A distributed approach to improve inland water transportation addressing privacy and incremental improvements

S. Geboers

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

A distributed approach to improve inland water transportation addressing privacy and incremental improvements

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

S. Geboers

November 29, 2022

Faculty of · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of for acceptance a thesis entitled

A distributed approach to improve inland water transportation addressing privacy and incremental improvements

by

S. Geboers

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: November 29, 2022

Supervisor(s):

Prof.dr.ir. B. De Schutter

Reader(s):

Dr.ir. T. van den Boom

Dr. V. Reppa

Abstract

Currently, inland waterway shipping mainly includes barges and bulk transportation with little to no variation in volume, product, and route. Since transport over water emits less CO_2 per tonne-km than road transport, a potential way to reduce CO_2 emissions is to transport more containers via inland waterways. Large delays cause unreliability, which in their turn cause an opportunity cost and increased operational costs. These increased costs have a negative impact on the competitiveness of inland water transport with regards to road transport.

An alternative to the way planning is handled currently is by solving a Vehicle Rotaton Planning Problem (VRPP). The problem is solved by minimizing a cost function. For example, the total time spent in a port area (called the sojourn time) can be minimized.

Solving the VRPP results in a rotation plan that dictates the order in which vessels visit terminals and how many containers they should transport from one terminal to another. There are multiple ways to solve the VRPP, one of which is through the use of the Distributed Pseudotree Optimization Protocol (DPOP), where the vessel operators reach an optimal solution together by sharing only information about those variables that share the same constraints with other variables from other agents. An attractive property of DPOP is that it is possible to include privacy constraints in the algorithm, such that vessel operators only have to share minimal information about their journey and not lose a competitive advantage to one another. For example, the operators might only want to share variables like the arrival time, and only want to share it with operators that have the approximately same arrival time.

To ensure the problem can be solved and that the message size does not run out of memory bounds, the memory-bounded version of DPOP is implemented. Because Memory-Bounded Distributed Pseudotree Optimization Protocol (MB-DPOP) requires an exponential number of messages when the inference width is above a certain threshold, an alternative way of formulating and solving the problem is considered using the Maximum Gain Messaging (MGM) algorithm and a formulation that includes how much vessel operators are willing to change their current rotation plan for a lower overall cost. With this formulation, when something changes in the problem definition, the entire calculation does not have to be done again because starting from the previous solution is possible.

Two simulated case studies in the port area of Rotterdam are done to performed the performance of the algorithms. The results show while MB-DPOP can guarantee some level of privacy, it does not scale very well and is thus not really useful in real life. While the algorithm yields a global solution and can guarantee a certain level of privacy, its execution time is in the order of hours, even for relatively small problem sizes. Conversely, the MGM algorithm with reformulation scales quite well and can possibly guarantee some level of privacy as well with extension. While the algorithm does not come up with a global solution, it can be a decent trade off between a level of privacy, and scalable algorithms that are centralized, such as a genetic algorithm.

Table of Contents

Preface

This is the final thesis report of my master thesis for the Delft Center for Systems and Control (DCSC) department of the Mechanical, Maritime and Materials Engineering (3mE) faculty over the course of 2020 till 2022 on using Distributed Constraint Optimization Problem (DCOP) algorithms to solve the VRPP. Being able to work on this long project was challenging, but has honed my skills both as a student, and as a person.

I would like to express my gratitude to my supervisor Prof.Dr.Ir B. De Schutter for always answering any questions I had, and pushing me to stay on track. During our biweekly meetings, you were always able to give valuable feedback, resulting in the birth of this thesis.

Next, I would like to thank my family for supporting me in pursuit of a degree. Thank you for putting up with me during stressful times. Thank you for letting me study abroad, choose my own path, and to encourage me to do what I love.

Lastly, I would like to thank my housemates from ds4. Thank you for providing support and distraction in tough times, and coffee and drinks in good times. Having lived with you has shaped me more than any other experience in my life.

Chapter 1

Introduction

This thesis has been produced in an attempt to increase the acceptability of distributed algorithms in inland waterway transport. Some problems are naturally distributed: when multiple parties are competing over a limited resource, some way of dividing the resource has to be implemented. It would be advantageous to come up with a good overall solution that is, in a way, fair. This would also allow to plan for a maximization of the capacity of a waterway, which could help reach the CO_2 emission reduction goals the world has set itself since transport over water has a low specific emission with 31 g $CO_2/\text{tonne} - \text{km}$ [2], with only transport over rail doing better. One way of doing such a thing is by centralizing all planning and operations. Many decision makers, however, are unwilling to share certain information about their plans. For example, in airport slot allocation, some airlines do not want competitors to find out which routes they plan to fly [3]. Similarly, inland water vessel operators can be unwilling to share information about their plans. Next to that, container shipment demands often change and new solutions have to be obatained. It would be unproductive to start from scratch again to come up with these solutions. Distributed algorithms can offer a solution to both these problems: when operators are unwilling to share certain information about their plans because they can guarantee a certain level of privacy, and some algorithms are able to use previously obtained results to come up with a new solution.

1-1 Problem Statement: Rotterdam Data

To reach the CO2 emission reduction goals in order to battle climate change, a lot of research and efforts go towards technologies that can lower carbon emissions. Because inland waterway transport has a low specific carbon emission, making sure a modal shift happens from road to waterway transport would be advantageous. However, currently, due to a lack of automation and communication, the full capacity of the waterways is not effectively used. A reasonable goal would be to minimize the rotation time (the total time spent in a port area). According to [4], [12], [16], the average rotation time in the port area of Rotterdam is 22.5 hours of which only 7.5 hours are used for loading and unloading, and only 62 % of the inland water

vessels leave the port area on time. It therefore is clear that there are still a lot of gains to be made. The problem statement leading the rest of this thesis will thus be: *How can distributed algorithms be made more adoptable such that they can be used in real-world scenarios in order to minimize the rotation time in port areas.*

1-2 Contributions

In this report, two main contributions are made to the literature. Firstly, whilst privacy options have been implemented with Distributed Pseudotree Optimization Protocol (DPOP) in [3], it has not been tested on a case study with many shared constraints. To do this, the memory-bounded variant of DPOP is implemented with privacy and tested on a larger scale problem with many shared constraints.

Secondly, the Maximum Gain Messaging (MGM) algorithm and problem formulation are extended with a dynamic utility, which allows for the problem to be dynamic whilst also giving the vessel operators a control parameter that indicates how willing they are to change their plans in exchange for a better overall solution. This way, both the dynamics of a free market and its advantages, as well as the advantages of coming up with a joint solution can be obtained.

1-3 Organization of the Report

Throughout the coming chapters, the given problem statement will be recast, reformulated, solved, and extended.

The report starts of by reformulating the problem statement as a Vehicle Rotaton Planning Problem (VRPP), which can be recast as a Distributed Constraint Optimization Problem (DCOP). After it is reformulated as a DCOP, multiple algorithms are listed to solve it. Among these are the MGM algorithm, which is an incomplete algorithm, and DPOP, which is a complete algorithm. DPOP is then extended with a parameter that bounds its memory usage, which is its main disadvantage.

Following the listing of the basic algorithms, they are extended in Chapter 3. The MGM algorithm is extended with variable utility and an unwillingness parameter whilst the memory-bounded version of DPOP is extended in a way that guarantees a certain level of privacy. Next to that, the problem formulation is extended such that water locks can be modeled as well.

After that, in Chapter 4, two case studies are done to compare the performance of the extended algorithms to each other and to other algorithms such as a genetic algorithm.

Finally, in Chapter 5, the conclusions of the report are given. Next to that, some recommendations are given as to where this research may be applied and what future points of research could be.

Chapter 2

Formulation of the Problem as a Vehicle Rotation Planning Problem

In this chapter, the problem introduced in Chapter 1 will be formulated as a Vehicle Rotaton Planning Problem (VRPP). This in turn allows us to cast it as a Distributed Constraint Optimization Problem (DCOP). There are multiple ways to solve the VRPP as a DCOP, such as the Maximum Gain Messaging (MGM) algorithm and the Distributed Pseudotree Optimization Protocol (DPOP). Both of these algorithms will be elaborated upon in this chapter.

2-1 Vehicle Rotation Planning Problem

With the goal of efficient ship deployment from Chapter 1 in mind, a more elaborate problem statement can be made, which is also done in [8]. This problem statement is called the VRPP and is widely used in the deployment of vehicles such as ships and trains [1]. It consists of deciding on the order in which terminals are visited, which is called the rotation plan, as well as the number of containers transported between terminals for each vessel whilst still satisfying the container transportation demand. The VRPP is a combinatorial optimization problem, as well as an integer problem. It is typically applied in a large port area, and with a planning horizon of several days. To find the most efficient vessel rotation plan, it is important to first define the efficient deployment of a fleet. Often, a cost function is used to minimize the total sojourn time[1]. Other cost functions are possible as well, e.g. fuel use, number of ships, or burdening of terminals. The reason we want to use the sojourn time is that the main goal is to make inland water transport competitive with road transport first, after which other cost functions can be implemented to drive down e.g. emissions. If the sojourn time is to be limited whilst the container demand is to be satisfied, then the weighted

[1]The sojourn time is the total time spent at the port. It includes the traveling time, the service time, and the waiting time of a vessel.

cost function for a single vessel is given as:

$$r_m = p_1 \left(\sum_{i=1}^{n} \sum_{j=1}^{n} z_{ij}^m (l_i + u_j) + \sum_{i=1}^{n} w_i^m + \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij}^m \tau_{ij}^m \right) + p_2 \left(\sum_{i=1}^{n} \sum_{j=1}^{n} (r_{ij}^m - z_{ij}^m) \right) \tag{2-1}$$

where i and j represent terminals, z_{ij}^m is the number of containers transported from terminal i to j by vessel m. The variable x_{ij}^m is a binary variable that indicates whether vessel m is traveling from terminal i to j. The variables l_i and u_i represent the loading and unloading time at terminal i respectively. The variable w_i denotes the waiting time at terminal i. Furthermore, τ_{ij}^m is the traveling time from terminal i to terminal j of vessel m, r_{ij}^m is the number of containers that need to be transported from terminal i to terminal j by vessel m. Finally, the weights p_1 and p_2 are used to make a trade-off between the sojourn time and containers not being delivered (inducing a so-called penalty time). In [8], a number of extra constraints are enforced for one vessel of which a list is shown below. Some of these constraints are based on assumptions, such as constraint 2, 3, and 4. Constraints 1 and 5 are added to ensure the solutions of the VRPP are sensible.

1. No containers should be transported unnecessarily, which means that the number of containers transported should be lower or equal to the required number of containers transported from one terminal to another terminal

2. Terminals are only visited once except for the entrance terminal

3. All terminals are connected.

4. The starting and end terminal are the same

5. There are no round-trips between two terminals

6. The container capacity of a ship is not exceeded

For multiple vessels, the objective becomes to minimize the sum of all these utility values:

$$\sum_{m=1}^{M} (r_m + r^{\mathrm{ia}_m}) \tag{2-2}$$

Here, r^m is the cost function of a single vessel (agent), and r^{ia_m} represents the so-called inter agent utility function which is a way to penalize overlap between the rotation plan of multiple vessels. In the context of a rotation plan to distribute containers around a port area this would be cost function with a high value when several agents have the same time slots at the same terminal. For example, when there are more vessels scheduled in slot than there is space.

