Solving Generalized Open Constraint Optimization Problem Using Two-level Multi-agent Framework

Hoong Chuin LAU[1]
School of Information Systems
Singapore Management University
hclau@smu.edu.sg

Lei ZHANG
The Logistics Institute Asia Pacific
National University of Singapore
tlizl@nus.edu.sg

Chang LIU
School of Computing
National University of Singapore
liuchang@comp.nus.edu.sg

Abstract

The Open Constraint Optimization Problem (OCOP) refers to the COP where constraints and variable domains can change over time and agents' opinions have to be sought over a distributed network to form a solution. The openness of the problem has caused conventional approaches to COP such as branch-and-bound to fail to find optimal solutions. OCOP is a new problem and the approach to find an optimal solution (minimum total cost) introduced in [1] is based on an unrealistic assumption that agents are willing to report their options in non-decreasing order of cost. In this paper, we study a generalized OCOP where agents are self-interested and not obliged to reveal their private information such as the order of their options with respect to cost. The objective of the generalized OCOP is to find a solution with low total cost and high overall satisfaction level of agents. A Two-Level Structured Multi-Agent Framework has been proposed: in the upper level, a neutral central solver allows agents report their preferred options in tiers and find a feasible initial solution from top tiers of options by constraint propagation and guided tiers expansion; in the lower level, agents form coalitions and negotiate among themselves on the initial solution by an argument of Persuasive Points. Experimental results have shown that this two-level structure yields very promising results that seek a good balance between the total cost of solution and the agents' overall satisfaction level in the long run.

1. Introduction

The advent of online business practices such as auctions and real-time resource allocation have given rise to Constraint Optimization Problems (COP) in an *open* environment, where agents' opinions are collected from a distributed network, and variable domains and constraints can change dynamically. In [1,2], the Open Constraint Optimization Problem (OCOP) was defined. The openness of OCOP has surfaced several new concerns that render the classical approaches for COP ineffective:

1. *Variable domains and constraints can change dynamically*. Most conventional approaches to solve COP reply on the closed-world assumption that the domains of the variable and constraints are known apriori. In OCOP, the central solver only has partial knowledge of agents' domains and constraints. Thus, branch-and-bound will no longer work and an optimal solution will in general be impossible to achieve as the best options may not have been revealed yet at the time of a variable is to be fixed.
2. *Agents are self-interested and autonomous*. Agents will make decision based on the current situation and their own strategies. This makes optimization much harder if not impossible. How to incentivize self-interested agents to cooperate remains a big challenge. In addition, as agents are self-interested, a further challenge is how to discourage agents from cheating and increase their overall satisfaction level.
3. *Agents have privacy concerns.* To prevent information leakage, agents are reluctant to reveal information such as the order of their options and costs to anyone, including the central solver. To address privacy concerns, agents should be required to provide information only when necessary. This also serves to reduce the communication cost, which in general is much higher than the computation cost in a distributed setting. The challenge is to use the least amount of information from agents to produce quality result.

In this paper, we propose a two-level multi-agent framework to solve OCOP for autonomous agents. In our framework, agents' privacies are protected as they do not need to report their options in any particular order. Agents are discouraged from cheating (see 6.4), but unfortunately we cannot ensure agents to be always truthful. We also seek to minimize communication cost as we only require agents to report their options when necessary. We also seek a balance between the total cost and the overall satisfaction level of all agents.

2. Literature review

To our knowledge, very few works on OCOP are available to date. [1] is the main paper we review here, where a refined version of branch-and-bound was proposed to find an optimal solution that minimizes the total cost. Since the variable domains are not completely

[1] Supported by The Logistics Institute Asia Pacific, under a grant from the Singapore Ministry of Defense

known to the central solver, the authors make an important (but unrealistic) assumption that the agents will report their options in decreasing order of preference to ensure the optimality of the solution. A Vickrey-Clark-Groves (VCG) tax mechanism is used to make it optimal for all agents to tell the truth about their options if they are self-interested. There are some limitations with this approach. First, this approach is basically a branch-and-bound method, which is computationally expensive and unscalable. In fact, the authors have claimed that this approach can only handle the test cases with up to 25 variables. Secondly, the agents are totally cooperative and play a passive role throughout. This is somewhat unrealistic in a real application. Agents tend to be self-interested and be cooperative only when there is incentive to do so. Lastly, there are concerns with the assumption that agents report their options in decreasing order of preference. We will discuss this more in Section 3.3.

