Comparing Multi-Objective Selection Methods using a Simulation of Dynamic Sensor Network

Maxime Clement[*1][*3] Tenda Okimoto[*2] Katsumi Inoue[*1][*3]

[*1]National Institute of Informatics [*2]Kobe University
[*3]SOKENDAI (The Graduate University for Advanced Studies)

Multi-Objective Distributed Constraint Optimization Problems (MO-DCOPs) can model problems where agents must coordinate to optimize multiple costs. As problems can involve conflicting objectives that cannot be minimized simultaneously, solving an MO-DCOP usually consists in finding a set of solutions offering various trade-offs of conflicting objectives. In real-world problems, a single solution is usually selected for implementation and many different methods have been proposed to perform this selection. If the problem is dynamic and changes over time, this selection can not only impact the costs for the current state of the problem but also for the future state of the problem. In this paper, we compare the impact of different multi-objective selection methods on a dynamic simulation of mobile sensor teams and our experiments confirm that the selection method can have an unforeseen impact on the quality of the system in the long run.

1. Introduction

Many real world problems involve multiple criteria that should be considered separately and optimized simultaneously. A *Multi-Objective Distributed Constraint Optimization Problem* (MO-DCOP) [2, 3] allows us to represent situations where some agents have to coordinate their values to minimize multiple objectives. In MO-DCOPs, since trade-offs may exist among the objectives, there does not generally exist a unique ideal assignment, which minimizes all objectives simultaneously. Thus, the optimal solution of an MO-DCOP is characterized by the concept of Pareto optimality. A solution is said to be *Pareto optimal* if there does not exist another solution better or equivalent for all objectives and strictly better for at least one objective. Solving a multi-objective problem is commonly seen as finding its set of Pareto optimal solutions.

While most multi-objective algorithms find a set of solutions, a single solution is needed in practice, often requiring an additional selection process that is well studied in multi-criteria decision making (MCDM). Ideally, this selection should reflect the preferences of a decision maker (DM), either by directly letting the DM pick the solution or by trying to model his preferences using mathematical functions, typically using weighted [6] or reference point-based [4] methods.

However, for dynamic problems that change over time and require to change the solution multiple time, the long term impact of the multi-objective selection process might be difficult to foresee by a human and should be studied in advance. For example, let us consider a mobile sensor network, a representative application of MO-DCOPs. We consider an environment with multiple targets that we want to track with each sensor being able to track the targets that are in sensing range. This problem is dynamic as the targets might move and we should adjust the position of the sensors accordingly. We want to optimize the tracking of the targets but at the same time, we do not want to move the sensors too much (to limit the use of energy and to increase their durability for example). Let us consider two different preferences for these two objectives. First, we can imagine that the decision maker strictly cares about tracking the targets, almost ignoring the cost of moving the sensors. In this case, we always select a solution that gives us the best tracking, meaning that if a single targets changed position, we are willing to drastically move the sensors just to obtain a small increase in the tracking quality. This can lead to very costly solutions in the long term, which might have been unforeseen by the DM. Second, we can imagine the opposite preferences where the DM mostly cares about the cost of moving the sensors. In this case, we might start selecting solutions whose quality are acceptable while only slightly moving the sensors. As time passes however, if several targets changes position too fast, the cost of moving the sensors might become high and the preference model might end up selecting solutions that do not keep track of any targets.

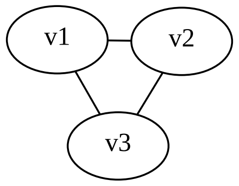
In this paper, we use a dynamic simulation of the Mobile Sensor Team [8] (MST) problem to show the different results obtained using different multi-objective decision making methods. We first present multi-objective decision making methods and present the ones we consider in our dynamic case. We then propose a multi-objective version of MST that we use to experiment with different decision methods and exhibit the impact it has on the system in the long run. We conclude by proposing future research topics that we feel are promising.