2-2 Distributed Constraint Optimization Problem

In real life, there are multiple vessel operators (agents) making their own decisions. For that reason, it might be appropriate to model our VRPP in a way that reflects this behavior. A way to do that is to formulate the VRPP as a DCOP. Often, these DCOPs operate on a so-called constraint graph, where an edge between two nodes indicates a common constraint and a node reflects a variable. These variables are controlled by so-called agents. From here onward, the terms variable, node, and agent can be used interchangeably. The main idea behind the use of distributed constraint optimization problem algorithms, and more specifically those involving a pseudo tree arrangement [2] is that because of the single-layer tree structure, every branch is relatively independent of other nodes in different branches and can thus be searched in parallel. An example of such a constraint graph or pseudo tree graph can be seen in Figure 2-1.

There exist several algorithms to solve DCOP problems. These algorithms usually have one agent controlling one variable at a time, so that more virtual agents are present in the problem solution than there are real-life agents. The authors of [9] divide the algorithms into two categories: incomplete and complete algorithms. Complete algorithms are guaranteed to give the optimal solution if it exists, while incomplete algorithms use local optimization to find locally optimal solutions. Whilst it may be beneficial to know a solution is also the global solution, often, in real life, incomplete algorithms are used because of their scalability. Some examples of complete algorithms are the SyncBB algorithm [20] [5], DPOP [14], and its memory-bounded version, Memory-Bounded Distributed Pseudotree Optimization Protocol (MB-DPOP) [15]. Some examples of incomplete algorithms are the MGM [11] algorithm, the DSA algorithm [22] and the DBA algorithm [21].

2-2-1 Distributed Pseudotree Optimization Protocol

The DPOP procedure was initially proposed in [14]. To understand how it works, it is worth explaining the Distributed Tree (DTREE) [13], which was in its turn based on the sum-product algorithm [6]. As explained above, a DCOP can be viewed as a so-called constraint graph that connects all the variables that share a utility function. Since there is a general utility function to be maximized (or minimized in case of a cost function), the variables such as arrival time, departure time, and order in which vessels visit terminals are connected. The DTREE algorithm is a so-called utility propagation algorithm, which has UTIL messages propagating through the network asynchronously. These UTIL messages contain the utility values associated with all the combinations of values variables can hold. The algorithm is initiated by the leaf nodes (the nodes with only one neighbor) in the tree, after which the so-called $n-1$ rule is used to propagate messages through the tree network. The $n-1$ rule entails that if a node has n neighbors, it will only send out a message to its kth neighbor after having received $n-1$ messages. That is, it only sends a message once it has received messages from all its neighbors except one, which is its parent. At some point, the node will receive a message back from its nth neighbor. When that happens, the node has enough information to choose its own optimal value, and will send out $n-1$ messages to its other

[2] A pseudo tree arrangement of a graph G is a rooted tree with the same vertices as G and the property that adjacent vertices from the original graph fall in the same branch of tree.

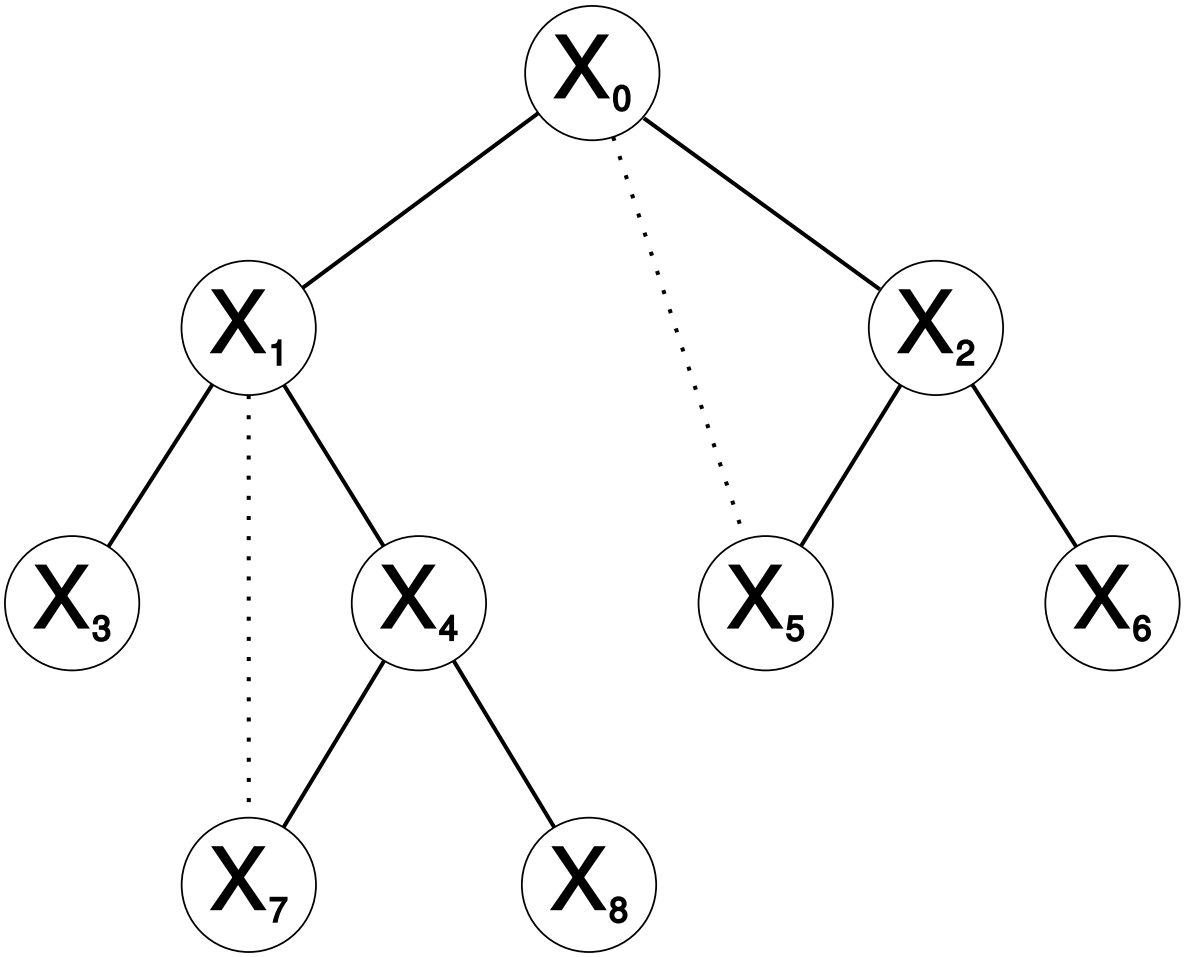

Figure 2-1: This figure illustrates the concept of a pseudo tree arrangement. A pseudo tree arrangement of a graph G is a rooted tree with the same vertices as G and the property that adjacent vertices from the original graph fall in the same branch of tree (e.g. X_0 and X_{11} in this figure). The solid lines are called edges and define the relationship between a child and a parent. The dotted lines are called backedges and define the relationship between pseudoparents and pseudochildren. The figure is based on [14].

neighbors, called its having received a message from that kth neighbor. This rule makes the tree dynamic because it has no root node, and allows for asynchronous processing of nodes. The algorithm requires a linear number of messages in the number of nodes of the tree. In these kinds of problems, where a tree-structured network is used to formulate a problem and polynomial-time algorithms such as the DTREE algorithm are used there are two sets: a set of agents X each responsible for a variable, and a set of agents A that are interested in the assignments made to these variables. For example, X_1 could represent the arrival time of a vessel, and A_1 would represent that vessel. Then, all the agents in A declare their relations R_i to the agents X_i concerned in those relations. Note again that the actual agents (vessels) are only used to set relations. They are not actively used in the optimization, since artificial agents are made for each separate decision variable.

In the DTREE algorithm, each agent X_i executes the following steps (asynchronously from

other agents):

1. Examine its own relations. That is, find out who the node is connected to through relations and hence who are its neighbors.

2. Determine whether it is a leaf in the constraint tree or not. That is, if it has a single neighbor. If X_i is a leaf node, it sends the UTIL value to its only neighbor.

3. Wait for incoming messages and respond to them. For example, a neighbor X_j of node X_i would send X_i a vector of all the optimal utilities that can be achieved for the subtree of X_i that contains X_j, and this for all the values X_i can take. The size of the message is thus (in a discrete case) equal to the $|\text{dom}(X_i)|$. Messages by agents are sent following the $k-1$ rule. At some point it will receive a message back from the kth neighbor, from which node X_i will be able to choose its optimal value and pass this message on to the other $k-1$ neighbors. This can only happen if the root node can choose an optimal value based on all incoming UTIL messages.

After these steps, the algorithm is finished for agent X_i, and the algorithm stops if all agents have finished the algorithm. For an example problem to which the algrorithm may be applied, refer to the pseudo tree containing solid edges in Figure 2-1 Here, the variables X_i are variables that can take on multiple values.

A major shortcoming of the DTREE algorithm is that it does not work on cyclic graphs (it only works on tree-shaped topologies). DPOP solves that issue by making an extension to arbitrary topologies using pseudo trees. Cyclic graphs can always be rearranged as pseudo trees. The easiest way to do this is by just doing a depth first search and then keep track of the relations by creating both direct relations such as children and parents conform the DTREE algorithm, and indirect relations such as pseudoparents and pseudochildren. These indirect relations signify that simply using direct relations would result in a cyclic graph and are indicated by dotted lines. For example, the parent of a parent of a node may also be the parent of that node itself (e.g. X_1, X_4, and X_7 in Figure 2-1).

DPOP starts by choosing a root node followed by a labeling protocol. This labeling protocol takes care of labeling all the parents, children, pseudoparents, and pseudochildren. Then, just like the DTREE algorithm, DPOP is initiated at the leaf nodes. In Figure 2-1, the leaf nodes are X_3, X_5, X_6, X_7, and X_8. Once the leaf nodes send UTIL messages towards their neighbors, the neighbors collect these UTIL messages in a table that contains all the possible combinations of values that its children and pseudochildren can take and their corresponding util value. It is here that the weakness of the DTREE algorithm and DPOP lies, since the biggest message size sent in the protocol scales exponentially with the highest induced width (the highest number of variables a certain variable has influence on). With small problems, as well as with problems that are very disconnected in nature, this is not a problem because some values can be inferred from the context and the dimension of the message can be collapsed to a smaller one again. Assuming everything goes as planned and no message is sent that is too large, the root node will eventually get a UTIL table out of which it can choose its optimal value. It will then communicate its value to its children, all the way back to the leaf nodes.

2-2-2 Memory-Bounded Distributed Pseudotree Optimization Protocol

Building on DPOP, MB-DPOP implements the feature of a tunable memory bound for the algorithm. The algorithm also uses a depth first search arrangement to perform search in parallel on independent branches, which it then combines. The algorithm is different with respect to DPOP because there is a memory bound which is controlled by a control parameter k (the maximum dimensionality of a message).

The algorithm behaves the same as the DPOP algorithm in areas with a low inferred width. In areas with width higher than k, clusters are formed with each cluster having a width lower or equal to k. The control parameter k specifies the maximal amount of so-called inference (the maximum message dimensionality). This could be based on the minimum amount of memory available at each node (i.e. the memory should be greater than d^k; this is because it can be proven that the message size is at most d^k in size). Here, d is a discrete, finite variable domain size of values, e.g. d contains the number of possible values for a given agent (variable). Subgraphs are identified where the width is higher than kk at that point clusters are made with cluster roots. Whilst MB-DPOP makes it possible to solve some problems by making the memory footprint manageable, it comes with a trade-off. Because it creates clusters in high-width areas, extra messages have to be sent to communicate between these clusters. This means that while the memory is bounded by d^ω, the number of messages sent in the procedure grows exponentially.

