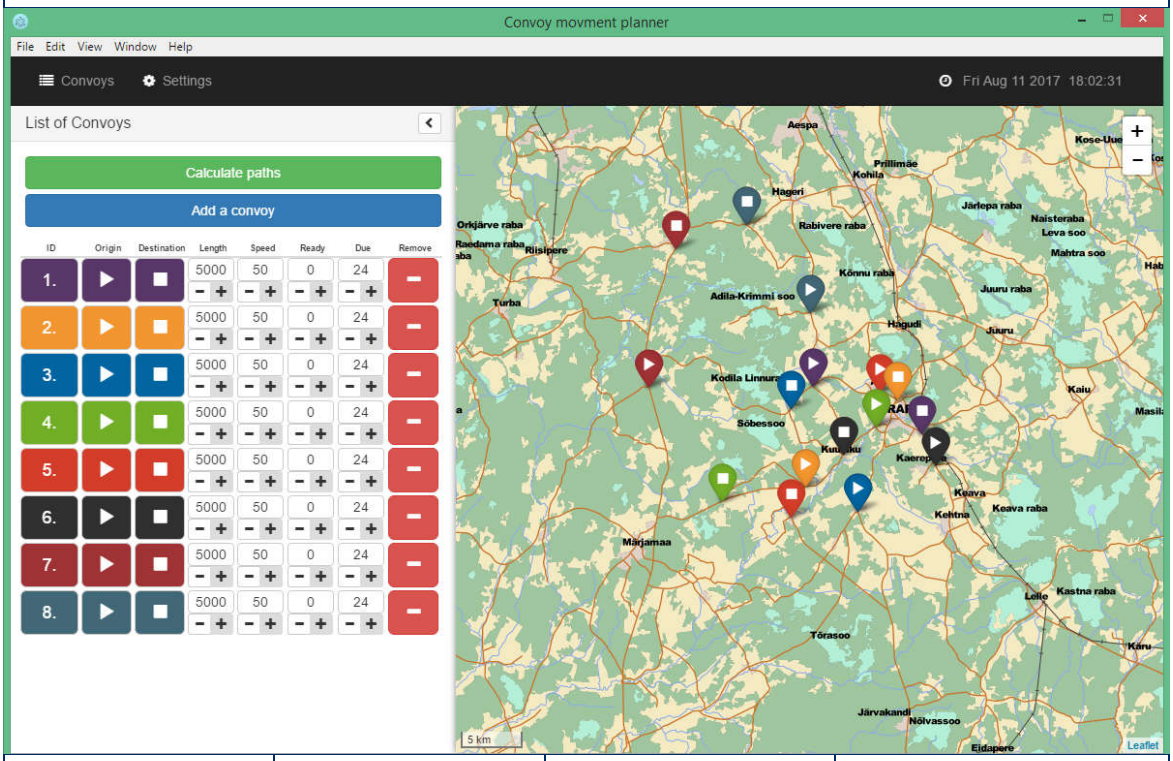
Number of convoys	Conflicts present	Number of alternative paths	Average number of nodes per optimal path
6	Yes	3	8.5