2. Preliminaries

2.1 Multi-objective Distributed Constraint Optimization Problem

Definition 1 (Multi-Objective DCOP) *A Multi-Objective Distributed Constraint Optimization Prob-*

Contact: maxime-clement@nii.ac.jp



v_1	v_2	*cost*	v_2	v_3	*cost*	v_1	v_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)

Figure 1: Example of bi-objective DCOP.

lem (MO-DCOP) [2, 3] is defined as a tuple MO-DCOP $= (X, V, \mathcal{D}, \mathcal{C}, \mathcal{F})$ *where* $X = \{x_1, \ldots, x_n\}$ *is a set of agents,* $V = \{v_1, \ldots, v_n\}$ *is a set of variables,* $\mathcal{D} = \{D_1, \ldots, D_n\}$ *is a set of domains,* $\mathcal{C} = \{C_1, \ldots, C_c\}$ *is a set of constraint relations, and* $\mathcal{F} = \{F_1, \ldots, F_m\}$ *is a set of sets of cost functions. For an objective* o $(1 \leq o \leq m)$*, a cost function* $f_j^o : \times_{\forall v_i \in C_j} D_i \rightarrow \mathbb{R}_{\geq 0}, f_j^o \in F_o$*, represents the cost generated by constraint* C_j *for the objective* l*.*

An instantiated set of variable $A \subseteq V$ *is called an assignment.* A *is said to be* partial *if* $A \subset V$ *and* complete *if* $A = V$*. When an assignment is complete, its quality can be represented by a cost vector* $R(A) = (R^1(A), \ldots, R^m(A))$ *where*

$$R^l(A) = \sum_{C_j \in \mathcal{C},} f_j^l(A)$$

Finding an assignment that minimizes all objective functions simultaneously is ideal. However, trade-offs can exist among the different objectives and there usually does not exist such an ideal assignment. Therefore, optimal solutions of an MO-DCOP are characterized using the concept of *Pareto optimality.*

Definition 2 (Dominance) *For an MO-DCOP and two cost vectors* $R(A)$ *and* $R(A')$*, we can 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., it holds* $R^l(A) \leq R^l(A')$ *for all objectives* l*, and there exists at least one objective* l'*, such that* $R^{l'}(A) < R^{l'}(A')$*.*

Definition 3 (Pareto optimal solution) *For an MO-DCOP, an assignment* A *is said to be a* Pareto optimal solution*, iff there does not exist another assignment* A'*, such that* $R(A') \prec R(A)$*.*

Definition 4 (Pareto Front) *For an MO-DCOP, the* Pareto front *is the set of cost vectors obtained by the Pareto optimal solutions.*

Solving an MO-DCOP consists in finding its Pareto front whose size is, in the worst case, exponential in the number of variables.

An MO-DCOP can be represented using a constraint graph where nodes and edges represent the variables and constraints of the problem.

Example 1 (MO-DCOP) *Figure 1 shows an example of MO-DCOP with two objectives, three binary constraints, and where each variable takes its value from a discrete domain* $\{a, b\}$*. The Pareto optimal solutions of this problem are* $\{\{(v_1, a), (v_2, a), (v_3, a)\}$ *and* $\{(v_1, a), (v_2, b), (v_3, b)\}\}$*, and the corresponding Pareto front is* $\{(6, 3), (10, 1)\}$*.*

2.2 Dynamic Multi-Objective Distributed Constraint Optimization Problem

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.

$$< \textit{MO-DCOP}_1, \textit{MO-DCOP}_2, ..., \textit{MO-DCOP}_t > . \quad (1)$$

In this paper, we consider a *reactive* case where a problem *MO-DCOP*$_i$ is unknown until a solution is implemented for problem *MO-DCOP*$_{i-1}$.

In case all the MO-DCOP in the sequence are known in advance (i.e., we know the nature and the order of all the changes), a proactive approach can be used, and was discussed in a previous work [1].