2-2-3 Maximum Gain Message Algorithm

The MGM [11] works by broadcasting a gain message to all its neighbors that represents the maximum change in its local utility if it is able to choose its own value based on the context. Note that each agent can for itself determine how much it values a change in its variable. It can be based on an (approximated) model, or on intuition/experience if a person is set in charge of this. If its gain message is larger than all the gain messages received by its neighbors, it is allowed to change its variable (act). It is clear that this algorithm will not always reach the global optimum but will reach a local optimum. Because of its ease of implementation and intuition, it allows for an adaptation of the cost function.

2-3 Conclusions

In this chapter the problem formulated in Chapter 1 was mathematically formulated as a VRPP. After formulating it that way, it was recast as a DCOP that can be solved by means of a distributed algorithm. Three such algorithms were elaborated upon: the DPOP as well as its memory-bounded version: MB-DPOP, and the MGM algorithm. DPOP is a complete algorithm and thus results in a globally optimal solution, but it can be computationally and memory expensive because it can require a memory size that exponentially scales with the highest induced width, or require an exponential number of messages in case the memory-bounded version is used with the formation of clusters. The MGM algorithm is easy to implement and intuitively leads to an optimum, but it is an incomplete algorithm and thus does not guarantee the global optimum.

Chapter 3

Possible Extensions to the Algorithms

As was mentioned in the previous chapter, there are some problems with distributed constraint optimization algorithms that need to be addressed. For example, whilst distributed algorithms such as Memory-Bounded Distributed Pseudotree Optimization Protocol (MB-DPOP) could be used for privacy-sensitive calculations, some changes still have to be made for this to happen. Furthermore, a port area is a very dynamic environment where requirements can quickly change. To support this, a way to incrementally come up with a good solution should also be elaborated upon. To solve these problems, some extensions to the algorithms can be made such that they could be better suited to real-life scenarios. In this chapter, the theoretical background for these extensions will be developed. Firstly, an extension to MB-DPOP is made to include privacy measures into the algorithm. These privacy measures can increase the willingness shippers have to share information and to take part in coming up with a collective solution. With these privacy measures, a minimum amount of privacy can be mathematically guaranteed. Secondly, an extension to the Maximum Gain Messaging (MGM) algorithm is made to include incremental improvements of the current solution. Since solving a Distributed Constraint Optimization Problem (DCOP) is often computationally expensive, it would be beneficial to be able to reuse a part of the current best solution instead of going back to the drawing table all over again. The extension to the MGM algorithm does exactly that: when the requirements change and a new solution has to be obtained, all vessel operators can indicate how willing they are to change their individual plans in exchange for an overall lower rotation time in port by means of a control parameter called the unwillingness. The resulting solution will look similar to the original one if there is only a small change, because largely the solution can be used again. In case there is a big change in the problem definition, the MGM algorithm works as usual. Because the cost of MGM algorithm is monotonically decreasing, it can easily be tweaked to make use of the old solution.

3-1 Privacy Extension

It would be beneficial to be able to guarantee a certain level of privacy towards the agents. However, let us first define privacy within a distributed optimization protocol context.

The authors in [3] define four types of privacy:

1. Agent privacy: no agent learns the identity of other agents. That is, no agent knows which variable is controlled by which agent.
2. Topology privacy: no agent learns the topological constructs such as constraints of agents it does not share a constraint with. That is, an agent does not know the identity of any other agent in the constraint graph except for its neighbors. It shares a constraint (i.e. has a neighbor) when it controls at least one variable in the constraint.
3. Constraint privacy: no agent learns the nature or content of a constraint of another agent. In an optimization setting, an agent cannot learn what value another agent attaches to a certain variable setting. Full constraint privacy is preserved when no such information is shared. Limited constraint privacy applies if the information that an agent can learn about such a combination of variable settings is bounded by a threshold ϵ.
4. Decision privacy: no agent learns the outcome of any decision that other agents make in the final solution. Full decision privacy is preserved if no agent can learn the values of any variable that it does not control in the final solution. Limited decision privacy occurs if no agent can learn the values of any variable that it does not control and that is not part of the neighborhood of a variable it controls.

Over the next few subsections, adaptations to MB-DPOP are given to include these privacy measures into the algorithm. As will be shown, some of the modes of privacy are already present in the algorithm, but some still require some work. Three phases are present in MB-DPOP: the LABEL phase, the UTIL phase, and the VALUE phase. All of these phases are possible candidates for increasing privacy.

3-1-1 LABEL phase

The LABEL phase in Distributed Pseudotree Optimization Protocol (DPOP) makes sure every node is labeled. After the LABEL phase every node knows who its parents, pseudo-parents, children, and pseudochildren are. In addition to the LABEL phase in DPOP, the LABEL phase in MB-DPOP also identifies clusters with an inferred width higher than a given value k, as well as cycle-cut nodes. Privacy can be implemented by performing some extra initialization. First, each agent creates a vector of random obfuscating keys for each constraint it encounters, after which it sends this vector to its neighbor that is also in that constraint. Its neighbor then does the same. After this, for each variable, the agent generates a codename as well as for each of the domain values of this variable, which it then shares with all its neighbors. When the initialization is finished, the agents anonymously elect a leader as the root of the Depth First Search (DFS) tree, which then starts the labeling phase as in normal DPOP.

3-1-2 UTIL phase

The UTIL phase works by sending UTIL messages starting from the leaves of the DFS tree, which were identified in the LABEL phase. Just like with MB-DPOP, the privacy-extended

algorithm also waits for the UTIL phases of all its children. It then deobfuscates these messages and combines them, as usual. After combining the messages, it then obfuscates its own UTIL message which it then sends to its parent.

3-1-3 VALUE phase

After the UTIL phase, the root node of the DFS tree can choose its optimal value. It then sends messages top-bottom towards the leaves. Every agent waits for the VALUE messages from its parent, after which the message is deobfuscated and its own optimal value can be calculated. It then obfuscates its own VALUE message, which it sends to its children.

3-1-4 Privacy Properties

By making a limited number of adaptations to MB-DPOP, some privacy guarantees for the different privacy modes can be given. Agent privacy is preserved because all agents are referred to by codenames, and codenames of variables are only communicated with agents that control a variable that shares constraints with the respective variables. Like agent privacy, topology privacy is also preserved because codenames are used. In case the constraint graph is cyclical, there is a backedge present from which a cyclical nature of the graph can be derived. Because the variable present in that backedge is the variable that can derive the cyclical nature of the graph itself, topology privacy is preserved. Limited constraint privacy is obtained, but the privacy that is leaked is necessary to inform neighbors.

An eavesdropping agent could theoretically derive a constraint variable, but that can be countered by including extra obfuscations to make the lost privacy arbitrarily small. Lastly, only limited decision privacy is obtained because variables learn what decisions their neighbors make because they use this information to make their own decisions. Note that this loss of privacy is necessary for the algorithm to work.

3-2 Incremental Improvements

Solving a DCOP is often time, resource, and energy intensive. Imagine having solved a DCOP but then something changes in the problem. Starting to calculate the solution from scratch would not be beneficial because it would require a lot of processing power, time, and maybe lead to a solution that is completely different from the current solution so that vessel operators will have to change their destination, which they would be unwilling to do unless there is a large benefit to it. To illustrate the idea of the implementation of willingness imagine 4 different phases in time:

- Phase 1: The cost equals a very high number and nothing is planned or done. To start the algorithm, random values are assigned to each variable. This just resembles an unsolved problem without any constraints added.
- Phase 2: Then, some constraints are added. For example: some containers may need to be delivered, which results in a cost value that penalizes containers not being delivered. The current cost, when no vessels are dispatched, is still very high.

- Phase 3: After the constraints are added, an attempt is made to dispatch vessels in order to decrease the total cost value. With the MGM algorithm, this cost would be monotonically decreasing. After a while, no better solution is found and a local optimum is reached.
- Phase 4: Something suddenly happens. For example, a higher container shipment demand occurs. Doing the complete calculation again would be unproductive because a solution similar to the current one but with a few tweaks will probably also be a good enough solution. We now propose an additional term in the cost calculation that represents the willingness of a certain vessel to change its route. This way, a control parameter is introduced that can change to what extent vessel operators are willing to change the current solution in exchange for an overall lower cost. The total cost value then becomes

$$\text{cost} = \text{sojourn time} + \sum_{m=1}^{n} \text{unwillingness}_{\text{m}} \tag{3-1}$$

  where n equals the number of vessels and the unwillingness is a positive value that can be calculated with

$$\text{unwillingness} = p_3(\hat{x} - x)^2 + p_4(\hat{z} - z)^2 \tag{3-2}$$

  where p_3 and p_4 are weighting factors, and $\hat{x}$ and $\hat{z}$ signify the vectors of the values of the currently best known solution. Note that this unwillingness is calculated per vessel, since x and z refer to a single vessel as per (2-1).
- Phase 5: If a vessel decides to change some of its values because the overall gain is large enough, the associated unwillingness cost goes away and is replaced by a new unwillingness such that we arrive back at Phase 4.

By tweaking the parameters p_3 and p_4, a vessel operator can indicate how willing it is to change their current plans in exchange for a better overall solution. In case the unwillingness value is too high, the vessel operator will not have their plan changed. However, because of this, any extra containers that can be shipped will also not be shipped by that operator because another operator might be more willing to change its plans in order to pick up certain containers, and thus get a competitive advantage.

3-3 Water Lock

To increase the willingness of vessel operators to adopt distributed algorithms such as MB-DPOP or MGM, it is beneficial to extend the problem formulation to water locks. With this extended problem formulation, the possibility to take a water lock can be implemented instead of only having the possibilities to go from one terminal to another via open waterways. The extension here is merely proposed in theory, and not actually tested in the results section like the previous two extensions. A water lock can be implemented by including it as an extra terminal that has no loading or unloading slots, but does have a certain time attached to it. This time refers to the time it takes to pass the water lock. Next to the variables x_{ij}^m and z_{ij}^m, an extra variable y_{ij}^m is then introduced. This variable can take the values 0 or 1, which indicates whether a water lock is taken or not. In the case the water lock is not taken, an

other, possible longer route has is taken. Note that the water lock is not just implemented as another terminal, because then the loading and unloading times of containers transported to the water lock would have to be taken into account. The use of this extra variable is best illustrated by means of an example. Say there is a vessel that needs to go from a terminal with index 1 to a terminal with index 3, with the travel time between the two terminals being 4 hours. If one takes the direct route from terminal 1 to terminal 3, this is indicated by setting $x_{13} = 1$. Now, let us introduce a water lock that can be used to get from terminal 1 to terminal 3 via an alternative route; we note this water lock as another terminal in the distance matrix, but there comes a transfer time cost associated with it in the total travel rotation time equation. The variable y_{ij} can then be used to signify whether a water lock will be used or not. If it equals 1, that signifies the route with the water lock is taken. In this case, $y_{13} = 1$, the travel length t from terminal 1 to terminal 3 then becomes $t = y_{13}(\tau_{12} + \tau_{23} - \tau_{13}) + \tau_{13}$. Note that when $y_{13} = 0$, the travel time is just the normal travel time from through open water. The cost function r_m then changes to

$$\begin{aligned} r_m = p_1 &\left(\sum_{i=1}^{n} \sum_{j=1}^{n} z_{ij}^m (l_i + u_j) + \sum_{i=1}^{n} w_i^m + \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij}^m (y_{ij}^m (\tau_{is}^m + \tau_{ks}^m - \tau_{ij}) + \tau_{ij}^m) \right) \\ &+ p_2 \left(\sum_{i=1}^{n} \sum_{j=1}^{n} (r_{ij}^m - z_{ij}^m) \right) \end{aligned} \tag{3-3}$$

Where s is the travel time matrix index corresponding to the water lock. This way, y_{ij}^m becomes a control parameter like x_{ij}^m and z_{ij}^m.

