

# Ensuring Privacy through Distributed Computation in Multiple-Depot Vehicle Routing Problems

Thomas Léauté and Brammert Ottens and Boi Faltings<sup>1</sup>

**Abstract.** The *Vehicle Routing Problem (VRP)* has been extensively studied over the last twenty years, because it is an abstraction of many real-life logistics problems. In its *multiple-depot* variant (MDVRP), the routes of vehicles located at various depots must be optimized to serve a number of customers. In this paper, we investigate how to protect the privacy of delivery companies, when each depot is owned by a different company with a limited view of the overall problem. Companies then need to exchange messages with each other to coordinate the assignment of customers to depots. We show how *Distributed Constraint Optimization (DCOP)* can be used to solve the assignment problem using distributed computation, and we study the guarantees that can be provided with respect to the protection of each company’s knowledge about the problem.

## 1 INTRODUCTION

Consider a delivery company that has subcontracted the delivery of packets to several independent contractors. These contractors are all interested in maximizing their own profit, while the global delivery company wants to serve its customers as efficiently as possible. Using a centralized approach to assign customers to the different contractors would involve the disclosure of sensitive information on the side of these contractors, which in general is not something that they are eager to accept. One could resort to the use of auctions to allocate customers to different contractors, but auctions are, in general, not able to give any guarantees on the quality of the solution found.

This paper proposes a distributed optimization approach that is able to find an efficient allocation of customers, while retaining a certain level of privacy for the different contractors. This approach makes use of the fact that contractors are generally geographically separated, meaning that not every contractor will be able to serve every customer, i.e. a contractor will not have to coordinate with every other contractor, greatly reducing the complexity of the problem.

The formalism used in the approach presented in this paper is the *Distributed Constraint Optimization (DCOP)* framework [21]. The DCOP framework is used to model loosely coupled multi-agent optimization problems, where the knowledge of the problem is distributed over different agents. Several algorithms exist that are able to solve DCOP problems. These algorithms can be sub-divided into two main families. One family of algorithms uses a form of distributed search to find an optimal solution (such as SynchBB [21], ADOPT [31], or AFB [16]), while the other family of algorithms is based on dynamic programming and uses inference to solve the problem (like DPOP [34], O-DPOP [36], or ASO-DPOP [32]).

We use the DCOP framework to address the problem of assigning customers to depots in a distributed variant of the *Multiple-Depot Vehicle Routing Problem (MDVRP)*. The *Vehicle Routing Problem (VRP)* and its numerous variants have been the subject of much research in the past few decades [47, 15]. The main purpose of this paper is to show the merits of distributed optimization in solving MDVRPs where each depot is controlled by a separate company. Special emphasis is put on protecting the privacy of the different contractors.

The rest of the paper is structured as follows. Section 2 first presents related work on the VRP and DCOPs. Section 3 then defines the problem, shows how it can be formulated as a DCOP, and solved using the DPOP algorithm. Section 4 discusses the privacy issues in DCOPs, and describes the P-DPOP algorithm that was designed to address these issues. Section 5 presents some other, *asynchronous* DCOP algorithms, and Section 6 compares algorithm performances based on benchmark problems. Section 7 finally concludes.

## 2 RELATED WORK

Section 2.1 first summarizes some of the previous work on solving the Multiple-Depot Vehicle Routing Problem. Section 2.2 then reviews the literature on Distributed Constraint Optimization.

### 2.1 The Multiple-Depot Vehicle Routing Problem

The *Vehicle Routing Problem* [47] is a well-known NP-hard problem in Operations Research: optimal routes must be designed for a fleet of vehicles so as to serve customers while satisfying certain constraints, e.g. on capacity or on route length. Variants of the problem have been proposed [15]; this paper focuses on the *multiple-depot* version.

Laporte *et al.* [23] proposed the first complete MDVRP algorithm, based on integer linear programming. Most algorithms proposed later are approximate heuristics that use a two-phase approach: 1) assign customers to depots, and 2) solve the resulting VRP. One of the first such heuristics was introduced by Chao *et al.* [6]. Other work proposed to use tabu search [40, 8], genetic clustering [46], large neighborhood search [39], or hybrid genetic algorithms [22].

Previous work also applied *Multi-Agent Systems* to solve various flavors of the VRP [9]. Some of this work is not truly applicable to the Distributed MDVRP we consider in this paper, because it uses *virtual* agents, simulated on a central computer that knows the whole problem; this includes memetic algorithms [4], or ant colony optimization [41]. A more relevant thread of research applied the *Contract Net Protocol* [42, 13] to solve VRPs in a truly distributed manner.

This paper proposes a comparable approach, but uses Distributed Constraint Optimization to solve the distributed problem of assigning customers to depots, and focuses on *privacy*. For the VRP part of the problem, we use an off-the-shelf, centralized algorithm.

<sup>1</sup> Artificial Intelligence Laboratory (LIA), École Polytechnique Fédérale de Lausanne (EPFL), Switzerland – email: [firstname.lastname@epfl.ch](mailto:firstname.lastname@epfl.ch)

## 2.2 Distributed Constraint Optimization

Distributed Constraint Optimization is used to model multi-agent optimization problems, in which problem knowledge is *distributed*: initially no agent has a global overview of the overall problem.

