Vessel Rotation Planning
A Layered Distributed Constraint Optimization Approach

Shijie Li, Rudy R. Negenborn and Gabriel Lodewijks
Faculty of Mechanical, Maritime and Materials Engineering,
Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands

Keywords: Multi-agent Systems, Constraint Optimization, Planning and Scheduling.

Abstract: Vessel rotation planning concerns the problem of assigning rotations to vessels over a number of terminals for loading and unloading containers in a large port. Vessel operators and terminal operators communicate with each other to make appointments about the rotation plans for the vessels. However, it happens frequently that these appointments cannot be met. Thus, it is important to generate the rotation plans for the vessel operators in an efficient automated way. In this paper, we propose an approach to solve the vessel rotation planning problem by modeling the problem as a layered distributed constraint optimization problem (DCOP). To evaluate the performance of the proposed approach, combinations of three DCOP algorithms are considered, namely, Asynchrounous Forward Bounding, Synchrounous Branch and Bound, and Dynamic Programming Optimization Protocol. We evaluate the solution quality and computational and communication costs of these three algorithms when solving the vessel rotation planning problem using the proposed layered formulation.

1 INTRODUCTION

A vessel rotation is the sequence in which a vessel visits different terminals in a large port. The vessel rotation planning problem (VRPP) concerns the problem of assigning rotations to a number of vessels over a number of terminals that they have to visit. The decision makers involved are vessel operators and terminal operators. Vessel operators are responsible for the voyage plan of vessels and coordinating inland shipping activities, while terminal operators are responsible for the transshipment of containers between deep sea vessels, trains, trucks, and inland vessels as well as the temporary storage of containers. As an example, on a typical day, around 25 inland vessels visit the Port of Rotterdam, and each vessel visits on average 8 different container terminals (Moonen et al., 2007).

Nowadays, vessel operators and terminal operators communicate with each other through telephone, fax and e-mail for making appointments. However, in practice, the appointments made often cannot be met (Melis et al., 2003). In the port of Rotterdam, the average rotation time for an inland vessel is approximately 22.5 hours, of which only 7.5 hours are used for loading and unloading, the rest of the time vessels are waiting and traveling (Moonen et al., 2007). A disturbance at one terminal can lead to the interruption of the operations of a vessel and terminal operators elsewhere. This can make it difficult for vessel operators to stick to the appointments made with terminal operators (Douma, 2008).

There have been several publications on the alignment of vessels and terminals in recent years. In (Schut et al., 2004), a multi-agent based and distributed planning system named APPROACH has been introduced. The outcomes of the software sometimes, however, contained routes considered unlogical and with longer sailing times than needed (Moonen et al., 2007). In (Douma et al., 2009) the same type of agents is used but with different interaction protocols and agent structure. The authors improve the multi-agent systems by considering design choices that could influence the acceptance of the end users and the extent to which users can optimize their operations in (Douma et al., 2012).

The problem considered in this paper consists of finding in an automated way the best solution to a VRPP rotation plan for a given set of vessels. Our approach proposes to use layered distributed constraint optimization (DCOP). DCOP is a theoretical model framework where several agents communicate with each other to take on values so as to minimize the sum of the resulting constraint costs. In (Li et al.,

Li S., R. Negenborn R. and Lodewijks ..
Vessel Rotation Planning - A Layered Distributed Constraint Optimization Approach.
DOI: 10.5220/0005202801660173
In *Proceedings of the International Conference on Agents and Artificial Intelligence* (ICAART-2015), pages 166-173
ISBN: 978-989-758-073-4

2014), we made the a first proposal for using DCOP for the VRPP. In this paper, we extended this work firstly proposing a layered DCOP structure that splits the rotation planning process into two layers so that relatively large problem instances can be considered; secondly, we propose a different model from (Li et al., 2014); we introduce waiting time variable in the objective function, so that the final rotation plans include the arrival, waiting, departure and travel time of each vessel, which makes the model more realistic; thirdly, the performance of combinations of three DCOP algorithms are compared with respect to the communication load and simulated time.

This paper is organized as follows. In Section 2, the definitions for DCOP and VRPP are introduced. In Section 3, the VRPP is defined and formulated as layered DCOP and solution algorithms are introduced. Section 4 presents the implementation of the proposed approach and experimental results. Conclusions and future work are given in Section 5.