3-4 Conclusions

In this chapter extensions to MB-DPOP and the MGM algorithm, as well as the way of modeling were given. The goal of these extensions is to increase the willingness of vessel operators to adopt a common scheduling system that can help with a better planning. In the first section it was shown that privacy can be added to MB-DPOP to ensure some privacy guarantees for the vessel operators. In the second section, a control parameter was introduced that gives shippers the opportunity to weigh in how willing they are to change their current plans in exchange for a slightly better deal both for themselves and for the overall solution. In the last section, the problem formulation is extended to water locks as well so that the algorithm can be used in a large port area with water locks as well.

Chapter 4

Case Studies: Simple and Complex Container Moving in the Port of Rotterdam

In Chapter 2, the theoretical background has been given to Distributed Constraint Optimization Problem (DCOP) algorithms. After that, in Chapter 3, some possible extensions were proposed to increase the acceptability of these distributed algorithms in industry. In this chapter, these extensions will be tested on some case studies and compared to a base case. First, two fictional case studies will be presented.

The first one is a simple case study in the port area of Rotterdam with only one vessel transporting containers and is covered in Section 4-2. This simple case study is slightly adapted depending on the algorithm it is used for. The base case study is used for the analysis of the Memory-Bounded Distributed Pseudotree Optimization Protocol (MB-DPOP) and the Maximum Gain Messaging (MGM) algorithm, and to compare it to a genetic algorithm. The reuse of a previous solution in the MGM algorithm is tested on a slightly modified version of the simple case study. Since only one vessel is present, it makes no sense testing the unwillingness in the simple case study.

The second case study is a more elaborate case study with two vessels able to transport containers and a higher container demand and will be covered in Section 4-3. Here again, the base case study is used for the analysis of the MB-DPOP and the MGM algorithm, and to compare it to a genetic algorithm. To test the reuse of a previous solution, as well as the unwillingness, two slightly modified case studies are used. That is, the problem definition that will be fed into the algorithm will differ slightly.

Both case studies have trivial solutions, such that the performance of the algorithms can be easily analyzed. Since the MB-DPOP is a complete algorithm and the MGM algorithm is an incomplete algorithm, they will be covered separately in the results sections of the case studies, after which they are both compared to the genetic algorithm at the same time. All the results shown in this chapter were obtained using a laptop running Linux with an i5-1240P at

a clock frequency of 4.4 GHz and a total memory of 64 GB. Lastly, some concluding remarks are given in Section 4-4.

4-1 Determining the Memory-Bounding Parameter k

The MB-DPOP requires us to choose a memory-bounding parameter k. This parameter limits the maximum message size to d^k, where d is the maximum finite domain size of a variable. Since the variable with the maximum domain size is z_{ij}^m, that variable is chosen to calculate the bounding parameter k. Given that the maximum number of containers that can be transported by a vessel is 96 and the minimum is 0, that gives us 97 discrete values. With a 64 bit size for each of these numbers, the corresponding maximum message size is bounded by $(64 \times 97)^k$, which has to be lower than the maximum amount of memory an agent has available. In this case, that is 64 GB of RAM, or 512×10^9 bits. Plugging in the results yields a bounding parameter $k = 3$, which is the bounding parameter used in the sections that follow. Note that this is quite a low bounding parameter, which follows from the fact that some of the variables used are not binary, but integers which can take on a lot of different values. This results in a faster growing exponential than would be the case with variables with a lower domain.

4-2 Simple Case Study

In this first case study, several terminals which are all in the general Rotterdam Port area are included. These terminals can be seen in Table 4-1.

Terminal	Index
Rotterdam World Gateway	1
Rotterdam Container Terminal	2
Broekman Distriport	3
CTT Rotterdam	4

Table 4-1: The terminals chosen for the simple case study.

The location of the terminals with respect to each other can be seen in Figure 4-1. With these terminals and indices, a distance matrix can be created that shows the distance from terminal i to terminal j. These distances can be seen in Table 4-2. Note that this distance matrix is symmetric.

For the simple case study, one vessel is chosen as an example. This vessel was picked randomly from marinetraffic.com, but because of privacy reasons the name and ENI number of the vessel are not shared. The vessel has a capacity of 96 TEU and its average speed was estimated from historical data to be 7.2 kts, or 13.3 km/h.

This average speed can be used to estimate an average traveling time from terminal i to terminal j. This is represented by the matrix T where each element τ_{ij} represents the traveling time from terminal i to terminal j in hours. The traveling times can be seen in Table 4-3.

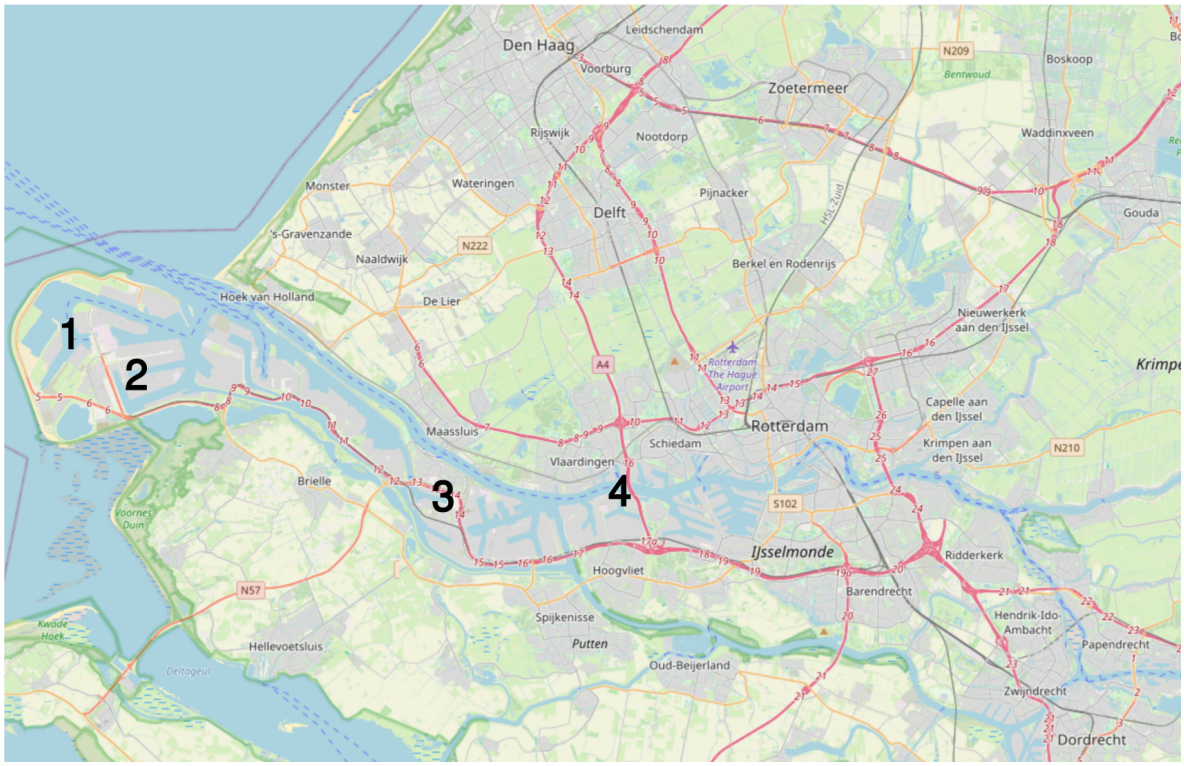

Figure 4-1: The terminals chosen for the simple case study, the numbers on the map denote the indices of the terminals. © OpenStreetMap Contributers

Now that the fixed parameters of the problem have been set, a container shipment demand can be constructed. An example container shipment demand could for example be:

- 5 containers from Rotterdam World Gateway to Rotterdam Container Terminal
- 10 containers from Rotterdam World Gateway to Broekman Distriport
- 5 containers from Broekman Distriport to CTT Rotterdam
- 5 empty containers from CTT Rotterdam to Rotterdam World Gateway

From this demand, a so-called demand matrix R can be calculated. Here, each element r_{ij} with indices ij represents the containers shipment demand from terminal i to terminal j. The

	j			
i	1	2	3	4
1	0	16.2	37.5	34.5
2	16.2	0	34.76	32.16
3	37.5	34.76	0	2.6
4	34.5	32.16	2.6	0

Table 4-2: The distances from a terminal with index i to a terminal with index j in km.

	j			
i	1	2	3	4
1	0	1.22	2.82	2.59
2	1.22	0	2.61	2.42
3	2.82	2.61	0	0.20
4	2.59	2.42	0.20	0

Table 4-3: The traveling time from a terminal with index i to a terminal with index j in hours.

demand matrix R is then given by:

$$R = \begin{bmatrix} 0 & 5 & 10 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 \\ 5 & 0 & 0 & 0 \end{bmatrix} \tag{4-1}$$

Note that conversely to the other matrices, this matrix is not symmetric due to the fact that containers do not necessarily have to be transported back from a certain terminal to another one.

The cost function r^mhat will have to be minimized is the one given in (2-1), but for the sake of clarity it is repeated here:

$$r^m = p_1 \left(\sum_{i=1}^{n} \sum_{j=1}^{n} z_{ij}(l_i + u_j) + \sum_{i=1}^{n} w_i + \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij}\tau_{ij} \right) + p_2 \sum_{i=1}^{n} \sum_{j=1}^{n} (r_{ij} - z_{ij}) \tag{4-2}$$

The waiting time w_i is taken to be 0 in this case study.

Here, the control variables for which the objective function is minimized are z_{ij}^m being a variable that represents how many containers are transported from terminal i to terminal j by vessel m and a binary variable x_{ij}^m that represents whether a vessel m is moving from terminal i to terminal j respectively. The variables l_i and u_j represent the loading and unloading time at terminals i and j respectively, which are assumed to be 1 min [7]. Furthermore, r_{ij} and τ_{ij} are read from the matrices R and T respectively. The weighting parameters p_1 and p_2 allow finding a balance between minimizing the sojourn time and making sure all containers are delivered. For example, if p_2 is taken very large, the problem values satisfying container shipment demand.

4-2-1 Results of the Simple Case Study

After the simple case study is defined, the extended algorithms can be tested. For the MB-DPOP algorithm, it measured how long it takes to come up with a solution. Since it is a complete algorithm, the solution found is also a global optimum. The unextended version of MB-DPOP takes 25,092 seconds to come up with the complete solution. The privacy extended version of this algorithm takes 26,773 seconds to finish. That means that whilst there is a relatively small overhead, it is still there. To come up with the solution, a total of 143,307 messages were sent.

To analyze the results of MGM algorithm and its extension, it is helpful to plot them together and look at what kind of impact the extension has. Let us imagine a case where, after an optimal solution of the original problem was found, a parameter in the problem formulation changes. In this case, let us imagine that the container demand from terminal 1 to terminal 2 changes from 5 to 10 containers. The results of such a simulation can be seen in Figure 4-2. The blue line signifies how the initial solution was found, after about 15 iterations, it arrives at a stable minimum. Because the case study is trivial, it can be checked that this is indeed the global optimum. Now, something such as the container demand changes. It might seem like a decent idea to just repeat the previous process. However, it can be beneficial to take into account the currently best known solution. Those two cases are signified by the orange and green line, respectively. From the figure, it can be concluded that there is a much faster convergence towards the optimal solution when the previous solutions are used in the process. The original MGM algorithm approached its optimum after 20 seconds of runtime. The MGM algorithm that used the previously obtained values obtained its optimum after 2.3 seconds of runtime.

