Aalto University
School of Science
Master's Programme in Mathematics and Operations Research

Petri Määttä

# Comparison of solution methods for the dynamic pickup and delivery problem with time windows

**School of Science**

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo, September 23, 2021

Supervisor: Professor Antti Punkka
Advisor: Heli Lampinen M.Sc. (Tech.)

Aalto University
School of Science
Master's Programme in Mathematics and Operations Research

ABSTRACT OF MASTER'S THESIS

**Author:** Petri Määttä

**Title:**
Comparison of solution methods for the dynamic pickup and delivery problem with time windows

| | | | |
|---|---|---|---|
| **Date:** | September 23, 2021 | **Pages:** | vii + 51 |
| **Major:** | Systems and Operations Research | **Code:** | SCI3055 |
| **Supervisor:** | Professor Antti Punkka | | |
| **Advisor:** | Heli Lampinen M.Sc. (Tech.) | | |

The Vehicle Routing Problem (VRP) is classical problem of combinatorial optimization, in which a fleet of vehicles and a set of locations are given, and the problem is to route the vehicles to visit the locations at minimum cost. Its dynamic counterpart (DVRP) and its variants have received a considerable amount of research during the past decades, owing to the advancement of information and communication technologies which have aided the process of vehicle routing. In a dynamic VRP, the information related to solving the problem is revealed over time, in contrast to the static VRP, where all information needed to solve the problem fully is known beforehand.

In this work a variant of the DVRP called the Dynamic Pickup and Delivery Problem with Time Windows (DPDPTW) is studied, and in particular, how different solution methods perform in solving instances of it. These methods consist of five different metaheuristics and four different first solution strategies. Metaheuristics are the most used solution methods in the literature of VRPs, and they can be described as high-level strategies that guide lower level heuristics to obtain better solutions. First solution strategies are simple methods which provide feasible initial solutions to the VRPs for the metaheuristics to start their search from.

The main results obtained were that the metaheuristic Guided Local Search (GLS) outperforms other methods in most scenarios, and that the choice of first solution strategy has a large impact on the final solution quality but depends on the problem parameters. Path Most Constrained Arc (PMCA) is the most reliable first solution strategy out of the ones compared. While the applicability of the results is somewhat limited, they are generally supported by the literature and are useful within their context.

| | |
|---|---|
| **Keywords:** | Vehicle routing problem, metaheuristics, first solution strategies |
| **Language:** | English |

Aalto-yliopisto
Perustieteiden korkeakoulu
Matematiikan ja operaatiotutkimuksen maisteriohjelma

DIPLOMITYÖN TIIVISTELMÄ

| | | | |
|---|---|---|---|
| **Tekijä:** | Petri Määttä | | |
| **Työn nimi:** | Ratkaisumenetelmien vertailu dynaamisessa ajoreititysongelmassa | | |
| **Päiväys:** | 23. syyskuuta 2021 | **Sivumäärä:** | vii + 51 |
| **Pääaine:** | Systeemitiede ja operaatiotutkimus | **Koodi:** | SCI3055 |
| **Valvojat:** | Professori Antti Punkka | | |
| **Ohjaaja:** | Diplomi-insinööri Heli Lampinen | | |

Ajoneuvon reititysongelma (Vehicle Routing Problem, VRP) on klassinen kombinatorisen optimoinnin ongelma, jossa joukko ajoneuvoja ja joukko sijainteja on annettu, ja ajoneuvoille tulee määrittää reitit siten että jokaisessa pisteessä käydään vain kerran ja että reittien kokonaiskustannukset minimoidaan. Vastaavaa dynaamista ongelmaa (DVRP) on tutkittu kasvavissa määrin viime vuosikymmenten aikana muun muassa kehittyvän teknologian ansiosta. Dynaamisessa ongelmassa ratkaisemiseen liittyvää informaatiota saadaan ajan myötä, toisin kuin staattisessa ongelmassa, jossa kaikki informaatio on saatavilla etukäteen.

Tässä työssä tutkitaan dynaamista lähettiongelmaa aikaikkunoilla, joka on eräs ajoreititysongelman variantti. Erityisesti tavoitteena on selvittää miten eri ratkaisumenetelmät — jotka koostuvat viidestä metaheuristiikasta ja neljästä konstruktiivisesta heuristiikasta — sopivat ongelman ratkaisemiseen. Metaheuristiikat ovat yleisin ratkaisumenetelmätyyppi ajoreititysongelmissa, ja niitä voidaan luonnehtia korkean tason strategoiksi, jotka ohjaavat matalamman tason heuristiikoita saavuttaakseen paremman laatuisia ratkaisuja. Konstruktiiviset heuristiikat ovat yksinkertaisia menetelmiä, jotka etsivät käypiä lähtöratkaisuja metaheuristiikoille.

Keskeisimmät johtopäätökset ovat, että metaheuristiikka Guided Local Search (GLS) suoriutuu muita paremmin useimmissa olosuhteissa, ja että konstruktiivisen heuristiikan valinta vaikuttaa vahvasti lopullisen ratkaisun laatuun. Path Most Constrained Arc (PMCA) on valituista konstruktiivisista heuristiikoista luotettavin. Olosuhteet, joissa näitä tuloksia voi sellaisenaan soveltaa, ovat jokseenkin rajoittuneet, mutta näissä olosuhteissa ne ovat hyödyllisiä.

| | |
|---|---|
| **Asiasanat:** | Ajoreititysongelma, metaheuristiikka, konstruktiivinen heuristiikka |
| **Kieli:** | Englanti |

# Acknowledgements

I wish to thank my advisor Heli Lampinen for her regular guidance, support and hospitality as well as for challenging me and offering new points of view. I thank the people at Länsiö Logistiikka Oy for giving me the opportunity to work on this complex and engaging problem – especially Juha Länsiö and Jari Aronen. Thanks also go to Teemu Lätti for his valuable guidance on the software side of things. Finally, I thank my supervisor, Professor Antti Punkka, for his precise and knowledgeable critique and advice on the scientific and written parts of the work.

Espoo, September 23, 2021

Petri Määttä

# Abbreviations and Acronyms

| | |
|---|---|
| VRP | Vehicle Routing Problem |
| CVRP | Capacitated Vehicle Routing Problem |
| DVRP | Dynamic Vehicle Routing Problem |
| VRPTW | Vehicle Routing Problem with Time Windows |
| PDP | Pickup and Delivery Problem |
| DPDPTW | Dynamic Pickup and Delivery Problem with Time Windows |
| TS | Tabu Search |
| GTS | Generic Tabu Search |
| GLS | Guided Local Search |
| GD | Greedy Descent |
| SA | Simulated Annealing |
| LCI | Local Cheapest Insertion |
| PCA | Path Cheapest Arc |
| PCI | Parallel Cheapest Insertion |
| PMCA | Path Most Constrained Arc |
| MS | Minimize sum (minimize sum of route lengths) |
| ML | Minimize longest (minimize length of longest route) |

# Contents

# Chapter 1

# Introduction

Research on the dynamic vehicle routing problem (DVRP) has seen a great increase over the last few decades (Psaraftis et al., 2016). During this time, the advancement of information and communication technologies has facilitated the use of more accurate data to aid the process of vehicle routing (Ritzinger et al., 2016). The vehicle routing problem (VRP) and its variants are extremely relevant in day-to-day logistics and business. Optimizing vehicle routing via mathematical optimization can give significant savings to a company by enabling more efficient use of the vehicle fleet and saving work hours by automating the routing process (Toth and Vigo, 2014). Efficiency in the use of vehicles naturally also has environmental benefits.

The vehicle routing problem, introduced originally by Dantzig and Ramser (1959), is a classical NP-hard problem of combinatorial optimization and integer programming (Toth and Vigo, 2002). It can be summarized as the following: given a fleet of vehicles and a set of locations to visit, what are the optimal routes for the vehicles to visit the locations at minimum cost? The VRP has both a static and a dynamic version. If all the information needed to route the vehicles is known beforehand and does not change, the problem is *static* (also *offline*). If on the other hand not all information related to the problem instance is known in advance, the problem is *dynamic* (also *online* or *real-time*) (Ritzinger et al., 2016). In a DVRP, some information is received during the execution of the routing plan, that is, while the vehicles are carrying out their routes. Possible dynamic events are for example the arrival of new transportation requests, and unknown service times, travel times or customer demands.

Another classification of VRPs is the stochastic-deterministic division (Toth and Vigo, 2014). In a stochastic setting, some information about the uncer-

tain elements of the problem is known in the form of probability distributions, while in a deterministic setting, these distributions are not available, meaning that nothing is known about future events until they present themselves.

The VRP has many variants that are based on often complex real-world constraints (Toth and Vigo, 2014). These include for instance the capacitated VRP (CVRP), where each vehicle has a carrying capacity and customer visits expend this capacity; VRP with time windows (VRPTW), where each location needs to be visited within a given time slot; pickup and delivery problem (PDP), where each transportation request has a separate pickup location and a delivery location. Of course many more variants exist, and in a real world setting, the actual problem instance is almost always a combination of multiple variants.

A large number of solution methods to the VRP and its variants have been developed over the years. Due to its combinatorial nature, exact solution methods are generally of limited use, except for simple variants and small problem instances (Bai et al., 2021). These methods are therefore not usually applicable to real world instances, which include complex constraints and large problem sizes. Thus, so-called metaheuristics are the most commonly used solution methods in practice (Elshaer and Awad, 2020). These methods have various strategies of getting out of local optima, but they do not guarantee global optimality of the solution. However they do provide solutions fast and usually within a reasonable gap of the global optimum.

In this thesis, we compare different local search based metaheuristic methods and first solution strategies in their performance of solving a DVRP using simulations. By this we hope to obtain information on which solution method one should choose in different VRP scenarios, which we feel is lacking in the current literature. The compared metaheuristics are Simulated Annealing (SA), Tabu Search (TS), Generic Tabu Search (GTS) and Guided Local Search (GLS). The first solution strategies used are Parallel Cheapest Insertion (PCI), Local Cheapest Insertion (LCI), Path Cheapest Arc (PCA) and Path Most Constrained Arc (PMCA). The problem parameters are varied in different ways and their effects on the solution characteristics are studied. Emphasis is especially placed on how the dynamism of the problem affects the solutions. The problem parameters that are varied are the dynamism of the problem, objective function, and the problem instance size.

This thesis is limited to the dynamic and deterministic VRP. The variant studied is a combination of CVRP, VRPTW and PDP with the added relaxation that vehicles may start and end in arbitrary locations, as opposed to starting and ending at a central depot which is most common in the VRP

literature (Toth and Vigo, 2014). The variant chosen is based on a real-world case study. The only dynamic element considered is the arrival of new transportation requests. As the problem type is deterministic, stochastic knowledge of future events is not used. However some knowledge from the case study is used to select parameters that generate random problem instances.

The rest of the thesis structured as follows. Chapter 2 gives an overview of the VRP variant under study and the methods used to solve it. Chapter 3 gives details on how the simulations for comparing the solution methods in different scenarios were carried out. The results and discussion on them are presented in Chapters 4 and 5, and the thesis is summarized in Chapter 6.

# Chapter 2

# Background

## 2.1 Dynamic Pickup and Delivery Problem with Time Windows

### 2.1.1 Overview

The VRP variant studied in this work is the Dynamic Pickup and Delivery Problem with Time Windows (DPDPTW). This variant models various real-life scenarios, such as less-than-truckload courier services, food delivery services or taxi services (Toth and Vigo, 2014), and was chosen for this work based on a case study. This variant is illustrated in Figures 2.1 and 2.2. The variant has the basic characteristics of a VRP: a fleet of vehicles and a set of locations are given and the problem is to find the minimum cost routes for the vehicles that visit all locations. Additionally the DPDPTW studied in this work has the following elements (Mitrović-Minić and Laporte, 2004; Toth and Vigo, 2014).

**Dynamic**. Not all information related to the routing problem is known initially but instead new information is received during the execution of the routing plan. In this case the new information is in the form of immediate transportation requests – these requests are pairs of pickup and delivery nodes, which are described below. Here it is assumed that requests arrive one at a time. Upon the arrival of a new request a new VRP needs to be solved, for which information about the past events is also needed. This includes the current cargo in each vehicle, the nodes that have been visited, and the next destinations for each vehicle, among other details. These dynamic subproblems can however be modeled and solved as essentially static problems that

include extra constraints for the dynamic elements of the problem (which is the case in this study). A commonly used measure for the dynamic nature of a DVRP is the degree of dynamism (Larsen, 2000), defined as

$$d = \frac{\text{Number of dynamic requests}}{\text{Number of total requests}} \tag{2.1}$$

Other measures for dynamism also exist, which for instance take into account the urgency of each request (in terms of its time window), but these measures are not used in this work. For comparison purposes, the static version of each dynamic problem is simulated as well.