**Definition 1 (DCOP)** A discrete Distributed Constraint Optimization Problem is defined as a tuple  $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ , where:

- $\mathcal{A} = \{a_1, \dots, a_{|\mathcal{A}|}\}$  is a set of agents;
- $\mathcal{X} = \{x_1, \dots, x_n\}$  are variables, each owned by an agent;
- $\mathcal{D} = \{D_1, \dots, D_n\}$  is a set of finite domains for the variables such that  $x_i$  takes values in  $D_i$ ;
- $\mathcal{C} = \{c_1, \dots, c_m\}$  is a set of soft constraints, where each constraint is a function  $c_i : D_{i_1} \times \dots \times D_{i_l} \rightarrow \mathbb{R}$  that defines a cost for each combination of assignments to a subset of variables.

A solution to the DCOP is an assignment to all variables that minimizes the sum of all costs  $\sum_i c_i$ .

Hard constraints that allow or disallow specific variable assignments can be modeled as soft constraints, using large positive costs. Any agent knows all the constraints expressed over at least one of its variables, but does not know any constraint that it is not involved in.

The DCOP formalism has been applied to a wide variety of problem domains, such as distributed meeting scheduling, sensor networks, or combinatorial auctions. Ottens *et al.* [32] also applied it to a distributed vehicle pickup and delivery problem.

Most DCOP algorithms are complete and based on distributed search. Some use a linear ordering of the variables (SynchBB [21], OptAPO [30], AFB [16]). Others use a *pseudo-tree* ordering to better exploit problem structure (ADOPT [31], DPOP [34], NCBB [7]). DPOP (Section 3.2.2) is the only one based on dynamic programming instead of search. Because DCOPs are NP-hard, approximate algorithms have also been proposed (DSA [49], MGM-2 [29]).

Several metrics can be used to assess the performances of complete DCOP algorithms. Since they are message-passing algorithms, two important metrics are the number of messages exchanged, and the total amount of information exchanged (as message sizes may vary significantly from one algorithm to another). Runtime can be estimated in terms of wall clock time (if each agent has a dedicated processor), number of *Non-Concurrent Constraint Checks* [17], or using *simulated time* [45].

Finally, the framework of *DCOP under stochastic uncertainty* [25] was recently proposed to model multi-agent optimization problems involving uncertain, stochastic problem data. This extended formalism can be applied for instance to distributed VRPs in which customer demands are stochastic.

## 3 PROBLEM DEFINITION

Section 3.1 first formally defines the problem; Section 3.2 then describes how it can be reformulated into a DCOP.

### 3.1 Distributed Multiple-Depot VRP

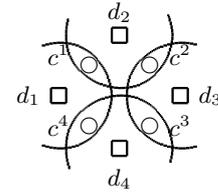
The Distributed Multiple-Depot VRP is simply a distributed version of the Multiple-Depot VRP, in which each depot is controlled by a specific company. The problem is defined more formally as follows.

**Definition 2 (DisMDVRP)** The Distributed, Multiple-Depot, Vehicle Routing Problem consists of a set  $D = \{d_1, \dots, d_{n_D}\}$  of depots in the Euclidian plane, each controlled by a different delivery

company. Each company owns a common number  $n_V$  of vehicles, with a common maximum load  $Q_{\max}$  and a common maximum route length  $L_{\max}$ .<sup>2</sup> The companies must serve a set  $C = \{c^1, \dots, c^{n_C}\}$  of customers at known locations, so that each customer  $c_i$  has a demand  $q_i \in \mathbb{N}$  and must be served by exactly one vehicle. Each vehicle must come back to its initial depot at the end of its route.

In addition to the usual MDVRP constraints, each company has a common visibility radius  $R \leq L_{\max}/2$  that defines the boundaries of its knowledge of the overall DisMDVRP. The company owning depot  $d_i$  is only aware of the customers that are within distance  $R$  of  $d_i$ , and only knows another company if their areas of visibility overlap.

The goal is for the companies to come to an agreement on who should serve which customers, using which vehicle routes, so as to serve all visible customers at minimal total route length.



**Figure 1.** A simple DisMDVRP with four depots and four customers.

Figure 1 shows a DisMDVRP instance with 4 depots and 4 customers. Each depot is at distance 1 of its two closest customers, and owns  $n_V = 2$  vehicles, with  $Q_{\max} = 1$  and  $L_{\max} = 2.5$ , so that a vehicle can only serve one customer, of common demand  $q = 1$ .

The visibility radius is  $R = 1.25$ , such that, for instance, depot  $d_1$  only knows customers  $c^1$  and  $c^4$ , and ignores the existence of  $c^2$  and  $c^3$ . It also knows the two companies  $d_2$  and  $d_4$ , and will have to coordinate with each of them to decide who serves  $c^1$  and  $c^4$ , but  $d_2$  and  $d_4$  do not know about each other, and neither do  $d_1$  and  $d_3$ .

The local nature of the agents' problem knowledge adds a new challenge to the traditional MDVRP. Any algorithm proposed to solve such a distributed, multi-agent problem will necessarily involve exchanging information between the agents, in order to efficiently assign customers to depots. While the performances of traditional MDVRP algorithms are usually compared based on runtime and solution quality, DisMDVRP algorithms should also be evaluated based on the amount of information they require the agents to exchange.