3. Multi-Objective Decision Making

It is usually considered that solving an MO-DCOP consists in finding its Pareto front and most MO-DCOP algorithms will find a set of solutions that offers different trade-offs of objectives. Since in practice only one solution can be implemented, it is assumed that an additional decision method will perform the selection of one solution from the Pareto front. This decision should be in accordance with the preferences of a decision maker (DM), either by creating some preference model or by interacting with the DM during the solving process.

Such methods have been reviewed in the context of evolutionary multi-objective optimization [5] and have been classified into *a priori*, *interactive*, and *a posteriori* methods. With *a priori* methods, the preferences of the DM are known in advance and are expressed using some mathematical function. This sometime allows us to transform the multi-objective optimization problem into a single-objective problem, using scalarization techniques for example. With *interactive* methods, queries to the DM are made during the run of the algorithm in order to help identify its preferred solution. With *a posteriori* methods, a selection is made after the set of optimal solutions was found, usually by asking the DM to select his preferred solution from this set.

4. Multi-Objective Decision Making for Dynamic MO-DCOPs

In dynamic problems, a new solution should be implemented as soon as possible after changes occurred to the problem. This makes *interactive* and *a posteriori* methods poorly suited for dynamic environments as a decision maker might not always be available or might take too much time to select a solution. Thus, we propose to use *a priori* methods that are automated and thus well suited for dynamic problems. We consider the weighted-sum, a popular aggregation technique that is widely used to tackle multi-objective problems. Such technique requires to provide a weight for each objective of the problem and effectively transforms the multi-objective problem into a mono-objective problem but guarantees to find a Pareto optimal solution.

Definition 5 (Weighted-Sum Selection) *Given a problem P and a set of weights $\Omega = (\omega_1, \dots, \omega_m)$, we select a solution S_Ω of P such that:*

$$S_\Omega = \underset{S}{\operatorname{argmin}} \sum_{o=1}^{m} R^o(S) \times \omega_o$$

The limit of such selection methods is that they are *static* and only consider the immediate utility obtained. However, when working in a dynamic environment, the selection method can have unforeseen consequences on the problem which can lead to a decrease of the solution quality over time. Thus, it can be important to use *dynamic* decision methods that alternate between different strategies.

Definition 6 (Alternating Multi-Objective Selection) *An* alternating multi-objective decision method *is a selection method that alternates between different strategies based on some conditions or probabilities.*

In this paper, we consider using multiple weighted-sum selection with equiprobability.

Definition 7 (Alternating Weighted-Sum Selection) *Given sets of weights $\{\Omega_i, \Omega_{i+1}, \dots\}$, an alternating weighted-sum method $A(\Omega_i, \Omega_{i+1}, \dots)$ is a selection method that randomly uses one of the set of weights to perform a weighted-sum selection.*

5. Experimental Evaluation

5.1 Simulation

To compare the multi-objective decision making methods discussed in the previous section, we use a dynamic simulation of the Mobile Sensor Team problem [8]. In this simulation, we consider a set of sensors $X = \{x_1, \dots, x_n\}$, and a set of targets $Y = \{y_1, \dots, y_l\}$, in a two dimensional Euclidean space using $d(.)$ as distance function. At a time step s ($1 \le s \le t$), the position of a sensor or target is denoted by $p_s(.)$. Each sensor x_i has a sensing range SR_i and a mobility range MR_i. Each target y_j has a tracking requirement TR_j indicating how many sensors should cover the target. We say that a sensor x_i *covers* target y_j if $d(p_s(x_i), p_s(y_j)) \le SR_i$ and denotes $SR_{(s,j)}$ the set of agents that cover target y_j at time step s. At each time step s of this problem, we need to find a new position $p_{s+1}(x_i)$ for each sensor such that $d(p_s(x_i), p_{s+1}(x_i)) \le MR_i$. The goal is then to find positions that minimize both the remaining tracking requirement $RTR_s = \sum_{j=1}^{l} \max(0, TR_j - |SR_{(s+1,j)}|)$, and the movement cost $MC_s = \sum_{i=1}^{n} d(p_s(x_i), p_{s+1}(x_i))$.

