Applying heuristic search strategies to plan with reconfigurable multi-robot systems

by

Rahul B. Shrestha

Bachelor Thesis in Computer Science

Submission: October 22, 2020
First reviewer: Prof. Dr. Francesco Maurelli, Jacobs University Bremen
Second reviewer: Prof. Dr. Frank Kirchner, University of Bremen
Mentor: Dr. Thomas Röhr, DFKI Bremen

Jacobs University Bremen | Department of Computer Science and Electrical Engineering

English: Declaration of Authorship

I hereby declare that the thesis submitted was created and written solely by myself without any external support. Any sources, direct or indirect, are marked as such. I am aware of the fact that the contents of the thesis in digital form may be revised with regard to usage of unauthorized aid as well as whether the whole or parts of it may be identified as plagiarism. I do agree my work to be entered into a database for it to be compared with existing sources, where it will remain in order to enable further comparisons with future theses. This does not grant any rights of reproduction and usage, however.

This document was neither presented to any other examination board nor has it been published.

Date, Signature

Abstract

This thesis investigates heuristic search strategies that can be applied to planning with reconfigurable multi-robot systems.

In the Vehicle Routing Problem (VRP), an optimal route for a set of vehicles to deliver a set of items to a set of customers is searched upon. Temporal Planning for Reconfigurable Multi-Robot Systems (TemPl) is a constrained-based mission planner which introduces a mission description as a generalisation of the Vehicle Routing Problem (VRP). Due to this generalisation, heuristic search strategies used to find solutions in VRP could also possibly be used on TemPl. TemPl contains multiple constraints (spatial, temporal, load, inter-route) that aren't considered by the classical VRP or its variants. Hence, heuristics used to solve VRP require to be adapted before applying to TemPl.

For this reason, this thesis intends to evaluate existing literature and find heuristic search strategies that can be applied to robot planning with TemPl. An implementation of the Very Large Neighbourhood Search algorithm serves as proof of concept for the applicability of heuristic search strategies for robot planning.

Contents

1 Introduction and Motivation of Research

Heuristic search strategies are used in combinatorial optimization problems to find solutions in large problem spaces. Approximate solutions are found by optimising for speed rather than accuracy as finding exact solutions can be computationally expensive.

The Vehicle Routing Problem (VRP) [9] is a widely studied combinatorial optimization problem. In VRP, an optimal route is to be found through which a set of vehicles deliver items to a set of customers. This optimal route must have the lowest possible cost, which could be dependent on total distance travelled, total energy consumed or total time spent delivering. There exists several variants of VRP depending on the constraints and characteristics of vehicles and customers. Some popular variants are Capicatated VRP(CVRP) [15], VRP with Time Windows (VRPTWW) [8] and VRP with Pickup and Delivery (VRPDD) [10].

Reconfigurable robots can physically merge to create dynamic robotic systems, leading to additional degrees of freedom. To operate such systems autonomously, mission planning is required to deal with the flexibility of composite agents. For this purpose, Roehr(2019) [26] introduced Temporal Planning for Reconfigurable Multi-Robot Systems (TemPl), a constrained-based mission planner. This planner introduces a mission description as a generalisation of the Vehicle Routing Problem (VRP). TemPl uses a combination of knowledge-based reasoning, constraint-based programming and linear programming [25] to convert a mission specification into a mission solution. TemPl translates a mission description into a multicommodity min-cost flow problem [3] to identify a locally optimal plan. This optimization is solved as a linear integer problem (LP) which can be solved using existing linear integer program solvers. The size of the multicommodity min-cost flow problem as a linear program depends on the characteristics of the underlying solution representation, as well as the number of active mobile and immobile agents that need to be routed. This optimal planning approach and the used LP solvers limits the scalability of this planning approach.

As the mission description is a generalisation of the VRP, heuristics used for VRP could also possibly be applied to TemPl. TemPl's mission specification consists of temporal, spatial and inter-route constraints as compared to classic VRPs, so heuristics applied to variants of VRP with multiple constraints has been considered in this thesis. Additionally, most heuristic search strategies for classical VRPs assume routes to be mutually independent and don't consider time windows, capacities, vehicle synchrnoisation and heteregenous agents [25]. Drexl(2013) [11] noted how research on VRP under multiple synchronisation constraints is scarce, and the use of heuristics to solve it is even scarcer.

Therefore, this thesis aims to explore and evaluate the use of heuristic search strategies such as Simulated Annealing [1], Tabu Search [30] or Very Large Neighbourhood Search (VLNS) [7] that could optimise the planner, TemPl.

This thesis is structured in the following way. Section 2 describes the technical background necessary to understand concepts used in this thesis. Section 3 investigates the state-of-the-art in heuristic search strategies. Section 4 describes the setup of the experiment and discusses the obtained results. Section 5 summarizes the results obtained and Section 6 suggests possible further investigation ideas.

2 Technical background

This section explains important technical concepts required to understand parts of this thesis. This thesis is about applying heuristic search strategies to reconfigurable multi-robot systems. So, we start with an explanation of reconfigurable mulit-robot systems and its system properties. Since TemPl introduces a mission description as a generalisation of the Vehicle Routing Problem, we explain the Vehicle Routing Problem and the planner, TemPl. Finally, we explain the representation of a mission specficiation and mission solution in TemPl. The stateof the art in ”Heuristic search strategies” is explained in the next section.

2.1 Reconfigurable multi-robot systems

Reconfigurable multi-robots can physically merge to form composite systems, offering additional capabilities as compared to the previously individual systems. Its modularity helps adapt to challenges in robotic missions. Each heterogeneous agent used to form a reconfigurable multi-robot system contains individual capabilities, functionalities and limitations. A reconfigurable robot contains a standardised interface which allows new hardware to be attached and removed, so resources can be shared as required.

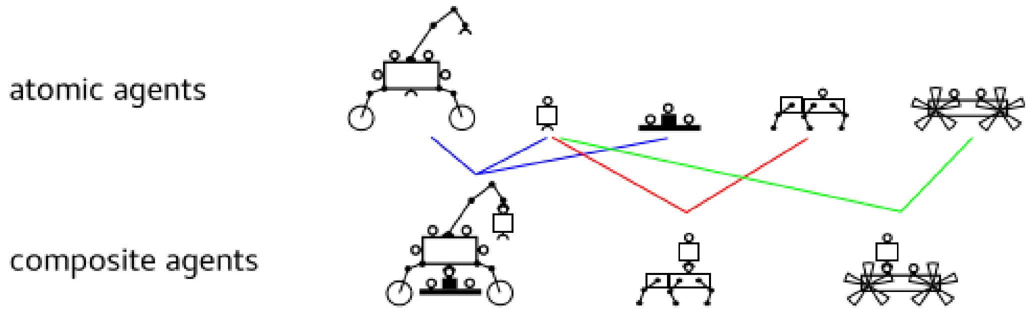

Figure 1: *Possible composite agents formed by combining atomic agents(Roehr 2019, Pg.29)* [26]

Composite agents may be required in certain scenarios of a mission. For example, an immobile agent can be combined with mobile agent to transport it from one location to another. Some functionality in composite agents appears through the ”super-additive effect” [25], when two or more atomic agents combine to form a single composite agent. Composite agents increase ”functional redundancy”, i.e. if one agent in a composite system fails, it can be replaced by another redundant agent. Adding relevant resources increases the redundancy and safety of an operation.