**Deterministic**. A VRP may be either stochastic or deterministic in terms of the uncertain information related to the routing problem. If the problem is stochastic, some information about the uncertain elements of the problem is known in the form of probability distributions, while if it is deterministic, these distributions are not available, that is, nothing is known about future events until they present themselves.

**Pickups and deliveries**. Each transportation request in the problem consists of exactly two nodes: a pickup node and a delivery node. The two nodes must be visited by the same vehicle and the pickup node must be visited before the delivery node. This variant of a Pickup and Delivery Problem (PDP) is called a *one-to-one* PDP (Cordeau et al., 2008). Other variants also exist which differ in, for example, the number of pickup nodes and delivery nodes per request.

**Capacitated**. The variant includes capacity constraints for the vehicles, meaning that each vehicle has a limited capacity to carry cargo, and visits to nodes expend this capacity. Each visit to a pickup node adds a certain amount to the load of a vehicle, and each corresponding visit to a delivery node decreases the load by the same amount.

**Time windows**. Each node must be visited within a given time window, and drivers each have an allowed work time window. The time windows allow for slack, that is, a vehicle can wait at a node if it arrives before the beginning of the node's window. The actual visit to the node must still take place within the window.

**Service times**. Each visit to a node takes a certain amount of time, during which the node is being serviced and the vehicle cannot move.

**No central depot**. In the variant studied in this work, there is no central depot from which the vehicles depart and where they end their routes, which is a modification to the more common case of having a depot. Instead each

vehicle may start its route in an arbitrary location and ends its route at any delivery node. This is mostly due to the dynamic nature of the problem, where as a new request arrives, the vehicles are almost certainly in arbitrary locations outside the depot, but also due to modeling for example the case where drivers may begin their routes at home, and may also travel directly home from their last drop-off point. Using this formulation has the added benefit that the case where all vehicles begin or end at a central depot is simply a special case and requires no additional modeling.

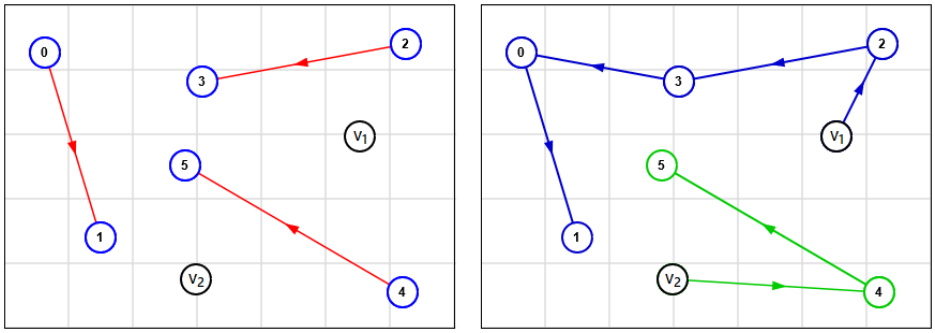


Figure 2.1: Illustration of a pickup and delivery problem (PDP). Vehicles are drawn in black while nodes that have to be visited are drawn in blue and all are drawn in the euclidean plane. Red arrows are drawn from pickup nodes to their corresponding delivery nodes. The left figure shows the problem to be solved while the right figure shows a feasible solution.

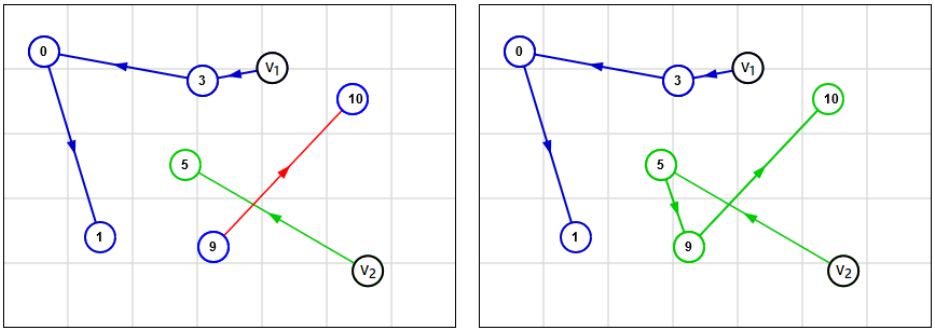


Figure 2.2: Illustration of a dynamic PDP. The left figure shows the situation after the solution seen in Figure 2.1 has been executed partway through, meaning that time has passed and the vehicles have moved along their routes and both have visited one (pickup) node. It also shows that a new dynamic request has arrived. The right figure shows a solution to the new problem and the changed route of vehicle 2.

**Optional nodes**. In some simulations, the nodes are optional, meaning that not visiting them produces a feasible solution to the problem. However, dropping even a single node adds a large penalty to the objective function, so the optimal solution is to drop as few nodes as possible.

### 2.1.2 Mathematical formulation

In general, there are multiple ways of formulating a VRP. Here, a vehicle flow formulation is presented, adapted from Toth and Vigo (2014). Let the set of pickup nodes $P = \{1, ..., n\}$, the set of delivery nodes $D = \{n+1, ..., 2n\}$, the set of vehicles $K = \{1, ..., |K|\}$, and the set of vehicle start locations $S = \{2n+1, ..., 2n+|K|\}$. Let also $V = P \cup D$ and $N = V \cup S$. We also introduce a dummy start depot and a dummy end depot, which have a distance 0 to all locations, with $d_s = 0$ being the start and $d_e = 2n + |K| + 1$ being the dummy end depot. The optimization problem is then

$$\min_{x} \quad \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} t_{ij} x_{ijk} \tag{2.2}$$

s.t.

$$\sum_{k \in K} \sum_{j \in V} x_{ijk} = 1 \qquad \forall i \in P \tag{2.3}$$

$$\sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{n+i,\, jk} \qquad \forall i \in P, k \in K \tag{2.4}$$

$$x_{d_s, 2n+k, k} = 1 \qquad \forall k \in K \tag{2.5}$$

$$\sum_{j \in S} x_{d_s jk} = 1 \qquad \forall k \in K \tag{2.6}$$

$$\sum_{i \in V} x_{id_e k} = 1 \qquad \forall k \in K \tag{2.7}$$

$$\sum_{j \in V} x_{jik} = \sum_{j \in V} x_{ijk} \qquad \forall i \in V, k \in K \tag{2.8}$$

$$T_{jk} \geq (T_{ik} + s_i + t_{ij}) x_{ijk} \qquad \forall i, j \in V, k \in K \tag{2.9}$$

$$Q_{jk} \geq (Q_{ik} + q_j) x_{ijk} \qquad \forall i, j \in V, k \in K \tag{2.10}$$

$$T_{n+i,k} \geq T_{ik} + s_i + t_{i,n+i} \qquad \forall i \in P \tag{2.11}$$

$$a_i \leq T_{ik} \leq b_i \qquad \forall i \in V, k \in K \tag{2.12}$$

$$\max(0, q_i) \leq Q_{ik} \leq \min(Q_k, Q_k + q_i) \qquad \forall i \in V, k \in K \tag{2.13}$$

$$x_{ijk} \in \{0, 1\} \qquad \forall i, j \in N, k \in K \tag{2.14}$$

The decision variables are $x_{ijk}$ which are 1 if vehicle $k$ goes from node $i$ to $j$ in the solution and 0 otherwise; $t_{ij}$ is the travel time from node $i$ to $j$; $T_{ik}$ is the time when vehicle $k$ arrives at node $i$; $s_i$ is the service time at node $i$; $[a_i,\ b_i]$ is the time window for node $i$; $q_i$ is the load to be picked up or delivered at node $i$ (positive for pickup, negative for delivery); $Q_k$ is the maximum capacity for vehicle $k$; and $Q_{ik}$ is the load of vehicle $k$ after visiting node $i$.

Constraints (2.3)–(2.4) ensure that a transportation request is served only once and that the pickup and delivery nodes are visited by the same vehicle. Constraints (2.6)–(2.9) ensure that all routes start and end at a dummy depot, while (2.10)–(2.11) ensure the consistency of time and capacity variables. Constraint (2.12) ensures that the pickup node of a request is visited before its corresponding delivery node, (2.13) enforces time window constraints, and (2.14) imposes capacity constraints.

The above objective minimizes the sum of the route lengths. When minimizing the length of the longest route, the objective function becomes

$$\min_{x} \max_{k \in K} \sum_{i \in N} \sum_{j \in N} t_{ij} x_{ijk} \tag{2.15}$$

## 2.2 Solution methods

The sections below describe first solution strategies, local search and metaheuristics in the context of solving VRPs. In general, metaheuristics are the main high-level strategies of solving VRPs, and they use the tools of local search within them. Some metaheuristics require first solution strategies (also referred to as constructive heuristics) to provide an initial feasible solution. The implementations used in this work require them, so they are discussed.

### 2.2.1 First solution strategies

First solution strategies are heuristic methods which are used to find a feasible initial solution to a VRP instance. This initial feasible solution can be improved by other methods. First solution strategies are less used today than in the past since metaheuristic methods nowadays are usually robust enough to not need a feasible solution to start the search from; instead any random solution will suffice (Toth and Vigo, 2014). Nowadays most papers

on VRP solution methods do not mention them at all.

Avdoshin and Beresneva (2019) describe and compare the performance of the most relevant first solution strategies. They also note that such a comparison has not been made – this is most likely due to the declining popularity of first solution strategies. Their findings indicate that first solution strategies have significant differences in their performance, with the savings algorithm (Clarke and Wright, 1964) being generally the most efficient. Van Breedam (2001) compares the Tabu Search (TS) and Simulated Annealing (SA) meta-heuristics in VRPs and states that the quality of the initial solution and thus the choice of first solution strategy has a significant impact on final solution quality, especially for TS. The methods assessed in the paper by Avdoshin and Beresneva (2019) that are relevant to this work are described below.

**Sequential insertion**. In this method, routes are constructed one vehicle at a time. The first step of the method is to initialize a route with a random unrouted node, such that the vehicle goes from the depot to the node and back. In the second step, another unrouted node is inserted into the route. This is done by choosing the node and its insertion position in the route such that the route length increase is minimized and such that the addition is feasible according to the constraints of the problem. Step two is then repeated until no further nodes can be added to the route, at which point a new route (for a new vehicle) is initialized according to step one, or all unrouted nodes have been incorporated into the routes.

**Parallel insertion**. This method is a modification of sequential insertion, with the routes being constructed in parallel. Step one is to initialize $k_{\min}$ feasible routes (as in sequential insertion) from the $k_{\min}$ closest unrouted nodes. Step two is to choose a random unrouted node and insert it into its best position among all the routes, with best meaning feasible and incurring the least increase in route length. Step two is then repeated until no node can be feasibly added into the routes, at which points a new route is initialized, or all nodes have been incorporated into routes.

**Nearest neighbor**. In this method routes are constructed one at a time. In step one, the closest unrouted node to the depot is chosen and a single route is initialized, going from the depot to the node and back. In step two, the closest unrouted node to the previously added one is inserted to the end of the route if it is feasible to do so. Step two is repeated until no new nodes can be added at the end of the route. A new route is then initialized if all nodes have not been included in routes.

In this work, four first solution strategies, or first solution strategies, are used

to find initial feasible solutions which are then improved by metaheuristics. These methods are very close to the ones described above but with slight alterations and differing names and they are as follows (Perron and Furnon, 2019).

**Local Cheapest Insertion (LCI)**. This method is essentially equivalent to the sequential insertion method described above. The cost of insertion is based on the arc cost function, that is, the travel times between nodes.

**Parallel Cheapest Insertion (PCI)**. This method is essentially equivalent to the parallel insertion method described above. The cost of insertion is based on the arc cost function, that is, the travel times between nodes.

**Path Cheapest Arc (PCA)**. This method is essentially equivalent to the nearest neighbor method described above.

**Path Most Constrained Arc (PMCA)**. This method is similar to PCA, but the node chosen for addition is not the nearest one but instead the most constrained one (or rather the most constrained arc, the connection between two nodes), based on the constraints defined by the problem instance, such as time windows, node demands, pickups and deliveries, and so on.

