Limiting Perturbations in Dynamic DCOP

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

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

Distributed Constraint Satisfaction Problem (DisCSP) and Distributed Constraint Optimization Problem (DCOP) are fundamental frameworks to model many multi-agent problems. In those frameworks, a solution is an assignment to a set of variables, each controlled by an agent, that either satisfies all constraints (DisCSP) or maximizes the sum of rewards induced by the constraints (DCOP). In many real world situations, a new solution is required whenever changes occur to the problem. However, a transition to a new solution induces an additional cost and a perturbation between the previous and new solutions should be considered. In this paper, we propose the Limited Perturbation Problem (LPP) where the goal is to find the best possible solution while limiting perturbations in a Dynamic DCOP. When limiting the number of variables that are allowed to change, we are able to provide an interesting a priori guarantee on the solution quality.

1. Introduction

Distributed Constraint Satisfaction Problem [9] (DisCSP) is a fundamental problem that can formalize various applications for multi-agent cooperation. In DisCSP, agents assign values to variables, attempting to generate a locally consistent assignment that is also consistent with all the constraints between agents. Distributed Constraint Optimization [3] (DCOP) is an extension of DisCSP where constraints yield a real number (reward) instead of being satisfied or unsatisfied. The goal of this problem is to find an assignment that optimizes the sum of rewards over all constraints.

Dynamic DisCSP can be represented as Minimal Perturbation Problem (MPP) [1]. When changes occur, the previous solution becomes invalid and the goal of the MPP is to find a new solution as close as possible to the previous one. In this work, what is considered as perturbations is the cost of changing the assignment.

Dynamic DCOP is usually represented using a sequence of static DCOPs [8]. This representation considers each problem to be independent and changing the assignment of a variable is free. This is an ideal view of a dynamic problem and it fails to take into account the perturbations that are considered in the MPP. In a Minimal Perturbation Problem, the initial case considered is a CSP with a solution already in place. This CSP is then changed and the goal is to find a new valid solution to the changed CSP that will generate the least perturbations when adopted. For example, a common application of this problem is the meeting scheduling problem. When you have a certain schedule in place, but some of the initial constraints are changed, it is usually best to compute a new schedule as similar as the initial one. Another work for Dynamic SAT (a problem close to CSP) considered the decision change cost when all the changes are known in advance [2]. The goal is then to find the sequence of valid assignments that minimizes the decision change costs. While decision change costs and perturbations are not exactly the same thing, the goal in those two works is to minimize a value induced when switching between assignments.

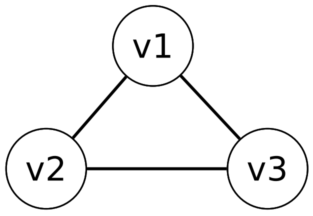
In our work, we want to handle similar situations as the MPP where we care about the cost of deviating from an initial assignment. However, compared to DisCSP, the goal of a DCOP is to optimize an objective function and all assignments can be considered valid.

In this paper, we want to give a new definition of Dynamic DCOP where some perturbations have to be taken into account. We consider that the previous definition of a Dynamic DCOP is designed for a specific case where changing the assignment to the variables is free. When we need to consider perturbations, we propose the Limited Perturbation Problem (LPP) which requires a cost function and a limit on how much of this cost is allowed. In the special case where this cost function is the Hamming distance (the number of changing variables), we are able to provide an interesting a priori guarantee on the solution quality.

2. DCOP

A *Distributed Constraint Optimization Problem* (DCOP) [3] is defined as a set of agents X, a set of variables V, a set of domains D, a set of constraint relations C and a set of reward functions F. Each agent is in control of one or multiple variables. A variable $v_i \in V$ takes its value from a finite, discrete domain $D_i \in D$. A constraint relation $c \in C, c \subset V$ means that there exists a constraint between all $v_i \in c$. The reward function $R_c(A) \in F$ represents the reward generated by constraint c when variables take values defined in assignment A. The quality of a value assignment to all variables A is evaluated by summing the rewards of all constraints: $R(A) = \sum_{c \in C,} R_c(a)$. Then, an optimal assignment A^* is given as $\arg\max_A R(A)$, meaning that A^* is an assignment that maximizes the sum of all reward functions. A DCOP can be represented using a graph, called a constraint graph [6], in which nodes represent variables and edges represent constraints.