2.2 Reconfigurable system properties

Roehr (2019) [26] uses the flexibility of reconfiguration in planning and characterises solutions based on three system properties: efficacy, efficiency and safety.

The efficacy of a reconfigurable system is its ability to produce a desired result. If a mission consists of performing N tasks, then efficacy describes the fraction of completed tasks. If all tasks are completed, the efficacy is 1.

The efficiency of a reconfigurable system is determined by its execution time and energy consumption. The lower the execution time and resources used, the higher the efficiency. Similarly to how carpooling in cities [19] helps reduce energy usage, a reconfigurable system can increase its efficiency by making full use of the capabilities of a single robotic system.

The safety of a reconfigurable system, as designed by Roehr (2019) [26], is dependent on the redundancy of resources. A higher number of available resources means the whole system has a lower probability of failing, as the single resource is replaceable. Safety is highly prioritised as robotic systems are deployed and operated remotely. Losing even a single robotic system is expensive.

Efficiency, efficacy and safety are dependent on each other. A system could increase its efficiency and efficacy by increasing its safety i.e. its rate of survival. Conversely, increasing safety could decrease efficiency as higher the redundancy of resources leads to higher energy usage. Therefore, it is important to find a balance between all three properties to achieve optimal results.

2.3 Vehicle Routing problem

In the Vehicle Routing Problem (VRP) [9], a set of vehicles and a set of items to be delivered are set. The objective is to find the optimal route used by the group of vehicles to deliver the items at a minimum cost. This is similar to deciding which vehicle delivers which item in which sequence such that all vehicle routes can be feasibly executed. The total cost depends on the total distance travelled by the vehicles and the number of used vehicles.

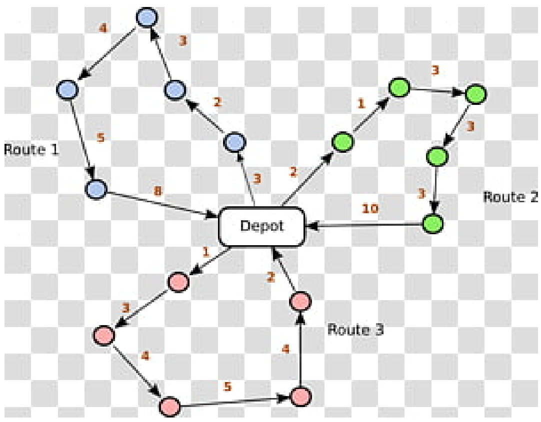

Figure 2: *Graphical representation of VRP, where each vehicle has its own route for delivering items* [22]

VRP and its variants has been extensively studied in the field of combinatorial optimization. Dantzig and Ramser(1959) [9] first described VRP as routing delivery trucks between stations and a terminal. Real world problems are more complex than the classical

VRP. The need for additional constraints need to be considered created several variants of VRP. In Capacitated VRP (CVRP) [15], each vehicle has a limit of items it can carry. In VRP with Time Windows (VRPTWW) [8], each delivery needs to be made within a certain time interval. In VRP with Pickup and Delivery (VRPDD) [10], vehicles need to pick up and deliver items from one location to another.

VRP is a NP hard combinatorial optimization problem, so solving large problems is not always feasible in CPU time. Heuristics are employed to find acceptable solutions in a short time.

2.4 Vehicle Routing problem with Trailers and Transshipments

A more relevant VRP variant to planning with reconfigurable multi-robot system problem. is the VRP with trailers and transshipments (VRPTT) [11]. VRPTT is representative of the class of vehicle routing problems with multiple synchronization constraints (VRPMSs). VRPMSs is similar to the mission description introduced by TemPl. VRPMSs contain temporal, spatial and load constraints. A change in one vehicle's route affects other vehicles' routes due to synchronization constraints, which is not the case in classical VRP. Most heuristic algorithms used to solve vehicle routing problems assume the routes to be mutually independent, which can't be used to solve VRPMSs. Changing a route could render the whole solution useless.

VRP with trailers and transshipments (VRPTT) was used to model a real world problem of collecting milk from farmyards [12]. The problem description is as follows. Vehicles are split into two categories: lorries (autonomous vehicles that can move on their own) and trailers (non-autonomous vehicles that require an autonomous vehicle to be moved). Additionally, there are "task vehicles", which visit customers and collect milk, and "support vehicles", which are used as mobile depots for task vehicles to transfer their load. Transfers occur in "transshipment locations". Selecting transshipment locations could also create additional costs per each transshipment.

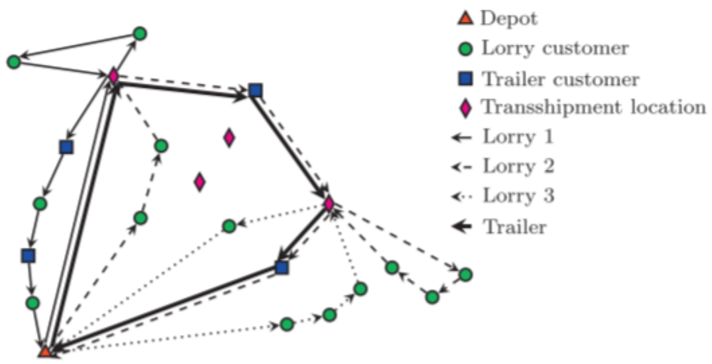

Figure 3: *Route plan of VRP with trailers and transshipments* [11]

As VRPTT is a subproblem of VRPMSs, multiple constraints exist. In addition to spatial, temporal and load constraints, vehicles may have accessibility constraints. Not every vehicle may be able to visit every location. The number of vehicles that can use a certain transshipment location and the number of transshipment operations that can occur in a transshipment locations is also limited. A time window for allowing transshipments can also exist.

Each customer has their own requirements. Customers that can be visited by lorries without a trailer are called “lorry customers”. Customers that can be visited by lorries with or without a trailer are called “trailer customers”.

The problem is to determine a route for the lorries and trailers such that the entire supply of customers is collected and dropped to a depot at the lowest possible cost. The final solution must respect all synchronization, temporal, spatial and load constraints.

2.5 Temporal Planning for Reconfigurable Multi-Robot systems (TemPl)

Roehr (2019) [26] introduced TemPl, a constraint-based mission planner for reconfigurable multi-robot systems. The planner uses a mission description as a generalisation of the Vehicle Routing Problem. TemPl converts planning with reconfigurable multi-robot systems into a multicommodity min-cost flow problem.

The mission planning problem is similar to the vehicle routing problem. “Mobile agents” are vehicles and “immobile agents” are items to be delivered. “Reconfiguration” is a transshipment between two vehicles. A key difference remains, however, that reconfigurable multi-robot systems can change their functional properties depending on the requirements of the mission. Reframing the mission planning problem as a vehicle routing problem allows it to be formalized as a multicommodity min-cost flow optimization problem so that locally optimal search can be applied. This is solved using linear programming in TemPl.