2 DCOP AND LAYERED DCOP

2.1 Distributed Constraint Optimization Problems

We use the DCOP formalism as defined in (Petcu, 2009). A DCOP is represented by a triple $\langle \mathcal{A}, \mathcal{COP}, \mathcal{R}^{\text{ia}} \rangle$, where, $\mathcal{A} = \{A_1, \ldots, A_N\}$ is a set of N agents; $\mathcal{COP} = \{COP_1, \ldots, COP_N\}$ is a set of disjoint, local Constraint Optimization Problems (COPs); COP_i is called the local sub-problem of agent A_i, and has to be solved by agent A_i; COP_i is a triple $\langle \mathcal{X}_i, \mathcal{D}_i, \mathcal{R}_i \rangle$, where $\mathcal{X}_i$ is a set of variables that belongs to A_i; $\mathcal{D}_i$ is a set of finite variable domains; $\mathcal{R}_i$ is a set of utility functions, which can also be used to represent objectives, hard and soft constraints; $\mathcal{R}^{\text{ia}} = \{r_1^{\text{ia}}, \ldots, r_{|\mathcal{R}^{ia}|}^{\text{ia}}\}$ is a set of inter-agent utility functions defined over variables of multiple agents. Each $r_l^{\text{ia}} : scope(r_l^{\text{ia}}) \rightarrow \mathbb{R}$ expresses the utility for joint decision obtained by the agents that have variables involved in r_l^{ia}. The agents involved in r_l^{ia} can decide on the values of the variables involved.

The objective of the agents solving a DCOP is to find the assignments to all variables X^* such that all the constraints are satisfied (expressed via utility functions) and the sum of values of all the utility functions representing the objectives is maximized,

$$X^* = \arg\max \sum_{i=1}^{N} \left(\sum_{v=1}^{|\mathcal{R}_i|} r_{iv}(X_{i1}, \ldots, X_{i|X_i|}) \right) + \sum_{l=1}^{|\mathcal{R}^{ia}|} r_l^{\text{ia}},$$

To find the optimal solution X^*, agents need to communicate and exchange messages that include information on the assignments of values to variables and the related utility values among each other. Thus, the total number of messages and the size of messages sent by the agents is an important performance metric for measuring the efficiency of DCOP algorithms.

Based on the completeness of the algorithms, DCOP algorithms can be categorized as complete and incomplete algorithms. Complete algorithms typically do an exhaustive search over the problem space: thus, they can guarantee finding an optimal solution, while incomplete algorithms usually use local search methods to find locally optimal solutions, and thus can potentially get trapped in local minima or fail to converge to any solution altogether. In (Petcu, 2009), a detailed description and comparison of the DCOP algorithms can be found. In this paper, we compare the performances of the combinations of three different complete algorithms, including DPOP, AFB and SyncBB.

2.2 Layered DCOP

As solving DCOP problems is NP-hard (Modi et al., 2005), increasing the number of variables increases the complexity of the problem exponentially. Therefore, using straightforward DCOP methods for solving VRPPs that consist of many vessels and terminals brings high computation and communication costs. Partitioning of the DCOP could significantly improve the solution process.

In (Hosseini et al., 2013), the authors consider a hierarchical DCOP setting for a target to sensor allocation problem, in which they change the original DCOP as a hierarchical set of smaller DCOPs with shared constraints in order to avoid significant computational and communication costs. The results show that compared with a non-hierarchical structure, the hierarchical modeling technique provides superior results, with the advantage becoming more significant under increased problem size and complexity.

Using a partitioned method prevents the creation of large DCOPs, potentially reducing the computational complexity greatly (Hosseini et al., 2013), although this can also be at the cost of lower solution quality. Thus, in this paper we propose a layered structure for the VRPP formulated as DCOP so that the computation and communication costs can be reduced. We split VRPP into two layers of DCOPs, the solutions from the upper layer will be the constraints in the lower layer. On the upper layer is the DCOP that involves all the vessels, while in the lower layer, there are sets of smaller DCOPs, each vessel having 1 DCOP.

3 LAYERED DCOP FOR VESSEL ROTATION PLANNING

3.1 Vessel Rotation Planning Problem

VRPP concerns selecting the rotation plan consisting of sequences and arrival and departure times of vessels to terminals in a port area. To model the VRPP as a DCOP, we make the following assumptions:

- Decisions are made at discrete time steps, so discrete *time slots* can be considered.
- Each *vessel* knows which *terminals* it needs to visit. The visiting sequence, is unknown. Each vessel has its own preference over the visiting sequence for different terminals.
- Distances between terminals are given.
- Service time of a vessel at a terminal is known.

The problem can now be formalized using the following variables:

- A set of vessels: $V = \{1,2,3,...,N\}$
- A set of terminals: $T = \{1,2,3,...,M\}$
- A set of discrete time slots: $D = \{1,2,3,...,P\}$
- x_{ij} represents the time slot at which vessel i is at terminal j
- a_{ij}, d_{ij} and w_{ij} represents the arrival, departure, and waiting time of vessel i at terminal j, respectively

We use the following parameters in our model:

- U_{ij}^k represents the preference of vessel i of being at terminal j during time slot k
- W_{ij}^k represents the utility value of variable w_{ij}
- s_{ij} represents service time of vessel i at terminal j
- t_{mn} represents traveling time from terminal m to n
- $[T_j^{k-1}, T_j^k]$ is a fixed time window for terminal j when it is visited by vessels during time slot k

U_{ij}^k is used to represent the preferences of vessel i at terminal j. For example, $U_{11}^1 = 5$, $U_{11}^2 = 4$, $U_{11}^3 = 3$ means that vessel 1 prefers to visit terminal 1 during time slot 1, since it has the highest utility value. These utilities are implemented in the DCOP framework by assigning different preference values (utility values that represents the preference of vessels) when $x_{ij} = k$, and assigning 0 to all the situation when $x_{ij} \neq k$.