The first solution strategies were chosen out of 12 available in the optimization software used (Perron and Furnon, 2019) based on performance in the simulated DPDPTW scenarios. These four performed the best while the others were deemed not viable to use in further comparison studies.

### 2.2.2 Local search

Local search is a method for iteratively finding approximate solutions to combinatorial optimization problems. It has proven to be effective in this context and many methods build upon it (Voudouris et al., 2010; Groër et al., 2010). Local search is based on exploring the neighborhood $N(x)$ of a solution $x$, where the neighborhood of a solution is defined by local search operators (also known as *moves* or *improvement heuristics*) that transform the solution $x$ in some way and yield a new solution $x'$. Figure 2.3 shows two examples of these operators in a VRP context.

In the context of a VRP, these operators can be divided into two categories: intra-route operators and inter-route operators (Toth and Vigo, 2014). In an intra-route operator, the order of nodes is modified within a single vehicle's route. Since only a single vehicle is involved, operators that have been developed for the travelling salesman problem can be readily utilized. A common

operator of this type is the $k$-opt exchange, where $k$ edges are removed and replaced by other edges in the route (Helsgaun, 2009).

In inter-route operators, nodes are exchanged between the routes of two vehicles, and these are necessary for obtaining high quality solutions. Common operators include the relocate operator, where $k$ consecutive nodes are moved from one route to another; the swap operator, where consecutive nodes are swapped between routes; and the 2-opt operator, where two edges are removed from two routes and they are connected in a different way. Iterating through a complete neighborhood defined by these standard operators requires $\mathcal{O}(n^2k^2)$ operations, so the neighborhood is usually pruned in some way (Toth and Vigo, 2014).

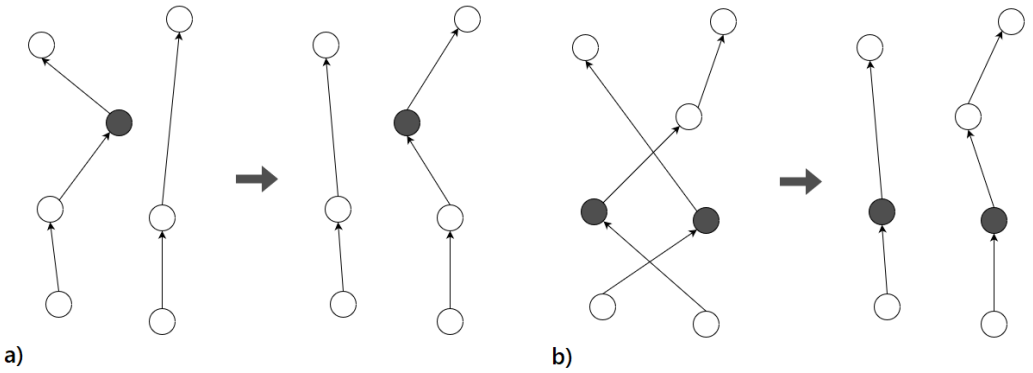


Figure 2.3: Two examples of local search operators in a VRP context. a) is the relocate operator and b) is the swap operator. Circles are nodes that have to be visited by vehicles and arrows indicate the routes of vehicles.

With a minimization problem of

$$\min \ f(x), \quad x \in S \tag{2.16}$$

where $S$ is the solution space, local search moves from a solution $x$ to a new solution $x' \in N(x)$ using the neighborhood $N(x)$ given by the local search operators (Groër et al., 2010). If only improving moves are considered, such that $f(x') < f(x)$ always, the search can be considered greedy, as in this way when the first local minimum is reached the search stops as no more direct improvements can be made. Worsening moves with $f(x') > f(x)$ can also be accepted to expand the search and to escape local minima, and this is the case with virtually all (local search based) metaheuristic methods (Boussaïd et al., 2013).

### 2.2.3 Metaheuristics

A significant amount of research has been done on metaheuristics in the context of VRPs and according to Elshaer and Awad (2020) they are the most popular type of solution method. This is in part due to the fact that while exact methods are available for some VRP variants, in general they are computationally unattractive, as VRPs are NP-hard combinatorial problems and real-life instances are often quite large (Bai et al., 2021).

Metaheuristics are methods which can solve hard optimization problems approximately without tailoring the method for each specific problem (Boussaïd et al., 2013). The prefix "meta" illustrates that the methods are higher level than problem-specific, low-level heuristics (the term "metaheuristic" was coined by Glover (1986)). Martí et al. (2006) describe a metaheuristic as a "master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality". Metaheuristics do not guarantee global optimality, and they do not use the gradient of the objective function, instead relying on other means such as local search (Boussaïd et al., 2013).

Metaheuristics can be divided into two categories: local search based methods (also *single solution* or *trajectory-based* methods) and population-based methods (Talbi, 2009). Population-based methods can be further divided into evolutionary and swarm intelligence methods. In this work the focus is solely on local search methods, so population methods are not discussed further.

A key concept relating to metaheuristics is the balance between intensification and diversification (Blum and Roli, 2003). Diversification means that the solution space is searched in a global, general and expansive way, and varied solutions are obtained. Randomization can be utilized in this. Intensification on the other hand means the opposite: the search is concentrated in a small region and one seeks to find local optima. As mentioned above, global optimality is not guaranteed, but if these two elements are balanced well, achieving a global optimum becomes more likely.

Elshaer and Awad (2020) state that important local search based methods include Simulated Annealing (SA), Tabu Search (TS), Large Neighborhood Search (LNS), Iterated Local Search (ILS), Guided Local Search (GLS), Variable Neighborhood Search (VNS), and Greedy Randomized Adaptive Search Procedure (GRASP). According to their review, out of these methods TS and VNS are applied most frequently in solving VRPs; LNS, SA, ILS and GRASP are used less often; and GLS is rarely used.

The performance of a metaheuristic depends on the VRP variant (Elshaer and Awad, 2020). Both local search based methods and population-based methods work well and are used frequently. Khouadjia et al. (2013) compare several population-based methods and one local search based method (TS) on DVRPs and conclude that a multi-population-based method (Multi-Adaptive Particle Swarm Optimization, MAPSO) outperforms others, which are quite level with each other.

Guided Local Search has been used and compared to other trajectory-based methods such as Tabu Search in some papers. Voudouris et al. (2010) compare GLS to TS in Travelling Salesman problems and find that GLS is either very competitive or much better in all cases. De Backer et al. (1997) compare GLS to TS in basic VRP settings and find that GLS outperforms TS most of the time. Zhong and Cole (2005) compare GLS to GRASP in VRPTWs with backhauls and customer precedence and find that GLS outperforms GRASP without precedence, while with precedence the opposite is true.

The solution methods of OR-tools are compared in DVRP problems by Abdirad et al. (2021). The problem instance generation and the dynamic problem modeling and solving are not disclosed however. Not all available first solution strategies are compared, but no reasons are given for this. According to their results, there is no clear best metaheuristic or first solution strategy. Apart from listing the best combination of first solution strategy and metaheuristic for each instance, no other data, comment, justification or insight is provided.

The metaheuristics that are used in this work (SA, TS, GLS) are described below. In addition to these, a greedy descent method (GD) that is used in the comparisons is described.

**Simulated Annealing**. Introduced by Kirkpatrick et al. (1983), this method is inspired by the physical process of annealing, where a material is brought to a high temperature, after which the temperature is lowered slowly to bring the material to a state of minimal energy, thereby avoiding local minima of energy in the process. In the context of optimization, the objective function to be minimized corresponds to the energy of the system. The algorithm is based on probabilistically accepting moves that worsen the objective function in order to escape local minima. The probability of moving to a worse solution depends on the temperature $T$, which is lowered gradually as the search continues. Therefore, the search is initially quite diverse and the solution space is explored more globally, and as the temperature lowers, the search becomes more intensified and focused on certain regions of the space.

Algorithm 1 gives an outline of the method (Boussaïd et al., 2013).

**Algorithm 1** Simulated Annealing

```
1: Generate initial solution x
2: Set initial temperature T
3: while stopping criterion is not met do
4:     repeat
5:         Generate new solution x' ∈ N(x)
6:         if f(x') ≤ f(x) then
7:             x ← x'
8:         else
9:             x ← x' with probability exp(−(f(x')−f(x))/T)
10:        end if
11:    until thermodynamic equilibrium
12:    Decrease T
13: end while
14: return best solution found
```

**Tabu Search**. This method, developed by Glover (1986), is memory-based as opposed to the probability-based Simulated Annealing. The central idea in it is that the features of previously visited solutions are not allowed (are made 'tabu') in subsequent solutions for a period of time. This diversifies the search and prevents the cycling of the same solutions. This is implemented with a tabu list in which last seen solutions are kept and the same or similar solutions are not visited again during the time they are in the list.

**Algorithm 2** Tabu Search

```
1: Generate initial solution x
2: Initialize empty tabu list
3: while stopping criterion is not met do
4:     Generate new solution x' ∈ N(x) not in the tabu list
5:     x ← x'
6:     Update tabu list
7: end while
8: return best solution found
```

The length of time that solutions remain in the list affects how diversifying or intensifying the search is. A long list leads to a diverse search since a high number of solutions are disallowed, while a short list intensifies the search to a narrower region. The length can also be varied during the search. To enhance the search further, so-called aspiration criteria are used to override

a solution being disallowed by the tabu list. The method is outlined in Algorithm 2 (Boussaïd et al., 2013).

**Guided Local Search**. This method was proposed by Voudouris et al. (2010). Like Tabu Search, it is memory-based. However, solution cycling is prevented with soft penalties in the form of an augmented objective function, instead of hard constraints. GLS uses a local search procedure that (greedily) finds a local minimum of the augmented objective. Every time a local minimum is reached, this augmented objective is updated, enabling the search to escape local minima.

**Algorithm 3** Guided Local Search

1: Select features $F_i$, $i = 1, ..., N$
2: Initialize penalties $p_i$ and costs $c_i$
3: Generate initial solution $x$
4: **while** stopping criterion is not met **do**
5: Find a local minimum $x^*$ of the augmented objective function using local search starting from $x$
6: Compute the utility $u_i$ of each feature $F_i$ of $x^*$
7: $j \leftarrow \arg\max_i u_i$
8: $p_j \leftarrow p_j + 1$
9: $x \leftarrow x^*$
10: **end while**
11: **return** best solution found

GLS defines a set of features $F_i$, $i = 1, ..., N$ and a function $I_i(x)$ which indicates whether a solution $x$ has the feature $i$. The features are given costs $c_i$ (which do not change during the search) and penalties $p_i$, which are initialized to zero and which increase during the search depending on which solutions are visited. The augmented objective is updated only by changing the penalties $p_i$, and the feature to be penalized at each iteration is selected by a utility computation. The utility of penalizing feature $i$ in a local optimum $x^*$ is $u_i(x^*) = I_i(x^*) \cdot \frac{c_i}{1+p_i}$. The purpose of this is to find the features that are the most impactful in terms of keeping the search from cycling. The augmented objective is defined as $g(x) = f(x) + \lambda \sum_i^N p_i I_i(x)$, where $f(x)$ is the unaltered objective function, and $\lambda$ is a parameter that adjusts the relative strength of the penalties. A large $\lambda$ incurs large penalties and diversifies the search, while a small $\lambda$ intensifies the search ($\lambda = 0$ corresponds to the unaltered objective). This feature penalization process focuses the search on promising regions. Algorithm 3 gives an outline (Boussaïd et al., 2013).

**Greedy Descent**. In addition to the metaheuristics described above, a greedy optimization method (Greedy Descent, GD) is also used in the performance comparisons. In this method one simply moves to the solution in the neighborhood of the current solution that yields the greatest improvement in the objective function, until a local minimum is reached. Algorithm 4 describes the method.

**Algorithm 4** Greedy Descent

1: Generate initial solution $x$
2: **repeat**
3: $x^* \leftarrow \arg\min_{x' \in N(x)} f(x')$
4: **if** $f(x^*) < f(x)$ **then**
5: $x \leftarrow x^*$
6: **end if**
7: **until** $f(x^*) \geq f(x)$
8: **return** $x$

# Chapter 3

# Simulation study description

## 3.1 Overview of simulations

In this work we carried out simulations of various DPDPTW scenarios based on a case study using different solution methods. The term scenario is used in this work for a particular combination of objective function, problem size and dynamism for a problem instance. This was done to determine the performance of each solution method in these different routing scenarios. These performance characteristics can then be used in decision recommendations on choosing the right solution method for a given problem in real-world VRP use cases.