Test case 7



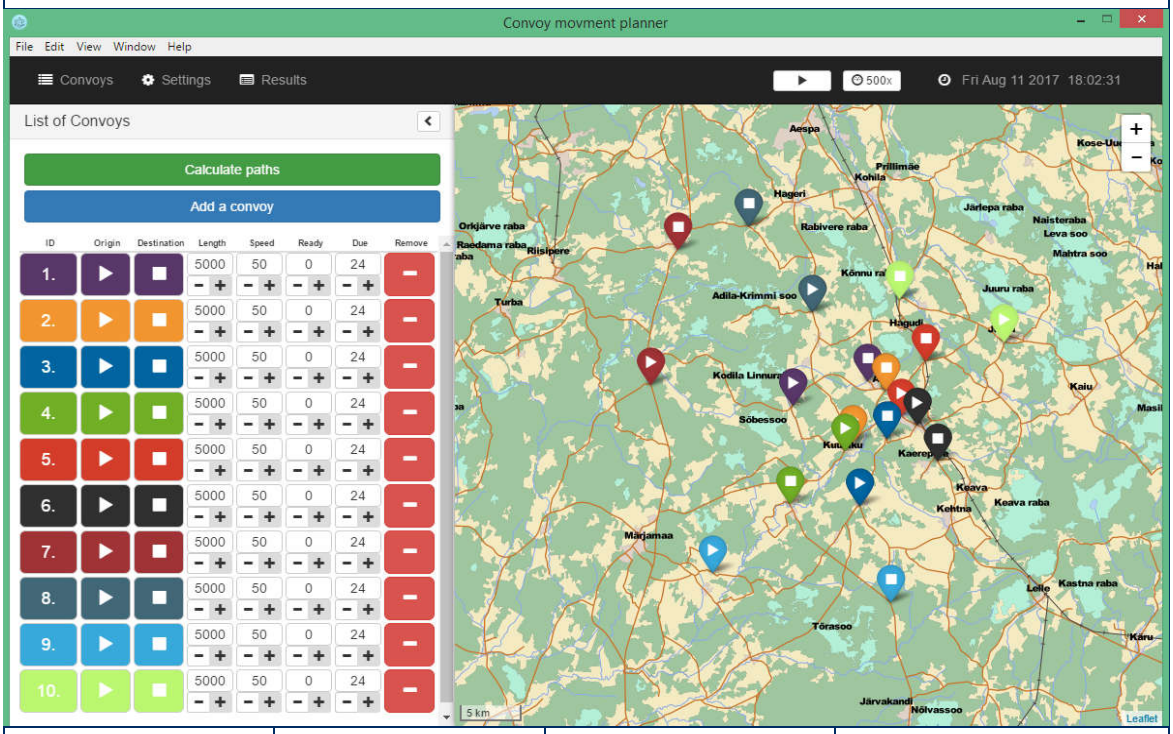
Number of convoys	Conflicts present	Number of alternative paths	Average number of nodes per optimal path
8	No	1	4.5

Test case 8



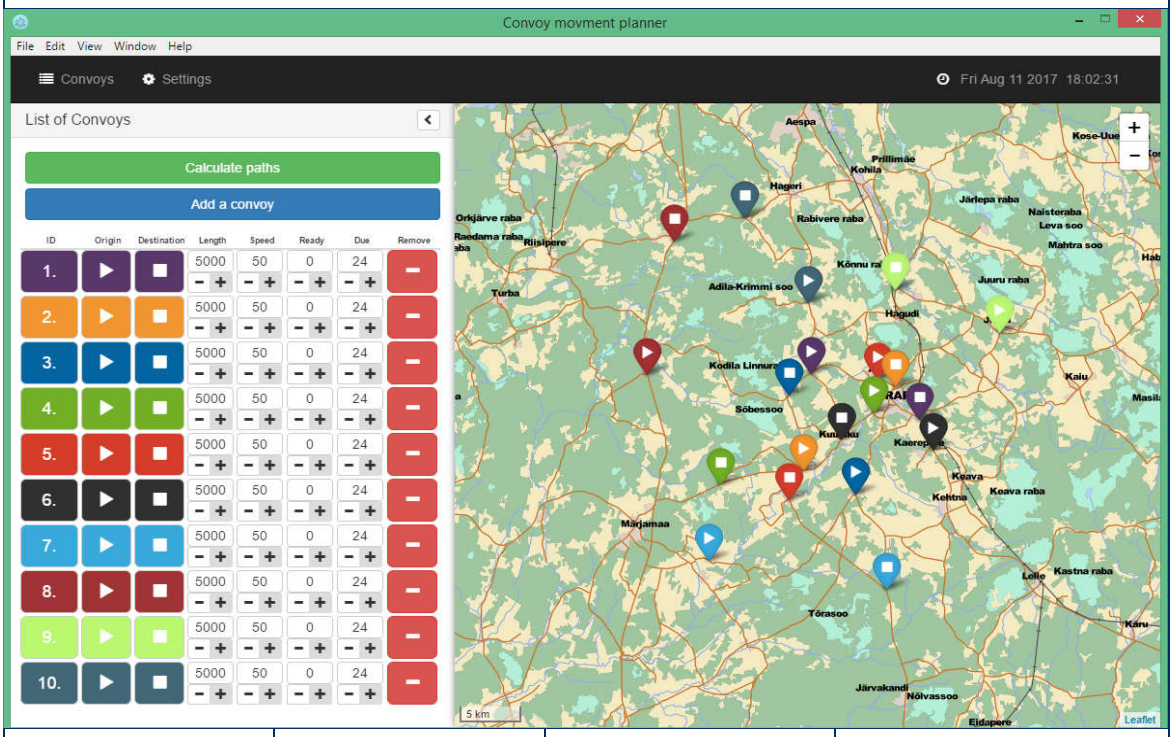
Number of convoys	Conflicts present	Number of alternative paths	Average number of nodes per optimal path
8	Yes	1	8

