Model and Algorithm for Dynamic Multi-Objective Distributed Optimization

Maxime Clement[1], Tenda Okimoto[3,4], Tony Ribeiro[2], and Katsumi Inoue[4]

[1] Pierre and Marie Curie University (Paris 6), Paris, France
maxime.clement@etu.upmc.fr
[2] The Graduate University for Advanced Studies, Tokyo, Japan
[3] Transdisciplinary Research Integration Center, Tokyo, Japan
[4] National Institute of Informatics, Tokyo, Japan
{tony-ribeiro,tenda,inoue}@nii.ac.jp

Abstract. Many problems in multi-agent systems can be represented as a Distributed Constraint Optimization Problem (DCOP) where the goal is to find the best assignment to variables in order to minimize the cost. More complex problems including several criteria can be represented as a Multi-Objective Distributed Constraint Optimization Problem (MO-DCOP) where the goal is to optimize several criteria at the same time. However, many problems are subject to changes over time and need to be represented as dynamic problems. In this paper, we formalize the Dynamic Multi-Objective Distributed Constraint Optimization Problem (DMO-DCOP) and introduce the first algorithm called DMOBB to handle changes in the number of objectives.

1 Introduction

A *Distributed Constraint Optimization Problem* (DCOP) [6, 8, 9] is a fundamental problem that can formalize various applications related to multi-agent cooperation. A DCOP consists of a set of agents, each of which needs to decide the value assignment of its variables so that the sum of the resulting costs is minimized. In the last decade, various algorithms have been developed to efficiently solve DCOPs, e.g., ADOPT [8], BnB-ADOPT [11], DPOP [9], and OptAPO [6]. Many multi-agent coordination problems can be represented as DCOPs, e.g., distributed resource allocation problems including sensor networks [4], meeting scheduling [5], and the synchronization of traffic lights [3].

A *Multi-Objective Distributed Constraint Optimization Problem* (MO-DCOP) [2, 7] is an extension of a mono-objective DCOP. Algorithms for solving an MO-DCOP provide all the solutions that offer an interesting trade-off between the different objectives Compared to DCOPs, there exists only two MO-DCOP algorithms, the Bounded Multi-Objective Max-Sum algorithm (B-MOMS) [2] and a distributed search method with bounded cost vectors [7] generalizes ADOPT for MO-DCOPs.

Now consider a dynamic environment where many changes can occur. Many real world problems take place in such environment but the previous models

G. Boella et al. (Eds.): PRIMA 2013, LNAI 8291, pp. 413–420, 2013.

(DCOP and MO-DCOP) do not take changes into account. There exists some works for dynamic DCOPs [1, 12], however, as far as the authors are aware, there exists no work on considering multiple criteria in a dynamic environment.

As an example, imagine a set of unmanned vehicles searching for survivors while maintaining a wireless communication network between them. Those vehicles care about several objectives such as the fuel consumption, the quality of the communication network, the distance to the base, etc. We do not know if changes to the problem might occur but assume the topology of the problem (the agents and their ordering) will not change. Now, while searching for survivors, the vehicles are warned about several dangerous areas in their research zone. The vehicles need to react to this new information in order to avoid dangerous spots and new solutions are required to take every objectives into account.

In this paper, we first propose a Dynamic Multi-Objective Distributed Constraint Optimization Problem (DMO-DCOP) which is the extension of an MO-DCOP and a dynamic DCOP. Furthermore, we develop the first algorithm called Dynamic Multi-Objective Branch and Bound (DMOBB) for solving a DMO-DCOP. This algorithm focuses on a change in the number of objectives and utilizes (i) a special graph structure called a *pseudo-tree*, which is widely used in DCOP algorithms, (ii) a Decentralized Synchronous Branch and Bound. We adapted it for MO-DCOPs and DMO-DCOPs.

The remainder of this paper is organized as follows. Section 2 and 3 provides some preliminaries on DCOPs and MO-DCOPs. Section 4 formalizes a DMO-DCOP and introduces a novel algorithm for solving a DMO-DCOP which can guarantee to find all Pareto solutions. Section 5 empirically evaluates our proposed algorithm. Finally, we conclude in Section 6 and provide some perspectives for future work.

2 DCOP

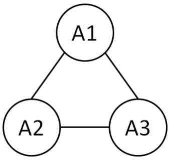
A *Distributed Constraint Optimization Problem* (DCOP) [8, 9] is a fundamental problem that can formalize various applications for multi-agent cooperation.