W_{ij}^k is used to represent the utility value for $w_{ij} = k$. A higher value of k represents a longer waiting time of vessel i at terminal j. Thus, the higher the value of k is, the lower the value of W_{ij}^k will be. Service time s_{ij} and traveling time t_{mn} are integer constants.

$[T_j^{k-1}, T_j^k]$ is a fixed time window for terminal j when it is visited by vessels during time slot k, for example, $[T_1^3, T_1^3] = [6, 10]$ means that if vessel i visit terminal 1 during time slot 3, the arrival time of vessel i at terminal 1 must be within the time window $[6, 10]$. We introduce the concept of time window to ensure that for the vessel that visits terminal j during time slot k, it can be serviced only if it arrives at the terminal within a specified time window $[T_j^{k-1}, T_j^k]$. T_j^k is the deadline for a vessel's arrival to be serviced at terminal j if it visits the terminal during time slot k.

The objective is to find the optimal rotation plan for vessels, which includes the sequences and departure and arrival time at each terminal, such that a set of utility functions are maximized.

3.2 Agents Structure

VRPP can be modeled as a DCOP by considering the vessels as agents, the constraints related with vessels as constraints of local problems of the agents. As the constraints for terminals usually involve variables from different vessel agents, these constraints can considered as inter-agent utility functions. In our model, agent A_i represents vessel i. In the local problem of A_i, values of variables x_{ij}, a_{ij}, d_{ij} and w_{ij} are determined by A_i. The inter-agent and local utility functions are defined as hard constraints related to a vessel i. The preference of vessels at terminals during different time slots are also considered as the utilities of the local COPs of the vessel agents.

For the DCOP we will formulate, the objective is find the assignments of values from domain D to variables x_{ij}, a_{ij}, d_{ij} and w_{ij}, that maximizes the sum of values of the utility functions of local COPs and the inter-agent utility functions. The inter-agent utilities are represented as the terminal capacity constraints.

Based on the above structure, we can provide a general definition of the VRPP as DCOP, as a tuple of the following form, $\langle \mathcal{A}, \mathcal{COP}, \mathcal{R}^{\text{ia}} \rangle$, in which $\mathcal{A}$ is the set of vessel agents; $\mathcal{COP}$ is the set of local constraint optimization problems for each vessel agent; $\mathcal{R}^{\text{ia}}$ is the set of inter-agent utility functions defined over variables from different vessel agents.

3.3 Layered DCOP for VRPP

Problem solving using a layered structure uses a top down approach. As we can see from Figure 1, the upper layer is what we define as assignment problem, which decides the sequence in which the vessels visit different terminals. When the solutions have been obtained at the higher layer, the lower layer, which we define as scheduling problem, will decide the exact arrival, departure and waiting time of each vessel at each terminal. There is one DCOP in the upper layer

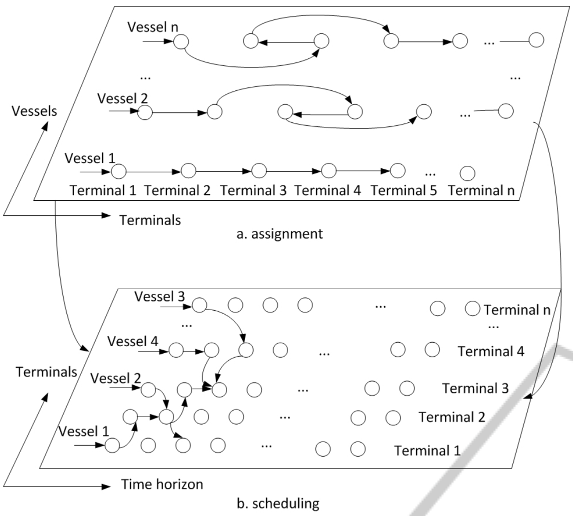

Figure 1: Layered DCOP structure.

that relates to all the vessels, while there are multiple smaller DCOPs in the lower layer, each relating only to 1 vessel. After finding an optimal solution for the upper layer DCOP problem, the lower layer's DCOP solver starts solving each problem separately. There are common variables between the upper layer DCOP and the lower layer DCOPs. In solving the problem of the lower layer, the algorithm is not concerned with the values maintained by the upper layer, but only with the solutions obtained from upper layer.