2.6 Mission planning with TemPl

The mission planner TemPl (Temporal Planning for Reconfigurable Multi-Robot Systems) introduces a mission description as a generalisation of VRP. This section explains how missions are represented and planned in TemPl.

Space exploration missions require robots to perform tasks such as collecting and analysing soil samples. Different robots have different functionalities, and some tasks may require multiple robots to be present. Tasks are assigned dynamically depending on the available robots. TemPl is based on this requirement of encoding tasks and inter-dependencies of each robot into a mission specification.

The mission can be seen as a task assignment for multi-robot systems, with the additional capability of atomic agents (single-robot systems) to combine and form composite agents.

2.6.1 Mission specification

The mission specification[25] of a reconfigurable multi-robot system defines the states of an agent in different stages of a mission.

A mission specification contains a spatial-temporal requirement, which states the functional and agent instance requirements at a certain space-timepoint. This specifies the time and location where an agent should be present.

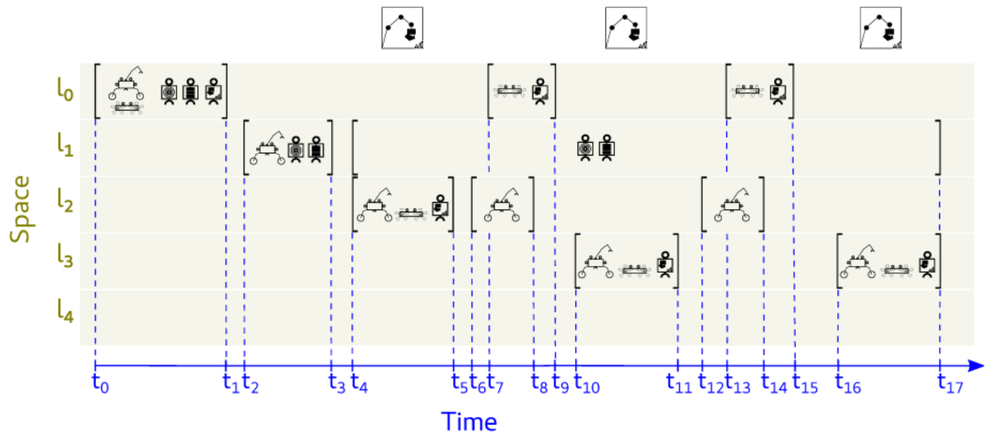

Figure 4: *Example of mission specification* [26]

Fig 4 shows a typical mission. Each agent needs to be in a specific space and time window. The required functionality is displayed on top of the timeline. For example, there are two mobile agents and three immobile agents between timepoints t0 and t1 and in location l0.

2.6.2 Mission solution

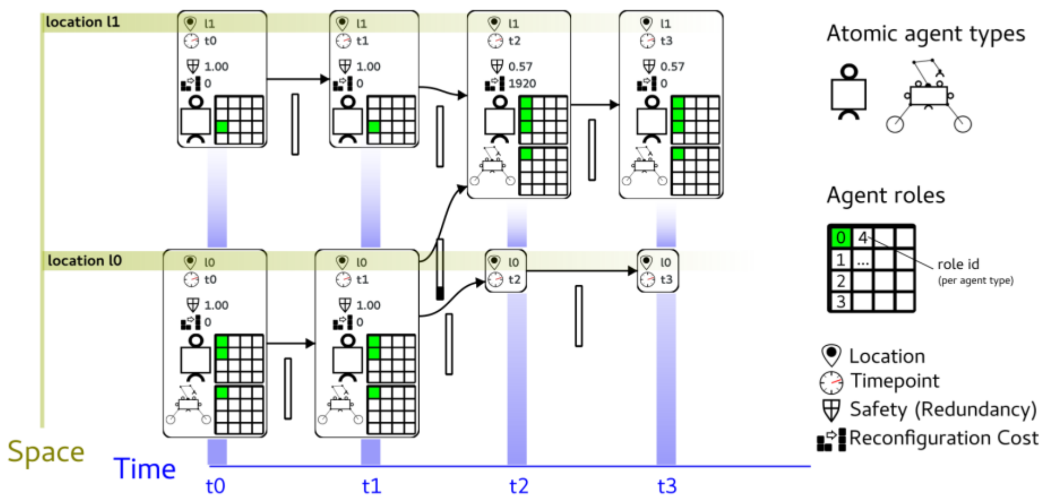

Figure 5: *Example solution (Roehr 2019, Pg.95)* [26]

Fig 5 is a visual representation of the mission solution generated by TemPl. A vertex corresponds to one spacetime tuple. The fillbars in the middle of tuples, represented by an upright rectangle, indicates the consumed capacity of the transitioning edge.

For example, the bottom-left tuple is in time t0 and location l0. It contains two atomic agent types. The first atomic agent type contains two green squares, which means there are two agents of that type. The second atomic agent type contains one green square, which means there is one agent of that type. The color green stands for the fulfilled requirement of each role agent. The color grey (not displayed in Fig 2.5) represents an agent role present without a requirement. The color red represents an unfulfilled requirement of a role agent.

2.6.3 Mission constraints in TemPl

Mission constraints in TemPl consist of:

(i) *Temporal*: Agent roles have lower and upper bound time intervals.

(ii) *Model*: Model constraints set the requirement of agent type and roles.

(iii) *Functionality and Property*: Functionality constraints state the number of required agents. To provide functionality, a minimum of one agent is always needed. Property constraints are for constraining the property of a functionality, e.g., an agent with a certain transport capacity may be demanded.

Further explanation on how each constraint is modelled can be found in Roehr (2019), pg. 96 [26].

2.6.4 Mission planning

The mission planner, TemPl, uses a planning algorithm to create a valid solution from a mission specification. This algorithm prioritizes creating a valid assignment before minimizing fleet size and total cost. The algorithm can be split into five steps:

(i) *Temporal ordering of timepoints*:

In this stage, the timepoints from the mission specification are temporally ordered to create valid timelines. Timepoints are ordered based on the relational operators $>$, $<$, $=$. For example, if timepoints t1 occurs before t2, the relation will be represented as t1 $<$ t2. If timepoints t1 and t2 occur at the same time, the relation will be represented as t1 $=$ t2.

The timepoints are ordered based on a "time expanded network" [27], a graph in which each of its vertex corresponds to a specific time and location.

(ii) *Bounding of agent type cardinality*:

The upper and lower bounding of agent type cardinality is done using a matrix representation of spatio-temporal requirements and agent types. In this stage, the least number of required set of agent roles is identified.

(iii) *Generating agent role timelines*:

In this stage, agent role assignments which fulfill all constraints are generated. The assignments should also form a path in temporally expanded network. A temporally expanded network is used to create a flow-based representation of the mission planning problem. The time expanded network, used to represent missions, allows edges only for vertices between neighbouring timepoints. For example, in Figure 5, an edge between two vertices is always from a lower timepoint t(N) to a higher timepoint t(N+1). Agent role variables assignments are sped up with the help of a path propagator [25] , which imposes a constrained path in the network.