A DCOP is defined with a set of agents S, a set of variables X, a set of constraint relations C, and a set of reward functions O. An agent i has its own variable x_i. A variable x_i takes its value from a finite, discrete domain D_i. A constraint relation (i, j) means there exists a constraint relation between x_i and x_j. For x_i and x_j, which have a constraint relation, the reward for an assignment $\{(x_i, d_i), (x_j, d_j)\}$ is defined by a reward function $r_{i,j}(d_i, d_j) : D_i \times D_j \to \mathbb{R}^+$. For a value assignment to all variables A, let us denote

$$R(A) = \sum_{(i,j)\in C, \{(x_i,d_i),(x_j,d_j)\}\subseteq A} r_{i,j}(d_i, d_j), \quad (1)$$

where $d_i \in D_i$ and $d_j \in D_j$. Then, an optimal assignment A^* is given as $\arg\max_A R(A)$, i.e., A^* is an assignment that maximizes the sum of the value of all reward functions. A DCOP can be represented using a constraint graph, in which a node represents an agent/variable and an edge represents a constraint.

Table 1. Example of MO-DCOP



A_1	A_2	cost	A_2	A_3	cost	A_1	A_3	cost
a	a	(5,2)	a	a	(0,1)	a	a	(1,0)
a	b	(7,1)	a	b	(2,1)	a	b	(1,0)
b	a	(10,3)	b	a	(0,2)	b	a	(0,1)
b	b	(12,0)	b	b	(2,0)	b	b	(3,2)

3 MO-DCOP

A *Multi-Objective Distributed Constraint Optimization Problem* (MO-DCOP) [2, 7] is the extension of a mono-objective DCOP. An MO-DCOP is defined with a set of agents S, a set of variables X, multi-objective constraints $C = \{C^1, \ldots, C^m\}$, i.e., a set of sets of constraint relations, and multi-objective functions $O = \{O^1, \ldots, O^m\}$, i.e., a set of sets of objective functions. For an objective l ($1 \leq l \leq m$), a cost function $f^l_{i,j} : D_i \times D_j \to \mathbb{R}$, and a value assignment to all variables A, let us denote

$$R^l(A) = \sum_{(i,j)\in C^l,\{(x_i,d_i),(x_j,d_j)\}\subseteq A} f^l_{i,j}(d_i, d_j),\ where\ d_i \in D_i\ and\ d_j \in D_j. \quad (2)$$

Then, the sum of the values of all cost functions for m objectives is defined by a cost vector, denoted $R(A) = (R^1(A), \ldots, R^m(A))$. Finding an assignment that minimizes all objective functions simultaneously is ideal. However, in general, since trade-offs exist among objectives, there does not exist such an ideal assignment. Thus, the optimal solution of an MO-DCOP is characterized by using the concept of *Pareto optimality.* Because of this possible trade-off between objectives, the size of the Pareto front is exponential in the number of agents, i.e., every possible assignment can be a Pareto solution in the worst case. An MO-DCOP can be also represented using a constraint graph.

Definition 1 (Dominance). For an MO-DCOP and two cost vectors $R(A)$ and $R(A')$ obtained by assignments A and A', we say that $R(A)$ *dominates* $R(A')$, denoted by $R(A) \prec R(A')$, iff $R(A)$ is partially less than $R(A')$, i.e., (i) it holds $R^l(A) \leq R^l(A')$ for all objectives l, and (ii) there exists at least one objective l', such that $R^{l'}(A) < R^{l'}(A')$.

Definition 2 (Pareto solution). For an MO-DCOP and an assignment A, we say A is the *Pareto solution*, iff there does not exist another assignment A', such that $R(A') \prec R(A)$.

Definition 3 (Pareto Front). For an MO-DCOP, the *Pareto front* is the set of cost vectors obtained by the Pareto solutions. *Solving an MO-DCOP is to find the Pareto front.*

Example 1 (MO-DCOP). We show a bi-objective DCOP using the example represented with Table 1. The table shows three cost tables among three agents. The Pareto solutions of this problem are $\{\{\{(A_1, a), (A_2, a), (A_3, a)\} \to (6, 3)\}, \{\{(A_1, a), (A_2, b), (A_3, b)\} \to (10, 1)\}\}$.

4 Dynamic Multi-Objective Distributed Constraint Optimization Problem

In this section, we formalize a Dynamic Multi-Objective Distributed Constraint Optimization Problem (DMO-DCOP). Furthermore, we develop the Dynamic Multi-Objective Branch and Bound (DMOBB), the first algorithm for solving a DMO-DCOP and provide its complexity.