3.3.1 Assignment Problem

At the upper layer, we only consider the sequence of vessels' visits to terminals. Thus, only variable x_{ij} which represents the time slot at which vessel i is at terminal j is involved. The exact arrival, departure and waiting time of vessels will not be included in this layer; these are considered in the lower layer.

The local COP_i of A_i is defined by a triple $\langle \mathcal{X}_i^{\mathrm{as}}, \mathcal{D}_i^{\mathrm{as}}, \mathcal{R}_i^{\mathrm{as}} \rangle$. Here, $\mathcal{X}_i^{\mathrm{as}} = \{x_{i1}, \dots, x_{i|\mathcal{X}_i^{\mathrm{as}}|}\}$ is a set of variables, in which x_{ij} represents the time slot at which vessel i is at terminal j; $\mathcal{D}_i^{\mathrm{as}} = \{d_{i1}, \dots, d_{i|\mathcal{D}_i^{\mathrm{as}}|}\}$ is a set of finite variable domains, for each $d_{ij} \in \{1,2,3,...,P\}$; $\mathcal{R}_i^{\mathrm{as}} = \{r_{i1}^{\mathrm{as}}, \dots, r_{i|\mathcal{R}_i^{\mathrm{as}}|}^{\mathrm{as}}\}$ contains utility functions that represent the constraints related with vessels and the their preferences over different terminals during different time slots. Thus, we have,

$$r_{i1}^{\mathrm{as}} = \begin{cases} U_{ij}^k : \text{if} x_{ij} = k \quad \forall i \in \{1,2,...,N\}, \\ \forall j \in \{1,2,...,M\}, k \in \{1,2,...,P\} \\ 0 : \quad \text{otherwise} \end{cases} \tag{1}$$

$$r_{i2}^{\mathrm{as}} = \begin{cases} 0 : \quad \textit{all-different}(x_{i1}, \dots, x_{iM}) \\ \forall i \in \{1,2,...,N\} \\ -\infty : \text{otherwise} \end{cases} \tag{2}$$

Utility function r_{i1}^{as} represents the preferences of vessel i being at terminal j during time slot k. U_{ij}^k is a constant defined for different combinations of i, j and k. Utility function r_{i2}^{as} is defined based on the *all-different* constraint as in (Rossi et al., 2006), it uses the *all-different* constraint to ensure that each vessel will only be at most at one terminal during a time slot.

In addition, we define the inter-agent utility functions as the capacity constraints of terminals, i.e. each terminal can serve only a limited number of vessels simultaneously. We incorporate the *cumulative* constraint from constraint programming to represent the inter-agent utility functions $r^{\mathrm{ia,as}}$ of the assignment problem,

$$r^{\mathrm{ia,as}} = \begin{cases} 0 : \quad \textit{cumulative}((x_{1j}, \dots, x_{Nj}), s_{ij}, 1, N_j) \\ \forall j \in \{1,2,...,M\} \\ -\infty : \text{otherwise} \end{cases} \tag{3}$$

In which, $(x_{1j}, \dots, x_{Nj})$ are the variables representing time slots at which the vessels that will be at terminal j. s_{ij} is the service time for a vessel at a terminal. Consumption rate c is set to 1 because a vessel will be serviced by one terminal. N_j is the number of vessels one terminal can serve simultaneously.

For the overall DCOP assignment problem, the objective is to maximize the sum of values of individual utility functions and the inter-agent utility functions, defined as:

$$\max \left(\sum_{i=1}^{N} (r_{i1}^{\mathrm{as}} + r_{i2}^{\mathrm{as}}) + r^{\mathrm{ia,as}} \right) \tag{4}$$

3.3.2 Scheduling Problem

When the solutions x_{ij} from the assignment problem have been obtained, we can determine the sequence of vessels to different terminal j and the time slots during which each vessel will be at terminals. We can formulate the scheduling problem based on DCOP similar to the assignment problem as a tuple $\langle \mathcal{A}^{\mathrm{sc}}, \mathcal{COP}^{\mathrm{sc}}, \mathcal{R}^{\mathrm{ia,sc}} \rangle$. The local problem COP_i^{sc} of each vessel agent A_i^{sc} is defined by a triple $\langle \mathcal{X}_i^{\mathrm{sc}}, \mathcal{D}_i^{\mathrm{sc}}, \mathcal{R}_i^{\mathrm{sc}} \rangle$, where, $\mathcal{X}_i^{\mathrm{sc}} = \{a_{i1}, \dots, a_{i|\mathcal{X}_i^{\mathrm{sc}}|}, d_{i1}, \dots, d_{i|\mathcal{X}_i^{\mathrm{sc}}|}, w_{i1}, \dots, w_{i|\mathcal{X}_i^{\mathrm{sc}}|}\}$ is a set of variables, and a_{ij}, d_{ij} represent the arrival, departure time of vessel i from terminal j, w_{ij} represents the waiting time of vessel i at terminal j; $\mathcal{D}_i^{\mathrm{sc}} = \{k_{i1}, \dots, k_{i|\mathcal{D}_i^{\mathrm{sc}}|}\}$is a set of variable domains, for each $k_{ij} \in \{1,2,3,...,P\}$; $\mathcal{R}_i^{\mathrm{sc}} = \{r_{i1}^{\mathrm{sc}}, \dots, r_{i|\mathcal{R}_i^{\mathrm{sc}}|}^{\mathrm{sc}}\}$ contains utility functions that represent the constraints for vessels' arrival, departure and waiting at terminals.