We chose 5 different metaheuristics, 4 different first solution strategies and 2 different objective functions to compare in the simulations, including all combinations of these. The metaheuristics are: Greedy Descent (GD), Guided Local Search (GLS), Simulated Annealing (SA), Tabu Search (TS) and Generic Tabu Search (GTS) (TS and GTS are both tabu search methods but with different parameters and so they are named differently for convenience; this is a feature of the software used). The first solution strategies are: Local Cheapest Insertion (LCI), Parallel Cheapest Insertion (PCI), Path Chepest Arc (PCA) and Path Most Constrained Arc (PMCA). The objective functions are: to minimize the sum of the lengths of all routes (Minimize Sum, MS), and to minimize the length of the longest route (Minimize Longest, ML).

The choice of metaheuristics and first solution strategies was mostly due to practical reasons and ease of implementation. These methods are readily available in the chosen optimization software which is Google's OR-tools

(Perron and Furnon, 2019). The metaheuristics used include all of the metaheuristics available in OR-tools, while the first solution strategies are a subset of the 12 available. The first solution strategies were compared in preliminary simulations which showed that the chosen 4 strategies were most viable, while most of the others were not usable due to their poor performance in the simulated problem instances.

In this work we assume that the metaheuristics and first solution strategies are independent of each other; the analyzed quantities are presented by metaheuristic or by first solution strategy but not by combinations of both. Interactions between metaheuristics and first solution strategies are possible but they are assumed negligible and are not studied in this work.

The chosen objective functions model real-world business incentives. With MS, the motivation is to minimize the total amount of paid work hours of all drivers, effectively minimizing the operational costs of the vehicle fleet. This (minimizing route cost) is one of the most common objective functions (Psaraftis et al., 2016). With ML, the motivation is to find a routing plan which completes the transportation requests in the shortest real time possible, disregarding the operational costs that this may incur.

The simulation study consists of two simulation types. The first simulation type gathers data about the average cost and success rate of the solution methods. The second type on the other hand gathers data about the proportion of nodes that are not visited when all nodes are optional.

It is important to note that the VRP variant and all of the problem instance parameters come from a case study. Therefore the goal of this work is not to study the effect of each parameter in an exhaustive manner, but rather to find the best performing method in various possible scenarios of the case study. This is evident for example with the capacities of the vehicles, which most of the time do not actually constrain the problem at all. This also means that the results of this study are applicable primarily for the case study and cannot be generalized for the most part.

### 3.1.1 Problem instances

For each combination of metaheuristic, first solution strategy and objective function, the same sets of problem instances were solved. These were custom-made DPDPTW instances which were designed such that a solution should practically always exist (although not proven). The problem sizes are shown in Table 3.1. In the tables, 'seeds' indicates the number of random instances

generated for each problem size. For each instance the number of transportation requests is half of the number of nodes, that is, half of the nodes are pickup nodes and half are delivery nodes.

Table 3.1: Problem sizes for the simulations.

| Nodes | Vehicles | Seeds |
|---|---|---|
| 40 | 4 | 10 |
| 80 | 8 | 10 |
| 120 | 12 | 10 |
| 160 | 16 | 10 |
| 200 | 20 | 10 |

For each problem instance of a given size, a static version and a dynamic version of the problem was generated. The degree of dynamism was 0.5 for all of the dynamic simulations.

Custom-made problem instances were used in this study instead of benchmark VRP instances, since this work does not present a novel solution method for VRPs and since the problem parameters come from the case study. Instead a number of solution methods are compared against each other in different VRP scenarios to determine which performs best under which circumstances. It is thus sufficient that the VRP instances are comparable to each other and that the range of scenarios is wide enough. If benchmarks were to be used in this work, they would have to be modified in some ways, defeating the purpose of using a benchmark. In this study for example the start and end locations of the vehicles are free, that is, the vehicles do not start and end at a central depot. This is not common in the literature and benchmarks for such problem instances do not seem to exist or at least have been used very little, which again counteracts the usefulness of benchmarks in this case.

### 3.1.2 Analyzed quantities

Each analyzed quantity was computed for each combination of metaheuristic, first solution strategy and objective function. For each problem size there were multiple random problem instances, and the values were averaged across these instances to obtain the final values presented.

**Static cost**. This value is the average best objective function value for the static version of the problem found by the solution method.

**Dynamic cost**. This value is the average best objective function value for the dynamic version of the problem found by the solution method.

**Success rate**. This value is the proportion of dynamic problems that are solved to completion. In dynamic problems, each arrival of a transportation request creates a new subproblem that has to be solved. If any of the subproblems are not solved successfully, the simulation run of the dynamic problem is counted as a failure and the run decreases the success rate of the specific combination of solution methods.

**Static drop proportion**. This value is the average proportion of nodes that are not visited in static problems when all nodes are optional with a penalty.

**Dynamic drop proportion**. Same as above but for dynamic problems.

We chose these specific quantities as they measure the reliability and performance of the different solution methods. Cost is the main performance metric in both dynamic and static problems. Success rate is the main reliability metric and it is quite essential since good performance (low cost) is meaningless if the reliability (success rate) is close to zero. The drop proportion is not a central measure of performance, but it provides useful information about the characteristics of the different methods. In a real-world setting it is often the case that some problems are infeasible in the sense that not all requests can be serviced, which in turn requires one to drop some requests. Knowing about the node dropping characteristics of different methods will then help in choosing the best method for such a situation.

It is common to use the so-called competitive ratio when comparing the performance of solution methods in static and dynamic problems (Golden et al., 2008). This ratio is defined as $cr = \sup_I \frac{z(I)}{z^*(I)}$ where $z(I)$ is the cost of a solution found by an algorithm to the dynamic problem instance $I$ and $z^*(I)$ is the optimal cost to the corresponding static problem. This quantity shows the effect on algorithm performance by the lack of full information about the problem. In its exact sense the ratio requires the globally optimal solution to each problem instance, so it can be used in this sense only in simple DVRPs. For more complex problems empirical estimations of this ratio are used.

The ratio is not however used for performance measurements in this study. Since the computed static cost is not the true optimum, in our case the ratio could be estimated in two ways for each method: one where the static cost is the best from the method in question; and one where the static cost is the best across all methods. In the first case, the competitive ratios are

not directly comparable across methods, since for example a high static cost improves the ratio for a given method. In the second case, comparing the ratios amounts to directly comparing dynamic costs, which is what is done here.

## 3.2 Simulation details

### 3.2.1 Convergence of metaheuristics

The metaheuristics used in this study require a stopping criterion which can be chosen in way that is most suited to the scenario at hand. In this case the solution time was chosen as the criterion for computational resource purposes; otherwise simulation running times can be highly variable and hard to predict which makes allocating high performance cloud computing resources more difficult. The amount of time given to solve each problem instance is based on the time it takes for the methods to roughly converge, as measured by the objective function value.

Auxiliary simulations were carried out to determine the convergence characteristics of each metaheuristic and how they relate to the problem instance size. Convergence in general can be defined quite arbitrarily and depends on the problem, but in these simulations convergence is defined as the objective function value improving by less than 0.1% (relative to the best known value) in the previous 10 seconds of computation time.

Table 3.2: Solver time limits for the problem instances.

| Nodes | Time limit |
|---|---|
| 40 | 20 s |
| 80 | 40 s |
| 120 | 60 s |
| 160 | 80 s |
| 200 | 100 s |

Based on the auxiliary simulations, the time until convergence seems to scale roughly linearly. The time limits used in the simulations proper for the static problems are given in Table 3.2. For the dynamic cases, the advance problems each have the same time limit as the corresponding static problem of the same size, while each dynamic subproblem (when each new request arrives) has a 5 second time limit regardless of problem size. Here, an advance problem

refers to the static part of a dynamic problem, that is, the requests that are known in advance.

### 3.2.2 Problem instance generation

The procedure to generate a random DPDPTW instance is given below. This general problem instance is then used in generating a static-dynamic pair of problem instances. The chosen parameters aim to model the case study.

**General problem instance generation.** The parameters required for this procedure are the number of vehicles $k$, the total number of nodes to be visited $n$ (half of which are pickup nodes and half are the corresponding delivery nodes), and a random seed.

First, locations are assigned to $n$ nodes and $k$ vehicles. The locations are uniformly random in a 40 by 40 unit square. The speed of a vehicle is 1 length unit in 1 time unit (time units can be considered to be minutes), and vehicles move in straight lines between nodes (as opposed to by Manhattan distance, for instance).

Second, time windows are assigned to nodes. The allowed visiting window for each node is 08:00—16:00 (480—960 in minutes), representing a normal working day. The vehicles start moving at 08:00 (earliest) and stop moving at 16:00 (latest). The time windows therefore serve the purpose of constraining the number of working hours for a single driver.

Third, service times are assigned to nodes from a uniformly random distribution. Each node takes 1 to 10 minutes to service during which the vehicle is not moving; this represents picking up or dropping off a package.

Finally, demands are assigned to nodes. Each pickup node adds 1 unit of cargo to the vehicle that visits it, and each corresponding delivery decreases the cargo by 1 unit. The maximum load of a vehicle is 50 units. Since the capacities of vehicles are large, most of the time they do not constrain the problem.

**Static-dynamic problem pair generation**. The degree of dynamism $d$ is a required parameter. This is the proportion of transportation requests that are dynamic, that is, arrive during the execution of the routing plan. Only one dynamic request is added at a time before rerouting, corresponding to 2 nodes (1 pickup node and 1 delivery node).

First, two auxiliary problem instances are generated by the general problem

instance generation. One contains only the requests known in advance ($n \cdot (1 - d)$ nodes) and one contains only the dynamic requests ($n \cdot d$ nodes).

Second, the time instants when the dynamic requests become known are generated. The instants are uniformly random in the interval 08:01—13:00 and are unique — in other words there is at most one dynamic request in each minute. No dynamic requests arrive during the last 3 hours of the working day; this is done to make the problem instances very likely to be feasible. For example an dynamic request arriving at 15:59 would almost certainly make the problem infeasible.

Now the dynamic version of the problem is one where the dynamic requests are not known beforehand, and the static version of the problem is one where the dynamic requests are known beforehand. Thus in the static problem the advance requests and dynamic requests are combined into one problem where everything is known in advance.

### 3.2.3 Cost computation

For the static version of the problem, the cost of the solution is simply the value minimized by the solver, since all information related to the problem is known at once. The static costs for each objective are

$$C_{\text{MS, static}} = \sum_{k=1}^{K} L_k \tag{3.1}$$

$$C_{\text{ML, static}} = \max_{k} L_k \tag{3.2}$$

where $K$ is the number of vehicles and $L_k$ is the route length for vehicle $k$. The route lengths are in units of time, in this case minutes.

In the dynamic problem, not all information is known and several subproblems need to be solved which then constitute the overall problem. In general the purpose of computing the dynamic total cost is to obtain a value that can be meaningfully compared to the cost of the corresponding static problem. The final cost is not however simply the sum of the costs of the solved subproblems.

This is illustrated in Figure 3.1. The bars for each $V_i$ denote the planned or executed lengths of routes for each vehicle. Each $T_i$ is a time instant when an dynamic transportation request becomes known, after which the routing problem is solved again which in turn may change the planned routes. As

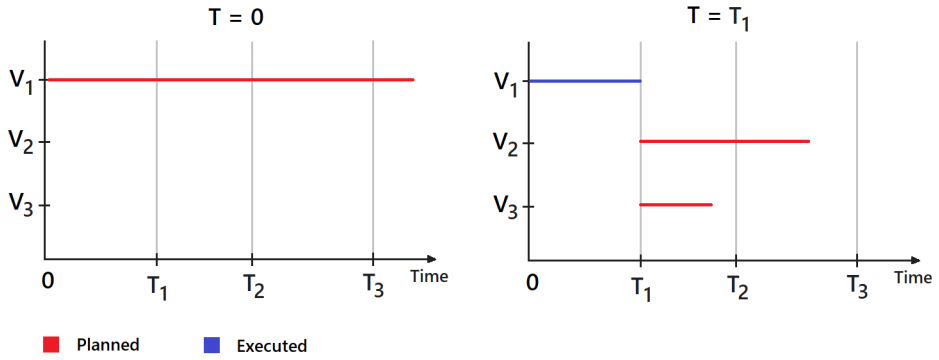


Figure 3.1: A possible assignment of routes to vehicles in a dynamic problem. The left figure shows the routing plan at time $T = 0$. At this time vehicle $V_1$ is planned to visit all the nodes. The right figure shows the plan at time $T = T_1$, when a new dynamic request has arrived and vehicle $V_1$ has executed part of its route. The routing plan has then changed and vehicles $V_2$ and $V_3$ have routes while the route of $V_1$ is empty.

time passes between some $T_i$ and $T_{i+1}$, the planned routes within that time interval are executed.