4.1 Model

A *Dynamic Multi-Objective Distributed Constraint Optimization Problem* (DMO-DCOP) is the extension of an MO-DCOP. A DMO-DCOP is defined by a sequence of MO-DCOPs.

$$< \mathit{MO\text{-}DCOP}_1, \mathit{MO\text{-}DCOP}_2, ..., \mathit{MO\text{-}DCOP}_k > . \tag{3}$$

In this paper, we assume that

- only the number of objective functions changes,
- the number of agents/variables, domains, and costs for current constraints does not change.

Solving a DMO-DCOP is to find a sequence of Pareto front

$$< PF_1, PF_2, ..., PF_k >, \tag{4}$$

where PF_i $(1 \leq i \leq k)$ is the Pareto front of $\mathit{MO\text{-}DCOP}_i$. Since we do not know how many objective functions will be removed/added in the next MO-DCOP, it is a reactive approach.

Definition 4 (Evolution of the Pareto Front). For an MO-DCOP$_i$ and its corresponding Pareto front PF_i, *adding* objectives to MO-DCOP$_i$ will result in a new Pareto front PF_{i+1} such that for all unique cost vectors in PF_i, one of the assignment yielding this cost will still be a Pareto solution in PF_{i+1}. However, if different assignments yield a same cost in PF_i, there is no guarantee that all assignments will still yield Pareto Solutions in PF_{i+1}. Similarly, in case several objectives are *removed*, there is no guarantee that all Pareto solutions of MO-DCOP$_i$ are also the Pareto solutions in MO-DCOP$_{i+1}$.

4.2 DMOBB Algorithm

To run DMOBB, we first order the agents into a *pseudo-tree* [10].

A pseudo-tree is a special graph structure widely used in DCOP algorithms. In a pseudo-tree, there exists a unique root node, and each non-root node has a parent node. For each node/agent i, we denote the parent node, and children of i as follows:

- $parent_i$, the parent of the agent i.

Algorithm 1. Search Algorithm for agent a_i

```
$i$: integer (agent id)
$children$: list of agents
$PF$: set of pairs of assignment and cost vector (local PF for all $context$)
$currentPF$: set of pairs of assignment and cost vector (local Pareto front for the current $context$)
$PF_c$: set of pairs of assignment and cost vector (children Pareto front)
$context$: vector of integers (ancestors assignment)
$UB$: set of cost vectors (local upper bounds)
$response$: integer
$d_i$: current value from domain $D_i$ being explored
$currentPF \leftarrow \emptyset$
if Root agent then// Root agent
    for each value $d_i$ of $D$ do
        $UB \leftarrow computeUB()$
        send $(d_i, \emptyset, UB)$ // Send Value message
        $response \leftarrow 0$ ;$PF_c \leftarrow \emptyset$
        while $response < |children|$ do // Receive Cost messages
            if message $= (PF_{c_i})$ then
                $PF_c \leftarrow (PF_c \bigoplus PF_{c_i}) + \delta_{assignment \cup d_i}$
                $response \leftarrow response + 1$
        $currentPF \leftarrow (currentPF \uplus PF_c)$
    send TERMINATE to all children
else
    while $message \neq$ TERMINATE do
        $message \leftarrow receive()$ // Receive Termination message
        if $message =$ TERMINATE then
            send TERMINATE to all children
        // Receive Value message
        if $message = (new\_context,\ \gamma_{new\_context},\ UB_p)$ then
            $context \leftarrow new\_context$ ; $currentPF \leftarrow \emptyset$ ; $PF_c \leftarrow \emptyset$
            for each value $d_i$ of $D$ do
                $UB \leftarrow computeUB()$
                $assignment \leftarrow context \cup d_i$
                $\gamma_{assignment} \leftarrow \delta_{assignment} + \gamma_{new\_context}$
                if $\gamma_{assignment}$ is not dominated by $UB$ then // Check bounds
                    send $(assignment, \gamma_{assignment}, UB)$ to all children
                    if Leaf agent then // Leaf agent
                        $currentPF \leftarrow (currentPF \uplus \delta_{assignment})$
                    else
                        $response \leftarrow 0$ ;$PF_c \leftarrow \emptyset$
                        while $response < |children|$ do // Receive Cost messages
                            if message $= (PF_{c_i})$ then
                                $PF_c \leftarrow (PF_c \bigoplus PF_{c_i})$
                                $response \leftarrow response + 1$
                        $currentPF \leftarrow (currentPF \uplus (PF_c + \delta_{assignment}))$
            send $currentPF$ to parent // Send Cost message
        add $currentPF$ to $PF$
```