Other related literatures include works on Constraint Satisfaction Problem (CSP), Dynamic CSP and Distributed CSP. CSP, Max-CSP and other variants are well-studied problems which often take exponential time to solve via exact search algorithms. Meta-heuristics such as tabu search has also been applied to solve Max CSP ([8]). Most algorithms are set in an environment where information is centralized and closed. In an open context, standard approaches will no longer work and cannot be readily applied for OCOP. For Dynamic CSP, constraints can change over time. [3] developed methods for adapting consistency computations to such changes, and [12] solved this problem by finding minimal change on the solution. These methods require that the solution domains be completely known apriori, which is not so in an open environment. The Distributed CSP has some similarities to our problem, where agents are not required to report complete variable domains beforehand. [5] has extended this problem to Distributed Constraint Optimization Problem, and proposed the ADOPT algorithm which guaranteed a user-defined optimality. It allows agents to execute asynchronously and in parallel with message passing. Agents can make local decisions without necessarily having global certainty. However, this algorithm relies on backtracking, which is not unscalable in an open setting. In [9], an optimal asynchronous algorithm based on cooperative mediation was presented. In [6], an interesting twist is presented for Distributed CSP where constraints are secrets and the idea of incentive auctions were proposed to tackle the problem.

3. Problem formulation

3.1. Problem formulation of COP

A Constraint Optimization Problem (COP) in a closed setting is defined by a tuple $<X, D, C, R>$ where:

- $X = \{x_1, x_2, .., x_n\}$ is the set of n variables
- $D = \{d_1, d_2, .., d_n\}$ is the set of corresponding domains
- $C = \{C_1, C_2, .., C_m\}$ is the set of m *constraints*, where a constraint C_i is associated with an ordered set of variables $\{x_{i1},..,x_{ik}\}$ and has a list of allowed tuples for the variables.
- $R = \{r_1, r_2, ..,r_m\}$ is the set of *relations,* where r_i maps an allowed tuple in C_i to a positive cost.

The goal is to find an optimal solution, i.e. a consistent assignment of all variables to domain values such that all constraints are satisfied and the total cost is minimized.

3.2. Hidden variable encoding

In the classical COP formulation, the variable domains and relations are treated separately. This makes the problem harder in the open settings, since we have to cope with both variable domains and relations which are revealed incrementally. To unify the two concepts, one approach proposed in [5] is to transform cost-measured constraints as variable tuples, so that the hard constraints need only enforce equality of values.

Definition 1. The reformulated COP is as follows:

- X is a set of $n+m$ *variables,* where the first n variables corresponds to the original variables and the $(n+i)^{th}$ variable represents an ordered *variable tuple* that includes all original variables associated with the original constraint C_i.
- D_i with the possible values (or value tuples) for each variable X_i.
- Constraints $C=\{(X_i, X_j) \mid X_i \in X_j\}$, where X_i *is a single variable*, X_j *is a variable tuple,* and each C_i enforces the equality between the value of variable X_i and the value of X_i that appears in the variable tuple X_j
- A set of m *relations* that maps a value assignment for all variables in one variable tuple to a positive value.

Fig 1 below shows an example of the reformulation.

In this paper, each variable tuple is represented as an agent i.e. the $(n+1)^{st}$ to $(n+m)^{th}$ variables are represented by agents a_1 to a_m respectively.