At each time step s, $MO\text{-}DCOP_s$ is constructed from the current state of the simulation A variable in the MO-DCOP corresponds to a sensor of the simulation and we use as domain all possibles positions offering coverage of a subset of the targets. To avoid too large domains, we discard subsets that are included in other subsets and if two positions offer the coverage of the same subsets, we keep the one that yield the minimum movement cost. Then, for each target y_j, a constraint is constructed between the sensors that can cover the target after any movement, i.e., $\{x_i | SR_i + MR_i \le d(p_s(x_i), p_s(y_j))\}$.

5.2 Settings

We implemented our simulation using NetLogo [7], using $n = 10$ sensors, $l = 10$ targets, $t = 40$ time steps, a sensing range of $SR_i = 5$ and a mobility range $MR_i = 5$ for all agents, a tracking requirement $TR_j = 3$ for all sensors, and a grid space environment of 10×10. The initial positions of the targets and sensors ($p_0(.)$) are randomly generated and describe the initial state of the problem. At each new time step, after the new positions of the sensors were adopted, we randomly move each target such that it moves away from the closest sensor over a distance of 3.

We consider three sets of weights for the weighted-sum selection. First, we consider that both objectives are of equal importance and use $\Omega_1 = (0.5, 0.5)$. Then, we want to put a bit more emphasis on the first objective (the remaining tracking requirement) and use $\Omega_2 = (0.75, 0.25)$. Finally, we consider the extreme case where the movement cost is completely ignored and only the tracking requirement is considered, using $\Omega_3 = (1, 0)$. In addition to these methods, we consider an alternating method $A(\Omega_1, \Omega_2)$ that uses Ω_1 or Ω_2 with equiprobability.

5.3 Results

Figure 2 and Figure 3 show the evolution of the remaining tracking requirement and movement cost over the 40 time steps. These results are averages over 50 simulation runs using different initial positions for the targets and sensors.

First, we notice that even though Ω_1 is supposed to care about both objectives equally, the remaining tracking requirement (RTR) increases significantly over the course of the simulation, from $RTR_1 \approx 18$ up to $RTR_{35} \approx 28$, while the movement cost (MC) decreases only slightly, varying between 5 and 0. In comparison, when we observe both Ω_2 and Ω_3, which care more about the remaining tracking requirement, we observe that both objectives are quite stable, remaining around the same values over the whole simulation. These observations indicate that the preferences represented by Ω_2 and Ω_3 are well matched throughout the simulation while Ω_1 cannot properly represent the given preference in the long run. This is further supported by the increase in the value of the weighted-sum when using Ω_1, which goes from around 11.5 to 14, indicating a clear