The inter-agent utility function $r^{\text{ia,sc}}$ for the scheduling problem is defined as follows:

$$r^{\text{ia,sc}} = \begin{cases} 0: & \text{if}\, x_{im'} = x_{im}+1, \text{then} \quad a_{im'} = d_{im} + t_{mm'} \\ \forall i \in \{1,2,...,N\}, \forall m,m' \in \{1,2,...,M\} \\ -\infty : \text{otherwise} \end{cases} \tag{5}$$

$r^{\text{ia,sc}}$ ensures that if vessel i travels from terminal m to m', the arrival time at terminal m' is the sum of departure time from m and the traveling time $t_{mm'}$.

For each vessel agent, we have the utility functions as follows (for the following constraints we have $k,k' \in \{1,2,3,...,P\}$),

$$r^{\text{sc}}_{i1} = \begin{cases} 0: & \text{if}\, x_{ij} = k, \text{then}\, a_{ij} \in [T_j^{k-1}, T_j^k] \\ \forall i \in \{1,2,...,N\}, \forall j \in \{1,2,...,M\} \\ -\infty : \text{otherwise} \end{cases} \tag{6}$$

$$r^{\text{sc}}_{i2} = \begin{cases} 0: & \text{if}\, d_{ij} = a_{ij} + w_{ij} + s_{ij} \\ \forall i \in \{1,2,...,N\}, \forall j \in \{1,2,...,M\} \\ -\infty : \text{otherwise} \end{cases} \tag{7}$$

$$r^{\text{sc}}_{i3} = \begin{cases} 0: & \text{if}\, x_{ij} = k, \text{then}\, w_{ij} = T_j^k - a_{ij} \\ \forall i \in \{1,2,...,N\}, \forall j \in \{1,2,...,M\} \\ -\infty : \text{otherwise} \end{cases} \tag{8}$$

$$r^{\text{sc}}_{i4} = \begin{cases} W_{ij}^{k'} : \text{if}\, w_{ij} = k' \quad \forall i \in \{1,2,...,N\}, \\ \forall j \in \{1,2,...,M\} \\ 0: \quad \text{otherwise} \end{cases} \tag{9}$$

Utility function r^{sc}_{i1} ensures that if vessel i visits terminal j during time slot k, the arrival time should be within the time window $[T_j^{k-1}, T_j^k]$ of time slot k. If vessel i cannot arrive within the time window, it will not be serviced during time slot k and has to wait to be serviced during the next time slot.

Utility function r^{sc}_{i2} ensures that the departure time of a vessel from a terminal equals the sum of the vessel's arrival time, waiting time and service time at the terminal. Utility function r^{sc}_{i3} ensures that the waiting time is the difference between the arrival time of a vessel i at terminal j and the start time for the vessel to be serviced at the terminal.

Utility function r^{sc}_{i4} ensures that the shorter the waiting time w_{ij} is, the higher the utility value $W_{ij}^{k'}$ will be. This utility is used to make sure that the objective of maximization of the sum of utility values will lead to the minimization of the sum of waiting times. As we can see, r^{sc}_{i1}, r^{sc}_{i3}, $r^{\text{ia,sc}}$ depends on the solutions for x_{ij} from the assignment problem.

The objective is to maximize the sum of the values of the individual utility functions and the inter-agent utility functions, defined as:

$$\max \left(\sum_{i=1}^{N} (r^{\text{sc}}_{i1} + r^{\text{sc}}_{i2} + r^{\text{sc}}_{i3} + r^{\text{sc}}_{i4}) + r^{\text{ia,sc}} \right) \tag{10}$$

3.4 Solution Algorithms

In this section, we will use DPOP introduced by (Petcu and Faltings, 2005a) as an example to illustrate the mechanism of solving vessel rotation planning problem based on DCOP.

Firstly, a depth first search (DFS) structure is established using a distributed DFS algorithm. Each variable is considered as a node. Each agent controls a set of variables. The nodes consistently label each other as parent/child nodes, and the edges are identified as tree/back edges. In our case, agent A_i^{as} and agent A_i^{sc} represent the same vessel i in the assignment problem and scheduling problem, respectively. This means that variables x_{ij} are owned by A_i^{as}, while a_{ij}, d_{ij} and w_{ij} are owned by A_i^{sc}.