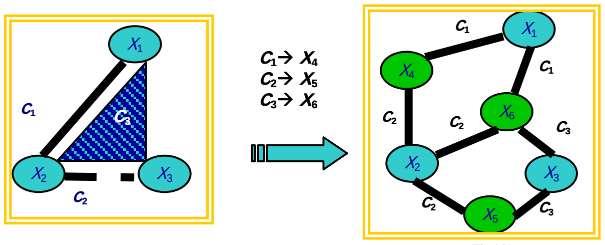
3.3. OCOP formulation and some concerns

OCOP is defined according to **Definition 1** by a tuple $<X, D^i, C, R^i>$ where the variable domains D^i and relations R^i are not completely known ([1]). Instead, they are collected from agents incrementally in the form of *options*.

Definition 2. An agent's *options* is the set of allowed value tuples for the variables associated with the agent. Within each agent, these options are assumed to be ranked according to the costs defined by its corresponding relation.

Given that agents reveal their options incrementally but not in the order of preference, it is conceivably impossible to find the optimal solution as a new value which is revealed in the later stage may lead to a better solution than the one found so far. [1] succeeded in finding an optimal solution by making an assumption that all the agents will report their options in the strictly non-decreasing order of cost. As discussed above, this assumption is unrealistic. First, the order of the options can be a sensitive piece of information that should not be revealed even to the central solver. Secondly, in open settings, there might be values that even the agent is not aware of at a certain time point that become part of the agent's domain later. Thus, the agent has no way to report their options in the non-decreasing order of cost apriori.

Furthermore, in the open settings, one is not only interested in minimizing the cost defined above, but also the *communication cost* among agents. Yet another concern is that, as agents are self-interested, a min-cost solution may not give a true picture of the level of *satisfaction* among all agents, as some agents might have sacrificed their own benefits for the sake of other agents.



(Fig 1-a) (Fig 1-b)

Fig 1: In Fig 1-a, there are 3 variables {X_1,X_2,X_3} and 3 constraints {X_1,X_2}, {X_2,X_3}, {X_1,X_2,X_3}. When transformed into Fig1-b, the variables becomes {X_1, …, X_6} where X_4, X_5 and X_6 are variable tuples that represent the 3 constraints. C_i now enforces the value consistency of variable X_i. Hence, the new constraint C_i now enforces equality of values between the single variable $X_{i,}$ and a variable tuples that contains $X_{i,}$.

3.4. Generalized OCOP and our objective

With the above considerations, we present the objective of our work. In our model, agents are not obliged to report their options in non-decreasing order of cost due to privacy reasons. Furthermore, agents are self-interested and autonomous for their actions while the central solver will act more like a broker to pass information among them. In addition, as agents become self-interested, they are not only concerned about costs but also *satisfaction level*. In this paper, we measure the overall agents' *satisfaction* level by the fairness of the solution, i.e. the rankings of the values in the solution should not differ too much with respect to the agents' own rankings. More precisely, if we assign indexes to the options (the option with lowest cost has index 1, the second lowest cost option has index 2, etc.), then over multiple rounds of negotiation, one can measure the standard deviation of the agent's option index against those in the solutions, and this quantity should be minimized. The essence behind this is that no self-interested agent is willing to sacrifice its own benefit to help others in the long term and thus a fair framework should ensure all agents perform equally good or bad in order to attract agents to stay in the system. Details of this will be elaborated in section 7.2.

We have thus extended the definition of OCOP given in [1], where we not only need to reduce the total cost of the solution, but also increase the agents' overall satisfaction level. We call this new problem the **Generalized OCOP**. There are several ways for accommodating these two objectives:

a: Given a satisfaction level expressed by the standard deviation $\boldsymbol{\sigma}$, find a solution with minimum cost that has the standard deviation less than $\boldsymbol{\sigma}$

b: Given a maximum overall cost C, find a solution with minimum standard deviation and has cost less than C

c: Find a mechanism that returns solutions with low cost C and small standard deviation $\boldsymbol{\sigma}$ in the long run (i.e. over multiple rounds of negotiation).