A simplistic approach could consist in the companies electing a leader, sharing with it all their respective information about the problem, and letting it solve the resulting MDVRP in a centralized fashion. This approach would have a number of important drawbacks:

- The companies must be willing to disclose all of their problem knowledge to an elected leader. This is unlikely to be the case in real life: even if they work under the same franchise, delivery companies would want to protect their trade secrets, such as the exact nature of their fleets, or their internal costs.
- This would be computationally very inefficient, since the entire computational burden would lie on the leader. An efficient DisMDVRP algorithm should distribute the effort between the participants in order to leverage their computational power.

<sup>2</sup> These restrictions are only made for simplicity and could be removed without making our approach invalid.

- This would also introduce a bottleneck in network bandwidth usage, since all agents must report their information to the leader.
- The algorithm would have a poor response time to changes in the problem, since the information about the changes would first have to propagate to the leader, which would have to find a new solution to the overall problem, and notify the other companies. A distributed approach should allow agents to directly react to changes, particularly if they only have a local impact.
- The leader becomes a central point of failure of the algorithm.

To solve the distributed part of the DisMDVRP, namely the assignment of customers to depots, we propose to use the DCOP approach, which has been successful at solving such multi-agent optimization problems. While we use a specific flavor of the VRP, this approach also applies to other variants of the problem, as long as they involve the necessity for multiple agents to coordinate their decisions. In particular, the issue of protecting the companies' private knowledge about the problem can become more important in variants of the VRP involving heterogeneous vehicle fleets.

## 3.2 Problem reformulation as a DCOP

Section 3.2.1 first presents how the DisMDVRP can be formulated as a DCOP. Section 3.2.2 then presents the DPOP algorithm.

### 3.2.1 Formal DCOP model

A DisMDVRP can be reformulated as a DCOP, with one agent per company/depot. Depot  $d_i$  owns one Boolean variable  $x_i^j$  for each visible customer  $c^j$ , modeling its decision to serve the customer ( $x_i^j = 1$ ) or not ( $x_i^j = 0$ ). The DCOP constraints are of two types:

1. For each customer  $c^j$ , if the set  $D^j$  of depots within visible distance  $R$  of  $c^j$  is non-empty, then the following  $|D^j|$ -ary hard constraint enforces that  $c^j$  must be assigned to exactly one depot:

$$\sum_{d_i \in D^j} x_i^j = 1. \quad (1)$$

2. For each depot  $d_i$ , if the set  $C_i = \{c^{j_1}, \dots, c^{j_n}\}$  of visible customers is non-empty, then the following  $|C_i|$ -ary soft constraint represents the cost of the optimal solution to  $d_i$ 's own VRP, as a function of whether it serves each customer in  $C_i$ :

$$vrp_i(x_i^{j_1}, \dots, x_i^{j_n}) = \text{optimal cost of } d_i\text{'s VRP} \in [0, \infty]. \quad (2)$$

Notice that evaluating a constraint  $vrp_i$  on a given assignment of values to its variables requires solving a (centralized) single-depot VRP. Such an evaluation can be seen as a call by agent  $d_i$  to a local subroutine, which does not require exchanging additional information with other agents, and can be implemented using any existing VRP algorithm. This modeling approach decouples the distributed, master problem of assigning customers to agents, solved using a DCOP algorithm, from the local, slave problem of routing a given agent's vehicles, solved using a traditional VRP algorithm.

The DCOP model for the DisMDVRP instance in Figure 1 is illustrated in Figure 2, in the form of a *constraint graph*. The nodes in the graph correspond to the variables in the DCOP, and constraints are represented by edges between the variables in the constraint scope. In this example, each agent (represented by a dotted box) owns two variables, whose respective values model whether or not the company is assigned each of the two customers in its area of visibility. Internally, each agent has to solve a VRP problem, represented by a  $vrp_i$  constraint over its variables. Inter-agent *sum* constraints enforce that each customer be served by exactly one company.

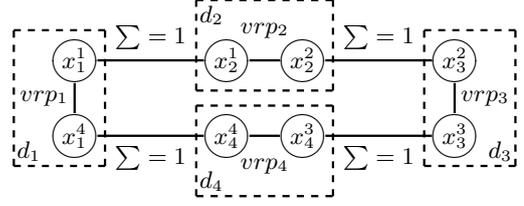


Figure 2. The DCOP constraint graph corresponding to Figure 1.

### 3.2.2 The DPOP algorithm

DPOP [34] is an instance of the general *bucket elimination scheme* from [10], adapted for the distributed case. It proceeds in four phases: the first two construct a *pseudo-tree* ordering of the variables (i.e. a tree allowing *back edges* between a variable and an ancestor), the third propagates information bottom-up along the pseudo-tree, and the last top-down phase infers optimal values for the variables.

**Leader election:** The agents first run a leader election algorithm to choose one variable as the root of the pseudo-tree ( $x_1^4$  in Figure 3). Each variable is initially assigned a score; the root is the one with the maximum score, computed by viral propagation. Various scoring heuristics have been proposed to generate pseudo-trees with specific good properties; the one that elects the most connected variable tends to produce low-width pseudo-trees.