In this work, the overall dynamic cost is the cost from all executed routes instead of the cost from all planned routes. Idle time of the vehicles (the time during which a vehicle is not assigned any nodes to visit) is not included in the cost. The optimization strategy in the dynamic case is to reoptimize using the previous solution upon the arrival of each new transportation request.

The total dynamic cost with the MS objective is the sum of all executed routes:

$$C_{\text{MS, dynamic}} = \sum_{j=1}^{J} \sum_{k=1}^{K} \min\left(L_{j-1,k}\ ,\ T_j - T_{j-1}\right) \tag{3.3}$$

where $J$ is the number of time instants when dynamic requests arrive, $T_j$ is the $j$th such instant, $K$ is the number of vehicles, and $L_{j,\,k}$ is the planned route length for vehicle $k$ at $T = T_j$ (the length is from $T_j$ on, not from $T_0$).

The real-world aim of the ML objective is to complete all transportation requests as fast as possible. As idle time is not included in the cost, the dynamic cost consists of the longest executed route for each segment $T_i, T_{i+1}$. In Figure 3.2 this amounts to $C_{\text{ML}} = T_1' + (T_2' - T_1) + (T_{\text{end}}' - T_2)$. In general

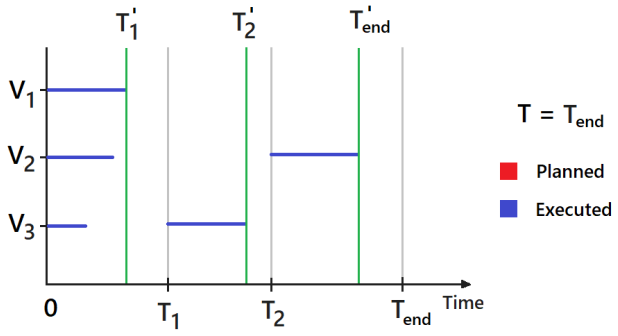


Figure 3.2: A possible assignment of routes to vehicles in a dynamic problem. The figure shows the executed routes at time $T = T_{\text{end}}$ when all requests have been completed.

the total dynamic cost with the ML objective is then

$$C_{\text{ML, dynamic}} = \sum_{j=1}^{J} \max_{k} \left(\min\ (L_{j-1,k}\ ,\ T_j - T_{j-1})\right) \tag{3.4}$$

where the notations are as in equation (3.3).

### 3.2.4 Simulation procedure: cost and success rate

The algorithmic procedure for computing the average static and dynamic costs as well as success rates is described below. This procedure is carried out for both objective functions, for each problem size, for each metaheuristic, for each first solution strategy, and for each random seed (problem instance). In this simulation all nodes must be visited; dropping even one node makes the solution infeasible. The problem instance sizes are described in Table 3.1 and they are designed to be feasible with a very high probability. In this simulation the recorded quantities are the cost of the static problem, the cost of the dynamic problem and the success or failure to route the dynamic problem.

**Simulation: cost, success rate**

1. Generate static-dynamic problem pair
2. Solve static problem and record statistics
3. Solve dynamic problem
   3.1. Solve advance problem

3.2. For each dynamic request:
   3.2.1. Advance time according to the request time instant
      – This involves moving the vehicles along their planned routes and visiting nodes
   3.2.2. Add the new dynamic request to the problem
      – The request consists of adding two new nodes to the problem: a pickup node and a delivery node
      – The two new nodes are added into the previous solution by assigning the request to the 'most idle' vehicle, namely the vehicle with the earliest route end time; the insertion is done at the end of the route
   3.2.3. Solve the subproblem based on the previous solution
      – Solution time for each subproblem is 5 seconds regardless of overall problem size since the amount of new nodes added is constant
      – If the by-hand insertion initial solution is not feasible according to the solver, it has to generate a feasible solution from scratch using a first solution strategy
   3.2.4. Increment total cumulative dynamic cost according to equation (3.3) or (3.4), depending on the objective function
3.3. Record statistics for the dynamic problem

4. Add the statistics of the simulation run to the overall pool of statistics
   – If the by-hand insertion and first solution strategy both fail after some dynamic request, the dynamic problem is not solved and the run is counted as a failure, which decreases the success rate of the specific combination of solution methods and objective function; in this case the obtained cost values are discarded as well
   – Otherwise the cost values contribute towards the average cost values and the success rate is increased

Failures to solve from a previous solution in the dynamic case may sometimes be due to implementation and software specific details. The initial solution, where the metaheuristic starts its search, is given in the software only as the sequences of visited nodes of each vehicle. Other variables pertaining to the solution, such as visit times and loads of the vehicles, are not given and have to be inferred by the solver; this is called the propagation of secondary variables. Occasionally the solver fails to infer the feasibility of an initial solution even though it may actually be valid. If the propagation fails, the solver has to start from scratch using a first solution strategy. In the dynamic case, the first solution strategies can thus be seen as a backup when the

initial solution fails, either by being actually infeasible or only according to the solver.

### 3.2.5 Simulation procedure: node drop proportion

In these simulations, all nodes are optional with a large penalty to the objective function. For each dropped node, a penalty of 10000 minutes is added to the objective. The problem instances are the same as for the cost and success rate simulations. While feasible solutions (solutions where no nodes are dropped) may be found if all nodes are required, these solutions can be missed if nodes are optional. The analyzed quantity is the proportion of nodes that are dropped in the solution. The described procedure is again done for both objective functions, for each problem size, for each heuristic, for each first strategy, and for each random seed (problem instance).

**Simulation: node drop proportion**

1. Generate problem by the general problem instance generation
   - Make all nodes optional with a penalty
2. Solve instance
   - In the static case this is straightforward
   - In the dynamic case, this involves solving each subproblem as described in the cost simulation
3. Record the total number of nodes that are dropped and add to the overall pool of statistics

### 3.2.6 Simulation software and hardware

The software suite OR-tools (Perron and Furnon, 2019) was used to carry out the optimization computations. This in turn also dictated which metaheuristics and first solution strategies were able to be studied in the work.

The parameters used for the metaheuristics were as follows. For SA, the initial temperature was set to 100 and a Cauchy annealing schedule was used, where the temperature at iteration $i$ is $T_i = T_0/i$. The relevant parameters for the two tabu search methods are the 'keep' list length, the 'forbid' list length and the tabu factor. A variable in the 'keep' list must keep its value for a given number of iterations and a variable in the 'forbid' list must not take its value in the list for a given number of iterations. The tabu factor

gives the number of violations to the tabu criterion that are allowed; a factor of 1 means no violations are allowed, and a factor of 0 means all violations are allowed. For TS, the parameters were: tabu factor 0.8, 'keep' iterations 10, 'forbid' iterations 10. For GTS: tabu factor 1, 'keep' iterations 0, 'forbid' iterations 100. The value used for the regularization parameter $\lambda$ for GLS was 5.

The simulations were ran on the Aalto Triton high performance computing cluster, using Intel Xeon Gold 6248 2.50GHz processors.

# Chapter 4

# Simulation study results

## 4.1 Success rate

Tables 4.1 and 4.2 show the success rates for each problem size and first solution strategy for both objective functions. These values are for the dynamic problems; the success rates for static problems are always 1. Based on the success rate values, the first solution strategies are more reliable for the ML objective. This is likely explained by the following. In general, the ML solutions tend to distribute nodes evenly to all the vehicles since the goal is to minimize the length of the longest route and giving all nodes to a single vehicle would counteract this. The MS objective on the other hand usually does the opposite: it tends to give all nodes to a single vehicle. For the ML objective subproblems then, generally a large number of vehicles are constrained to visit at least some nodes which in turn means that only a small number of nodes are constrained to any given vehicle. This then likely makes finding a feasible route for a single vehicle easier. For the MS objective subproblems, the situation is the opposite: most of the nodes are usually constrained to very few vehicles, which in turn can make finding a feasible route for a single vehicle more difficult.

Both objectives show a drop-off with increasing problem size which is more pronounced with the MS objective. This is intuitive since a larger problem size entails a more complex and difficult problem to solve.

Also for both objectives it is clear that some first solution strategies are more reliable than others. PCI and PCA are not very robust for large problem sizes, while PMCA is especially robust. PMCA is clearly the best performing method as it finds an initial solution 100% of the time for both objectives.

Table 4.1: Success rate with ML objective by first solution strategy in the dynamic case.

| Nodes<br>First solution | 40 | 80 | 120 | 160 | 200 |
|---|---|---|---|---|---|
| LCI | 1.00 | 1.00 | 0.96 | 0.56 | 0.44 |
| PCI | 1.00 | 1.00 | 0.96 | 0.58 | 0.18 |
| PCA | 1.00 | 0.94 | 0.58 | 0.00 | 0.00 |
| PMCA | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |

Table 4.2: Success rate with MS objective by first solution strategy in the dynamic case.

| Nodes<br>First solution | 40 | 80 | 120 | 160 | 200 |
|---|---|---|---|---|---|
| LCI | 0.92 | 0.44 | 0.18 | 0.04 | 0.00 |
| PCI | 0.72 | 0.08 | 0.00 | 0.00 | 0.00 |
| PCA | 0.10 | 0.00 | 0.00 | 0.00 | 0.00 |
| PMCA | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |

PCA is a particularly weak method for the MS objective, even for small problem sizes. The robustness of PMCA is no doubt due to its most-constrained-first method of assigning values to variables and thus of finding a solution. Since the dynamic subproblems can be large and very highly constrained, methods which assign values to variables in a cheapest-first approach (especially PCA) often fail to find a solution. This is especially clear with the MS objective, where as mentioned above there can be a large number of nodes already constrained to a single vehicle.

## 4.2 Static cost

Figures 4.1, 4.2, 4.3 and 4.4 show the cost values for static problems from the simulations. For each figure, the values are the differences to the mean within each problem size group. The plain cost values are included in Appendix A.

From Figures 4.1 and 4.2 one can see that the relative differences in cost between metaheuristics decrease as the problem size increases. This result takes into account the fact that a larger problem requires more time to solve

to convergence, see Section 3.2.1. One can also see that most of the time, Guided Local Search (GLS) is the best performing metaheuristic.

Figures 4.3 and 4.4 show the static costs by first solution strategy. With the MS objective as seen in Figure 4.3, the impact of the strategy seems to increase with growing problem size, but the differences in cost between the methods are relatively small. For the ML objective, there seems to be a drastic difference between a problem with 40 nodes and larger problem sizes. The variation in cost between methods and also within a single method between problem sizes is significant. LCI and PCA seem to perform the best, at least for large problem sizes.

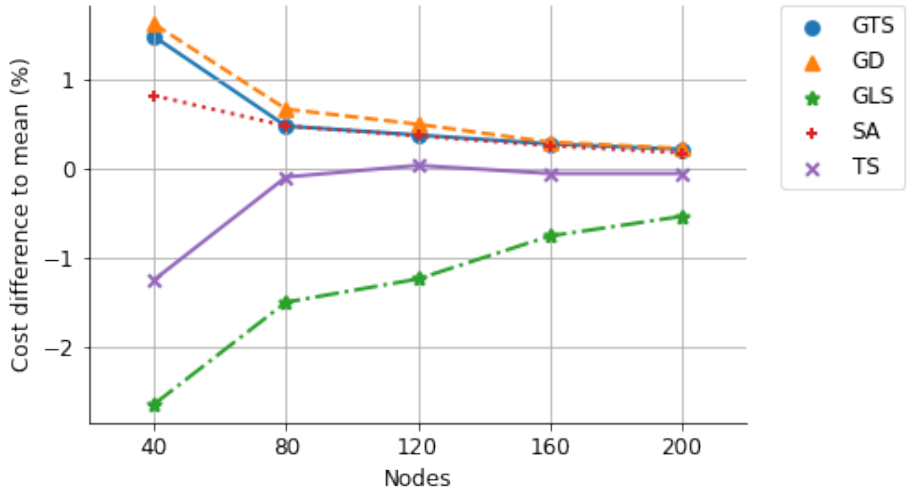


Figure 4.1: Average static cost by metaheuristic with MS objective.