Contact: maxime-clement@nii.ac.jp



v_1	v_2	$reward$	v_2	v_3	$reward$	v_1	v_3	$reward$
0	0	2	0	0	1	0	0	1
0	1	1	0	1	2	0	1	1
1	0	0	1	0	1	1	0	0
1	1	2	1	1	1	1	1	3

Figure 1: Example of Constraint Optimization Problem

Example 1 (DCOP) *Consider the DCOP represented in Figure 1 where 3 variables v_1, v_2 and v_3 are linked by binary constraints. Each variable can take the value 0 or 1 and the resulting rewards are shown in the reward table. The optimal solution of this problem is $A = \{(v_1, 1), (v_2, 1), (v_3, 1)\}$ and the optimal value (i.e. sum of rewards obtained by A) is 6. This reward is computed by checking the reward of each individual constraint: $R_{\{1,2\}}(1, 1) = 2$, $R_{\{2,3\}}(1, 1) = 1$ and $R_{\{1,3\}}(1, 1) = 3$. The resulting reward is $R(A) = 2+1+3 = 6$.*

3. Dynamic DCOP

A Dynamic Constraint Optimization Problem (DynDCOP) [8] can be represented as a sequence of static DCOPs. This assumes that each problem is independent and that reassigning variables is free. We call such representation a Resource-Unbounded Dynamic DCOP (RU DynDCOP).

Definition 1 (Resource-Unbounded Dynamic DCOP) *A Resource-Unbounded Dynamic DCOP is a sequence $\langle DCOP_0, DCOP_1, \ldots, DCOP_l\rangle$ where the goal is to find the optimal solution of each problem in the sequence.*

In this work, we propose to tackle Cost-Sensitive Dynamic DCOP (CS DynDCOP) where adopting an assignment has a cost that needs to be taken into account.

Definition 2 (Cost-Sensitive Dynamic DCOP) *A Cost-Sensitive Dynamic DCOP is a sequence $\langle DCOP_0, DCOP_1, \ldots, DCOP_l\rangle$ and a set of perturbation functions $\Delta = \{\delta_0, \delta_1, \ldots, \delta_l\}$ where a function δ_i measures the cost of adopting an assignment for the problem $DCOP_i$.*

The goal is to find a solution for each problem such that it (i) maximizes the solution quality and (ii) minimizes the perturbations induced, i.e., the cost of adopting the solution.

RU DynDCOP can be seen as a CS DynDCOP where the sole focus is to maximize the solution quality. With Definition 2, we can represent the cost of changing the assignment between two problems by having δ_i function of both the solution of the previous problem $DCOP_{i-1}$ and the new assignment, allowing to model the cost of changing from an assignment to another.

Existing DCOP and RU DynDCOP approaches cannot be used to solve a CS DynDCOP as they completely ignore the minimization of perturbations. One possibility is to use a multi-objective approach [4], either to find all possible trade-offs between quality and perturbations, or to find one solution corresponding to a preference between the two objectives.

Because perturbations often represent real-life resources, the approach we will propose in this paper is to limit the amount of perturbations allowed.

4. Limited Perturbation Problem

In this paper, we work on DCOP and we do not assume any hard constraint. We consider the dynamic case where we solved an initial DCOP and adopted an initial assignment. The problem was then modified and we now want to find a new solution. If we consider the Minimal Perturbation Problem for Dynamic DCOP, then we want a solution as close as possible from the previous one. Since all assignments are valid in a DCOP without hard-constraints, with the priority of the Cost-Sensitive Dynamic DCOP being to minimize perturbations, we will always keep the same assignment as before since it produces no perturbations. Doing so completely ignores the optimization of the quality and can lead to very bad solutions.

In this section, we introduce the Limited Perturbation Problem (LPP) which can be used to solve a CS DynDCOP by setting a limit on how much perturbations are allowed. We then explain the relationship between solutions of the LPP and k-size optimal solutions and show that we can reuse guarantees designed for k-size optimal solutions for the solutions of the LPP.

4.1 Definition

We now propose the *Limited Perturbation Problem* where the goal is, given an initial assignment α and a maximum allowed perturbations d, to find the best solution within this limit. The amount of perturbations generated by adopting a new assignment A is measured using the function $\delta(\alpha, A)$.

Definition 3 (Limited Perturbation Problem) *The Limited Perturbation Problem (LPP) is defined as a tuple*

$$\Pi = (\Theta, \alpha, \delta, d),$$

where Θ is a Constraint Optimization Problem, α is an assignment for Θ that is called initial assignment, δ is a function that defines a distance between two assignments and d is the maximum acceptable distance between two assignments.

A solution to an LPP is an assignment A for Θ such that (i) $\delta(\alpha, A) \leq d$ and (ii) there does not exist another assignment A' such that $\delta(\alpha, A') \leq d$ and $R(A') > R(A)$.