(iv) *Flow optimization*:

In this stage, optimization is performed to the agent role assignments after the timeline has been generated. After roles are assigned, the agent role timelines are converted to a multi-commodity min-cost flow problem [3]. Mobile agents are transports, immobile agents are commodities and edges are connections in a route. Edges created due to a mobile agent transition have an upper bound capacity set by the transport capacity of that mobile agent. All available mobile agents (vehicles) span a flow network in which immobile agents (commodities) are transported to their target destination. Agents don't have single target destinations only, so requirements for a routes of each agents are only partially defined. So, immobile agent requirements are represented by minimum trans-flow requirements by the flow network.

The network flow problem is then solved. It is translated to a CPLEX LP standard representation so that it can be applied to LP solvers (e.g. GLPK [17]). Feasible and optimal solutions to the network flow problem is feasible but not necessarily an optimal solution to the mission assignment problem.

(v) *Quantification of timepoints*:

A qualified temporal network is converted into a quantitative simple temporal network. In this new network, a transition from one location to another is based on time taken by mobile systems to transition and form composite agents. ”min” and ”max” duration constraints are also applied.

3 Description of the Investigation

This section explains the state of the art in heuristic search strategies used to solve vehicle routing problems. A heuristic found to be promising in terms of performance results is picked to be implemented with the planner, TemPl.

3.1 Workflow

In this subsection, we will be describing the two different workflows used during testing different heuristics. In the first workflow, we apply heuristics to the mission solution. In the second workflow, we change constraints in the mission description itself before planning.

3.1.1 Modifying the mission solution

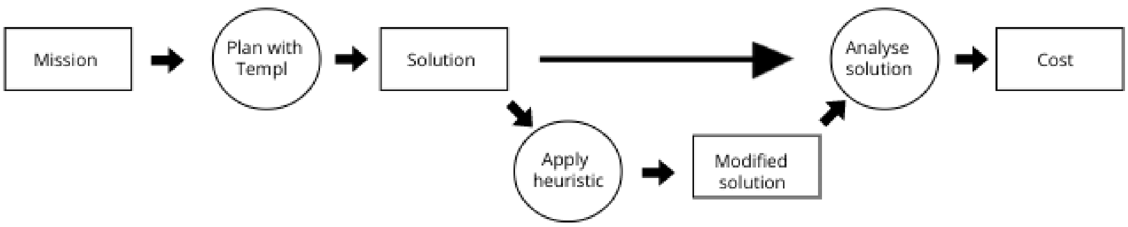

Figure 6: *Workflow for applying heuristics by changing the mission solution*

Figure 6 describes the workflow undertaken in measuring the performance of different heuristics. The workflow starts with a randomly generated mission. The randomly generated mission is fed into the TemPl planner. TemPl uses linear programming to create a solution from the mission. A certain heuristic (e.g. Very Large Neighbourhood Search) is applied to the solution. This modified solution is analysed using a cost function (explained in Section 4.1). The cost before and after applying the heuristic is used to observe if the solution improved or not.

3.1.2 Changing constraints in the mission description

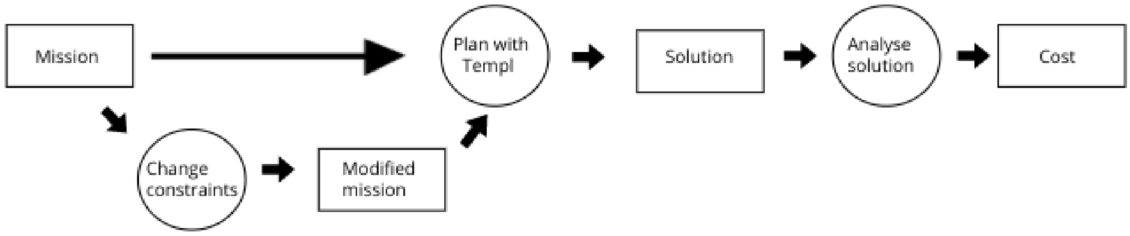

Figure 7: *Workflow for applying heuristics by changing the constraints in the mission description*

Figure 7 describes the workflow undertaken in measuring the performance of different heuristics by changing constraints in the mission description. The workflow starts with a randomly generated mission. The constraints in this mission are changed. This new mission is fed into the TemPl planner. The solution generated by TemPl is analysed using a cost function (explained in Section 4.1). The cost before and after applying the heuristic is used to observe if the solution improved or not.

3.2 State of the art research

An evaluation of the state of the art in heuristic search strategies was performed from which the most promising ones are listed below. The research presented here is mostly relevant for the workflow presented in Section 3.1.1 i.e. modifying the mission solution itself.

3.2.1 Constructive heuristics

Constructive heuristics are used to create a starting solution for other improvement heuristics to use. They aren't used anymore as most metaheuristics can now be initialized from an existing feasible or infeasible solution.

Afshar-Nadjafi (2019)[2] used a constructive heuristic to solve the time-dependent multi-depot vehicle routing problem. The heuristic minimized the total heterogeneous fleet cost by assuming travel time between two different locations depends on the departure time of the vehicle. It managed to outperform Lingo [16], a software tool for solving optimization models, on 180 test problems.

A classical example is the Clarke and Wright savings heuristic [6]. It creates multiple routes back and forth before gradually merging them to reduce the total cost.

Constructive heuristics have largely been replaced by more robust metaheuristics. Also, constructive heuristics creates a starting solution but since we will be dealing with an existing mission solution generated by TemPl, this isn't relevant to our problem.

3.2.2 Classical improvement heuristics

Classical improvement heuristics split and combine routes to achieve optimal results.

The best performing classical improvement heuristic was found [13] to be the Lin and Kernighan 2-opt algorithm. Thhis algorithm swaps edges to make the final route shorter. A modified version of the Lin and Kernighan algorithm by Helsgaun [14] managed a running time of $O(n^{2.2})$. Unfortunately, it was considered impractical for large problems containing more than 100,000 vertices.

Classical improvement heuristics are older, less robust and don't scale well with the problem size. Hence, this will be not used in the investigation.

3.2.3 Metaheuristics

Baugh (1998) [5] used metaheuristics to improve suboptimal results of dynamic programming for pickup and delivery problems. Toth and Vigo stated [31] most metaheuristics tend to be similar to each other and are hybrids of several metaheuristics. The more distinct ones are detailed below:

Local Search Algorithms Local search algorithms improve an initial solution by moving to the neighbourhood of another solution in each iteration. "Cycling" occurs in local search algorithms when the algorithm keeps iterating within the same neighbourhoods. Local Search Algorithms do not require the cost of the next solution to be lower than the cost of the current solution, which could lead to an endless loop.

Simulated Annealing In the classical version of simulated annealing by Osman (1993) [21], degraded solutions are occasionally accepted in the hopes of escaping the local optimum. The algorithm uses randomized search, acceptance and stopping criteria on a local search algorithm to prevent cycling. Osman observed new best solutions for upto 163 vehicles but a large variance still existed in terms of computation time and solution quality. This algorithm was also implemented on the classical version of VRP, which has no constraints.

Afifi, Dang, Moukrim (2013) [1] used simulated annealing for VRP with Time Windows and Synchronization Constraints (VRPTWSyn). In this variant of VRP, each customer has a time window for receiving items. If a vehicle arrives before the time window, it must wait. If a vehicle arrives later than the time window, it is rejected. Customers requiring two visits from two vehicles also exist. For cases like these, visits must be synchronized i.e. both vehicles must have the same starting time.