Algorithm 2. Algorithm to build UB

```
$UB$: set of cost vectors (local upper bounds)
$UB_p$: set of cost vectors (upper bounds received from the parent)
$currentPF$ set of pairs of assignment and cost vector (local Pareto front for the current $context$)
$previousPF$ set of pairs of assignment and cost vector (local Pareto front for the previous search)
$context$: vector of integers (ancestors assignment)
$addedObjMax$: vector of integers with the local maximal value for each newly added objectives.
$UB \leftarrow currentPF \uplus UB_p$
//Find the maximal acceptable cost for the new objectives
for each added objective $m$ do
    for each $cost \in UB$ do
        $addedObjMax[m] \leftarrow max(addedObjMax[m], cost[m])$
//Reuse previous bound
for each $(assignment, cost) \in previousPF$ do
    if $assignment$ compatible with $context$ then
        $UB \leftarrow UB \uplus (cost \cup addedObjMax)$
```

- *children*$_i$, the set of children of i.

We assume that this operation is done as a preprocessing step. Since adding or removing objectives has no impact on the topology of the problem, the ordering will stay the same throughout the execution.

We show the pseudo-code of DMOBB in Algorithm 1 and 2. During the search phase, the solution space will be explored to completely determine the Pareto solutions. The search can start without any prior knowledge or it can use the Pareto front found during the previous search.

To communicate information between the agents in the pseudo-tree, we use the following three message types :

Value message: Sent from an agent i to its children, it contains the *context* currently being explored, the gamma cost $\gamma_{context}$ and the bounds used by the parent (UB_p).

Cost message: Sent from an agent i its parent, it contains the local Pareto front $PF_{context}$ found for the given context *context*.

Terminate message: Sent from parent to children to indicate the search is over.

Furthermore, we define 2 operators, the first one is the direct sum for two Pareto fronts which makes use of the direct sum between two vectors.

$$PF_1 \bigoplus PF_2 = \left\{ \forall (X, Y) \in PF_1 \times PF_2, X \bigoplus Y \right\} \tag{5}$$

The second operator is the union of two Pareto fronts that keeps only the non-dominated cost vectors.

$$A \uplus B = A \cup B \setminus \{a < b\} \cup \{b < a\}, a \in A, b \in B. \tag{6}$$

We also define the delta cost $\delta_{context+d_i}$ and the gamma cost $\gamma_{context+d_i}$. The delta cost is the sum of constraint costs of all constraints that involve both i and one of its ancestors for the current value d_i and the values of ancestor agents contained in the current *context*. The gamma cost is the sum of ancestors' delta cost plus the local delta cost for context $context + d_i$.

Theorem 1. *With DP the DMO-DCOP we want to solve, n the number of variables, m the number of objectives and* $|d|$ *the domain size for the variables, the memory use of an agent to solve DP is given by* $O(2m|d|^n)$. *The total time required to solve DP is given by* $O(m^2|d|^{3n}|DP|)$.

5 Experimental Evaluation

In this section, we evaluate the performances of DMOBB and compare them with the naive method where each MO-DCOP is solved independently. All the tests are made with a domain size of 2 and a density of 1 (a variable always share a constraint with all the other variables). We show the results obtained when

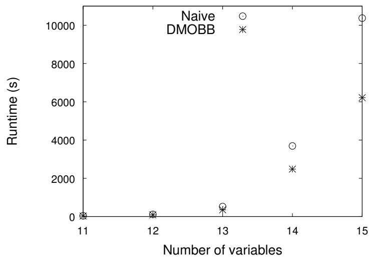


Fig. 1. Varying number of variables

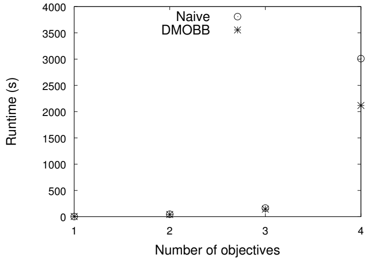


Fig. 2. Varying number of objectives

varying the number of variables and when varying the number of objectives. We implemented our algorithm in Java using the Jade framework and all tests were run on 6 cores running at 2.6GHz with 12GB of RAM.