Note that c differs from a and b in that it seeks a balance between two objectives and emphasizes the performance of the system in the long run. In this paper, we adopt c. Thus, the objective is to find feasible solutions with (1) promising quality in terms of cost, and (2) improved overall satisfaction level of agents in the long run.

4. Two-level multi-agent framework

4.1. Overview of the framework

In [4], a two-level framework for task allocation problem involving agent coalition formation and negotiation has been proposed. Coalition formation has been proposed as an effective strategy for self-interested agents to perform a common task (see for instance [10]). Negotiation is a fundamental technique in conflict resolution among self-interested agents [7]. The strength of a two-level framework lies in the ability of the two levels to concentrate on different aspects of the problem goal, and ultimately achieve a good balance between them. More explicitly, the upper level of the framework concentrates on the quality of the solution and uses a centralized approach to find an initial solution with low cost; in the lower level, agents will negotiate among themselves to further improve the initial solution according to their own strategies which will in turn increase agents' interests to stay in the system.

Motivated by the two-level framework proposed in [4], we propose here a two-level framework to solve the generalized OCOP. The key idea is that instead of asking agents to report their options in non-decreasing order of cost, we relax the requirement to ask agents to report their preferred options in *tiers*. For example, the agent may be asked to report its top 10% of most preferred options to the central solver without telling the actual *order* among these options and the costs associated. This tier information is much less sensitive than the exact ranking of the options, since the central solver or other agents may never guess the actual costs of the options reported nor the relevant importance accrued to them by the agent. Hence, the agent is more assured of privacy. Furthermore, since tiers are themselves reported in sequence (as we shall see later), if the central solver can find a feasible solution from the topmost tier of options reported, this initial solution will serve as a very good starting point for the lower-level negotiation process.

Here is the overview of the framework:

In the upper-level initial solution construction phase, the central solver (henceforth called **CS**) will ask agents to report their top-tier options and attempt to find a feasible solution. Note that this reduces the problem to a standard CSP which can be solved using any conventional CSP algorithm. If a feasible solution is found, it will be passed to the lower-level negotiation phase as an initial solution; otherwise, CS will first identify the failed variable which is the deepest node reached in the search with inconsistency detected (see [2]), and collect the next tier of options from only agents that have the failed variable in their options. The reason for identifying the failed variable is to reduce the amount of information collected from agents and focus on those agents that caused the failure first. A second CSP is solved with the enlarged domains. If this also fails, then the next iteration is invoked where all agents are asked to report the next tier of options. The process is repeated until a feasible solution is found.

The lower-level negotiation phase can be seen as an improvement phase where agents will form coalitions and those who are not satisfied with the initial solution will propose new options to CS. The new proposals will be negotiated among all affected agents and a consensus is achieved through an exchange of *persuasive points*. As agents are autonomous in the negotiation, their satisfaction level will be clearly higher than the upper-level phase where the agents merely take orders from CS.

4.2. Persuasive points

Persuasive Points (***PP*** for short) represents the agent's persuasive power during negotiation and serves as the "goodies" which can be exchanged with other agents in order to achieve some consensus. A similar idea of persuasion was recently proposed in [11]. Initially, each agent will obtain equal amount of ***PP*** which will be accumulated and carried over throughout the whole negotiation process. During a negotiation phase, each agent can persuade other agents to accept its proposal by attaching some ***PP*** which can later be distributed among agents who buy into this proposal. Agents will be compensated with some ***PP*** if they make some compromise. For example, when CS sends two options <Op1, Op2> to agent A who prefers Op2, A will return <Op2, *p*> back to CS where *p* is the amount of ***PP*** it is willing to give out if Op2 is chosen eventually.

5. Upper level: initial solution construction

The objective of upper-level initial solution construction phase is to find a feasible solution from the top tiers of options reported by the agents which will be later passed to the lower-level negotiation phase. The quality of the initial solution is important because it serves as the basis for optimization in the lower level negotiation phase. A high quality initial solution will help to produce a better final solution.