This was one of the few papers where simulated annealing was applied to VRPTWSyn. This algorithm was able to find all known optimal solutions in short computational times. The algorithm can be divided into three parts: construction heuristic, diversification process and local search procedure.

In the construction heuristic, a solution is built from scratch. In each iteration of the heuristic, visits with lower insertion costs are picked to be inserted in the associated route. More routes will only be added after it is impossible to insert any remaining visits to existing routes. The algorithm terminates after all visits are routed. As it's necessary to adhere to synchronization constraints, an update is sent to different routes after every insertion.

In the diversification process, a random number of visits are removed. The solution is built again using the construction heuristic. This “destroy and repair” operation helps obtain a new solution without losing much quality compared to the older solution. Each visit is recorded with a priority number of 0. For every insertion of a visit that creates an extra route, the priority number is increased by 1. This way, visits that cause lots of extra routes are identified as ”critical”. Critical visits are prioritized during insertion with the construction heuristic. The algorithm can always move back to a feasible solution when it reaches an infeasible one.

In the local search procedure, a random neighbourhood w is selected from a set W and initialized to 2-opt* or or-opt in every iteration. The picked neighbourhood is removed from W and applied to the current solution multiple times. If an improvement occurs with the random neighbourhood w, the other neighbourhood will be reinserted into W (if it was removed). This process recurs until there are no more neighbourhoods left in W.

The local search procedure uses two types of operations: 2-opt and or-opt.

2-opt* (exchange of path between routes): In this operator, two edges are swapped with two others in the same route and checked if it improved the solution. This version of 2-opt is different from the classical one as it wouldn’t have been possible to find an improvement due to the preset order of visits from time windows. In this version of 2-opt, a subset of visits is selected. For each pair of visits (v^i, v^j), the pairs v^i, v^{i+1} and v^j, v^{j+1} are considered (v^i denotes the visit v at position i; v^i and v^j are in two different routes). If an exchange is feasible, the new cost is computed and recorded. After multiple random selections, the lowest cost is selected and the corresponding visits are swapped.

Or-opt (exchange of visits in the same route): In this operator, consecutive visits are moved to another location in the same route. A random selection is followed by a feasibility check, which is partially similar to 2-opt*.

Deterministic Annealing As compared to simulated annealing where a solution is picked randomly, deterministic annealing uses deterministic methods to pick a solution. In the search process, some directions have a higher chance of finding an optimal solution. The stochastic search process used by the algorithm is guided towards these directions to save time and computational cost.

Baranwal et al. (2016) [4] implemented, to our knowledge, the first ever deterministic-annealing based approach to VRP with Time Windows (VRPTW). This paper gave a high-level explanation of its algorithm.

Tabu Search Tabu search starts with a solution and moves to the best non-tabu solution in every iteration. Several variants of tabu search exist. Potvin et.al. (1996) [23] implemented the tabu search heuristic based on 2-opt* and or-opt exchanges. Rochat and Taillard (1995) [24] used a tabu search heuristic which exploits a neighbourhood by exchanging customers between routes. An ”adaptive memory” records the best routes found from each search. This memory is used to create new starting solutions for tabu search.

Edge-exchange heuristics: This is similar to 2-opt* and or-opt exchanges. A staring solution is picked. All solutions in its neighbourhood are generated using one of several heuristics. The best solution is picked and defined to be the starting solution. This repeats for a number of times. Increasing the graph size increases the total number of iterations taken to reach a local optimum.

Gendrau et. al.(1997) [30] uses a modified version of this edge-exchange heuristic to find neighbourhoods for its solution. Moves are evaluated by calculating the difference in cost (measured by distance travelled) between the neighbouring and current solution. The size of the neighbourhood is decreased by getting rid of moves that do not improve the solution. Initially, this tabu search heuristic creates a solution from the routes found in the adaptive memory. The solution is broken down into subset of routes. Tabu search is applied on each subset. The new routes formed by tabu search are merged to create a new solution. The new routes are stored in the adaptive memory, and the entire process is repeated.

The algorithm has the following components:

Initialization: A randomized insertion heuristic combines routes from the adaptive memory. The set of routes is broken down into subsets, which will be fed to the tabu serch heuristic.

Stopping criterion: The tabu search stops after a number of iterations, depending on the decomposition/reconstruction cycle. Nodes are removed and reinserted into the solution in this cycle. The entire solution improves incrementally after every cycle, so a higher number of iterations creates a better solution. Gendrau et. al. [30] uses the following formula to calculate the stopping criterion. $A \times (1 + \frac{DR-1}{B})$

where, A and B are parameters and DR is the current iteration of the decomposition and reconstruction cycle.

Tabu list: The tabu list is an array where each element is associated with a solution. A solution is assigned into the list by this formula:(cost of the solution MODULO length of tabu list). An element of the list stores the number of iteration at which the solution will lose its "tabu" status. The (cost of the solution) modulo (length of tabu list) is calculated after every solution generation. The number of iterations of this new solution is compared with the one in tabu list. If the value is greater, then the current move is accepted, else, it is considered as "tabu".

Diversification: Diversification occurs by preventing exchanges that are frequently performed. An equation is used to penalize exchanges:

$p_e = x\times\ \Delta max.iter\times\ \frac{fr_e}{fr_{max}}$

where, x is a random value between 0 and 0.5. $\Delta max.iter$ is the maximum difference in cost between two consecutive solutions upto the iteration $iter$. fr_e is the frequency of an exchange e, fr_{max} is the maximal frequency obtained. fr_{max} normalizes frequencies as it is lower for larger neighbourhoods.

Reordering of routes: Each vehicle needs to deliver items to customers in a route. When a best solution is found, the customers within each individual route may be reordered. Gendrau et. al. uses Solomon's I1 insertion heuristic [29]. This heuristic works on a given route the following way. The customers in a single route are unrouted. The customer that is farthest from the depot is selected as a "seed customer" i.e a vehicle starts at the depot, delivers to the seed customer, and then goes back to the depot. The customers that were unrouted are then reinserted in succession into the new route. The customer which maximizes a generalized savings measure is picked first to be reinserted. The picked customer is reinserted in a location which minimizes a weighted sum of detour in space and time. Solomon's I1 insertion heuristic is applied N times, and the best solution is picked. If the new route generated by the heuristic is worse in terms of cost or if the new route doesn't cover all the customers that were unrouted, then the original route is restored.

3.3 Selecting a heuristic

From our evaluation of the state of the art in heuristic search strategies, Cotlin and Veloso's [7] VLNS-T algorithm was found to be more promising. Their VLNS algorithm improved state of the art solutions for pickup and delivery problems (PDP) by a significant margin with the use of transfers. The VRP problem in which the VLNS-T algorithm is applied to is similar to the mission description provided by TemPl. VLNS-T uses the concept of "transfers" where a route of vehicle is split and shared with other vehicles. This is similar to a "reconfiguration" in reconfigurable multi-robot systems as two or more atmoic agents (robots) may combine to complete a task. Thus, we've selected this VLNS algorithm to be applied to the planner, TemPl.