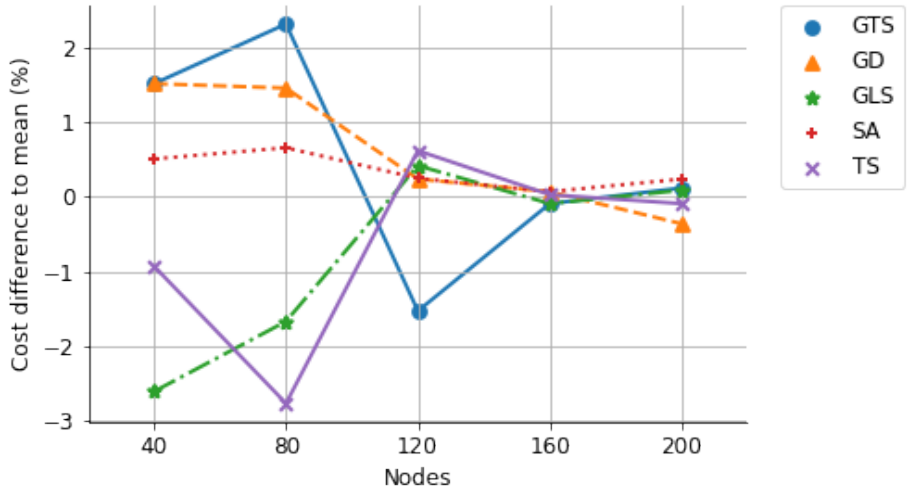


Figure 4.2: Average static cost by metaheuristic with ML objective.

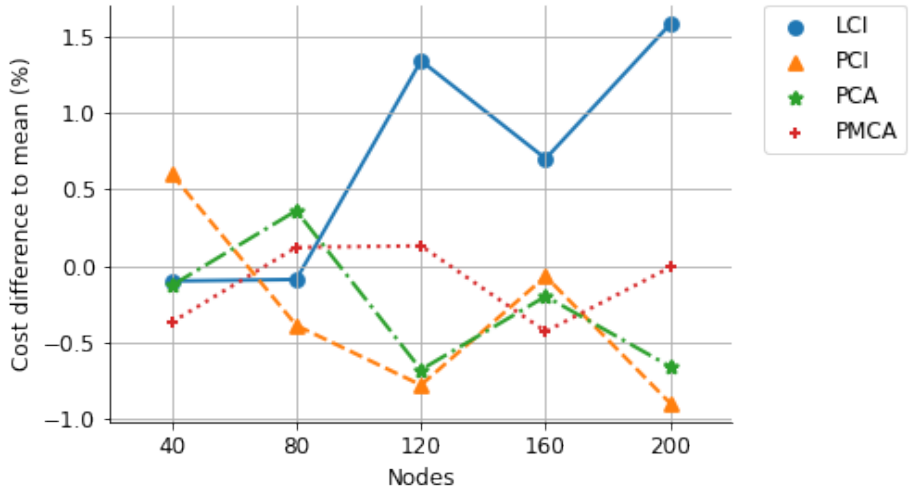


Figure 4.3: Average static cost by first solution strategy with MS objective.

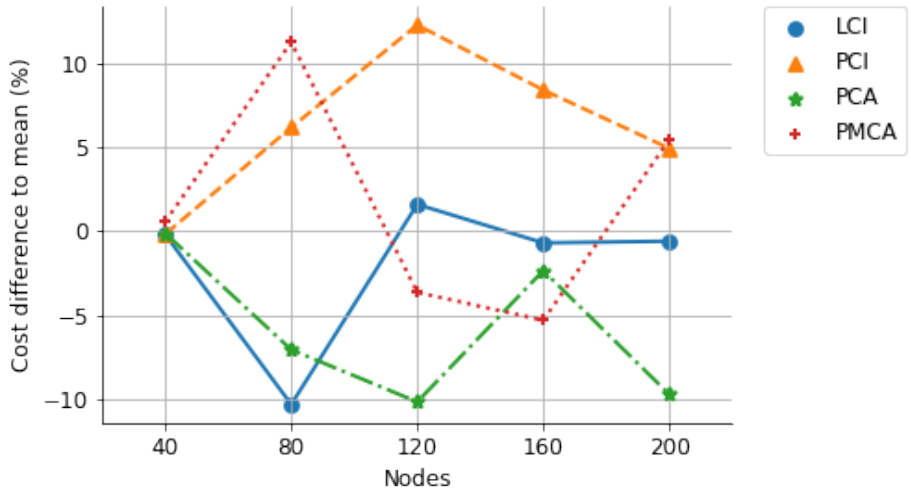


Figure 4.4: Average static cost by first solution strategy with ML objective.

## 4.3 Dynamic cost

Figures 4.5, 4.6, 4.7 and 4.8 show the cost values for dynamic problems from the simulations. For each figure, the values are the differences to the mean within each problem size group. In these figures, missing data points indicate that the success rate for a particular solution method and problem size is too low to include the cost value as a reliable data point. A cost value is omitted if the success rate is below 80% for a given combination. The plain cost values are included in Appendix A.

The data in Figure 4.5 seems very noisy and cannot be interpreted very clearly. Figure 4.6 shows data that seems more well-behaved and indicates that GLS is again a well-performing method.

The interpretation of Figures 4.7 and 4.8 is quite clear: PMCA is very robust in terms of its success rate in dynamic scenarios, while other methods are not.

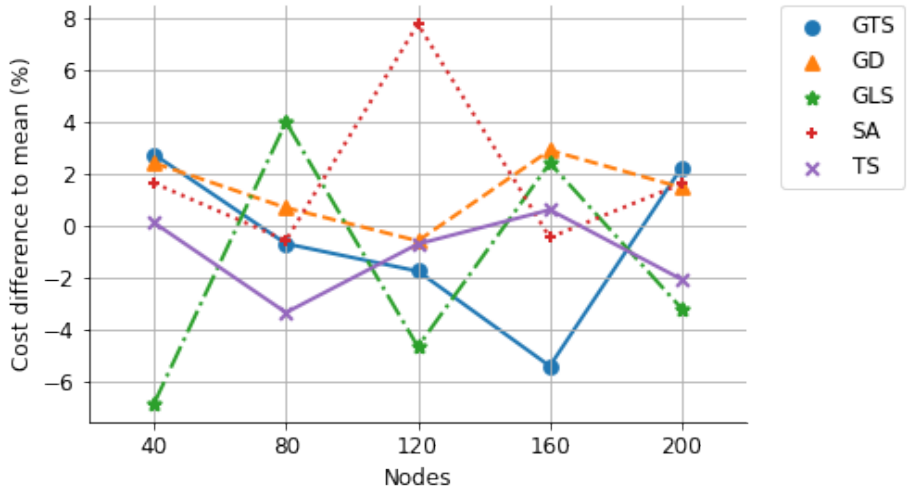


Figure 4.5: Average dynamic cost by metaheuristic with MS objective.

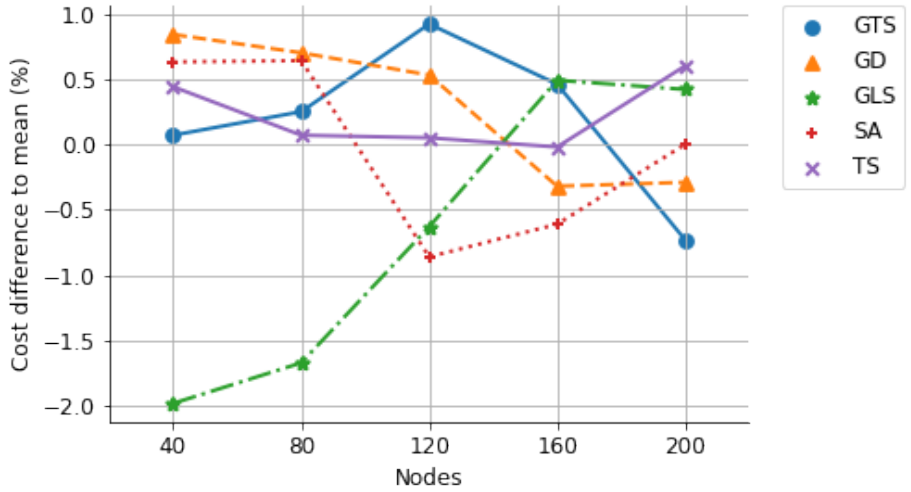


Figure 4.6: Average dynamic cost by metaheuristic with ML objective.

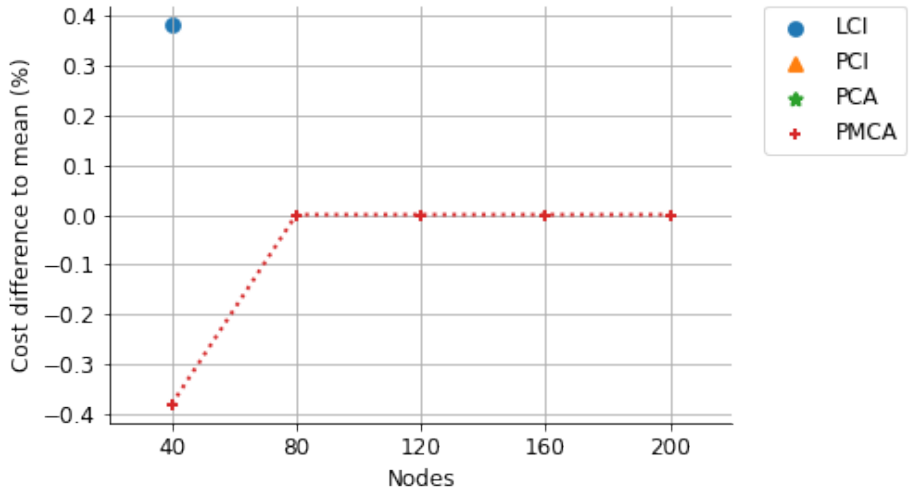


Figure 4.7: Average dynamic cost by first solution strategy with MS objective. Missing data points indicate that the success rate for a particular solution method and problem size is too low (below 0.80) to include the cost value as a reliable data point.

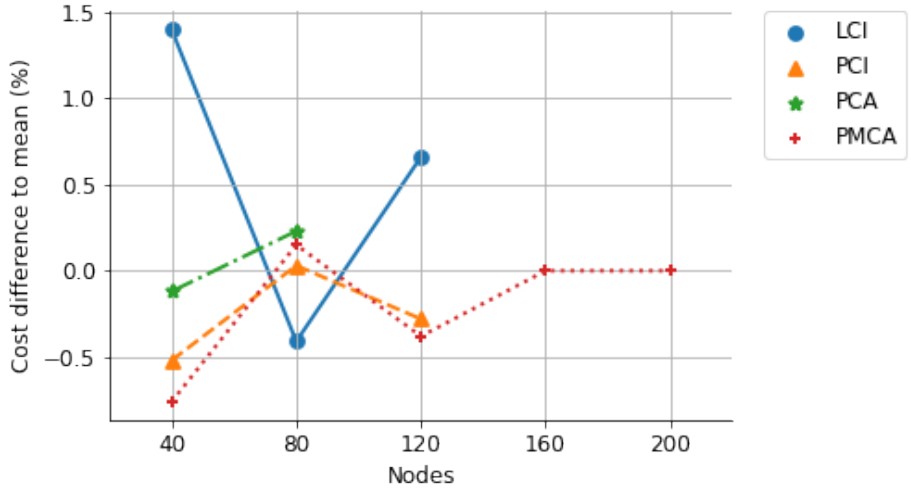


Figure 4.8: Average dynamic cost by first solution strategy with ML objective. Missing data points indicate that the success rate for a particular solution method and problem size is too low (below 0.80) to include the cost value as a reliable data point.

## 4.4 Drop proportion

Tables 4.3 and 4.4 show the drop proportions in the dynamic case. The drop proportions for the static case are all zero. Since the static success rates are always one, this is to be expected but it is not entirely self-evident. The drop proportions differ little when differentiated by metaheuristic so the values are only shown for the first solution strategies.

We can see that the drop proportions contain very similar information to the success rates. PMCA is again is a very well performing method, with its drop proportions being zero in all instances with ML and close to zero with MS. As the success rates are higher with the ML objective, the drop proportions are likewise lower with ML.