The second phase is UTIL propagation, the objective of which is to propagate aggregated utilities up the DFS tree. Initially, messages travel up in the DFS tree, propagating information about the aggregated optimal costs/utilities. For example, for each variable x_{ij} belonging to agent A_i^{as}, agent A_i^{as} joins the constraints involving x_{ij} together, while ignoring all constraints that involve at least one descendant of x_{ij}. Then agent A_i^{as} waits for the reception of a UTIL message from each of the child nodes of x_{ij}, and joins them all together with its constraints. Finally, agent A_i^{as} eliminates x_{ij} from the join, and sends the resulting utility to the parent node of x_{ij}. This utility corresponds to the maximum achieved utility for the whole subtree rooted at x_{ij}, as a function of the value for the parent node of x_{ij} and also the values for other variables higher than x_{ij} in the DFS tree.

The third phase of DPOP is VALUE propagation. At the end of the UTIL propagation, the root variable has obtained its own local optimal value based on the messages it received. The agent that controls the value of the root variable sends this optimal value to each of the child nodes of the root variable through VALUE messages. Recursively, for each variable x_{ij}, when the corresponding agent receives the VALUE message from the parent node of that variable x_{ij}, is able to look up its own corresponding optimal value during the UTIL propagation phase. To each of the children nodes of x_{ij}, agent A_i^{as} sends not only the optimal value of x_{ij}, but also the optimal values for all the variables in x_{ij}'s children node's separator(parent and pseudo-parents nodes), which are contained in the VALUE message it receives. Optimal decisions are hereby propagated down the DFS tree, until all variables have been assigned optimal values.

SyncBB and AFB are alternative DCOP algorithms that use a linear ordering of agents instead of a DFS tree. For SyncBB, at each step an agent

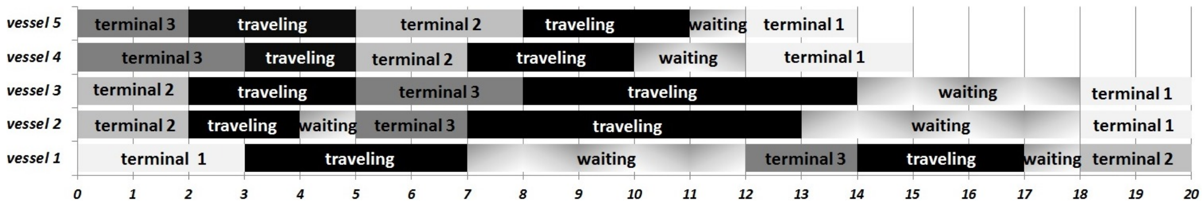

Figure 2: Rotation plan (5 vessels, 3 terminals)

Table 1: Comparison of the total number of messages for different cases and algorithms (by type)

	DPOP+ DPOP	AFB+ AFB	SyncBB+ SyncBB	DPOP+ SyncBB	SyncBB + DPOP	AFB + SyncBB	SyncBB+ AFB	DPOP+ AFB	AFB + DPOP
Case 1*	127	4,923	4,685	2,013	2,815	4,746	4,853	2,574	2,885
Case 2*	307	14,562	11,559	3,509	8,357	12,412	13,706	5,659	9,120
Case 3*	392	24,132	23,520	4,650	8,927	21,230	22,964	6,752	10,237

* The total number of messages exchanged is the same for the three different tests of the same case.

tries to assign a value to the current assignment without causing the lower bound to reach the upper bound while for AFB, agents perform asynchronous concurrent checks of bounds. More details of SyncBB and AFB are given in (Gershman et al., 2009) and (Hirayama and Yokoo, 1997).

4 EXPERIMENTS

The DCOP algorithms we experiment with are implemented in the FRODO2 toolbox (Léauté et al., 2009). Our tests are performed on an Intel Core i5-2400 CPU with RAM 4GB under Windows 7. Values reported here are averages of 10 repetitions of the simulation.

4.1 Setup

To test the effectiveness of the DCOP algorithms, we choose 3 different types of DCOP algorithms, including DPOP, AFB and SyncBB. We apply different algorithms in the two layers, the assignment problem and the scheduling problem. In order to evaluate the communication load and simulated time of the three algorithms, three performance metrics are measured: total number of messages, total size of messages, and average simulated time.

We also differ cases concerning the number of vessels and terminals. We summarize this below:

- Case 1: 3 vessels, 3 terminals, each vessel needs to visit 3 terminals, 36 variables, 55 constraints;
- Case 2: 5 vessels, 3 terminals, each vessel needs to visit 3 terminals, 60 variables, 93 constraints;
- Case 3: 7 vessels, 5 terminals, each vessel needs to visit 3 terminals, 80 variables, 157 constraints.