3.3.1 Very Large Neighbourhood Search

In Very Large Neighbourhood Search (VLNS) algorithms, a solution is destroyed and repaired multiple times in the hope of finding a better solution. Searching a large neighbourhood leads to a local optima of high quality. This could be time consuming so VLNS algorithms use filtering techniques to limit the neighbourhood searched.

Cotlin, Veloso (2014) [7] introduced the Very Large Neighbourhood Search with Transfers (VLNS-T) algorithm which improved the previously known benchmark of pickup and delivery problem solutions. This is similar to the Adaptive VLNS algorithm for PDP without transfers by Ropke and Pisinger (2006) [28]. It can also be considered as a variant of the Simulated Annealing algorithm from Section 3.1.3. Items are removed and reinserted with different heuristics to find "neighbors".

This algorithm introduces transfers to reduce cost of solutions. Masson (2013) et al. [18] showed the effectiveness of introducing transfers to improve solutions. Through transfers, routes of vehicles are split and shared with other vehicles. Transfers with different metrics minimizes delivery time of vehicles.

Algorithm 1: vlns(S): Use VLNS to form a new schedule from previous schedule S

1 $S_{best} \leftarrow S$
2 $T \leftarrow \frac{w}{-ln0.5} cost(S, \gamma = 0)$
3 **for** $i \in 1, ..., N$ **do**
4     $q \leftarrow$ random integer in $[min(4, |P|), min(100, \xi|P|)]$
5     $R \leftarrow$ random-items(S, q)
6     $S' \leftarrow$ remove-items(S, R)
7     $S' \leftarrow$ insert-items(S', UND(S))
8   **if** $cost(S') < cost(S_{best})$ **then**
9         $S_{best} \leftarrow S'$
10   **end**
11   **if** $random() < exp - (cost(S') - cost(S))/T$ **then**
12         $S \leftarrow S'$
13   **end**
14     $T \leftarrow cT$
15 **end**
16 **return** S_{best}

The explanation of the algorithm is as follows.

- (Line 5): A random number of items are picked from the existing schedule.
- (Line 6): The picked items are removed from the schedule using one of several removal heuristics.
- (Line 7): Items that are not part of the schedule are reinserted using one of several insertion heuristics.
- (Line 8, 9): If the cost of the new solution is lesser than that of the best known solution, the new solution becomes the best solution.
- (Line 11, 12): The likelihood of the new schedule being accepted decreases with time.

There are two relevant heuristics used in this algorithm to modify the existing schedule: Removal (remove-items) and Insertion (insert-items). The removal heuristic takes out items from the solution. The insertion heuristic puts items back into the solution. Each heuristic uses different techniques to achieve this, which gives different results. An explanation of each type of heuristic is given below.

Removal heuristics : The remove-items function uses three different heuristics to remove picked items from the existing schedule. The *Shaw* removal heuristic picks items that are similar in distance, time and demand. The *Random* removal heuristic picks selects *q* items randomly to remove. The *Worst* removal heuristic picks items by calculating the cost before and after its removal from the solution. Items that are the most "expensive" have a higher probability of being removed from the solution.

Insertion heuristics : The insert-items function uses two different heuristics to insert items into the schedule. Both use 'transfers' so that routes can be split to achieve a better cost. The *Split Routes* insertion heuristic takes a vehicle's existing route and splits it with another vehicle. The *Insert item with transfer* heuristic searches the solution for a possible insertion but with max one transfer.

3.3.2 Designing a VLNS algorithm for TemPl

Cotlin and Veloso's [7] VLNS algorithm needs to be adapted to work with TemPl. Even though the mission description used by TemPl is similar to the VRP problem, TemPl consists of additional constraints that need to be considered while applying heuristics. For example, TemPl deals with inter-route constraints, so there are limits to the removal and insertion heuristics that can be applied to the solution.

For this VLNS algorithm, we've employed one removal heuristic and one insertion heuristic, both based on randomness.

Algorithm 2: vlns(S): Use VLNS to form a new solution from previous solution S

1 $S_{best} \leftarrow S$
2 $T \leftarrow \frac{w}{-ln0.5} cost(S, \gamma = 0)$
3 **for** $i \in 1, ..., N$ **do**
4     $q \leftarrow$ random agent from list of agents $[1......roles]$
5     $S' \leftarrow$ remove-items(S, q)
6     $S' \leftarrow$ insert-items(S, q)
7   **if** $cost(S') < cost(S_{best})$ **then**
8         $S_{best} \leftarrow S'$
9   **end**
10   **if** $random() < exp - (cost(S') - cost(S))/T$ **then**
11         $S \leftarrow S'$
12   **end**
13     $T \leftarrow cT$
14 **end**
15 **return** S_{best}

The explanation of the algorithm is as follows.

- (Line 3): The algorithm iterates N number of times. Increasing N can help find better solutions but will be computationally more expensive.
- (Line 4): One random agent in a space-timepoint is picked from the list of agents in a solution.
- (Line 5 and 6): The picked agent is removed from the solution and reinserted in a different location.
- (Line 7, 8): If the cost of the new solution is lesser than that of the best known solution, the new solution becomes the best solution.
- (Line 11, 12): The likelihood of the new schedule being accepted decreases with time.

Similar to the VLNS algorithm used by Cotlin, Veloso, we've employed Removal and Insertion heuristics. The difference is, a random agent in a specific spacetime point is picked and moved into another location of the same timepoint. It is not possible to move an agent across timepoints, only locations.

An explanation of the removal and insertion heuristics used is given below:

Removal heuristics : The remove-item function is a random removal heuristic to remove agents from a solution. An agent is picked in random from a list of available agents. An agent can exist in multiple space-timepoint tuples (Section 2.6) but it is only removed from one specific space-timepoint tuple. This is because each agent contains multiple dependencies in the mission solution and requiring every single one would make the problem complicated.

Insertion heuristics : The insert-item function reinserts the removed agent back into another tuple of the same timepoint but different location. Agents aren't allowed to exist in two different timepoints in the same location. It is not possible for a robot to be in two places at the same time.

After an agent is removed and reinserted, its cost is compared to that of the current solution. The new solution is accepted if the cost decreased.

4 Evaluation of the Investigation

4.1 Evaluation criteria

Mission solutions are evaluated based on a cost function to analyse the effectiveness of heuristics. Traditional VRP problems use "total distance travelled by all vehicles" as a cost function. Dealing with reconfigurable multi-robot systems is more complicated so additional variables need to be considered.

As stated in Section 2.1, Roehr (2019)[26] uses three system properties to analyze the flexibility of reconfigurable multi-robot systems: efficacy, efficiency and safety. Our cost function to analyse solutions is based on these three properties.

Efficiency: In VRP, efficiency is the highest when travelled distance is lowest. Reconfigurable mutli-robot systems are heterogenous. Each robot consumes energy differently. Moving robots with higher energy consumption will decrease efficiency as compared to moving robots with lower energy consumption. Therefore, efficiency can't be measured solely by the distance travelled. The total energy consumed by all robotic systems in a solution also needs to be considered.

Reconfigurable multi-robot systems also share energy when reconfigured into one composite agent. This reconfiguration affects the efficiency of the system.

Safety: A higher redundancy of resources in the same space-timepoint increases safety. If an agent fails, it can be replaced by another agent of the same type.