Example 2 (LPP) Let us consider the DCOP as in Example 1 where $A = \{(v_1, 1), (v_2, 1), (v_3, 1)\}$ is the solution. Now let us imagine the dynamic case where the constraint between v_1 and v_3 was removed, generating Θ, the new DCOP to solve.

We consider the resulting Limited Perturbation Problem with initial assignment $\alpha = \{(v_1, 1), (v_2, 1), (v_3, 1)\}$ and $\delta = \mathcal{H}$, the Hamming distance.

We will now evaluate the solution of this LPP for different values of d. The case where $d = 0$ is straightforward. Since we allow no change from the initial assignment, the solution of the LPP is α whose reward is now $R(\alpha) = 3$.

For $d = 1$, we need only to consider, in addition to α, assignments where one variable has a different value compared to α: $\{(v_1, 1), (v_2, 1), (v_3, 0)\}$ yields a reward of 3,
$\{(v_1, 1), (v_2, 0), (v_3, 1)\}$ yields a reward of 2
and $\{(v_1, 0), (v_2, 1), (v_3, 1)\}$ yields a reward of 2.
Since $R(\alpha) = 3$, the LPP can have two solutions for $d = 1$, $\{(v_1, 1), (v_2, 1), (v_3, 0)\}$ or α.

Finally, let us consider $d = 2$, allowing two variables to change. The new assignments to consider (in addition to α and the one we considered for $d = 1$) are: $\{(v_1, 1), (v_2, 0), (v_3, 0)\}$ yields a reward of 1, $\{(v_1, 0), (v_2, 1), (v_3, 0)\}$ yields a reward of 2 and $\{(v_1, 0), (v_2, 0), (v_3, 1)\}$ yields a reward of 4. The best reward over all the considered assignments is 4. The solution of the LPP for $d = 2$ is thus $\{(v_1, 0), (v_2, 0), (v_3, 1)\}$.

To find a solution for the problem $DCOP_i$ in a CS DynDCOP, we consider the LPP $\Pi_i = (DCOP_i, \alpha, \delta_i, d)$ where α is the assignment adopted for problem $i-1$ (or the initial state of the system if $i = 0$).

Definition 4 (LPP-Based CS DynDCOP) *An LPP-Based Cost-Sensitive Dynamic DCOP is a sequence*

$$\langle \Pi_0, \Pi_1, \ldots, \Pi_l \rangle,$$

such that for each $\Pi_i = (DCOP_i, \alpha_i, \delta_i, d_i)$, α_i is solution of the LPP Π_{i-1} for $i > 0$ or the initial state of the system for $i = 0$.

Such representation is reactive as we only consider the cost of change from one problem to the next.

4.2 Relationship With k-Size Optimality

k-size optimality [5] is a solution criterion for DCOP which can provide the quality bound of a solution a priori, meaning the bound is obtained before we actually run the algorithm. An assignment is k-size optimal if its reward cannot be improved by changing the values of k or less of its variables.

Definition 5 (k-Size Optimality) *Considering $\mathcal{H}(A, A')$ the Hamming distance between two assignments, an assignment A is classified as k-size optimal if $R(A) \geq R(A')$ for all A' such that $\mathcal{H}(A, A') \leq k$.*

The value of k can vary between 0 (all solutions are 0-size optimal) and $|V|$, the number of variables in the problem (only an optimal solution of the problem is $|V|$-size optimal). In the LPP, when we consider perturbations as the Hamming distance between the two assignments ($\delta = \mathcal{H}$), there exists some strong relationships between the solution of a Limited Perturbation Problem and a k-size optimal solution.

Property 1 *A k-size optimal solution A of the problem Θ is a solution of the LPP $\Pi = (\Theta, A, \mathcal{H}, k)$.*

We can also use the same guarantee for the Limited Perturbation Problem as for k-size optimality.

Proposition 1 *For a Limited Perturbation Problem $\Pi = (\Theta, \alpha, \mathcal{H}, d)$ where Θ is a DCOP with n variables and a maximum constraint arity of m, we can express the following relation between the reward of the LPP solution A and the reward of the optimal solution A^* of Θ:*

$$R(A) \geq \frac{\binom{n-m}{d-m} R(A^*)}{\binom{n}{d} - \binom{n-m}{d}} \tag{1}$$

The proof for this proposition is very similar to the one for k-size optimality in [5]. It is based upon the number of constraints that can be optimized like in A^*. For the LPP, we count those constraints based on the distance from α whereas for k-size optimal solutions, it is based on the distance from A. Due to space limitation, we omit the proof here.