4-2-2 Genetic Algorithm

Both the MB-DPOP and the MGM algorithm will be compared to a base case algorithm. In this case, a genetic algorithm was chosen. The parameters for this algorithm are elaborated upon in this subsection. All parameters for the problem formulation are taken the same as with the simple case study given earlier.

For the genetic algorithm to be implemented, we need a way to model the possible solutions and a way to determine the fitness of each algorithm. Since the variables x_{ij}^m and z_{ij}^m always come in pairs, it makes sense to pair them up per subscript. Looking at their domains, together they can be represented as one byte. An example representation of an $x_{ij}^m - z_{ij}^m$ pair is $x_{ij}|z_{ij} = 1|0000011$ where a vessel travels from terminal i to terminal j and takes three containers from i to j.

Gluing all the encoded pairs together gives the total solution. In total there are n^2 $x_{ij}^m - z_{ij}^m$ pairs, where n is the number of terminals which means that each solution can be encoded in n^2 byte. The fitness of said solution can then be calculated with the cost function when the pairs are decoded and plugged into (4-2).

Now that the encoding and cost function have been constructed, the genetic algorithm can be implemented. There are multiple choices that can be made to come to a solution, in this report it is chosen to vary the selection procedure of the parents and the number of crossover points; all other parameters are chosen to be constant with respect to the tested variation. All variations of the genetic algorithm below have five children per couple of parents, a population size of 200, a time span of 50 generations, and a mutation chance of $\frac{1}{8\times n\times n}$. Here, n is the number of terminals; the mutation rate is chosen such that on average there is exactly one mutation per parent, which corresponds to a chance of 0.8 % bit mutation chance for the simple case study. All variations are run ten times.

For the selection of the parents, two variations were implemented. These variations are the tournament selection [19], and the roulette wheel selection [10]. The tournament selection works by choosing a number of contestants (in this report this number is taken to be 10) from the population randomly, the two winners from that tournament are then chosen as parents

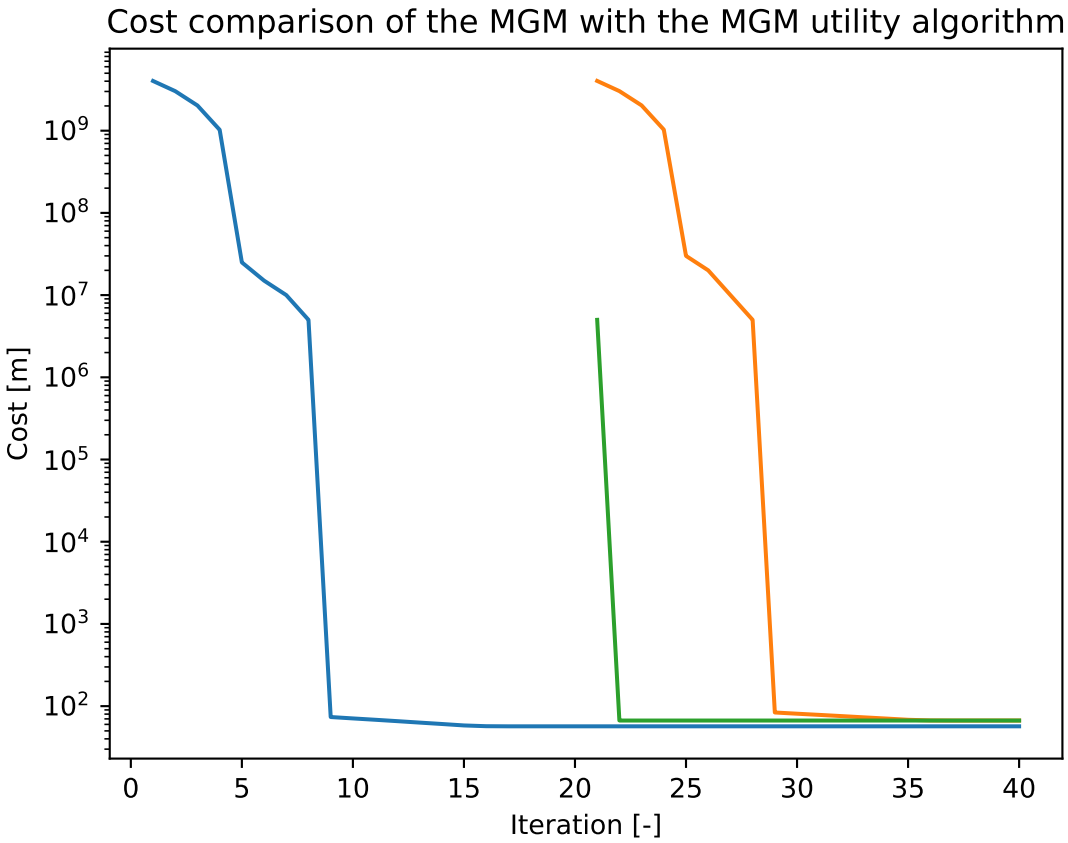

Figure 4-2: This figure shows how the cost function progresses throughout different iterations of the MGM algorithm. Three different lines are shown: two normal MGM cost progressions, and a cost progression that takes into account the current best solution. The blue and orange lines correspond to MGM ran with randomized starting values, the green line corresponds to reusing the solution obtained at iteration 20 as the starting values for MGM

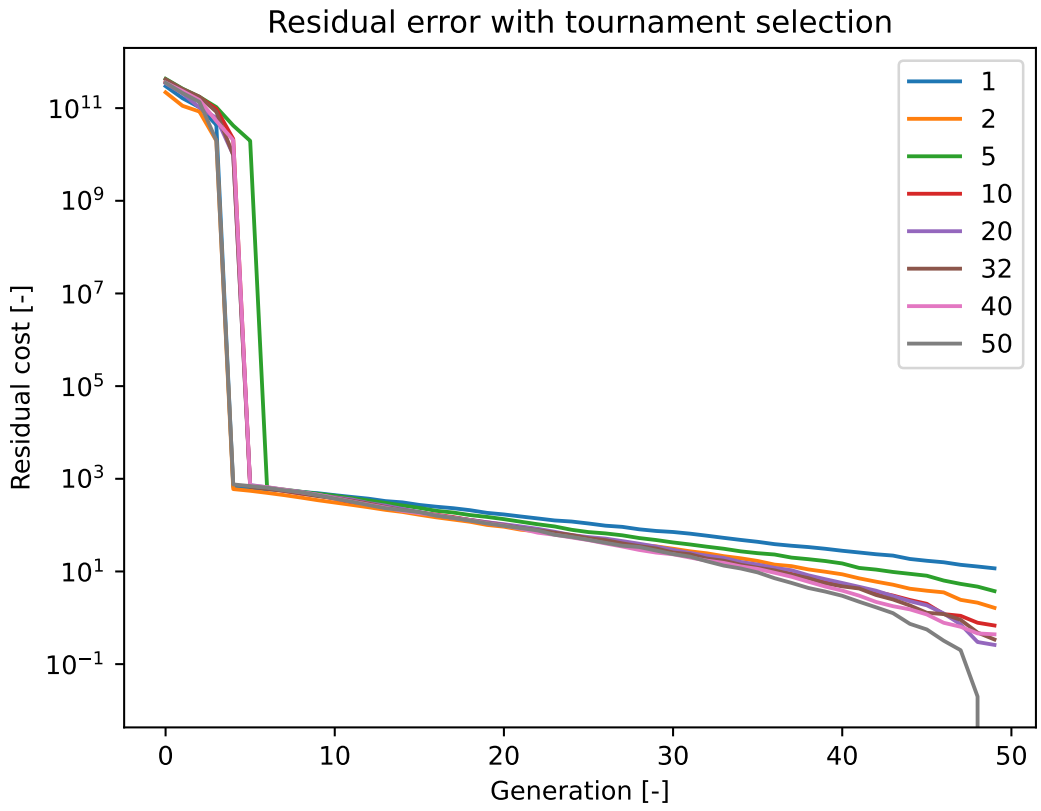

Figure 4-3: The residual error of the genetic algorithm with tournament selection. The different lines represent the varying number of crossovers.

for their 5 children. This process is repeated until the population is filled again. The roulette wheel selection works by choosing parents from the population with a given probability. The probability to choose a parent from a population is directly proportional to its fitness; the higher its fitness, the higher the chance it is being chosen. For the purpose of the roulette wheel selection, the cost of each parent is inverted.

The other thing that will be varied is the number of crossovers. The crossover type is chosen to be a k-point crossover [18] at random places, where k is varied. The variations are chosen from the discrete list 1, 2, 5, 10, 20, 32, 40, 50. The reason why so many were chosen in such a large range is to see what the effect is on the rate of convergence and the final solution of the genetic algorithms. The number of crossovers 32 was chosen because it exactly equals one crossover per variable on average.

The averages for all converging iterations were obtained to plot Figure 4-3 and Figure 4-4. From Figure 4-3, it can be concluded that for tournament selection, an increase in the number of crossovers leads to a faster convergence up to a certain point. An even further increase beyond this point (32) leads to a deterioration of the convergence as can be seen in the Figure 4-3. This deterioration is also present and more visible in Figure 4-4.

Both the tournament selection and the roulette wheel selection were able to get close to

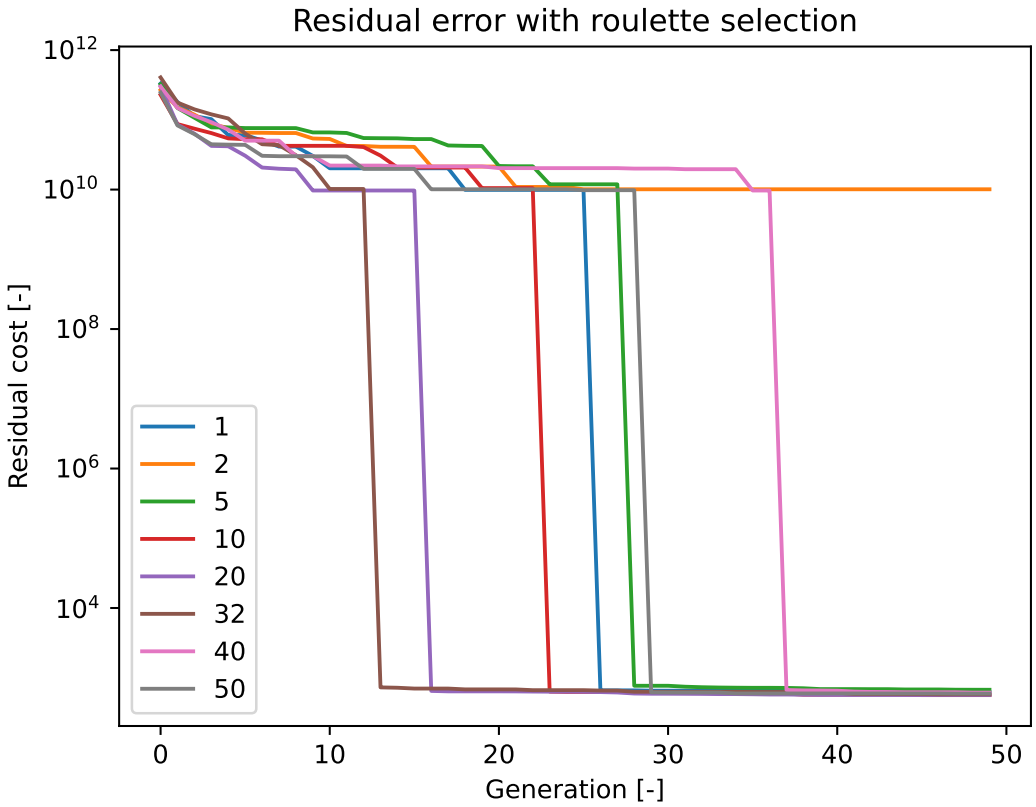

Figure 4-4: The residual error of the genetic algorithm with roulette selection. The different lines represent the varying number of crossovers.