**DFS traversal:** The root variable then initiates a depth-first traversal of the constraint graph to compute a spanning tree, using a token-passing algorithm described in [11]. The *tree edges* of the resulting pseudo-tree correspond to the edges in the spanning tree, and the *back edges* to the remaining edges in the constraint graph. The pseudo-tree in Figure 3 consists of a single branch; however, in the more general case, the pseudo-tree may have multiple branches, each executing the remaining two phases in parallel.

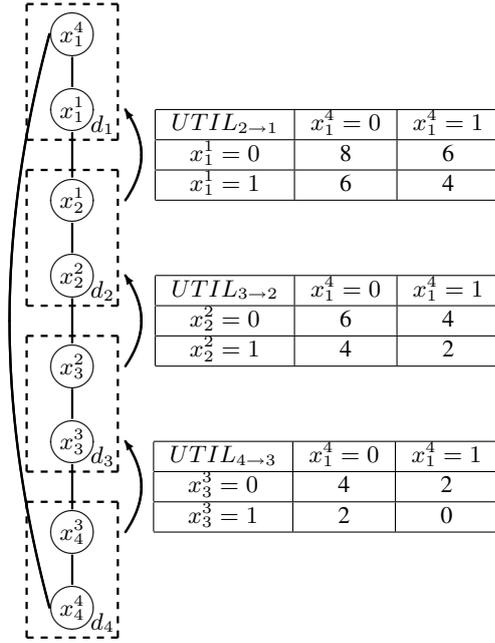
**UTIL propagation:** During this bottom-up phase, each variable sends a *UTIL* message to its parent with the aggregated, optimal cost achievable by its whole subtree, as a function of some ancestor variables. In Figure 3, the message  $UTIL_{3 \rightarrow 2}$  sent by  $d_3$  contains the optimal cost of  $d_4$  and  $d_3$ 's combined VRPs, as a function of whether  $d_2$  serves  $c^2$  and whether  $d_1$  serves  $c^4$ .

**VALUE propagation:** After adding the costs of its local VRP to  $UTIL_{2 \rightarrow 1}$ , agent  $d_1$  can infer the optimal values for its variables that minimize the overall objective function. In the example, due to the symmetries in Figure 1, all combinations of decisions have cost 8. Agent  $d_1$  can choose to serve both customers  $c^1$  and  $c^4$  ( $x_1^1 = x_1^4 = 1$ ); these decisions are then sent downwards in a top-down phase, after which all variables have been assigned optimal values. For instance, agent  $d_2$  can infer from  $d_1$ 's decisions and from  $UTIL_{3 \rightarrow 2}$  that its optimal decisions are  $x_2^1 = 0$  and  $x_2^2 = 1$ .

DPOP's computational bottleneck is in the sizes of the UTIL messages: they are exponential in the width of the pseudo-tree. [34]

## 4 PROVIDING PRIVACY GUARANTEES

In many real-life situations, companies want to keep their problem knowledge private. While the DCOP approach does not require full disclosure to one agent, agents can still make inferences based on received messages. Section 4.1 presents the types of information that can be leaked, and previous work on this issue. Section 4.2 then describes a recent variant of DPOP that improves the state of the art.



**Figure 3.** A possible pseudo-tree for the constraint graph in Figure 2, with the bottom-up UTIL messages exchanged by the agents in DPOP.

#### 4.1 Privacy leaks in DCOP algorithms

Faltings *et al.* [11] defined *semi-private information* as the part of an agent’s knowledge that might be considered private, but that can *inevitably* be leaked by the solution chosen to the DCOP. In Figure 2, assume that, after running a DCOP algorithm,  $d_1$  chooses  $x_1^1 = 0$ . Although the algorithm might not explicitly reveal to  $d_1$  the value that  $d_2$  chose for its variable  $x_2^1$ , agent  $d_1$  can infer it from the constraint  $x_1^1 + x_2^1 = 1$ . In other words, since  $c^1$  must be served by either  $d_1$  or  $d_2$ , if  $d_1$ ’s decision is to not serve  $c^1$ , then  $d_1$  can infer the identity of the agent serving  $c^1$ . However, if three companies can serve a customer, the identity of the assigned agent is no longer semi-private information, and may or may not be protected by the algorithm.

Putting aside semi-private information, Faltings *et al.* [11] also defined four types of privacy that may be desirable, based on four categories of problem knowledge that agents may want to protect.

**Agent privacy:** No agent should discover the existence of agents it does not share constraints with. In Figure 2,  $d_1$  and  $d_3$  should not discover each other’s existence. Linear variable orderings (SynchBB, AFB) violate agent privacy (Table 1): they can require message exchanges between non-neighboring variables. Pseudotrees (DPOP, ADOPT, NCBB) do not, but some agent privacy can still be leaked. For instance, the presence of variable  $x_1^4$  in the message  $UTIL_{4 \rightarrow 3}$  reveals to agent  $d_3$  the existence of agent  $d_1$ .

**Topology privacy:** No agent should discover topological constructs (variables, constraints, cycles...) it is not involved in. Agent  $d_1$  must not learn how many customers  $d_2$  can see. This can be leaked by the variable ordering heuristic: the *most connected* heuristic reveals to all agents the degrees of their neighboring variables.