Definition 3. The *choice set* is the set of domains composed by the union of all tiers of options reported by the agents.

Clearly, the choice set provides fixed domains for all variables resulting in a standard CSP. Note that the choice set will grow as agents are asked to report lower tiers of options and eventually become the entire set of agents' options. For simplicity, we assume a *feasible* solution always exists when that happens.

Definition 4. A variable is called *tight* if its domain becomes empty when a CSP is being solved.

Definition 5. The *degree* of the variable is the number of constraints associated with this variable in the constraint system.

The function **`g-search`** describes how to construct an initial solution from the tiers of options reported by the agents in the upper-level. There are many CSP solvers available and here we use **`CSP-Solver`** to denote any efficient CSP solver that solves the reformulated CSP in Definition 1 induced by *X*, *A*, and the domains given by the current choice set.

```
function g-search
$X \leftarrow \{x_1, x_2 \ldots x_n\}$;  $A \leftarrow \{a_1, a_2 \ldots a_m\}$
Sort X in non-increasing order of degree
loop
   $ChoiceSet \leftarrow \bigcup_{i=1,2\ldots m} nextTier(a_i)$
   $s \leftarrow$ CSP-Solver()
   if s is empty
        $v^t \leftarrow$ first tight variable detected
        $A^t \leftarrow \{a_i \in A \mid v^t \in a_i\}$
        $ChoiceSet \leftarrow ChoiceSet \cup \bigcup_{a_i \in A^t} nextTier(a_i)$
        $s \leftarrow$ CSP-Solver()
   if s is not empty return s
end loop
```

6. Lower level: agent negotiation

6.1. Agent coalition

Intuitively, the change of one variable's domain may affect the value assignment of some other variable if both appear in a constraint. Thus, we want to find a way to gather agents into coalitions such that the change of a variable's domain will only affect the variables within the same coalition.

We ask agents to form coalitions when they have common variables in their option domain. The variables then become the variables of this coalition, and formed the ***coalition variables set***. For example, if agent A_1 has ***(X, Y)*** as its constraint and agent A_2 has ***(Y, Z)*** as its constraint, then A_1A_2 will form a coalition with coalition variable set ***(X, Y, Z)***. If an agent A_3 has ***(W, Y)*** as its constraint, A_3 will be added to the coalition of A_1A_2 and the coalition variables set will now become ***(W, X, Y, Z)***. In other words, agents can be added to a coalition if they have variables that are elements of the existing coalition variable set. If the newly added agent have variables that are not yet in the coalition variable set, the variables will be added to the coalition variable set.

Since changing the value of any variable will have a knock-on effect only on the remaining variables in the coalition variable set, we can divide agents into disjoint coalition. CS will only communicate with agents who are concerned with the change and thus reduce the amount of information needed to be transferred. In the following parts of paper, we will only elaborate the negotiation process within the same coalition.

6.2. Agent negotiation

Definition 6: One *negotiation round* is defined as the process that each agent in the coalition re-proposes a new option to CS if the agent is not satisfied with the value assignment of its original option in the current solution.

Definition 7: The *current solution* is the latest solution that is agreed by all the agents within the coalition. The initial solution constructed in the upper-level is the first current solution used in the lower-level. The current solution will change after each negotiation round if a better solution is agreed by all the agents.

When the initial solution is passed to the lower-level negotiation system, only CS knows the whole solution while agents will only be notified about their options selected. By this way we protected agent's personal information and could set a more fair ground for the following operations.