For each case, we choose 10 different groups of values for U_{ij}^k and W_{ij}^k. Thus, we run 30 experiments in total. The range of the utility value for U_{ij}^k and W_{ij}^k is arbitrary in our tests, but higher value means higher priority. In addition, we assume that one terminal can serve two vessels simultaneously ($N_j = 2$).

4.2 Results

4.2.1 Generated Schedule Plan

Based on the solutions we have obtained from the experiments, we can get the arrival time, departure time, service time and travel time for each vessel from variable a_{ij}, d_{ij}, s_{ij} and t_{mn}, respectively. Due to space limitations, here we only present one of the rotation plans generated using DPOP in both assignment and scheduling problem in the Case 2 as an example, which is shown in Figure 2. The figure shows the rotation plan for 5 vessels, and 3 terminals. In this figure, different colors represent different terminals. As we can see, when a vessel does not arrive at the terminal within the corresponding time window, it has to wait at the terminal until the next available time window of the terminal. From this rotation plan, each vessel operator can determine the sequence in which to visit the different terminals and exact arrival and departure time.

4.2.2 Communication Load

Table 1 and Table 2 show the comparison of total number of messages and size of messages. The number and size of messages calculated here include both the inter-agent and intra-agent messages. With respect to the total number of messages, applying only the DPOP algorithm results in the lowest number of

Table 2: Comparison of the total size of messages for different cases and algorithms (/bytes).

	DPOP + DPOP	AFB + AFB	SyncBB+ SyncBB	DPOP+ SyncBB	SyncBB+ DPOP	AFB+ SyncBB	SyncBB+ AFB	DPOP+ AFB	AFB+ DPOP
Case 1*	31,342	139,423	106,323	72,986	65,391	120,949	128,757	142,964	76,057
Case 2*	5,927,266	726,259	265,855	6,009,593	183,528	471,473	520,641	6,264,379	389,146
Case 3*	8,926,286	809,437	484,567	8,462,975	2,987,432	575,984	620,453	8,764,275	487,276

* The total size of messages exchanged is the same for the three different tests of the same case.

Table 3: Comparison of average simulated time for different cases and algorithms (/ms).

	DPOP+ DPOP	AFB+ AFB	SyncBB+ SyncBB	DPOP+ SyncBB	SyncBB+ DPOP	AFB+ SyncBB	SyncBB+ AFB	DPOP+ AFB	AFB+ DPOP
Case 1*	61	64	110	86	108	59	115	89	57
Case 2*	2,352,094	163	343	2,352,130	298	165	341	2,352,114	120
Case 3*	6,361,192	280	395	6,308,543	354	290	451	6,254,391	357

* The simulated time is the average of calculations based on all three tests for each case.

messages generated in all three cases. The reason is that the number of messages is linear in the number of agents and depends on the height of the DFS tree associated with the problem being modeled (Petcu and Faltings, 2005b). When the number of variables and agents increase, the total number of messages generated increases linearly. For all three cases, AFB generate the largest number of messages.

However, with respect to the size of messages, the situation is different. For small size problem as in Case 1, DPOP has much smaller size of messages than the others, however, with the increase of agents and variables, the size of messages for DPOP increase exponentially, while for AFB and SyncBB, the size of messages does not increase as significantly as DPOP as in Case 2 and Case 3. The reason is that complexity of DPOP is given by the size of the largest UTIL message it produces, aggregated from the communication between parent agents and children agents in the DFS tree, which is exponential in the induced width of the DFS ordering used. Thus, DPOP sends out exponentially larger messages (but only a linear number of them), while the other algorithms send out an exponential amount of messages but of only linear size.

4.2.3 Simulated Time

Table 3 shows the simulated time of DCOP algorithms. For smaller cases like Case 1, the simulated time for each algorithm does not differ much. SyncBB has slightly longer simulated time. However, with the increase of problem size, in Case 2 and Case 3, DPOP has the longest simulated time in either the upper layer or the lower layer. For SyncBB and AFB, the simulated does not have steep increases as for DPOP. AFB has a shorter simulated time than SyncBB. This happens because the memory requirement for DPOP is exponential in the induced width of the DCOP problem, which depends on the number of backedges (the links between nodes and their pseudo-parents) in the DFS tree (Petcu and Faltings, 2005b). It can be as large as the number of agents minus one of the constraint graph is fully connected and every agent is thus constrained with every other agent. For SyncBB and AFB, the memory requirements are polynomial in the number of agents.

To conclude, DPOP outperforms AFB and SyncBB in the total number of messages exchanged. However, this performance is achieved with a high increase in message size and simulated time (the growth of which is exponential in the DFS tree-width) which makes it inapplicable to DCOPs with large tree-widths and larger problem sizes. Our results suggest that SyncBB offers the best tradeoff between communication load and simulated time on this problem class. These results seem to indicate that AFB is not the best suited algorithm for the VRPP. This could be due to the problem's variety and complexity.