Varying Variables Figure 1 shows the runtime when varying the number of nodes. Those results are obtained for the complete solving of a $DMO\text{-}DCOP =< MO\text{-}DCOP_1, MO\text{-}DCOP_2, MO\text{-}DCOP_3 >$ with the first MO-DCOP having 3 objectives, the second one 4 and the last one 5. We can see the expected exponential growth of the runtime making larger problems quickly uncomputable. However, we can see that the growth when using DMOBB is reduced. The costliest operation in our algorithm is the comparison of Pareto fronts. Our algorithm, even in the worst case, can prune some solutions in the leaf nodes. This reduces the size of the Pareto fronts that comes up the tree, decreasing the runtime significantly. We now consider the influence of the number of objectives on the runtime. For this test, we solved a $DMO\text{-}DCOP =< MO\text{-}DCOP_1, MO\text{-}DCOP_2 >$ such that $MO\text{-}DCOP_1$ has m objectives and $MO\text{-}DCOP_2$ has $m+1$ objectives. We show in figure 2 the runtime it takes to solve $MO\text{-}DCOP_2$ for a problem with 14 variables. We varied m from 1 to 4 and we can see that with bigger m the improvement compared to the naive method increases. DMOBB has almost no impact for small problems but we see that we get 30% speedup when solving a problem with 5 objectives and reusing the previous solutions.

To conclude the experimental part, we have shown that the larger the problems, the more efficient DMOBB is compared to the naive resolution. However, on smaller problems, DMOBB offers no advantages compared to the naive method and can even be less efficient. Note that those results were obtained on the worst case (random cost vectors and density 1) and that depending on the problem, better results can be expected.

6 Conclusion

In this paper, we introduced the Dynamic Multi-Objective Distributed Constraint Optimization Problem (DMODCOP) and proposed DMOBB, the first algorithm to solve such problem in a reactive approach. We showed how DMOBB is more efficient than the naive method where each problem in the sequence is solved independently.

As future works, we want to want to abandon the assumption of this paper that considers only changes in the number of objectives. Since Pareto fronts are of exponential size in the worst case, we also want to develop an incomplete algorithm for DMO-DCOPs in order to solve large-scale problem instances.

References

[1] Billiau, G., Chang, C.F., Ghose, A.: SBDO: A new robust approach to dynamic distributed constraint optimisation. In: Desai, N., Liu, A., Winikoff, M. (eds.) PRIMA 2010. LNCS, vol. 7057, pp. 11–26. Springer, Heidelberg (2012)

[2] Fave, F.M.D., Stranders, R., Rogers, A., Jennings, N.R.: Bounded decentralised coordination over multiple objectives. In: Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems, pp. 371–378 (2011)

[3] Junges, R., Bazzan, A.L.C.: Evaluating the performance of DCOP algorithms in a real world, dynamic problem. In: Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems, pp. 599–606 (2008)

[4] Lesser, V., Ortiz, C., Tambe, M. (eds.): Distributed Sensor Networks: A Multiagent Perspective, vol. 9. Kluwer Academic Publishers (2003)

[5] Maheswaran, R.T., Tambe, M., Bowring, E., Pearce, J.P., Varakantham, P.: Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In: Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems, pp. 310–317 (2004)

[6] Mailler, R., Lesser, V.R.: Solving distributed constraint optimization problems using cooperative mediation. In: Proceedings of the 3rd International Conference on Autonomous Agents and Multiagent Systems, pp. 438–445 (2004)

[7] Matsui, T., Silaghi, M., Hirayama, K., Yokoo, M., Matsuo, H.: Distributed search method with bounded cost vectors on multiple objective dcops. In: Proceedings of the 15th International Conference on Principles and Practice of Multi-Agent Systems, pp. 137–152 (2012)

[8] Modi, P., Shen, W., Tambe, M., Yokoo, M.: Adopt: asynchronous distributed constraint optimization with quality guarantees. Artificial Intelligence 161(1-2), 149–180 (2005)

[9] Petcu, A., Faltings, B.: A scalable method for multiagent constraint optimization, pp. 266–271 (2005)

[10] Schiex, T., Fargier, H., Verfaillie, G.: Valued constraint satisfaction problems: Hard and easy problems. In: Proceedings of the 14th International Joint Conference on sArtificial Intelligence, pp. 631–639 (1995)

[11] Yeoh, W., Felner, A., Koenig, S.: BnB-ADOPT: An asynchronous branch-and-bound DCOP algorithm. Journal of Artificial Intelligence Research 38, 85–133 (2010)

[12] Yeoh, W., Varakantham, P., Sun, X., Koenig, S.: Incremental dcop search algorithms for solving dynamic dcops. In: AAMAS, pp. 1069–1070 (2011)