**Constraint privacy:** No agent should discover the costs in any constraint it is not involved in. DPOP violates constraint privacy: the message  $UTIL_{4 \rightarrow 3}$  reveals to agent  $d_3$  the optimal costs of

agent  $d_4$  for serving customers  $c^3$  and  $c^4$ .

**Decision privacy:** If no agent can learn the values taken by another agent’s variables, then the algorithm protects *full decision privacy*. *Partial decision privacy* allows agents to learn the values only of neighboring variables. DPOP violates both, since, in Figure 3, the VALUE propagation phase reveals to all agents the value of  $x_1^4$ .

Most of the literature on privacy in DCOP focuses on constraint privacy. Frameworks have been proposed to measure leaks of constraint privacy [14, 28, 19]. Methods have been developed to partially protect constraint privacy in search-based algorithms [5] and in DPOP [18]. Cryptography has also been applied to fully protect constraint privacy [48, 43], but the algorithms do not scale to realistic problems. The following section describes a more recent algorithm that scales better, while addressing all four types of privacy.

**Table 1.** The privacy guarantees of various complete DCOP algorithms.

	agent	topology	constraint	decision
SynchBB, OptAPO, AFB				
DPOP, ADOPT, NCBB	partial	partial		
P-DPOP	full	full	partial	partial
P <sup>2</sup> -DPOP	full	full	full	full

#### 4.2 The P-DPOP algorithm

P-DPOP [11] is a variant of DPOP that guaranties all four types of privacy defined in Section 4.1, through the following techniques:

**Codenames:** To protect agent privacy, P-DPOP obfuscates names and values of back-edge variables using *codenames* (Table 3).

**Anonymous leader election:** To provide topology privacy, agents are allowed to lie a random number of times during the viral propagation of scores, to hide the position of the root. A randomized scoring heuristic is used to avoid leaking topological information.

**Cost obfuscation:** Constraint privacy is leaked by the message  $UTIL_{4 \rightarrow 3}$ , which reveals to  $d_3$  the dependency of  $d_4$ ’s optimal cost function on  $x_4^1$ . To obfuscate this dependency,  $d_4$  adds a secret, large number to each column; for instance 1234 for  $x_4^1 = 0$ , and 2345 for  $x_4^1 = 1$  (Table 3). Agents  $d_3$  and  $d_2$  can then perform their operations on the obfuscated costs without decrypting them; only  $d_1$  does so by subtracting the secret numbers.

**Table 3.** Obfuscated message  $UTIL_{4 \rightarrow 3}$  in P-DPOP.

	$\alpha = \beta$	$\alpha = \gamma$
$x_3^3 = 0$	1238	2347
$x_3^3 = 1$	1236	2345

Silaghi *et al.* [44] showed that most DCOP algorithms also violate constraint privacy through their runtime, because it depends on the *tightness* of the constraints, i.e. how soft or hard they are. Since they are based on dynamic programming instead of search, DPOP and P-DPOP do not suffer from this flaw.

Finally, codenames also guarantee *partial* decision privacy. *Full* decision privacy is however violated, since each variable learns the value of its upper neighbors in the pseudo-tree. The P<sup>2</sup>-DPOP algorithm [26] provides a patch to this privacy leak (Table 1), but this comes at a high price in terms of memory and runtime complexity.

**Table 2.** Experimental results for various DCOP algorithms on some of the Cordeau instances from [2]. ADOPT timed out on all instances.

prob.	D	R	C	wall clock time (ms)					information exchanged (bytes)				
				SynchBB	DPOP	O-DPOP	ASO-DPOP	P-DPOP	SynchBB	DPOP	O-DPOP	ASO-DPOP	P-DPOP
p01	4	14	30	<b>535</b>	802	712	850	1 437	23 900	<b>6 525</b>	<b>6 427</b>	8 405	28 616
		16	32	<b>1 177</b>	2 182	1 934	2 237	4 035	53 507	<b>7 772</b>	<b>7 984</b>	11 405	45 612
		18	35	9 061	<b>6 939</b>	7 232	<b>6 881</b>	11 141	992 643	<b>10 099</b>	11 505	19 001	186 823
		20	38	-	<b>103 153</b>	-	-	-	-	<b>22 249</b>	-	-	-
p03	5	10	35	<b>224</b>	373	385	397	701	7 803	<b>1 981</b>	<b>1 911</b>	2 338	7 731
		12	40	<b>273</b>	636	713	813	1 266	8 278	<b>2 450</b>	<b>2 497</b>	3 375	13 225
		14	46	<b>1 443</b>	4 028	3 813	4 037	5 804	47 671	<b>7 543</b>	<b>7 840</b>	10 979	45 059
		16	58	<b>55 320</b>	99 209	-	-	268 804	10 192 397	<b>12 824</b>	-	-	14 603 058
p11	5	22	45	<b>323</b>	846	985	975	1 437	8 171	<b>1 981</b>	<b>1 911</b>	2 338	7 867
		24	53	<b>338</b>	5 763	6 753	6 996	8 932	9 133	<b>2 450</b>	<b>2 497</b>	3 232	95 051
		26	60	<b>449</b>	19 186	45 370	43 872	33 513	9 365	<b>2 450</b>	<b>2 497</b>	3 435	13 055
		28	68	<b>751</b>	91 185	-	-	165 211	12 905	<b>3 374</b>	-	-	59 348
		30	72	<b>875</b>	197 478	-	-	476 268	14 981	<b>3 970</b>	-	-	352 032

P-DPOP has the same complexity as DPOP, i.e. it is exponential in the width of the pseudo-tree used. However, because P-DPOP cannot use elaborate heuristics to choose the pseudo-tree that would leak private information, the pseudo-trees produced by P-DPOP have on average a higher width than for DPOP.