Efficacy: Efficacy is described as the ability of a reconfigurable multi-robot system to provide a particular functionality. Higher the number of fulfilled requirements of a mission, higher the efficacy of the solution.

The cost function we use combines the three system properties:

$$cost = \alpha \times efficiency + \beta \times efficacy + \epsilon \times safety \tag{1}$$

α, β and ϵ are used to balance the parameters: efficiency, efficacy and safety. Efficacy and safety is a value between [0,1] so α accounts for normalisation to [0,1] for the energy cost. Therefore, α is calculated with the use of E_{max}^{-1}, where E_{max} is the maximum energy cost allowed or observed in an existing mission solution [25].

4.2 Data setup

In order to evaluate the performance of the Very Large neighbourhood Search with TemPl, we've used missions of different sizes and complexity.

TemPl was intially used to search the solutions. Then, a the VLNS heuristics was applied. The application of heuristic is analyzed.

The tested mission consists of 14 timepoints and 10 locations. The coordinates of each location is given below. The distance between two coordinates is measured in metres.

We've used a value of 1 for α, β and ϵ in the cost function. We aren't concerned about balancing the parameters during our experiments as we want to observe the change in cost with respect to each system property. Varying α, β and ϵ could lead to invalid conclusions.

The coordinate of each location in our mission is as follows.

Location	**Latitude**	**Longitude**
lander	-83.82009	87.53932
base1	-84.1812	87.60494
base2	-83.58491	85.98319
b1	-84.1812	87.60494
b2	-83.96893	86.75471
b3	-83.66856	87.42557
b4	-83.54570	87.09851
b5	-83.82009	84.66000
b6	-83.77371	84.70960
b7	-83.34083	84.64467

The timepoints use relational values, unlike the coordinate of locations. The timepoints are ordered in the following way,

```
t1 < t2 < t3 < t4 < t5 < t6 < t7 < t8 < t9 < t10 < t11 < t12 < t13 < t14
```

The operator " $<$ " shows the temporal relationship between two different timepoints. When $t1 < t2$, $t1$ occurs before $t2$.

4.3 Results

The Very Large Neighbourhood Search (VLNS) algorithm from section 4.3 was implemented to mission solutions of different sizes for an iteration of N = 100. The cost was calculated three times, averaged and recorded in the table below.

In the first set of experiment, the VLNS algorithm with random insertion and removal heuristic was implemented. As described in Algorithm 2, the search strategy optimises for cost of the solution. This led to the efficacy of solutions to continuously decrease. This isn't optimal. An efficacy value of 0 will lead to high efficiency but could leads to missions with lots of unfulfilled requirements.

It is important to balance the three system properties of the cost function. To prevent efficacy from falling to 0, a stopping criterion was added to not accept solutions with efficacy lower than 0.33. The results of the experiment are displayed below.

Experiment no.	No. of mobile agents	Cost with TemPI	Cost after applying VLNS
1	5	2478.93	2474.04
2	7	7580.76	7575.91
3	5	4806.33	4804.27
4	7	6681.61	6676.97
5	5	3501.63	3502.63
6	6	6604.69	6603.89
7	7	8312.93	8306.86
8	7	8629.58	8626.47
9	5	5092.88	5088.76
10	7	6682.77	6676.83

Table 1: Analysis of VLNS with solutions of minimum efficacy ≥ 0.33

Optimising for cost lead to solutions with low efficacy. After each iteration, the efficacy of the solution would decrease, which meant requirements weren't being fulfilled. We implemented the VLNS again with a new condition in the heuristic. This time, solutions which lowered the efficacy wouldn't be accepted. This way, we hoped to maintain an efficacy of 1.0 even after the random removal/insertion.

Experiment no.	No. of mobile agents	Cost with TemPI	Cost after applying VLNS
1	5	2478.93	2477.41
2	7	7580.76	7578.17
3	5	4806.33	4807.09
4	7	6681.61	6678
5	5	3501.63	3503.73
6	6	6604.69	6607.9
7	7	8312.93	8314.09
8	7	8629.58	8628.79
9	5	5092.88	5092.88
10	7	6682.77	6674.1

Table 2: Analysis of VLNS with solutions of minimum efficacy ≥ 1.0

From our results, we can observe that applying a random insertion and removal insertion heuristic succeeded in lowering the solution cost. Decreasing efficacy seems to be the best way to reduce total cost of the entire solution. Solutions generated with minimum efficacy of 0.33 performed better than solutions with minimum efficacy of 0.1.

We intentionally chose the values, 0.33 and 1.0 of minimum efficacy accepted for solutions. Lowering the efficacy even more for better solution could be possible, but, would harm the completeness of solutions.

4.4 Summary

This section summarizes the experiment from the previous section.

The Very Large Neighbourhood Search algorithm from Section 4.3 was adapted to robot planning with TemPl. To test its effectiveness, we applied the algorithm to a solution generated by TemPl. The cost before and after the application of the algorithm was compared. A positive improvement was noticed in the solutions in terms of total cost.

The most promising results came when the efficacy was the lowest. As efficacy is the percentage of fulfilled requirements, removing every single agent from a solution would lead to a minimum cost. But, this isn't feasible in real life scenarios as we would want our mission requirements to be at least partially fulfilled. Therefore, we performed two different experiments with a minimum efficacy of 0.33 and 1.0 for accepted solutions.

We observed from our experiments that lowering the minimum efficacy created solutions of lower costs. It now depends on the person applying heuristics to decide the proportion of unfilled requirements they are willing to tolerate when applying this heuristic. Balancing the three system properties of efficiency, efficacy and safety.

4.5 Discussion of results

This section discusses the results we obtained from our evaluation.

5 Conclusion

This thesis intends to evaluate heuristic search strategies that can be applied to planning with reconfigurable multi-robot systems. We evaluated existing state-of-the-art heuristics that has been applied to solve Vehicle Routing Problems (VRPs), picked a promising heuristic and implemented it with TemPl. We picked the metaheuristic, ”Very Large Neighbourhood Search (VLNS)” [7], and adapted it to deal with planning with reconfigurable multi-robot systems.

The VLNS algorithm was applied to a mission solution and its results were evaluated. A mission consisting of 14 timepoints and 10 locations was used to generate 10 solutions with TemPl. Each solution generated contained different number of mobile agents and cost. VLNS with random insertion and removal heuristic was applied to the solutions. We observed a decrease in the cost of the solutions after applying VLNS, but the efficacy decreased after every iteration. So, we set a minimum efficacy of accepted solutions at 0.33 and 1.0. The VLNS algorithm was applied again. We observed a decrease in cost, though, setting the minimum accepted efficacy at 0.33 produced solutions with lower cost than at 1.0.

The satisfactory results observed by this thesis displayed the applicability of the VLNS metaheuristic and heuristic search strategies in general to planning with reconfigurable multi-robot systems.

6 Future Work and Possible Improvements

There are two further research areas that can be explored to continue the work introduced by this thesis.

Metaheuristics are the most robust and well-performing heuristic search strategies. The other metaheuristics could be implemented and its performance compared with the VLNS algorithm used in this thesis. Heuristics that had promising results in solving VRP wasthe Tabu heuristic [30] and Iterated Local Search [20].