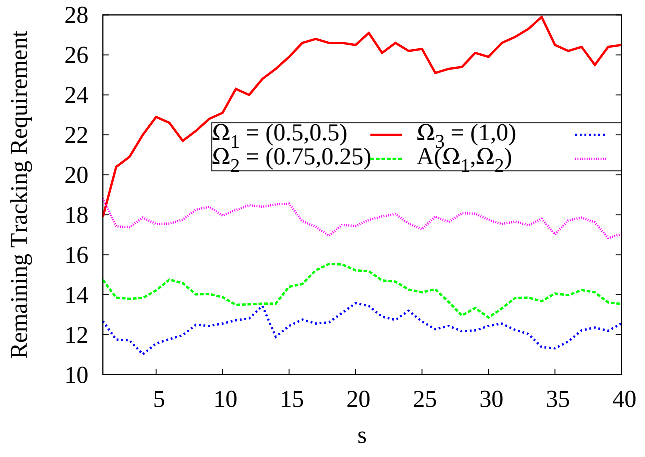


Figure 2: Comparison of Remaining Tracking Requirement

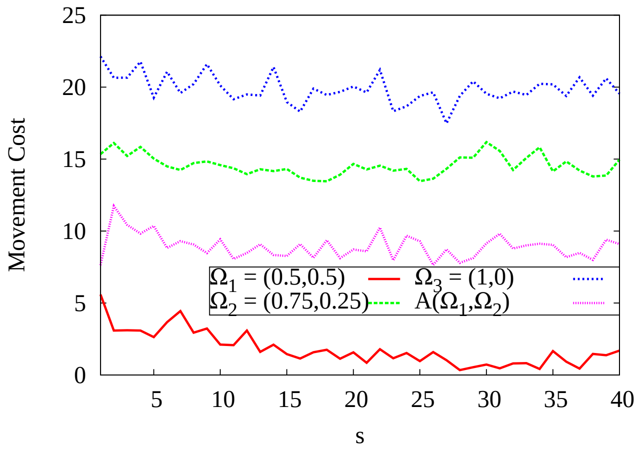


Figure 3: Comparison of Movement Cost

decrease in solution quality over time.

Let us now look at our alternative method which randomly uses Ω_1 or Ω_2. With this method, both objectives are quite stable, with $RTR \approx 18$ and $MC \approx 10$. We notice that the value of RTR starts at about the same value as when using Ω_1 alone but without increasing with time. When compared to the results we obtained with Ω_2 alone, we see that the value of MC is reduced by around 5 while the value of RTR is increased by around 4. This shows that by alternating between two multi-objective decision methods, we can obtain a new trade-offs of the objective that is more stable than when using a single decision method.

6. Conclusion

In this paper, we discussed the importance of defining multi-objective decision making methods in dynamic problems. Using Multi-Objective Distributed Constraint Optimization Problems and the Mobile Sensor Team application, we considered various methods to select the solution to implement within the Pareto set. We then used a simulation to experiment with different methods and showed that, in dynamic problems, simple selection methods are not always suited to represent the preferences of a decision maker. We showed that some trade-offs of objectives can become harder to obtain or can lead to poor solutions quality. However, we showed that our proposed alternating selection method can obtain solutions close to those trade-offs while providing a stable quality.

In future works, we want to study more cases where multi-objective decision making has a strong impact on a dynamic problem. This can be the opportunity to evaluate in more depths the limit of traditional decision making and propose additional methods that would prove more suited to dynamic problems.

References

[1] M. Clement, T. Okimoto, N. Schwind, and K. Inoue. Finding resilient solutions for dynamic multi-objective constraint optimization problems. In *ICAART 2015 - Proceedings of the International Conference on Agents and Artificial Intelligence, Volume 2, Lisbon, Portugal, 10-12 January, 2015.*, pages 509–516, 2015.

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

[3] T. Matsui, M. Silaghi, K. Hirayama, M. Yokoo, and H. Matsuo. 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*, pages 137–152, 2012.

[4] T. Okimoto, N. Schwind, M. Clement, and K. Inoue. Lp-norm based algorithm for multi-objective distributed constraint optimization. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems*, pages 1427–1428, 2014.

[5] R. C. Purshouse, K. Deb, M. M. Mansor, S. Mostaghim, and R. Wang. A review of hybrid evolutionary multiple criteria decision making methods. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1147–1154, 2014.

[6] N. Schwind, T. Okimoto, S. Konieczny, M. Wack, and K. Inoue. Utilitarian and egalitarian solutions for multi-objective constraint optimization. In *Proceedings of the 26th IEEE International Conference on Tools with Artificial Intelligence*, pages 170–177, 2014.

[7] S. Tisue and U. Wilensky. Netlogo: A simple environment for modeling complexity. In *International conference on complex systems*, volume 21, pages 16–21. Boston, MA, 2004.

[8] R. Zivan, H. Yedidsion, S. Okamoto, R. Glinton, and K. Sycara. Distributed constraint optimization for teams of mobile sensing agents. *Autonomous Agents and Multi-Agent Systems*, 29(3):495–536, 2015.