## 5 ANY-TIME BEHAVIOR VIA ASYNCHRONY

Although DPOP is a very efficient algorithm in terms of the number of messages that are exchanged, the sizes of the messages are exponential in the width of the pseudo-tree. Faltings *et al.* [12] observed that, in general, one does not need full knowledge of a constraint problem to find the optimal solution. Although this observation was made for centralized problems, it is also valid in a distributed setting. Since DPOP sends information on *all* possible assignments, there is obvious room for improvement. In Open-DPOP (O-DPOP) [36], the UTIL propagation phase is replaced by a new phase in which an agent reports the entries in its UTIL table one by one, and in a best first order (i.e. lowest costs first), until its parent has enough information to determine its own next best assignment. An optimal solution to the DCOP is found when the root finds its first best assignment.

One disadvantage of both DPOP and O-DPOP is that they are not any-time algorithms, i.e. there are no intermediate solutions and a decision can only be made at termination. To mitigate this problem, ASO-DPOP [32] was proposed. The difference between O-DPOP and ASO-DPOP is twofold. First of all, the UTIL propagation phase and the VALUE propagation phase run concurrently, allowing agents to make decisions before the termination of the algorithm. Furthermore, agents are also allowed to report partial information, speeding up the propagation of information through the network, enabling the agents to make better informed intermediate decisions. The resulting privacy properties are the same as for DPOP (Table 1).

## 6 EXPERIMENTAL RESULTS

Simulations were run with the FRODO platform [27], on the Cordeau MDVRP benchmarks from [2]. To solve the local VRPs, we used the OR-Objects library [1], with the best out of randomized versions of the Clarke & Wright and the Gillett & Miller algorithms (5 iterations, strength 10) as construction algorithm, and the 3-Opt TSP improvement algorithm [24, 3]. The simulations were carried out on a 2.9 GHz, 16-core PC (i.e. one agent per core) with 2 GB of Java heap, and a timeout of 15 min. The results are presented in Table 2.

One can first observe that SynchBB is almost always faster than all other algorithms, except for two instances with large  $R$ , including one that DPOP was the only algorithm to solve. This can be explained by the fact that algorithms in FRODO are implemented at the variable level, not at the agent level. This means that variables within the

same agent exchange virtual messages (not counted in the information exchange metric), as if they were independent agents. Therefore, DPOP-based algorithms perform *Dynamic Programming (DP)* also inside each agent, while SynchBB performs *Branch & Bound (B&B)*, which is more efficient in the centralized setting.

At the inter-agent level, however, DP proves to be dramatically more efficient than B&B to reduce network bandwidth usage: all DPOP-based algorithms (except P-DPOP) send much less information than SynchBB. On p03 and for  $R = 16$ , DPOP even sends three orders of magnitude less information than SynchBB, and is only twice slower. This is due to the fact that, by sending few large messages instead of many small messages, DPOP saves on the overhead of sending a single message. Future work should investigate the use of a DP/search hybrid such as *PC-DPOP* [37].

Also, note that in these simulations, agents exchanged messages simply by accessing shared memory space. Therefore, the runtime metric does not take into account the cost of sending messages. It is very probable that, in a more realistic setting in which agents would communicate over a network, SynchBB's runtime performance would be significantly degraded by the large amounts of information (and the large numbers of messages) it exchanges.

When it comes to the performance of O-DPOP in terms of information exchange, O-DPOP actually sends just as much information as DPOP. This is probably due to the fact that, while O-DPOP needs to explore fewer combinations of assignments to the variables, and does not report all the information contained in DPOP's UTIL messages, it has to pay the overhead of sending a *context* in each of its UTIL messages. As far as ASO-DPOP is concerned, it sends only slightly more information; future work should study its convergence properties compared to other any-time algorithms like SynchBB.

Finally, the price that P-DPOP has to pay compared to DPOP in order to provide privacy guarantees is only moderate as far as runtime is concerned. P-DPOP however sends significantly more information.

## 7 CONCLUSION AND FUTURE WORK

In this paper, we have introduced the Distributed, Multiple-Depot Vehicle Routing Problem, in which each depot is controlled by a separate company with only partial information. We have proposed to use Distributed Constraint Optimization to formalize the distributed assignment of customers to depots. We have evaluated the performances of various DCOP algorithms on distributed versions of existing MDVRP benchmarks, in terms of runtime, information exchange, and any-time behavior. We have also put a special emphasis on the guarantees that can be provided in terms of protecting the companies' private information.