The negotiation process works as follows: in each negotiation round, agents who are not satisfied with their options in the current solution can re-propose a new option to CS in the form of $<Op_{new}, p>$ where Op_{new} is the new option and *p* is the amount of *PP* the agent is willing to give out if Op_{new} is adopted. CS will then pass the new option as well as the old option in the current solution to the rest of the agents without telling the agents the cost difference between these two in the form of $<Op_{original}, Op_{new}>$. Agents have to make a choice between the two options and feed back to CS in the form of $<Op', p>$ where Op' is the preferred option and *p* is the amount of *PP* to be given out if Op' is adopted. CS will calculate the ***PP*** attached to both options and the new option will be chosen as one candidate option if it has more *PP* attached. At the end of one negotiation round, CS will examine all the candidate options, get the option costs for the candidate options and the one with the most significant decrease in the overall cost will be chosen to replace the current solution. CS will return twice of *PP* to those agents who oppose the new option adopted as the compensation and keep the rest of *PP* to itself.

Here, CS only acts as a broker to pass information among agents and adjust agents' ***PP*** after a new assignment of variable is agreed within the coalition. CS will also facilitate the communication among different coalitions so that a better and yet consistent assignment will be found.

6.3. Agents' strategies

As mentioned above, after receiving a new proposal from CS, the agent need to feedback to CS which option it prefers and the amount of ***PP*** that it is willing to give out if its preferred option is adopted. The amount of ***PP*** that it can give out is determined by the agent's own strategy and could differ from each other in real application. In this paper, to make the problem easy to model, we assume all agents will use the same strategy to calculate the amount of ***PP*** to give out. We assume the cost of the agent's option in the original solution is $C_{original}$. If the new proposal from CS is adopted, the agent will need to change its option to a new one in order accordingly if they have some common variables. We assume the cost of the new option to be C_{new}. The amount of ***PP*** the agent is willing to give out for the more preferred option is $|C_{original} - C_{new}|$.

6.4. Redistribution of *PP*

Upon receiving feedbacks from agents for the new proposal, CS will compare the ***PP*** associated with both the original and new option. We define the sum of ***PP*** for the original option as $PP_{original}$ and the ***PP*** for the new proposal as PP_{new}. If $PP_{new} > PP_{original}$, the new option will be chosen as one candidate change to the original solution in the current negotiation round and discarded otherwise. Agents who failed to get their options adopted will get back their *PP* and be compensated with the same amount of ***PP*** as they have associated with their preferred options before. Agents who succeed to get their options adopted will give out the ***PP*** they have associated with their preferred option as a compensation for those agents who lose in the negotiation. The absolute difference between the ***PP***$_{\text{original}}$ and ***PP***$_{\text{new}}$ will be kept by CS and distribute evenly to all the agents after certain pre-defined negotiation rounds.

Agents are discouraged from cheating under this protocol. For example, if agent A preferred the new proposal and reported a ***PP*** higher than the actual amount that should be reported, agent A is at its own risk that it has to give out more ***PP*** if the new proposal is adopted. Same situation happens when agent A reported a ***PP*** lower than the actual amount. In this case, agent A is at the risk that its preferred option may be discarded. As agent A has no way to predict the consequence of reporting a fake ***PP***, it will not be encouraged to cheat during negotiation.

6.5. Repeated negotiation rounds

As the negotiation proceeds in rounds, the result and ***PP*** distribution in one negotiation round will affect the result and ***PP*** distribution in the next round. In order to prevent a variable being always the focus of the negotiation process, we decide that when a variable value is agreed in the current round, it will not change in the following rounds unless the agents who want to change its value can afford to pay the ***PP*** to persuade all the agents and the ***PP*** that previous agents have paid on changing the variable to its current value in the previous round.

6.6. Negotiation termination condition

As the communication cost is expensive, CS will terminate the negotiation when a pre-defined number of negotiation rounds is reached. The best solution reached will become the final solution.