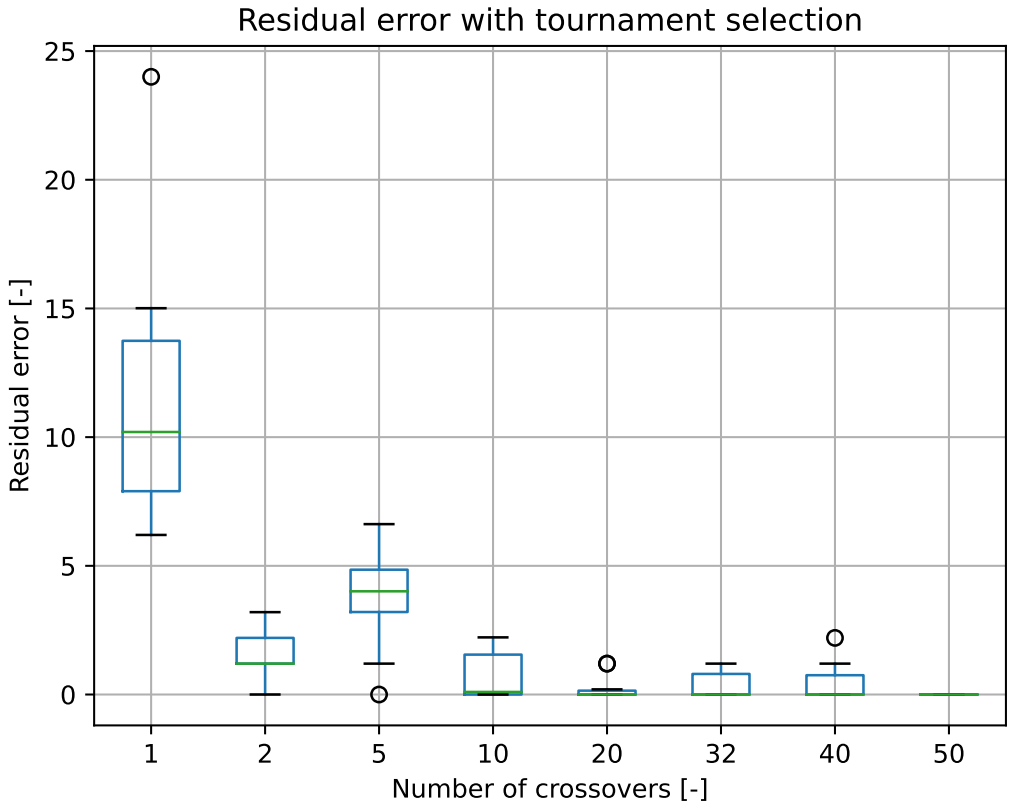

Figure 4-5: Box plots of the residual error after 50 generations with tournament selection. Each plot corresponds to a different number of crossovers.

the known optimum within 50 generations. For each of the ten iterations per variation, the residual error of each best member at the end of the 50 generations was logged and used to make the box plot shown in Figure 4-5. Whilst the genetic variation with roulette wheel selection often came close to the actual solution, it often stops at a certain point. It is theorized that the reason for this is that the roulette wheel favors local minima so much that there is no evolution anymore, as is also the case in [17]. This is also the reason the box plot for the roulette selection was omitted, because it sometimes results in high-cost outliers, skewing the plot. A possible way to prevent this would be to use stochastic universal sampling. The tournament selection variation has almost always a negligible residual error from ten crossovers onward and has a small standard deviation comparing to the roulette wheel selection method.

From the residual error plots and the residual error box plot, it can clearly be seen that the tournament selection variation is the better choice. Whilst the roulette wheel selection method is simple to implement, it introduces stochastic noise and tends to be stuck at local minima.

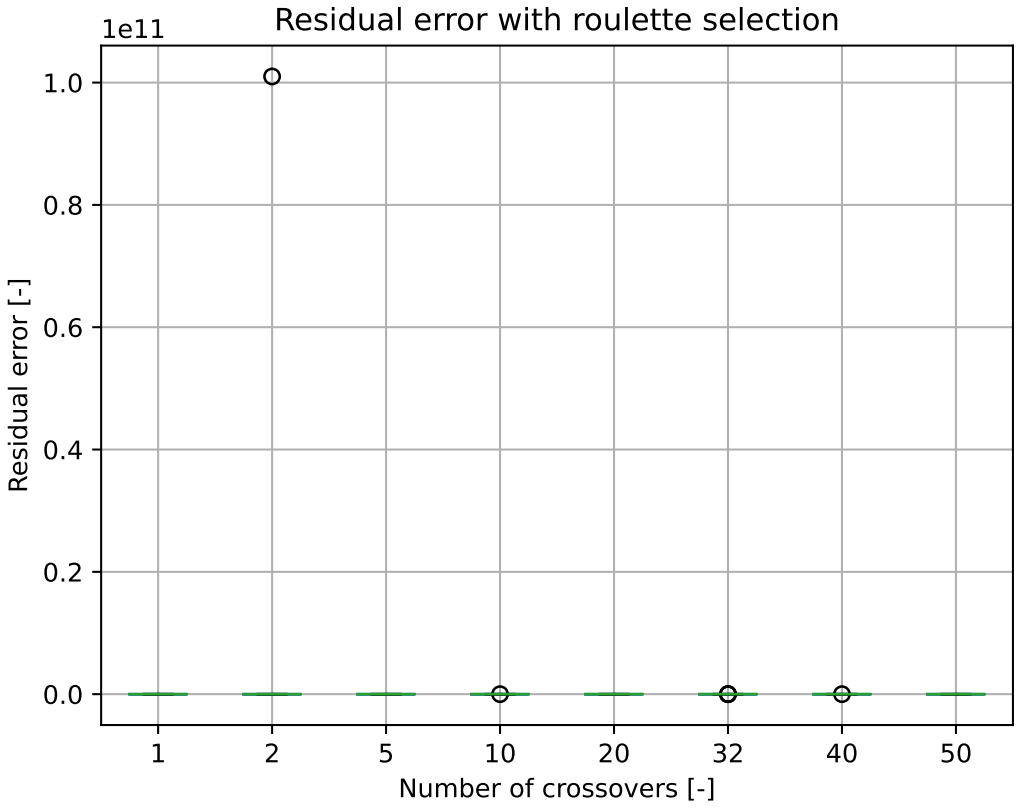

Figure 4-6: Box plot of the residual error after 50 generations with roulette selection. Each plot corresponds to a different number of crossovers. Note that the scale is messed up because of an outlier.

Algorithm	Execution Time [s]	Optimum	Privacy
MB-DPOP	25,092	global	no
MB-DPOP (P)	26,773	global	some guaranteed
MGM	20.3	local	some possible
Genetic algorithm	45	local	no

Table 4-4: This table shows a comparison of the algorithms for the simple case study.

4-2-3 Comparison with Other Algorithms

In this subsection, the extended MB-DPOP and MGM algorithms will be compared to a genetic algorithm. Comparing the genetic algorithm to extended MB-DPOP, the genetic algorithm comes up with a solution significantly faster. Compared to MGM, it comes up with a solution in about the same time range. Whilst the solution time is better, the solution is not guaranteed to be a global solution with both the genetic algorithm and the MGM algorithm. The MB-DPOP algorithm does guarantee a global solution, and can guarantee a certain level of privacy, but this comes at the cost of an exponential number of messages when the inferred width of the constraint graph is too high. Looking at the solving time versus optimality and privacy trade-off between MB-DPOP and the genetic algorithm, the extended MGM algorithm lies somewhere in the middle. While it can not guarantee a global solution, it can be tweaked to include some measures of privacy, and it makes problems still very tractable to solve. See Table 4-4 for an overview.

4-3 Complex Case Study

Now that the simple case has been presented, a more complex case can be considered as well. Like the simple case study, this study too is in the port area of Rotterdam, with an extra inland terminal at a large distance and the waiting time w_i will be taken to be 0. The study is made more complex by adding a terminal located far inland, having a higher shipment demand and especially by introducing a second vessel, to test the unwillingness parameter. The terminals chosen for this study and their respective indices can be seen in Table 4-5; their locations with respect to each other can be seen in Figure 4-7. Note that the only difference with regard to the terminals is an extra terminal farther inland.

Terminal	Index
Rotterdam World Gateway	1
Rotterdam Container Terminal	2
Broekman Distriport	3
CTT Rotterdam	4
Wijk bij Duurstede	5

Table 4-5: The terminals chosen for the complex case study.

To force multiple ships to be used, a higher container demand is simulated. This container

Figure 4-7: The terminals chosen for the complex case study, the numbers on the map denote the indices of the terminals. © OpenStreetMap Contributers

demand matrix R is given by

$$R = \begin{bmatrix} 0 & 50 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 \\ 50 & 0 & 0 & 0 & 50 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4-3}$$

The traveling times are given by Table 4-6.

	j				
i	1	2	3	4	5
1	0	1.22	2.82	2.59	8.51
2	1.22	0	2.61	2.42	8.34
3	2.82	2.61	0	0.20	6.12
4	2.59	2.42	0.20	0	5.92
5	8.51	8.34	6.12	5.92	0

Table 4-6: The traveling time from a terminal with index i to a terminal with index j in hours.

4-3-1 Results of the Complex Case Study

After the simple case study is defined, the extended algorithms can be tested. Just like with the simple case study, for the MB-DPOP algorithm it measured how long it takes to come up

with a solution. The unextended version of MB-DPOP takes 39,206 seconds to come up with the complete solution. The privacy extended version of this algorithm takes 43,597 seconds to finish. Note again here that there is a notable overhead, and that even with a small increase in problem difficulty, the computation time almost doubles. To come up with the solution, a grand total of 283,052 messages were sent.

Mirroring the simple case study, here to the results of the MGM algorithm as well as its extension are plotted together to analyze what the impact of the extension is. The change in the base case study is taken a bit differently since the original case study is also a bit different. To test the reuse of previous results for the MGM algorithm, the container shipment demand from terminal 4 to terminal 5 is changed from 50 containers to 60 containers. The results of such a simulation can be seen in Figure 4-8. The blue line corresponds to how the initial solution was found. Again, after about 15 iterations, it arrives at a relatively stable minimum. Even though the cost still diminishes after the 15th iteration, the drop in cost per iteration becomes a lot less. When the previous results are fed into the MGM algorithm, a quasi-instant convergence to the optimum is obtained. From the figure, it can be concluded that there is a much faster convergence towards the optimal solution when the previous solutions are used in the process.