Table 4.3: Drop proportion in the dynamic case with MS objective.

| Nodes<br>First solution | 40 | 80 | 120 | 160 | 200 |
|---|---|---|---|---|---|
| LCI | 0.00 | 0.09 | 0.29 | 0.37 | 0.46 |
| PCI | 0.03 | 0.22 | 0.40 | 0.49 | 0.49 |
| PCA | 0.00 | 0.29 | 0.49 | 0.49 | 0.50 |
| PMCA | 0.00 | 0.00 | 0.02 | 0.02 | 0.10 |

Table 4.4: Drop proportion in the dynamic case with ML objective.

| Nodes<br>First solution | 40 | 80 | 120 | 160 | 200 |
|---|---|---|---|---|---|
| LCI | 0.0 | 0.0 | 0.0 | 0.00 | 0.32 |
| PCI | 0.0 | 0.0 | 0.0 | 0.00 | 0.40 |
| PCA | 0.0 | 0.0 | 0.0 | 0.05 | 0.45 |
| PMCA | 0.0 | 0.0 | 0.0 | 0.00 | 0.00 |

## 4.5 Aggregated data

Table 4.5 shows the success rate values aggregated by dynamism, objective and problem size. Tables 4.6 and 4.7 show the aggregated cost values. The size of the problem has been combined into two categories: small, corresponding to the value for 40 nodes; and large, corresponding to an average of the values for 80, 120, 160, and 200 nodes. This is primarily for summarization purposes, but it also represents somewhat of a split in the performance of the methods across problem sizes, at least for some scenarios (such as those seen in Figures 4.1, 4.4, 4.7, 4.8 and Table 4.2). Bold values indicate the best performing method in each scenario, and missing values indicate that the success rate in that particular scenario is too low (below 0.80) to include the cost value as a reliable data point. The data from which the aggregation was done is included in Appendix A.

Table 4.5: Aggregated success rates by first solution strategy in different scenarios.

| Objective | Dynamism | Problem size | PCI | LCI | PCA | PMCA |
|---|---|---|---|---|---|---|
| MS | dynamic | small | 0.72 | 0.92 | 0.10 | **1.00** |
| MS | dynamic | large | 0.02 | 0.17 | 0.00 | **1.00** |
| ML | dynamic | small | **1.00** | **1.00** | **1.00** | **1.00** |
| ML | dynamic | large | 0.68 | 0.74 | 0.38 | **1.00** |

Table 4.6: Aggregated costs (percentage difference to mean) by first solution strategy in different scenarios.

| Objective | Dynamism | Problem size | PCI | LCI | PCA | PMCA |
|---|---|---|---|---|---|---|
| MS | static | small | 0.60 | -0.10 | -0.13 | **-0.37** |
| MS | static | large | **-0.54** | 0.88 | -0.30 | -0.05 |
| MS | dynamic | small | — | 1.84 | — | **1.06** |
| MS | dynamic | large | — | — | — | **1.87** |
| ML | static | small | -0.19 | **-0.22** | -0.14 | 0.56 |
| ML | static | large | 7.92 | -2.51 | **-7.32** | 1.92 |
| ML | dynamic | small | -0.52 | 1.40 | -0.12 | **-0.76** |
| ML | dynamic | large | — | — | — | **4.70** |

Table 4.7: Aggregated costs (percentage difference to mean) by metaheuristic in different scenarios.

| Objective | Dynamism | Problem size | GTS | TS | GLS | SA | GD |
|---|---|---|---|---|---|---|---|
| MS | static | small | 1.47 | -1.25 | **-2.64** | 0.81 | 1.61 |
| MS | static | large | 0.33 | -0.04 | **-1.01** | 0.31 | 0.41 |
| MS | dynamic | small | 2.74 | 0.10 | **-6.88** | 1.63 | 2.41 |
| MS | dynamic | large | **-1.27** | -1.23 | -0.21 | 2.22 | 1.27 |
| ML | static | small | 1.52 | -0.93 | **-2.61** | 0.51 | 1.52 |
| ML | static | large | 0.20 | **-0.55** | -0.31 | 0.30 | 0.35 |
| ML | dynamic | small | 0.07 | 0.44 | **-1.98** | 0.63 | 0.84 |
| ML | dynamic | large | 0.22 | 0.17 | **-0.34** | -0.20 | 0.15 |

# Chapter 5

# Discussion

## 5.1 Application of results

The results presented in the previous section have been summarized into decision recommendations in Table 5.1, based on the aggregated data in Tables 4.5, 4.6 and 4.7. The problem scenarios are divided into categories by whether they are static or dynamic, by objective function and by problem size. The recommendations are purely a broad summary of the results.

Table 5.1: Solution method recommendations for PDPTW scenarios. 'small' denotes a problem with 40 or fewer nodes and 'large' a problem with more than 40 nodes. (*) indicates that the choice is crucial to reliability or performance.

| Objective | Dynamism | Problem size | First solution | Metaheuristic |
|---|---|---|---|---|
| MS | static | small | PMCA | GLS |
| MS | static | large | PCI | GLS |
| MS | dynamic | small | PMCA | GLS |
| MS | dynamic | large | PMCA* | GTS |
| ML | static | small | LCI | GLS |
| ML | static | large | PCA | TS |
| ML | dynamic | small | PMCA | GLS |
| ML | dynamic | large | PMCA* | GLS |

These recommendations are based on the success rate and cost of a particular method, since the drop rates contain similar information to the success rates. As mentioned in Section 3.1, first solution strategies and metaheuristics are assumed independent and therefore the recommendations for them

are independent as well. The 'crucial' recommendations do not have a strict definition; however, for instance in the large dynamic case with MS objective it is quite obvious that the PMCA method performs best while other methods are not very viable.

The recommendations are to be applied principally for the case study, and possibly other scenarios which are very similar. Generalizing beyond these circumstances is problematic.

The recommendations assume that the parameters of the VRP scenario, such as the degree of dynamism and total problem size, are known beforehand. It is possible for instance that the total number of requests received is unknown, in which case these recommendations are more difficult to apply. However it seems more likely that in most real-world use cases the parameters are known with at least some degree of confidence, owing to the existence of historical data. However, it is possible to modify the recommendations such that they are given in terms of the current dynamic subproblem to be solved. In this way the nature of the overall problem would not be necessary to know beforehand but instead only the problem at hand.

Overall we can see that Guided Local Search (GLS) is the metaheuristic of choice for most situations out of the ones compared. The best first solution strategy depends on the scenario, but PMCA is clearly the most reliable method. The two methods marked with (*) are the most significant choices; not choosing the recommended alternatives in these scenarios will likely lead to worse results.

In general, finding a feasible initial solution to a VRP with time windows is NP-hard. Naturally this initial solution becomes harder to find with an increasingly large and complex problem. Based on this work, the method by which one attempts to obtain a first feasible solution to a DPDPTW is very important. The choice of first solution strategy has a large impact on, firstly, whether a solution is found at all; and secondly, what the final value of the objective function is after optimization by a heuristic.

In dynamic problem settings, there is always a certain amount of real-world time between the arrival of dynamic requests. If these times are short, in the order of seconds, utilizing the previous solution is important as this saves the effort of always computing a feasible solution from scratch to highly complex problems. If the previous solution is not used, the choice of first solution strategy is essential. This is because it is then not simply a backup to using the previous solution but is used for every dynamic subproblem. In these circumstances it is important to choose a method such as PMCA which will

in most cases yield a solution even if the dynamic subproblem is large and highly constrained. It is also possible to switch solution methods on the fly: for instance one may choose to use the most efficient first solution strategy as a default and upon its failure switch to a more reliable one.

## 5.2 Evaluation

This study was a reasonable success all things considered. There are no major issues that would thoroughly undermine the work, and most of the obtained results are very useful to the real-world case study which the simulation parameters were based on.

The results for cost values for different methods seem quite reliable, at least for the static case. Since the dynamic cost values depend heavily on the success rate values, the values themselves may not be as informative and have room for interpretation. Some random noise is always present but this could be mitigated by increasing sample sizes. The drop rate results again may have some noise in them but seem quite reliable.

The time window and capacity constraints used in this work constrain the problem only moderately since they need to model the case study. The capacities especially do not constrain the problem at all. The time windows would have to be made narrower and the capacities smaller to properly constrain the problem and to generalize the obtained results.

Some results are more implementation and software specific than others. The results that relate to the specific performance differences of the first solution strategies can mostly be applied only if one is using the exact implementations provided by the OR-tools software package, although they are based on methods well known in the literature. Still, the conclusion can be drawn that in any complex VRP one should choose very carefully the method of finding an initial feasible solution. For the metaheuristics however, all of the methods studied are very well known and studied, and their performance differences in this work can be more readily generalized.

## 5.3 Future work

The expansion of these kinds of simulations would widen the usefulness and range of applications for their results. As it stands, the results are only

applicable for the case study. These expansions would naturally consume significantly more computational resources but we believe this would be a worthwhile investment. Possible extensions include the following.

**No first solution strategies**. The implementations of the metaheuristics required first solution strategies, which is not the norm in the literature. This makes the obtained results less readily generalizable and more dependent on the software. A future goal could be to use implementations which are more standard in the literature, which would allow for more direct comparisons.

**More solution methods**. Implementing more metaheuristics and first solution strategies would be laborious but fruitful. This would increase the number of use cases where the results are applicable and enable more comprehensive comparisons between methods. As mentioned in Section 3.1, a possible extension would be to also study the interactions between metaheuristics and first solution strategies.

**Larger sample sizes**. The number of random problem instances for each combination of problem parameters could be increased, which would make the results obtained more statistically robust and less prone to random noise.

**More variations of problem parameters**. This includes for instance increasing the number of problem sizes, adding different possible time window configurations, using different node location distributions, or using different dynamism characteristics for the problem.

**Different VRP variants**. One could study simpler variants, such as those without time windows, without vehicle capacities or without pickups and deliveries. One could also study even more specialized variants of the DPDPTW. Results about more general variants would perhaps be more useful: the more specific the variant, the less real-world use cases there are for the results. Still, if a very specific use case has simulation data tailored exactly for it, the data is of course more valuable than more generic VRP simulation data.

**Benchmark problem instances**. Using benchmarks would require slight alterations to the parameters of the VRP variant but it would enable more direct comparison to other similar work in the literature.

# Chapter 6

# Conclusion

In this work we studied a VRP variant called the Dynamic Pickup and Delivery Problem with Time Windows (DPDPTW), and in particular, how different solution methods perform in different scenarios of this problem variant. This variant was chosen based on a case study on the subject and includes time window constraints for each visited node, capacity constraints for each vehicle and a pickup node – delivery node pair for each transportation request, among other details.

We compared five metaheuristics and four first solution strategies (which provide a feasible initial solution for the metaheuristics) with two objective functions. The metaheuristics were Simulated Annealing (SA), Tabu Search (TS), Generic Tabu Search (GTS) and Guided Local Search (GLS). The first solution strategies were Parallel Cheapest Insertion (PCI), Local Cheapest Insertion (LCI), Path Cheapest Arc (PCA) and Path Most Constrained Arc (PMCA). The objective functions were: to minimize the sum of the lengths of all routes, and to minimize the length of the longest route. These methods were compared in DPDPTW scenarios which differed in the degree of dynamism and problem size.

The main results obtained were that Guided Local Search (GLS) outperforms other metaheuristics in most scenarios, and that Path Most Constrained Arc (PMCA) is the most robust first solution strategy. A key result is also that the choice of first solution strategy can have a large impact on the final solution quality, depending on the scenario. These results are mostly supported by the literature, especially the performance of GLS. First solution strategies have been declining in popularity in VRPs due to metaheuristic implementations generally not needing them, so current literature on them is quite scarce.

The applicability of the results is limited. Naturally they are most relevant if the use case is the same as in the study, with metaheuristic implementations that require first solution strategies, the same VRP variant, similar problem parameters and so on. Generalizing beyond these circumstances is problematic. However when the circumstances do match the results can potentially be very useful, such as for the case study which this work was based on. Still, the result relating to the performance of GLS can tentatively be generalized across different VRP settings as it has backing in the literature.

# Bibliography


Maryam Abdirad, Krishna Krishnan, and Deepak Gupta. A two-stage metaheuristic algorithm for the dynamic vehicle routing problem in industry 4.0 approach. *Journal of Management Analytics*, 8(1):69–83, 2021.

S.M. Avdoshin and E.N. Beresneva. Constructive heuristics for Capacitated Vehicle Routing Problem: a comparative study. *Trudy ISP RAN*, 31(3), 2019.