Test case 9



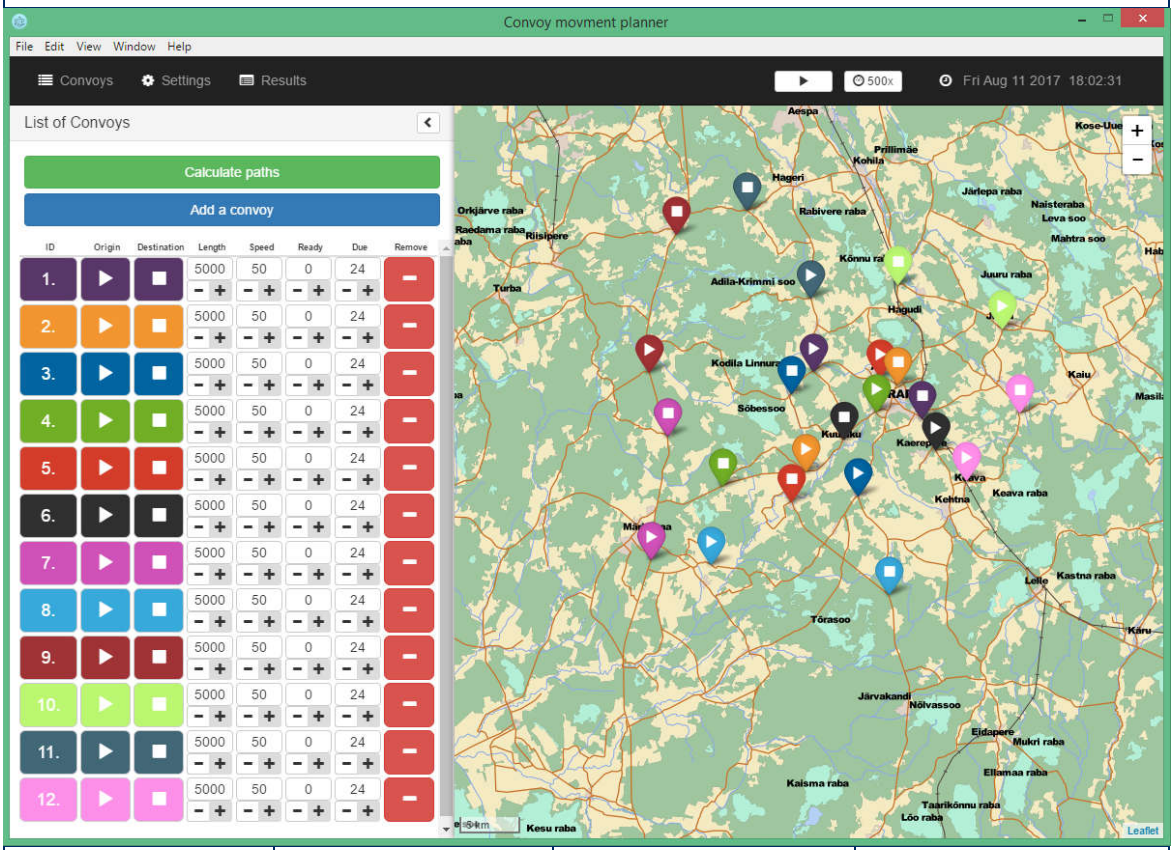
Number of convoys	Conflicts present	Number of alternative paths	Average number of nodes per optimal path
10	No	1	4.6

Test case 10



Number of convoys	Conflicts present	Number of alternative paths	Average number of nodes per optimal path
10	Yes	1	7.4

Test case 11



Number of convoys	Conflicts present	Number of alternative paths	Average number of nodes per optimal path
12	Yes	3	6.8
10	Yes	3	8.5

VII. License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Meelis Tapo**,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:
    1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
    1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

Optimization of Military Convoy Routing,

supervised by **Bahman Ghandchi** and **Dirk Oliver Theis**.

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **14.08.2017**