Lastly, the unwillingness is tested. Because two different vessels are present in the case study, and they start from the same depot, the MGM algorithm yields the result that gives the least sojourn time. This result only uses one vessel. When an additional container demand is added from terminal 4 to terminal 5, the MGM algorithm again results in a solution that only utilizes one vessel. In practice, this vessel may not be willing to pick up the extra 10 containers because it may interfere with their already made plans. This is not ideal, because we want the best solution to the Vehicle Rotaton Planning Problem (VRPP) overall. To solve this problem, the first vessel can choose a certain unwillingness value to indicate how willing it is to change its plans by choosing two weighting parameters p_3 and p_4. For example, the vessel operator might choose $p_3 = p_4 = 100$. In this example, the operators choose their weighting values every 20 iterations, but this can also be done at any other number of iterations. That way, the MGM algorithm can be tricked into coming up with a solution that better suits the needs of the operators. The result of vessel operator 1 choosing the weighting parameters p_3 and p_4 can be seen in red in Figure 4-8. Note that the cost on which it stabilizes is significantly higher than the other graphs, this is because there is an extra (high) unwillingness added by a vessel operator, which we still need to subtract to come at the eventual cost value. By choosing lower values for p_3 and p_4, at some point a tip-over point will be reached again where the first operator would be willing to change its plans, resulting in the overall lowest sojourn time.

This works as expected, and could even be used with multiple vessel operators indicating their own different costs.

4-3-2 Genetic Algorithm

Just like with the simple case, the MB-DPOP and the MGM algorithm will be compared to a genetic algorithm. The parameters for the genetic algorithm are chosen to be the same as with the simple case study. Mirroring the simple case study, here to the genetic algorithm is analyzed to try and pick the best parameters. The plots and figures look similarly to the

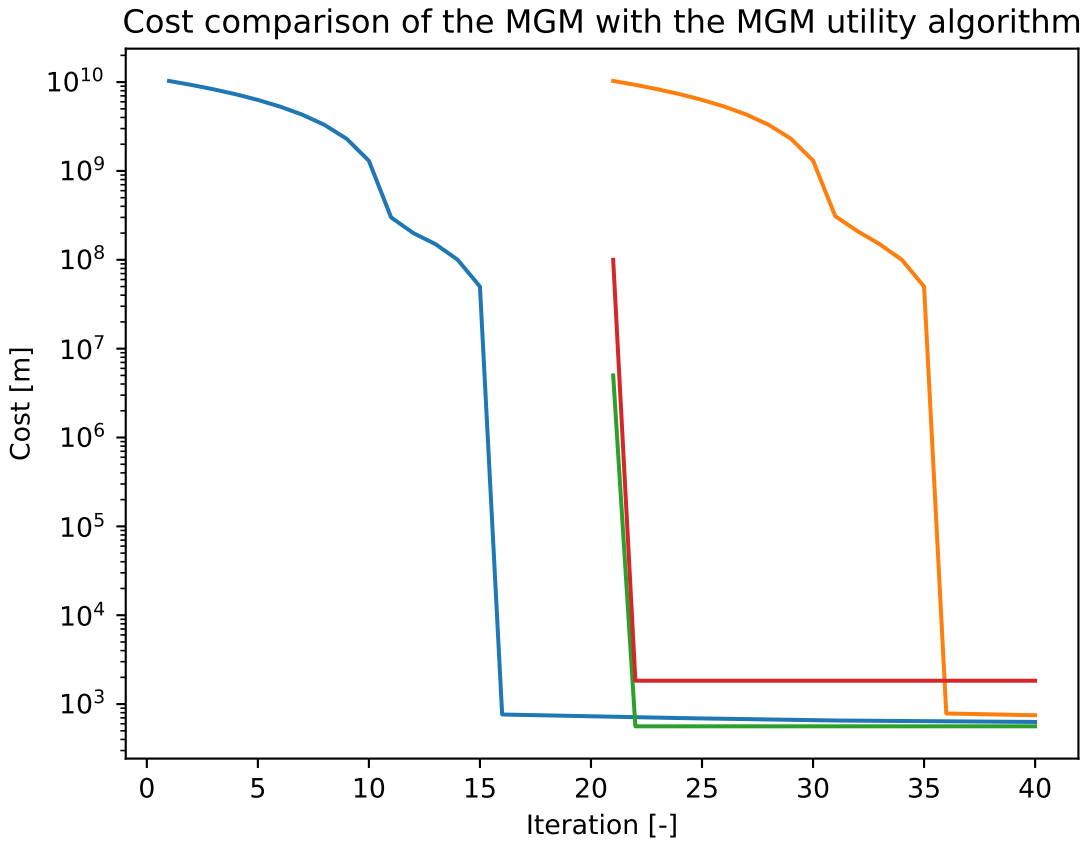

Figure 4-8: This figure shows how the cost function progresses throughout different iterations of the MGM algorithm. Three different lines are shown: two normal MGM cost progressions, and a cost progression that takes into account the current best solution. The blue and orange line correspond to two different initial starts of the MGM algorithm. The green line corresponds to an MGM algorithm using the previously obtained results (at iteration 20) as starting values.

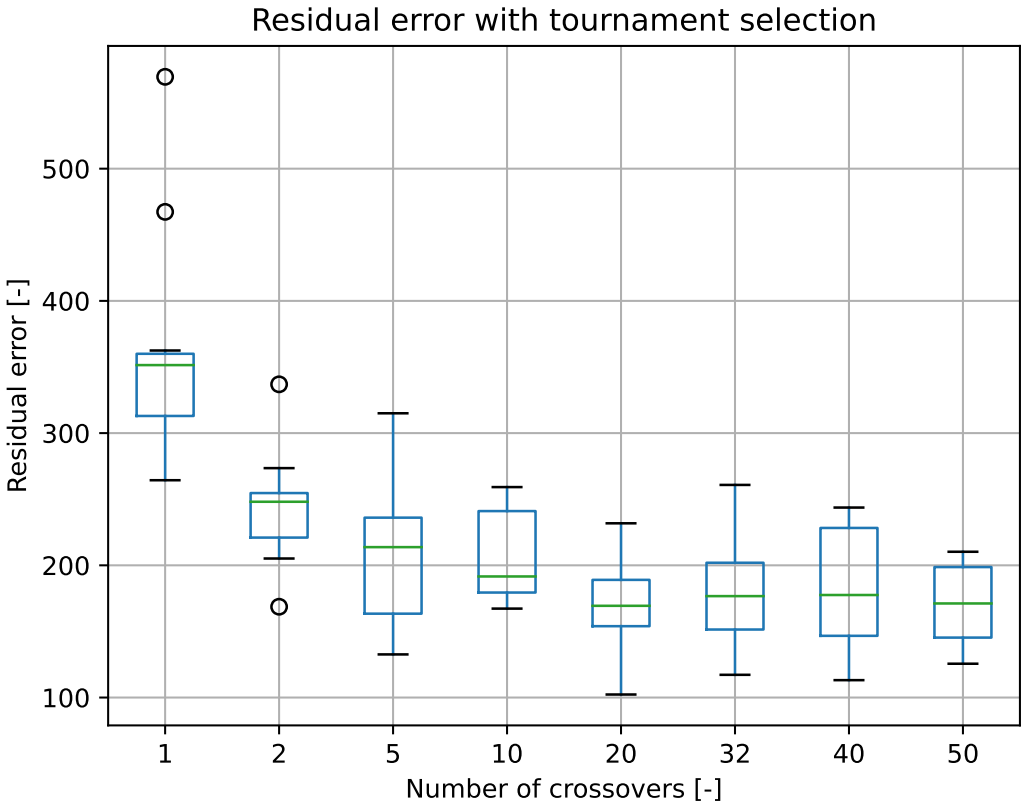

Figure 4-9: Box plots of the residual error after 50 generations with tournament selection. Each plot corresponds to a different number of crossovers. Note that the residual error here is higher than was the case with the simple case study. It is hypothesized that this is because 50 iterations might not be enough to fully converge to a local optimum given the larger solution space. This hypothesis is backed up by the fact that the residual error has not stabilized yet at iteration 50 in Figure 4-10.

ones in the simple case study, so not all of them are repeated here. In this case too, the tournament selection comes out on top, of which a box plot can be seen in Figure 4-9. The residual error plot can be seen in Figure 4-10.

4-3-3 Comparison with other Algorithms

Comparing the genetic algorithm to the extended MB-DPOP, the genetic algorithm comes up with a solution significantly faster. Compared to MGM, the genetic algorithm comes up with a solution in about the same time range. Whilst the solution time is faster, the solution is not guaranteed to be a global solution with both the genetic algorithm and the MGM algorithm. The MB-DPOP algorithm does guarantee a global solution, and can guarantee a certain level of privacy, but at the cost of an exponential number of messages when the inferred width of the constraint graph is too high. Looking at the solving time versus optimality and privacy trade-off between MB-DPOP and the genetic algorithm, the extended MGM algorithm lies

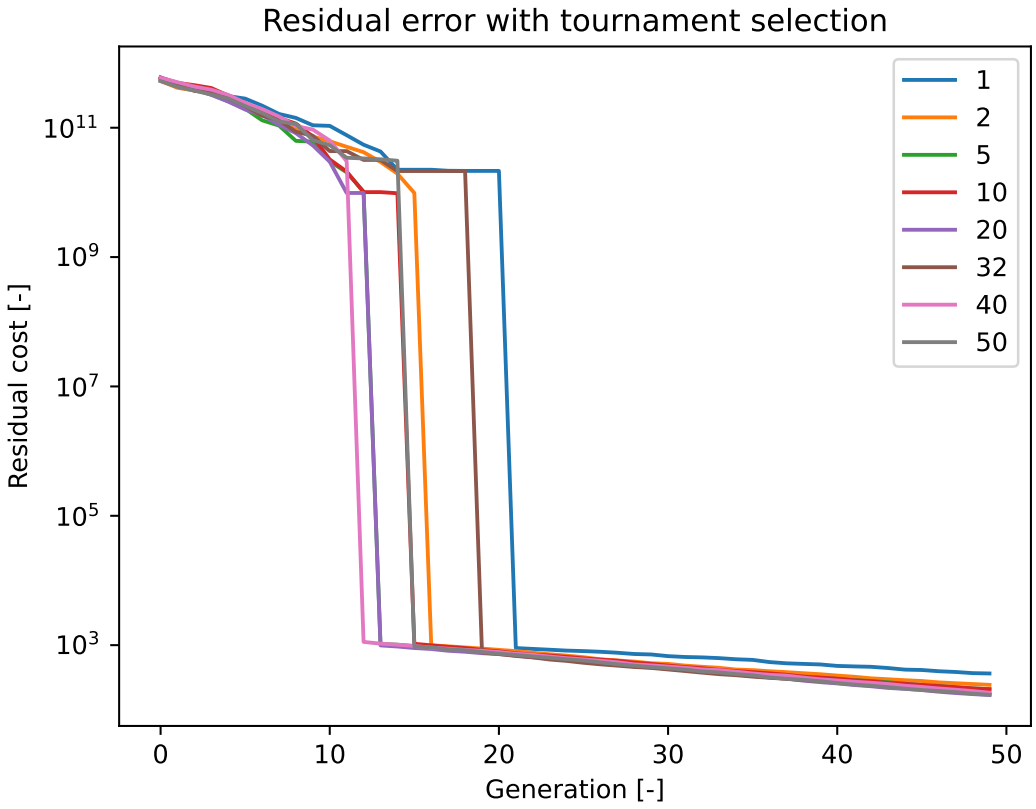

Figure 4-10: Residual error with tournament selection with different numbers of crossovers.

somewhere in the middle. While it can not guarantee a global solution, it can be tweaked to include some measures of privacy, and it makes problems still very tractable to solve.

See Table 4-7 for an overview.

Algorithm	Execution Time [s]	Optimum	Privacy
MB-DPOP	39,206	global	no
MB-DPOP (P)	43,597	global	some guaranteed
MGM	42.3	local	some possible
Genetic algorithm	45	local	no

Table 4-7: This table shows a comparison of the algorithms for the complex case study.