Future work should evaluate the DCOP approach against the ap-

proach based on the CNP. One could also investigate *dynamic* MD-VRPs in which the companies must react to changes such as arrivals of new customers; this could be done using *S-DPOP* [35]. Another thread of research is designing *incentives* for the companies to honestly report their true costs. Petcu *et al.* [38] already proposed the *M-DPOP* algorithm as an incentive-compatible DCOP algorithm.

## REFERENCES

- [1] Operations Research – Java Objects. <http://opsresearch.com>.
- [2] VRP Web. <http://neo.lcc.uma.es/radi-aeb/WebVRP/>.
- [3] *Local Search in Combinatorial Optimization*, eds., Emile Aarts and Jan Karel Lenstra, Wiley, 1st edn., 1997.
- [4] Dariusz Barbucha and Piotr Jędrzejowicz, 'An agent-based approach to vehicle routing problem', *WASET*, **26**, 36–41, (2007).
- [5] Ismel Brito and Pedro Meseguer, 'Distributed forward checking may lie for privacy', In Pearce [33].
- [6] I-Ming Chao, Bruce L. Golden, and Edward Wasil, 'A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions', *American Journal of Mathematical and Management Sciences*, **13**(3–4), 371–406, (1993).
- [7] Anton Chechetka and Katia Sycara, 'No-commitment branch and bound search for distributed constraint optimization', in *Proceedings of AAMAS'06*, pp. 1427–1429, Hakodate, Japan, (May 8–12 2006).
- [8] Jean-François Cordeau, Michel Gendreau, and Gilbert Laporte, 'A tabu search heuristic for periodic and multi-depot vehicle routing problems', *Networks*, **30**(2), 105–119, (December 7 1998).
- [9] P. Davidson, L. Henesey, L. Ramstedt, J. Tornquist, and F. Wernstedt, 'An analysis of agent-based approaches to transport logistics', *Transportation Research Part C*, **13**, 255–271, (2005).
- [10] Rina Dechter, *Constraint Processing*, Morgan Kaufmann, May 5 2003.
- [11] Boi Faltings, Thomas Léauté, and Adrian Petcu, 'Privacy guarantees through distributed constraint satisfaction', in *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'08)*, pp. 350–358, (December 9–12 2008).
- [12] Boi Faltings and Santiago Macho-Gonzalez, 'Open constraint programming', *Artificial Intelligence*, **161**(1–2), 181–208, (January 2005).
- [13] Klaus Fischer, Jörg P. Müller, and Markus Pischel, 'Cooperative transportation scheduling: An application domain for DAI', *Applied Artificial Intelligence*, **10**(1), 1–34, (February 1996).
- [14] M. S. Franzin, E. C. Freuder, Francesca Rossi, and R. Wallace, 'Multi-agent constraint systems with preferences: Efficiency, solution quality, and privacy loss', *Computational Intel.*, **20**(2), 264–286, (May 2004).
- [15] Michel Gendreau, Jean-Yves Potvin, Olli Bräysy, Geir Hasle, and Arne Løkketangen, 'Metaheuristics for the vehicle routing problem and its extensions : A categorized bibliography', Technical Report 2007-27, CIRRELT, (August 2007).
- [16] Amir Gershman, Amnon Meisels, and Roie Zivan, 'Asynchronous forward-bounding for distributed constraints optimization', in *Proceedings of ECAI'06*, pp. 103–107, (August 29–September 1 2006).
- [17] Amir Gershman, Roie Zivan, Tal Grinshpoun, Alon Grubshtein, and Amnon Meisels, 'Measuring distributed constraint optimization algorithms', in *Proceedings of DCR'08*, pp. 17–24, (May 13 2008).
- [18] Rachel Greenstadt, Barbara Grosz, and Michael D. Smith, 'SSDPOP: Using secret sharing to improve the privacy of DCOP', In Pearce [33].
- [19] Rachel Greenstadt, Jonathan P. Pearce, and Milind Tambe, 'Analysis of privacy loss in distributed constraint optimization', in *Proceedings of AAAI'06*, pp. 647–653, (2006).
- [20] Katsutoshi Hirayama, William Yeoh, and Roie Zivan, eds. *Proc. Distributed Constraint Reasoning Workshop (DCR'09)*, July 13 2009.
- [21] Katsutoshi Hirayama and Makoto Yokoo, 'Distributed partial constraint satisfaction problem', in *CP'97*, volume 1330, pp. 222–236, (1997).
- [22] William P. C. Ho, George T. S. Ho, Ping Ji, and Henry C. W. Lau, 'A hybrid genetic algorithm for the multi-depot vehicle routing problem', *Engineering Applications of AI*, **21**(4), 548–557, (June 2008).
- [23] G. Laporte, Y. Nobert, and D. Arpin, 'Optimal solutions to capacitated multidepot vehiclerouting problems', *Congressus Numerantium*, **44**, 283–292, (1984).
- [24] *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, eds., E. L. Lawler, Jan Karel Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, Wiley, August 1985.
- [25] Thomas Léauté and Boi Faltings, 'E[DPOP]: Distributed constraint optimization under stochastic uncertainty using collaborative sampling', In Hirayama *et al.* [20], pp. 87–101.