7. Experimental results

7.1. Experiment setup

We set the number of variables (i.e. size of the variable tuple) associated with each agent to be within the range [2, 4]. We first generate a fixed number of agents and a fixed number of variables, and then randomly associate each agent with variables from the variable pool. We then generate all possible options by enumerating the value assignments of all the variables, and assign a random cost in the range of [0, 20] to these options. In order to make the number of options lie within a reasonable computational range, we set the domain size for all variables to be a small value of 5. Within each agent, we sort its options in non-decreasing order of cost, and indicate its order by an *option index*. For a fixed solution (i.e a fixed assignment of values to variables), it is easy to determine the option index of this solution with respect to a given agent (simply by locating the corresponding value tuple on the agent's rank table).

7.2. Results

We first benchmark our results against the optimal solutions obtained in [1] in terms of cost with a fixed tier-size and later show the effect of different tier-size on both the upper level result and the final result. When the tier-size is set to 1, agents will actually report their options in non-decreasing order of cost and thus the upper level will work in the same way as [1] and hence find the optimal solution. To make the comparison more meaningful, we need to choose any tier-size greater than 1. In this case, tier-size of 3 is chosen. As introduced in 3.4, we introduce the notion of *option index standard deviation* $\boldsymbol{\sigma}$ to measure the fairness of the solution which is defined as follows:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{\sum_{r=1}^{R} I_i^r}{R} - \frac{\sum_{r=1}^{R} \mu^r}{R}\right)^2}$$

where $\boldsymbol{N}$ denotes the agent number, R the total number of negotiation rounds tested, I_i^r the option index for agent $\boldsymbol{i}$ at negotiation round r, μ^r the average option index over all agents (i.e. all variable tuples) at negotiation round r. We use $\boldsymbol{\sigma}$ as an overall measure on the overall agents' satisfaction level. We plot different problem instances (with agent numbers ranging from 7 to 20 and variable pool size ranging from 10 to 20) against the two metrics.

From Fig 2, we can see that the total cost computed by our approach is reasonably near to optimality. The worst-case ratio between our result and the optimal solution is 21%, and the average ratio is 13%. In terms of standard deviation however, Fig 3 shows that that $\boldsymbol{\sigma}$ has a value within the low range of [1.4, 2.1], implying that agents do not have a large deviation from the average option index and thus illustrates the fairness of our approach.

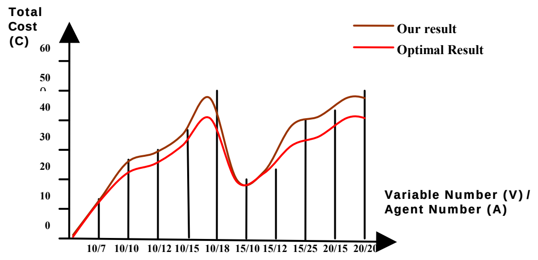


Fig 2: Comparison with respect to total cost

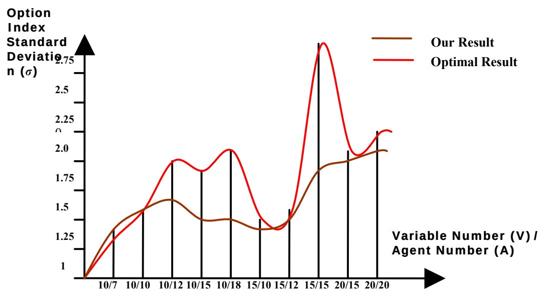


Fig 3: Comparison with respect to standard deviation

Tier-size	Total cost in Optimal Solution	Total cost in Upper Level Soln	Upper Level Solution Ratio (%)	Total cost in Final Solution	Final solution Ratio (%)	σ in Upper Level Solution	σ in final solution
1	19	19	0	21	10.53	1.46	1.33
2	19	21	9.52	20	5.26	1.53	1.47
3	19	21	9.52	20	5.26	1.53	1.47
5	19	22	15.79	22	15.79	1.89	1.78
8	19	28	47.37	22	15.79	1.87	1.84
10	19	32	68.42	23	21.05	2.23	1.97
15	19	42	121.05	27	42.11	2.17	2.12
20	19	47	147.37	28	47.37	2.21	2.14