5. Experimental Evaluation

As an example application of the LPP, let us consider the problem of forming a set of efficient teams from a set of agents [7]. We have, for each pair of agents, a potential synergy when they work together. The goal is then to assign every agent to a team such that the global productivity is maximized. We model this problem as a DCOP Θ where agents are represented as variables that takes their values from the set of possible teams. We find the solution α to Θ and form our teams accordingly.

Next, the synergies inside a team change. and we update our knowledge of the problem. We obtain a new DCOP Θ' for which the previous solution was α. The different teams being located in various locations around the world, the travel expenses between those countries have to be taken into account when reallocating the agents. We represent this problem as a Limited Perturbation Problem $\Pi = (\Theta', \alpha, \delta, d)$ where δ gives us the costs of transferring researchers and d is our maximum budget.

We generated a random initial problem Θ with a given number of variables $|V| = 15$ (researchers) and a domain size of three (corresponding to teams in Australia, France and the USA). We consider a complete constraint graph and rewards are randomly generated following a graph coloring model. For the distance function δ, we use a cost of 500 to transfer a researcher between France and the USA and between Australia and the USA. For a transfer between Australia and France, we use a cost of 1000. If a researcher does not change team, the resulting cost is 0.

change intensity	LPP												Complete	
	Time (s) / Quality													
	$d = 0$		$d = 1000$		$d = 2000$		$d = 3000$		$d = 4000$		$d = 5000$			
0.25	0	305	0.07	324	1.36	340	11.7	349	55.7	356	165	361	665	367
0.5	0	301	0.07	330	1.39	345	11.9	354	56.6	361	167	366	662	374
0.75	0	295	0.07	326	1.51	344	13.2	353	62	361	180	367	664	373
1.0	0	299	0.07	338	1.31	357	11.1	370	52.7	379	157	382	665	383

Table 1: Results of using the LPP on a team formation problem.

The original problem is solved using a complete DCOP algorithm and we consider α the solution of Θ. When we modify Θ, we select a fixed percentage of all pair of variables sharing the same value in α (the researchers in a same team). For each selected pair, we then generate a new reward function.

Table 1 shows the average results over 20 instances where each line corresponds to an intensity of change. For each value of d and for the complete solving, we show the average time taken to solve the problem with a naive brand and bound algorithm and the corresponding quality. One of the main observation we can make is that the more the problem was changed, the bigger the impact of allowing more budget. It thus seems that when a problem went under very little changes, it might not be necessary to spend numerous resources to change the previous solution. When the problem underwent drastic changes however, it can be very interesting to allow many changes to be made in order to reach a much better solution.

6. Conclusion

Dynamic problems are a challenging topic concerning a wide array of applications and often involving a cost for implementing a new solution. In this paper, we proposed the definition of a Cost-Sensitive Dynamic DCOP (CS DynDCOP) which allows to take this cost into consideration. Greatly inspired by the Minimal Perturbation Problem for CSP, we introduced the Limited Perturbation Problem (LPP), a model that aims at limiting the perturbations induced by changing the assignment in a CS DynDCOP. In the LPP, we set a limit on that cost and aim at finding the resulting best possible solution. We also use the LPP to quickly find a new solution to a Resource-Unbounded Dynamic DCOP. By changing parts of the previously adopted assignment, we can expect to easily find a new good solution. Through the adjustable parameter d, we can then obtain different trade-offs between quality and computation time.

As future works, we want to use an approach to CS DynDCOP opposite of the LPP where we would give a limit on the minimum quality allowed and then minimize the perturbations. We will also consider a proactive approach to the Cost-Sensitive Dynamic DCOP. In a CS DynDCOP, the choice of assignment for the current problem has an impact on the next problem. With this in mind, we can prepare for the next steps of the problem by adopting an assignment that does not require many changes to be repaired.

References

[1] R. Bartàk, T. Müller, and H. Rudov. A new approach to modeling and solving minimal perturbation problems. In *In Recent Advances in Constraints*, pages 233–249. Springer, 2004.

[2] D. Hatano and K. Hirayama. Dynamic sat with decision change costs: Formalization and solutions. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 560–565, 2011.

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

[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] J. P. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1446–1451, 2007.

[6] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 631–639, 1995.

[7] S. Ueda, A. Iwasaki, M. Yokoo, M. C. Silaghi, K. Hirayama, and T. Matsui. Coalition structure generation based on distributed constraint optimization. In *AAAI Conference on Artificial Intelligence*, 2010.

[8] W. Yeoh, P. Varakantham, X. Sun, and S. Koenig. Incremental DCOP search algorithms for solving dynamic DCOPs. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 1069–1070, 2011.

[9] M. Yokoo and K. Hirayama. Algorithms for distributed constraint satisfaction: A review. *Autonomous Agents and Multi-Agent Systems*, 3(2):185–207, 2000.