4-4 Conclusions

In this chapter, two case studies were done to evaluate the performance of the MB-DPOP algorithm and the MGM algorithm while solving the VRPP. Even though MB-DPOP has a control parameter that bounds the maximum size of messages sent, it only gains that advantage by making a trade-off in the number of messages sent. When agents share numerous constraints, as often happens in the real world, the algorithm is not that scalable and becomes very impractical to use. Even with relatively small case studies as are used in this report, the execution times are in the order of multiple hours. Conversely, the MGM algorithm is more scalable and gives reasonable (albeit local) solutions within the order of a few seconds or minutes. In industry, it seems that the extended version of MGM would be the best option because it is both scalable, and can be tweaked to have some level of privacy. This can be done by encrypting messages it shares with its neighbors. The genetic algorithm can also be used, and gives results in a reasonable time-frame. However, it has no way to guarantee privacy because a central institution would have to collect all the data. A final comparison of the algorithms is shown in Table 4-8.

Algorithm	Execution Time	Optimum	Privacy
MB-DPOP	very slow	global	no
MB-DPOP (P)	very slow	global	some guaranteed
MGM	fast	local	some possible
Genetic algorithm	fast	local	no

Table 4-8: This table shows a comparison of the algorithms.

Chapter 5

Conclusions and Future Recommendations

In this thesis, implementations were tested that could increase the willingness of vessel operators to adopt a unified approach in their scheduling of their rotation plans in a port area. That way, a more efficient use can be made of the inland waterway capacity present. In this chapter, first a summary of the thesis is given. After that, the conclusions from Chapter 4 are aggregated and repeated. Finally, some recommendations for future research are given.

5-1 Summary

In Chapter 1, an introduction to the current problem was given. Because vessel operators often do not want to share specifics of their plans in fear out of losing a competitive advantage, it is useful if a certain level of privacy can be guaranteed whilst not needing a central agency that could potentially leak data. Next to that, the container demands often change and it would be beneficial to use previous solutions because they do not differ that much. Apart from the problem, the contributions as well as the report layout were listed.

After that, in Chapter 2, it was shown that the problem of efficiently routing containers to terminals can be recast as a Vehicle Rotaton Planning Problem (VRPP), which in turn can be recast as a Distributed Constraint Optimization Problem (DCOP). Distributed algorithms can then be used to determine a solution to the vessel rotation planning problem, and have the possibility to guarantee a certain level of privacy. To achieve this, some other issues had to be addressed first such as the scaling of the algorithms by introducing Memory-Bounded Distributed Pseudotree Optimization Protocol (MB-DPOP), a memory bounded variant of the Distributed Pseudotree Optimization Protocol (DPOP). The scalable version of the algorithm was introduced in Chapter 2, as well the Maximum Gain Messaging (MGM) algorithm, which iteratively gives a local solution instead of a global solution.

In Chapter 3, some extensions to the MB-DPOP and the MGM algorithms were given that might increase the acceptability of them in the real world. In this chapter, a privacy im-

plementation was done that guarantees a certain level of privacy for the agents. Next to a certain level of privacy, the problem formulation was extended such that water locks can be implemented as well. Lastly, a varying utility method was proposed that introduces a control variable called the unwillingness into the MGM algorithm that makes it possible for vessel operators to weigh how much they are willing to change their current plan for a better plan.

Lastly, in Chapter 4, two case studies were done, a simple one and a more complex one, both of which simulated a container shipment demand in the port area of Rotterdam. The simple case study included only one vessel and a relatively small number of containers to be transported, whilst the more complex case study included multiple vessels and a larger shipment demand. With these simple case studies, the algorithms were compared to each other (MB-DPOP and MGM), as well as to other algorithms such as a genetic algorithm.

5-2 Conclusions

In the simulations section, some conclusions were drawn with respect to the speed and feasibility of implementation in real life. Whilst the possibility for a guaranteed level of privacy makes the MB-DPOP algorithm attractive, it is not very tractable when a large number of agents are present. In real life, where agents often share many constraints with one another, the number of constraints quickly becomes too high. Even though MB-DPOP has a control parameter to limit the maximum size of the messages sent, it is only gained at the cost of a trade-off with regards to the number of messages sent. Conversely, the MGM algorithm is a more realistic option. The algorithm iteratively gives local solutions, and with a utility implementation the algorithm is able to combine both market forces due to vessel operators being able to specify their unwillingness, as well as a collective resulting solution with a low cost.

5-3 Recommendations

Following from the conclusions, some recommendations can be made with regards to possible applications and research areas. Because many problems are distributed in nature, the algorithms could be applied to many problems in different environments. Examples of such environments are scheduling in railway systems, airport slot reservations, large terminals (internally), and sensor placement. Some of these environments such as container yards are high paced and would benefit more from a local optimization algorithm that yields a reasonable solution fast and scales well. Conversely, others might benefit from knowing the global solution. In a system where different railway operators want access to a railway track that has to be provided by a governing body, different railway companies might attach different costs to different combinations of railway obtained track. These railway operators would probably want to keep this information private, which is where the privacy-extended MB-DPOP version could be used.

Below, recommendations will be given for three different time horizons: short term, medium term, and long term. The short term recommendations are recommendations that could be implemented in a few weeks. The medium term would be around the size of a new MSc thesis study, and the long term recommendations would be even longer than that.

5-3-1 Short Term

- Check whether the MB-DPOP algorithm can be improved upon further. The authors of the algorithm in [15] have already come up with some improvements to the algorithm that makes it more scalable.
- A way to lower the complexity of the problem would be to do some kind of local optimization, such that the inferred width of the problem becomes less. This could result in a non-global optimum, but it is deemed worthwhile investigating.
- Investigate if there is a threshold that can make MGM be stuck in a bad local optimum. That is, when is a starting value so bad that we can better start with a random initialization, or any other initialization.
- Investigate how much privacy is lost according to the 4 levels of privacy. How can MGM be adapted to guarantee as much privacy as possible, and how much privacy would that be?

5-3-2 Medium Term

- Combine multiple algorithms with the goal to negate some of the disadvantages of the algorithms while also trying to keep the advantages of the algorithms. For example, an computationally less expensive MB-DPOP (as obtained from the recommendation above) could be combined with a MGM algorithm that reuses old solutions. The global optimum of the MB-DPOP could then be used to check the correctness of the solutions that are provided by the MGM algorithm, as well as provide the starting values for the MGM algorithm. Because the MB-DPOP takes way longer to yield a result, it could be run each time a number of iterations of the MGM algorithm has run.
- What is the ideal number of iterations that MGM would have to be run to implement the above recommendation? What parameters would be in the trade-off?

5-3-3 Long Term

- Investigate how the DCOP framework can be unified and tied in further with other optimization algorithms. What conversions are possible, and what can those conversions mean for the speed at which certain algorithms can be used?

Bibliography

[1] Ralf Borndörfer, Markus Reuther, Thomas Schlechte, and Steffen Weider. Vehicle Rotation Planning for Intercity Railways. page 22.

[2] ECTA-CEFIC. Guideline for measuring and managing CO2, March 2011.

[3] B. Faltings, T. Léauté, and A. Petcu. Privacy guarantees through distributed constraint satisfaction. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 350–358, 2008.

[4] RIL Foundation. Functioneel ontwerp bargeplanning.nl. Technical report, 2000.

[5] K. Hirayama and M. Yokoo. Distributed partial constraint satisfaction problem. In *International Conference on Principles and Practice of Constraint Programming*, pages 222–236. Springer, 1997.

[6] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, February 2001.

[7] Chung-Lun Li and George L. Vairaktarakis. Loading and unloading operations in container terminals. *IIE Transactions*, 36(4):287–297, April 2004.

[8] S. Li. Coordination for efficient transport over water. In *2015 IEEE 19th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 389–394, Calabria, Italy, May 2015.

[9] S. Li. Distributed constraint optimization for addressing vessel rotation planning problems. *Engineering Applications of Artificial Intelligence*, 48:14, 2016.

[10] Adam Lipowski and Dorota Lipowska. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6):2193–2196, March 2012.

[11] R. T. Maheswaran. A family of graphical-game-based algorithms for distributed constraint optimization problems. In *Coordination of Large-Scale Multiagent Systems*, pages 127–146. Springer, 2006.

[12] H. Moonen. Agent Technology supports Inter-Organizational Planning in the Port. *ERIM*, page 24, April 2005.

[13] A. Petcu and B. Faltings. A distributed, complete method for multi-agent constraint optimization. 2004.

[14] A. Petcu and B. Faltings. DPOP: A scalable method for multiagent constraint optimization. *IJCAI*, 05:266–271, 2005.

[15] A. Petcu and B. Faltings. MB-DPOP: A new memory-bounded algorithm for distributed optimization. In *IJCAI*, pages 1452–1457, 2007.

[16] Port of Rotterdam. Nextlogic: Chain Optimization Container Barging, 2013.

[17] Wuwen Qian, Junrui Chai, Zengguang Xu, and Ziying Zhang. Differential evolution algorithm with multiple mutation strategies based on roulette wheel selection. *Applied Intelligence*, 48(10):3612–3629, October 2018.

[18] Walchand College of Engineering, Umbarkar A.J., Sheth P.D., and Government College of Engineering, Karad. CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW. *ICTACT Journal on Soft Computing*, 06(01):1083–1092, October 2015.

[19] Jiaping Yang and Chee Kiong Soh. Structural Optimization by Genetic Algorithms with Tournament Selection. *Journal of Computing in Civil Engineering*, 11(3):195–200, July 1997.

[20] M Yokoo, E H Durfee, T Ishida, and K Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. page 13.

[21] Makoto Yokoo and Katsutoshi Hirayama. Distributed Breakout Algorithm for Solving Distributed Constraint Satisfaction Problems. page 9.

[22] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. Distributed stochastic search and distributed breakout: Properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1-2):55–87, January 2005.

Glossary

List of Acronyms

3mE	Mechanical, Maritime and Materials Engineering
DCOP	Distributed Constraint Optimization Problem
DCSC	Delft Center for Systems and Control
DFS	Depth First Search
DPOP	Distributed Pseudotree Optimization Protocol
DTREE	Distributed Tree
MB-DPOP	Memory-Bounded Distributed Pseudotree Optimization Protocol
MGM	Maximum Gain Messaging
VRPP	Vehicle Rotaton Planning Problem

List of Symbols

τ_{ij}^m	Traveling time from terminal i to terminal j of vessel m
d	Discrete variable domain size of a variable
i	Departure terminal index
j	Destination terminal index
k	MB-DPOP memory bounding parameter
l_i	Loading time at terminal i
n	Number of neighbors
p_1	Sojourn time trade-off weight
p_2	Undelivered container trade-off weight
p_3	Change of vector x trade-off weight
p_4	Change of vector z trade-off weight
r^m	VRPP cost function for a vessel m
r^{ia_m}	Inter agent utility function
r_{ij}	Container shipment demand matrix entry, denoting how many containers should be transported from terminal i to terminal j
s	Travel time matrix index corresponding to a water lock
t	Travel length
u_i	Unloading time at terminal j
w_i	Waiting time at terminal i
X_i	Agent i controlling a variable
x_{ij}^m	Binary variable that indicates whether vessel m goes from terminal i to terminal j
y_{ij}^m	Binary variable that denotes whether a vessel m takes a water lock going from terminal i to terminal j
z_{ij}^m	Integer variable denoting how many containers vessel m transports from terminal i to terminal j
R	Container shipment demand matrix
T	Traveling time matrix