Table 1: Effects of different tier sizes

We next investigate the effects of tier-size on our approach for a 10-agent 15-variable problem. Table 1 presents the results. We can see that the general trend is the ratio between the total cost of the initial solution and the optimal solution increases as tier-size increases. This is not surprising as when the tier-size increases, information is lost and hence widening the gap to the optimal solution. In the lower level, standard deviation ***σ*** will decrease as compared with ***σ*** achieved in the upper level. This shows that the lower level negotiation is effective and improves the agents' overall satisfaction level. As for the total cost, the general trend is that the lower level will yield better reduced cost for larger tier-sizes (for which the upper-level solution has high total cost). Unfortunately, there is no guarantee for all tier-sizes, as agents are more interested to improve their overall satisfaction level rather than the total cost in the negotiation. For example, when tier-size is set to 1, the upper level will find the optimal solution, while the lower level will take place to improve the overall satisfaction level (improved from 1.46 to 1.33 as shown in the table) at the cost of increase of total cost (from 19 to 21).

8. Conclusions and Future Work

In this paper, we studied a generalized version of OCOP where the agents are self-interested and the objective is not only to reduce the total cost but also improve the agents' overall satisfaction level in the long run. We proposed a two-level framework that searches for the solution based on the tiers of options reported by agents in order to better protect the agents' privacy. Experimental results show that the two-level framework has achieved a good balance between the total cost of the solution and the agents' overall satisfaction level in the long run. We regret that we cannot achieve a smaller ratio when benchmarked against [1] in terms of total cost.

Possible future work includes the study of the generalized OCOP with respect to other objectives such as those discussed in section 3.4. It is also interesting to investigate other metrics for agent's satisfaction level other than the one proposed in this paper. In addition, more experiments should be performed to measure the communication cost under this framework and how it scales with the size of input data. Finally, we await more effective approaches from the community to solve the generalized OCOP proposed in this paper.

References

[1] B. Faltings and S. Macho-Gonzalez, "Open Constraint Optimization", Proc. *CP-2003*, pp.303-317

[2] B. Faltings and S. Macho-Gonzalez, "Open Constraint Programming", *Artificial Intelligence*, 161(2005), pp.181-208, 2005

[3] C. Bessiere, "Arc-Consistency in Dynamic Constraint Satisfaction Problems.", Proc. AAAI, pp.221-226, 1991

[4] H. C. Lau and L. Zhang, "A Two-level Framework for Coalition Formation via Optimization and Agent Negotiation", Proc. *IEEE/WIC/ACM Int'l Conf. Intelligent Agent Technology*, 441-445, Beijing, China, 2004

[5] M. Yokoo, "Algorithms for Distributed Constraint Satisfaction: A Review", *Autonomous Agents and Multi-Agent Systems*, September, 2003

[6] M. Silaghi, "Incentive Auctions and Stable Marriages Problems Solved with [n/2]-Privacy of Human Preferences", Technical Report, Florida Inst Tech, CS-2004-11

[7] N. Jennings, et al., "Automated Negotiation: Prospects, Methods and Challenge", *J. Group Decision and Negotiation*, 2000

[8] P. Galinier and J. Hao, "Tabu Search for Maximal Constraint Satisfaction Problems", Proc. *CP-1997*, pp.196-208.

[9] R. Mailler and V. Lesser, "Solving Distributed Constraint Optimization Problems Using Cooperative Mediation", Proc. *AAMAS-04*, New York, USA

[10] T. Sandholm et al, "Coalition Structure Generation with Worst Case Guarantees." *Artificial Intelligence*, 111(1-2), pp. 209-238, 1999

[11] S. Ramchurn, "Multi-Agent Negotiation Using Trust and Persuasion", Ph.D Thesis, University of Southampton, December, 2004

[12] Y. Ran, N. Roos, J. van den Herik, "Approaches to find a near-minimal change solution for Dynamic CSPs", Proc. *CPAIOR*, pp.373-387, 2002