Increasing the constraints, e.g. load and reconfiguration, can also be interesting to observe. Planning with reconfigurable multi-robot systems consists of temporal, model, functionality and property constraints. We didn't consider all of these constraints while applying VLNS. Further changes to the metaheuristic in this thesis is required to adapt for all constraints. Different results, as compared to solutions without constraints, can also be expected.

References

[1] Sohaib Afifi, Duc-Cuong Dang, and Aziz Moukrim. "A Simulated Annealing Algorithm for the Vehicle Routing Problem with Time Windows and Synchronization Constraints". In: *7th International Conference, Learning and Intelligent Optimization (LION 7)*. Vol. 7997. Catania, Italy, Jan. 2013, pp. 259–265. DOI: 10.1007/978-3-642-44973-4_27. URL: https://hal.archives-ouvertes.fr/hal-00916972.

[2] Behrouz Afshar-Nadjafi and Alireza Afshar-Nadjafi. "A constructive heuristic for time-dependent multi-depot vehicle routing problem with time-windows and heterogeneous fleet". In: *Journal of King Saud University - Engineering Sciences* 29 (Jan. 2017), pp. 29–34. DOI: 10.1016/j.jksues.2014.04.007.

[3] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. "Network flows". In: (1988).

[4] Mayank Baranwal et al. "Vehicle Routing Problem with Time Windows: A Deterministic Annealing approach". In: *2016 American Control Conference (ACC)* (2016), pp. 790–795.

[5] John W Baugh Jr, Gopala Krishna Reddy Kakivaya, and John R Stone. "Intractability of the dial-a-ride problem and a multiobjective solution using simulated annealing". In: *Engineering Optimization* 30.2 (1998), pp. 91–123.

[6] G. Clarke and J. W. Wright. "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points". In: *Oper. Res.* 12.4 (Aug. 1964), pp. 568–581. ISSN: 0030-364X. DOI: 10.1287/opre.12.4.568. URL: https://doi.org/10.1287/opre.12.4.568.

[7] Brian Coltin and M. Veloso. "Scheduling for transfers in pickup and delivery problems with very large neighborhood search". In: *Proceedings of the National Conference on Artificial Intelligence* 3 (Jan. 2014), pp. 2250–2256.

[8] Jean-François Cordeau et al. "VRP with Time Windows". In: Jan. 2002, pp. 157–193. DOI: 10.1137/1.9780898718515.ch7.

[9] G. B. Dantzig and J. H. Ramser. "The Truck Dispatching Problem". In: *Management Science* 6.1 (1959), pp. 80–91. URL: https://EconPapers.repec.org/RePEc:inm:ormnsc:v:6:y:1959:i:1:p:80-91.

[10] Guy Desaulniers et al. "VRP with Pickup and Delivery". In: Jan. 2002, pp. 225–242. DOI: 10.1137/1.9780898718515.ch9.

[11] Michael Drexl. "Applications of the vehicle routing problem with trailers and transshipments". In: *European Journal of Operational Research* 227 (June 2013), pp. 275–283. DOI: 10.1016/j.ejor.2012.12.015.

[12] Michael Drexl. "On Some Generalized Routing Problems". In: (Jan. 2007).

[13] G. Gutin and A. P. Punnen. "Experimental analysis of heuristics for the STSP, in The Traveling Salesman Problem and Its Variations". In: *Kluwer, Dordrecht, 2002* (2002), pp. 369–443. DOI: http://dimacs.rutgers.edu/archive/Challenges/TSP/papers/stspchap.pdf.

[14] Keld Helsgaun. "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristic". In: *European Journal of Operational Research* 126 (Oct. 2000), pp. 106–130. DOI: 10.1016/S0377-2217(99)00284-2.

[15] Abdullahi Ibrahim, Rabiat Abdulaziz, and Jeremiah Ishaya. “CAPACITATED VEHICLE ROUTING PROBLEM”. In: *International Journal of Research - GRANTHAALAYAH* 7 (Apr. 2019), pp. 310–327. DOI: 10.5281/zenodo.2636820.

[16] Anne Hummel Kris Thornburg. *Lingo 8.0 tutorial*. URL: http://www.columbia.edu/~cs2035/courses/ieor3608.F06/lingo-tutorial.pdf.

[17] A. Makhorin. *GLPK (GNU Linear Programming Kit)*. URL: https://www.gnu.org/software/glpk/glpk.html.

[18] Renaud Masson, Fabien Lehuédé, and Olivier Péton. “An Adaptive Large Neighborhood Search for the Pickup and Delivery Problem with Transfers”. In: *Transportation Science* 47 (Aug. 2013), pp. 344–355. DOI: 10.1287/trsc.1120.0432.

[19] Paul Minett and John Pearce. “Estimating the Energy Consumption Impact of Casual Carpooling”. In: *Energies* 4 (Dec. 2011). DOI: 10.3390/en4010126.

[20] Vinicius Morais, Geraldo Mateus, and Thiago Noronha. “Iterated local search heuristics for the Vehicle Routing Problem with Cross-Docking”. In: *Expert Systems with Applications* 41 (Nov. 2014), pp. 7495–7506. DOI: 10.1016/j.eswa.2014.06.010.

[21] Ibrahim Osman. “Meta-strategy simulated annealing and Tabu search algorithms for the vehicle routine problem”. In: *Annals of Operations Research* 41 (Dec. 1993), pp. 421–451. DOI: 10.1007/BF02023004.

[22] pngguru.com. URL: https://www.pngguru.com/search?png=vehicle+Routing+Problem.

[23] Potvin et al. “The vehicle routing problem with time windows”. In: *Part I: Tabu search. INFORMS Journal on Computing 8* (1996), pp. 158–164.

[24] Yves Rochat and Éric Taillard. “Taillard, E.D.: Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. Journal of Heuristics 1(1), 147-167”. In: *Journal of Heuristics* 1 (Sept. 1995), pp. 147–167. DOI: 10.1007/BF02430370.

[25] Thomas Roehr. “A Constraint-based Mission Planning Approach for Reconfigurable Multi-Robot Systems”. In: *Inteligencia Artificial* 21 (Sept. 2018), p. 25. DOI: 10.4114/intartif.vol21iss62pp25-39.

[26] Thomas Roehr. “Autonomous Operation of a Reconfigurable Multi-Robot System for Planetary Space Missions”. PhD thesis. Nov. 2019. DOI: 10.13140/RG.2.2.30628.83844.

[27] Thomas Roehr and Frank Kirchner. “Spatio-Temporal Planning for a Reconfigurable Multi-Robot System”. In: June 2016.

[28] Stefan Ropke and David Pisinger. “An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows”. In: *Transportation Science* 40 (Nov. 2006), pp. 455–472. DOI: 10.1287/trsc.1050.0135.

[29] M. Solomon. “Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints”. In: *Oper. Res.* 35 (1987), pp. 254–265.

[30] Éric Taillard et al. “A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows”. In: *Transportation Science* 31 (May 1997), pp. 170–186. DOI: 10.1287/trsc.31.2.170.

[31] Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Jan. 2002. DOI: 10.1137/1.9780898718515.