Ruibin Bai, Xinan Chen, Zhi-Long Chen, Tianxiang Cui, Shuhui Gong, Wentao He, Xiaoping Jiang, Huan Jin, Jiahuan Jin, Graham Kendall, et al. Analytics and Machine Learning in Vehicle Routing Research. *arXiv preprint arXiv:2102.10012*, 2021.

Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

Ilhem Boussaïd, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013.

Geoff Clarke and John W Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

Jean-Françcois Cordeau, Gilbert Laporte, and Stefan Ropke. Recent Models and Algorithms for One-to-one Pickup and Delivery Problems. In *The Vehicle Routing Problem: Latest Advances and New Challenges*, pages 327–357. Springer, 2008.

George B Dantzig and John H Ramser. The Truck Dispatching Problem. *Management Science*, 6(1):80–91, 1959.

Bruno De Backer, Vincent Furnon, P Prosser, P Kilby, and Paul Shaw. Local search in constraint programming: Application to the vehicle routing

problem. In *Proc. CP-97 Workshop Indust. Constraint-Directed Scheduling*, pages 1–15. Schloss Hagenberg Austria, 1997.

Raafat Elshaer and Hadeer Awad. A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers & Industrial Engineering*, 140:106242, 2020.

Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533–549, 1986.

Bruce L Golden, Subramanian Raghavan, and Edward A Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43. Springer Science & Business Media, 233 Spring Street, New York, NY 10013, USA, 2008.

Chris Groër, Bruce Golden, and Edward Wasil. A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation*, 2(2):79–101, 2010.

Keld Helsgaun. General k-opt submoves for the Lin–Kernighan TSP heuristic. *Mathematical Programming Computation*, 1(2):119–163, 2009.

Mostepha R Khouadjia, Briseida Sarasola, Enrique Alba, El-Ghazali Talbi, and Laetitia Jourdan. Metaheuristics for Dynamic Vehicle Routing. In *Metaheuristics for Dynamic Optimization*, pages 265–289. Springer, 2013.

Scott Kirkpatrick, C Daniel Gelatt, and Mario P Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

Allan Larsen. *The Dynamic Vehicle Routing Problem*. PhD thesis, Institute of Mathematical Modelling, Technical University of Denmark, 2000.

Rafael Martí, Manuel Laguna, and Fred Glover. Principles of scatter search. *European Journal of Operational Research*, 169(2):359–372, 2006.

Snežana Mitrović-Minić and Gilbert Laporte. Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B: Methodological*, 38(7):635–655, 2004.

Laurent Perron and Vincent Furnon. OR-Tools, 2019. URL `https://developers.google.com/optimization/`.

Harilaos N Psaraftis, Min Wen, and Christos A Kontovas. Dynamic vehicle routing problems: Three decades and counting. *Networks*, 67(1):3–31, 2016.

Ulrike Ritzinger, Jakob Puchinger, and Richard F Hartl. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, 54(1):215–231, 2016.

El-Ghazali Talbi. *Metaheuristics: From Design to Implementation*, volume 74. John Wiley & Sons, 111 River Street, Hoboken, New Jersey, NJ 07030, USA, 2009.

Paolo Toth and Daniele Vigo. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, 3600 University City Science Center, Philadelphia, PA 19104-2688, USA, 2002.

Paolo Toth and Daniele Vigo. *Vehicle Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics, 3600 Market Street, 6th Floor, Philadelphia, PA 19104-2688, USA, 2014.

Alex Van Breedam. Comparing descent heuristics and metaheuristics for the vehicle routing problem. *Computers & Operations Research*, 28(4): 289–315, 2001.

Christos Voudouris, Edward PK Tsang, and Abdullah Alsheddy. Guided Local Search. In *Handbook of Metaheuristics*, pages 321–361. Springer, 2010.

Yingjie Zhong and Michael H Cole. A vehicle routing problem with backhauls and time windows: a guided local search solution. *Transportation Research Part E: Logistics and Transportation Review*, 41(2):131–144, 2005.

# Appendix A

# Additional cost data

Table A.1: Cost values by first solution strategy. Omitted values indicate that no data exists, that is, the success rate is zero.

| Objective | Dynamism | Problem size | PCI | LCI | PCA | PMCA |
|---|---|---|---|---|---|---|
| MS | static | 40 | 473.8 | 470.5 | 470.3 | 469.2 |
| MS | static | 80 | 850.8 | 853.4 | 857.2 | 855.2 |
| MS | static | 120 | 1217.4 | 1243.4 | 1218.6 | 1228.5 |
| MS | static | 160 | 1570.1 | 1582.3 | 1568.1 | 1564.6 |
| MS | static | 200 | 1903.6 | 1951.2 | 1908.2 | 1920.7 |
| MS | dynamic | 40 | 620.2 | 613.0 | 566.2 | 608.3 |
| MS | dynamic | 80 | 1011.5 | 1171.9 | — | 1183.7 |
| MS | dynamic | 120 | — | 1864.4 | — | 1825.9 |
| MS | dynamic | 160 | — | 2243.5 | — | 2559.5 |
| MS | dynamic | 200 | — | — | — | 3404.0 |
| ML | static | 40 | 158.9 | 158.9 | 159.0 | 160.1 |
| ML | static | 80 | 199.5 | 168.6 | 174.7 | 209.0 |
| ML | static | 120 | 382.8 | 346.5 | 306.4 | 328.6 |
| ML | static | 160 | 425.9 | 390.1 | 383.6 | 372.0 |
| ML | static | 200 | 451.2 | 427.4 | 388.3 | 453.3 |
| ML | dynamic | 40 | 319.6 | 325.7 | 320.8 | 318.8 |
| ML | dynamic | 80 | 350.6 | 349.0 | 351.3 | 351.0 |
| ML | dynamic | 120 | 372.4 | 376.0 | 382.0 | 372.1 |
| ML | dynamic | 160 | 382.2 | 380.1 | — | 414.8 |
| ML | dynamic | 200 | 388.1 | 388.4 | — | 475.1 |

Table A.2: Cost values (percentage difference to mean) by first solution strategy. Omitted values indicate that no data exists, that is, the success rate is zero.

| Objective | Dynamism | Problem size | PCI | LCI | PCA | PMCA |
|---|---|---|---|---|---|---|
| MS | static | 40 | 0.60 | -0.10 | -0.13 | -0.37 |
| MS | static | 80 | -0.39 | -0.09 | 0.36 | 0.12 |
| MS | static | 120 | -0.78 | 1.34 | -0.68 | 0.13 |
| MS | static | 160 | -0.07 | 0.70 | -0.20 | -0.43 |
| MS | static | 200 | -0.90 | 1.58 | -0.66 | -0.01 |
| MS | dynamic | 40 | 3.03 | 1.84 | -5.93 | 1.06 |
| MS | dynamic | 80 | -10.51 | 3.68 | — | 4.73 |
| MS | dynamic | 120 | — | 1.16 | — | -0.93 |
| MS | dynamic | 160 | — | -9.14 | — | 3.66 |
| MS | dynamic | 200 | — | — | — | 0.00 |
| ML | static | 40 | -0.19 | -0.22 | -0.14 | 0.56 |
| ML | static | 80 | 6.15 | -10.31 | -7.06 | 11.22 |
| ML | static | 120 | 12.24 | 1.58 | -10.16 | -3.65 |
| ML | static | 160 | 8.38 | -0.71 | -2.36 | -5.31 |
| ML | static | 200 | 4.91 | -0.61 | -9.71 | 5.41 |
| ML | dynamic | 40 | -0.52 | 1.40 | -0.12 | -0.76 |
| ML | dynamic | 80 | 0.03 | -0.41 | 0.23 | 0.15 |
| ML | dynamic | 120 | -0.84 | 0.09 | 1.70 | -0.95 |
| ML | dynamic | 160 | -2.58 | -3.14 | — | 5.72 |
| ML | dynamic | 200 | -6.98 | -6.89 | — | 13.87 |

Table A.3: Cost values by metaheuristic.

| Objective | Dynamism | Problem size | GTS | TS | GLS | SA | GD |
|---|---|---|---|---|---|---|---|
| MS | static | 40 | 477.9 | 465.1 | 458.6 | 474.8 | 478.6 |
| MS | static | 80 | 858.2 | 853.3 | 841.4 | 858.2 | 859.8 |
| MS | static | 120 | 1231.5 | 1227.3 | 1211.7 | 1231.4 | 1233.0 |
| MS | static | 160 | 1575.6 | 1570.4 | 1559.3 | 1575.2 | 1575.9 |
| MS | static | 200 | 1925.0 | 1919.8 | 1910.5 | 1924.1 | 1925.2 |
| MS | dynamic | 40 | 618.4 | 602.5 | 560.5 | 611.7 | 616.4 |
| MS | dynamic | 80 | 1125.4 | 1095.4 | 1178.7 | 1126.9 | 1141.2 |
| MS | dynamic | 120 | 1801.0 | 1820.3 | 1747.4 | 1976.0 | 1822.4 |
| MS | dynamic | 160 | 2354.8 | 2505.2 | 2550.1 | 2478.4 | 2562.5 |
| MS | dynamic | 200 | 3480.1 | 3333.0 | 3294.4 | 3458.7 | 3453.7 |
| ML | static | 40 | 161.6 | 157.7 | 155.0 | 160.0 | 161.6 |
| ML | static | 80 | 192.3 | 182.7 | 184.8 | 189.2 | 190.7 |
| ML | static | 120 | 335.9 | 343.2 | 342.5 | 341.9 | 341.9 |
| ML | static | 160 | 392.6 | 393.0 | 392.6 | 393.2 | 393.2 |
| ML | static | 200 | 430.6 | 429.7 | 430.4 | 431.1 | 428.5 |
| ML | dynamic | 40 | 321.4 | 322.6 | 314.8 | 323.2 | 323.9 |
| ML | dynamic | 80 | 351.3 | 350.7 | 344.6 | 352.7 | 352.9 |
| ML | dynamic | 120 | 379.1 | 375.8 | 373.3 | 372.4 | 377.6 |
| ML | dynamic | 160 | 394.2 | 392.3 | 394.3 | 390.0 | 391.1 |
| ML | dynamic | 200 | 414.2 | 419.7 | 419.0 | 417.2 | 416.0 |

Table A.4: Cost values (percentage difference to mean) by metaheuristic.

| Objective | Dynamism | Problem size | GTS | TS | GLS | SA | GD |
|---|---|---|---|---|---|---|---|
| MS | static | 40 | 1.47 | -1.25 | -2.64 | 0.81 | 1.61 |
| MS | static | 80 | 0.47 | -0.10 | -1.50 | 0.47 | 0.66 |
| MS | static | 120 | 0.37 | 0.03 | -1.24 | 0.36 | 0.49 |
| MS | static | 160 | 0.27 | -0.06 | -0.76 | 0.25 | 0.29 |
| MS | static | 200 | 0.21 | -0.06 | -0.54 | 0.17 | 0.22 |
| MS | dynamic | 40 | 2.74 | 0.10 | -6.88 | 1.63 | 2.41 |
| MS | dynamic | 80 | -0.43 | -3.09 | 4.28 | -0.30 | 0.96 |
| MS | dynamic | 120 | -2.28 | -1.23 | -5.19 | 7.22 | -1.12 |
| MS | dynamic | 160 | -4.63 | 1.46 | 3.28 | 0.37 | 3.78 |
| MS | dynamic | 200 | 2.24 | -2.09 | -3.22 | 1.61 | 1.46 |
| ML | static | 40 | 1.52 | -0.93 | -2.61 | 0.51 | 1.52 |
| ML | static | 80 | 2.32 | -2.77 | -1.67 | 0.66 | 1.46 |
| ML | static | 120 | -1.53 | 0.62 | 0.42 | 0.25 | 0.24 |
| ML | static | 160 | -0.09 | 0.03 | -0.09 | 0.07 | 0.07 |
| ML | static | 200 | 0.12 | -0.09 | 0.09 | 0.24 | -0.36 |
| ML | dynamic | 40 | 0.07 | 0.44 | -1.98 | 0.63 | 0.84 |
| ML | dynamic | 80 | 0.25 | 0.07 | -1.67 | 0.64 | 0.70 |
| ML | dynamic | 120 | 0.92 | 0.05 | -0.63 | -0.86 | 0.53 |
| ML | dynamic | 160 | 0.46 | -0.02 | 0.49 | -0.61 | -0.32 |
| ML | dynamic | 200 | -0.73 | 0.60 | 0.42 | -0.00 | -0.29 |