- [26] Thomas Léauté and Boi Faltings, 'Privacy-preserving multi-agent constraint satisfaction', in *Proc. of the 2009 IEEE Intl Conf. on PrivAcy, Security, riSk and Trust (PASSAT'09)*, pp. 17–25, (August 29–31 2009).
- [27] Thomas Léauté, Brammert Ottens, and Radoslaw Szymanek, 'FRODO 2.0: An open-source framework for distributed constraint optimization', In Hirayama *et al.* [20], pp. 160–164. <http://liawww.epfl.ch/frodo/>.
- [28] Rajiv T. Maheswaran, Jonathan P. Pearce, Emma Bowring, Pradeep Varakantham, and Milind Tambe, 'Privacy loss in distributed constraint reasoning: A quantitative framework for analysis and its applications', *Autonom. Agents and Multi-Agent Systems*, **13**(1), 27–60, (July 2006).
- [29] Rajiv T. Maheswaran, Jonathan P. Pearce, and Milind Tambe, 'Distributed algorithms for DCOP: A graphical-game-based approach', in *Proceedings of ISCA PDCS'04*, pp. 432–439, (September 15–17 2004).
- [30] Roger Mailler and Victor R. Lesser, 'Solving distributed constraint optimization problems using cooperative mediation', in *Proceedings of AAMAS'04*, volume 1, pp. 438–445, (July 19–23 2004).
- [31] Pragnesh J. Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo, 'ADOPT: Asynchronous distributed constraint optimization with quality guarantees', *Artificial Intelligence*, **161**, 149–180, (2005).
- [32] Brammert Ottens and Boi Faltings, 'Coordinating agent plans through distributed constraint optimization', in *Proceedings of the ICAPS'08 Multiagent Planning Workshop (MASPLAN'08)*, (September 14 2008).
- [33] Jonathan P. Pearce, ed. *Proceedings of the 9th Intl Workshop on Distributed Constraint Reasoning (CP-DCR'07)*, September 23 2007.
- [34] Adrian Petcu and Boi Faltings, 'DPOP: A Scalable Method for Multi-agent Constraint Optimization', in *Proc. 19th Intl Joint Conf. on Artificial Intelligence (IJCAI'05)*, pp. 266–271, (July 31 – August 5 2005).
- [35] Adrian Petcu and Boi Faltings, 'S-DPOP: Superstabilizing, fault-containing multiagent combinatorial optimization', in *Proc. of the 20th National Conference on A.I. (AAAI'05)*, pp. 449–454, (July 9–13 2005).
- [36] Adrian Petcu and Boi Faltings, 'O-DPOP: An algorithm for open/distributed constraint optimization', in *Proc. of the 21st National Conf. on Artificial Intelligence (AAAI'06)*, pp. 703–708, (2006).
- [37] Adrian Petcu, Boi Faltings, and Roger Mailler, 'PC-DPOP: A new partial centralization algorithm for distributed optimization', in *Proceedings of IJCAI'07*, pp. 167–172, (January 6–12 2007).
- [38] Adrian Petcu, Boi Faltings, and David C. Parkes, 'M-DPOP: Faithful distributed implementation of efficient social choice problems', *Journal of Artificial Intelligence Research (JAIR)*, **32**, 705–755, (July 2008).
- [39] David Pisinger and Stefan Ropkea, 'A general heuristic for vehicle routing problems', *Computers and OR*, **34**(8), 2403–2435, (August 2007).
- [40] Jacques Renaud, Gilbert Laporte, and Faye F. Boctor, 'A tabu search heuristic for the multi-depot vehicle routing problem', *Computers and Operations Research*, **23**(3), 229–235, (March 1996).
- [41] A.E. Rizzoli, R. Montemanni, E. Lucibello, and L.M. Gambardella, 'Ant colony optimization for real-world vehicle routing problems: From theory to applications', *Swarm Intelligence*, **1**, 135–151, (Sept. 2007).
- [42] Tuomas Sandholm, 'An implementation of the contract net protocol based on marginal cost calculations', in *AAAI'93*, pp. 256–262, (1993).
- [43] Marius-Călin Silaghi, Boi Faltings, and Adrian Petcu, 'Secure combinatorial optimization simulating DFS tree-based variable elimination', in *Proc. of the 9th Intl Symp. on A.I. and Math.*, (January 4–6 2006).
- [44] Marius-Călin Silaghi and Debasis Mitra, 'Distributed constraint satisfaction and optimization with privacy enforcement', in *Proceedings of IAT'04*, pp. 531–535, (September 20–24 2004).
- [45] Evan A. Sultanik, Robert N. Lass, and William C. Regli, 'DCOPolis: A framework for simulating and deploying distributed constraint optimization algorithms', In Pearce [33].
- [46] Sam R. Thangiah and Said Salhi, 'Genetic clustering: An adaptive heuristic for the multidepot vehicle routing problem', *Applied Artificial Intelligence*, **15**(4), 361–383, (2001).
- [47] *The Vehicle Routing Problem*, eds., Paolo Toth and Daniele Vigo, SIAM, 2001.
- [48] Makoto Yokoo, Koutarou Suzuki, and Katsutoshi Hirayama, 'Secure distributed constraint satisfaction: Reaching agreement without revealing private information', *Artif. Intel.*, **161**(1–2), 229–245, (Jan. 2005).
- [49] 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', *JAIR*, **161**(1–2), 55–87, (January 2005).