5 CONCLUSIONS AND FUTURE RESEARCH

This paper proposes a new method for solving the vessel rotation planning problem using a DCOP approach. We use a non-binary variable representation to reduce the number of variables and constraints, and incorportate a layered structure to divide the main problem into smaller ones and simplify the complex and hard problem. Three DCOP algorithms are compared over three different cases. Results show that it is possible to verify the effectiveness and efficiency of DCOP algorithms in the proposed application.

To take the DCOP approach into application, the problem size should at least be increased to 22 vessels, 8 terminals. In addition, the constraints from the perspective of terminal operators should be taken into consideration for future work.

Although the layered problem solving decreases computation and communication cost , addressing the problem in a layered way may lead to finding sub-optimal solutions. The reason is that some values are selected for the common variables in the upper level and these selections may impose extra constraints on these common variables in the lower layer DCOPs. Lower layer DCOP then have to keep previously selected values unchanged. On the hand, considering the reduction of communication and computation complexity of the layered approach, in dynamic situations when DCOP algorithms do not have enough time to reach the optimal solution, the benefit of using a layered approach strongly outweighs its costs drawback.

ACKNOWLEDGEMENTS

This research is supported by the China Scholarship Council under Grant 201206680009 and the VENI project "Intelligent multi-agent control for flexible coordination of transport hubs" (project 11210) of the Dutch Technology Foundation STW, a subdivision of the Netherlands Organization for Scientific Research (NWO). The authors would also like to thank Dr.Thomas Léauté for his constructive suggestions regarding FRODO2 toolbox.

REFERENCES

Douma, A. (2008). *Aligning the operations of barges and terminals through distributed planning*. PhD thesis, University of Twente, Enschede, The Netherlands.

Douma, A., Schutten, M., and Schuur, P. (2009). Waiting profiles: An efficient protocol for enabling distributed planning of container barge rotations along terminals in the port of rotterdam. *Transportation Research Part C: Emerging Technologies*, 17(2):133–148.

Douma, A., van Hillegersberg, J., and Schuur, P. (2012). Design and evaluation of a simulation game to introduce a multi-agent system for barge handling in a seaport. *Decision Support Systems*, 53(3):465–472.

Gershman, A., Meisels, A., and Zivan, R. (2009). Asynchronous forward bounding for distributed cops. *Journal of Artificial Intelligence Research*, 34(1):61–88.

Hirayama, K. and Yokoo, M. (1997). Distributed partial constraint satisfaction problem. In *Proceedings of the 3rd International Conference on Principles and Practice of Constraint Programming*, pages 222–236. Springer, Linz, Austria.

Hosseini, S., Samaneh, and Basir, O. A. (2013). Target to sensor allocation: A hierarchical dynamic distributed constraint optimization approach. *Computer Communications*, 36(9):1024–1038.

Léauté, T., Ottens, B., and Szymanek, R. (2009). FRODO 2.0: An open-source framework for distributed constraint optimization. In *Proceedings of the 21th International Joint Conference on Artificial Intelligence*, pages 160–164, Pasadena, California, USA.

Li, S., Negenborn, R., and Lodewijks, G. (2014). A distributed constraint optimization approach for vessel rotation planning. In *Proceedings of the 5th International Conference on Computational Logistics*, pages 61–80, Valparaso, Chile.

Melis, M., Miller, I., Kentrop, M., Van Eck, B., Leenaarts, M., Schut, M., and Treur, J. (2003). Distributed rotation planning for container barges in the port of rotterdam. *Intelligent Logistics Concepts*, pages 101–116.

Modi, P. J., Shen, W.-M., Tambe, M., and Yokoo, M. (2005). Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1):149–180.

Moonen, H., Van de Rakt, B., Miller, I., Van Nunen, J., and Van Hillegersberg, J. (2007). Agent technology supports inter-organizational planning in the port. *Managing Supply Chains: Challenges and Opportunities*, pages 201–225.

Petcu, A. (2009). *A class of algorithms for distributed constraint optimization*. PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne, Switherland.

Petcu, A. and Faltings, B. (2005a). Approximations in distributed optimization. In *Principles and Practice of Constraint Programming*, pages 802–806. Springer.

Petcu, A. and Faltings, B. (2005b). A scalable method for multiagent constraint optimization. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, volume 5, pages 266–271, Edingurgh, Scotland.

Rossi, F., Van Beek, P., and Walsh, T. (2006). *Handbook of constraint programming*. Elsevier.

Schut, M. C., Kentrop, M., Leenaarts, M., Melis, M., and Miller, I. (2004). Approach: Decentralised rotation planning for container barges. In *Proceedings of the 16th European Conference on Artificial Intelligence*, pages 755–759, Valencia, Spain.