Coalition Structure Generation Based on Distributed Constraint Optimization

Suguru Ueda and **Atsushi Iwasaki** and **Makoto Yokoo**
Kyushu University
{ueda@agent., iwasaki@, yokoo@}is.kyushu-u.ac.jp

Marius Călin Silaghi
Florida Institute of Technology
msilaghi@fit.edu

Katsutoshi Hirayama
Kobe University
hirayama@maritime.kobe-u.ac.jp

Toshihiro Matsui
Nagoya Institute of Technology
matsui.t@nitech.ac.jp

Abstract

Forming effective coalitions is a major research challenge in AI and multi-agent systems (MAS). Coalition Structure Generation (CSG) involves partitioning a set of agents into coalitions so that social surplus (the sum of the rewards of all coalitions) is maximized. A partition is called a coalition structure (CS). In traditional works, the value of a coalition is given by a black box function called a characteristic function. In this paper, we propose a novel formalization of CSG, i.e., we assume that the value of a characteristic function is given by an optimal solution of a distributed constraint optimization problem (DCOP) among the agents of a coalition. A DCOP is a popular approach for modeling cooperative agents, since it is quite general and can formalize various application problems in MAS. At first glance, one might imagine that the computational costs required in this approach would be too expensive, since we need to solve an NP-hard problem just to obtain the value of a single coalition. To optimally solve a CSG, we might need to solve $O(2^n)$ DCOP problem instances, where n is the number of agents. However, quite surprisingly, we show that an approximation algorithm, whose computational cost is about the same as solving just one DCOP, can find a CS with quality guarantees. More specifically, we develop an algorithm with parameter k that can find a CS whose social surplus is at least $\max(k/(w^*+1), k/\lfloor n/2 \rfloor)$ of the optimal CS, where w^* is the tree width of a constraint graph. When $k = 1$, the complexity of this algorithm is about the same as solving just one DCOP. These results illustrate that the locality of interactions among agents, which is explicitly modeled in the DCOP formalization, is quite useful in developing an efficient CSG algorithm with quality guarantees.

Introduction

Coalition formation is an important capability in automated negotiation among self-interested agents. Coalition Structure Generation (CSG) involves partitioning a set of agents into coalitions so that social surplus (the sum of the rewards of all coalitions) is maximized. A partition is called a coalition structure (CS). This problem has become a popular research topic in AI and multi-agent systems (MAS). Possible applications of CSG include distributed vehicle routing (Sandholm and Lesser 1997), multi-sensor networks (Dang et al. 2006), etc. Solving CSG is equivalent to a *complete set partition problem* (Yeh 1986), and various algorithms for solving CSG have been developed (Rahwan et al. 2007; Rahwan and Jennings 2008; Rothkopf, Pekeč, and Harstad 1998; Sandholm et al. 1999; Yeh 1986). In these traditional works, the value of a coalition is given by a black box function called a characteristic function. A notable exception is (Ohta et al. 2009). In this work, the value of a coalition is calculated by applying a set of rules, such as MC-nets (Ieong and Shoham 2005), rather than a single black box function. The motivation for using a set of rules rather than a black box function is to represent a characteristic function more concisely. However, it remains unclear what kinds of problem domains can be concisely represented by such rules.

Let us reconsider the meaning of the value of a coalition. It represents the optimal gain achieved by agents in the coalition when they work together. Thus, it is natural to think that the value is obtained by solving some optimization problem among the agents of the coalition. This idea is also pointed out in (Sandholm et al. 1999). Deng, Ibaraki, and Nagamochi (1997) also present a concept called combinatorial optimization games, in which the value of a coalition is given by solving various graph-related combinatorial optimization problems, such as the maximum flow problem.

After a pioneering work by (Modi et al. 2003), a Distributed Constraint Optimization Problem (DCOP) has become a popular approach for modeling cooperative agents. Various algorithms have been developed, including DPOP (Petcu and Faltings 2005), OptAPO (Mailler and Lesser 2004), NCBB (Chechetka and Sycara 2006), etc. In DCOP, each agent has a choice of actions (values). Reward/cost is determined by the combination of values. The goal is to maximize/minimize the sum of the rewards/costs. This framework is quite general and can represent various application domains in cooperative MAS. In this research, we assume that the value of a coalition is given by an optimal solution of a DCOP among the agents of the coalition.

Let us introduce some motivating examples that can be formalized as CSG based on DCOPs. Consider the problem of forming rescue groups in a disaster area. There exists a set of people T. Each person has following different capabilities: providing medical treatment, driving a vehicle, acting as a firefighter, etc. We want to create as many rescue groups as possible, but at the same time, we need to

make sure that the members of each group have enough capabilities. Having people with identical capabilities in one group can be wasteful. We can assume that each person is a variable, whose domain consists of the possible combinations of his/her capabilities to perform. Thus, there exist positive relations/rewards between complementary capabilities, and there exist negative relations/rewards between substitutable capabilities. Also, the vehicle routing problem described in (Sandholm 1993) can be formalized as CSG based on DCOPs, where geographically dispersed dispatch centers of different companies cooperate.

Although a DCOP is a very general and powerful framework, the required computational costs might be too expensive, because we need to solve an NP-hard problem just to obtain the value of a single coalition. To optimally solve a CSG, we might need to solve $O(2^n)$ DCOP problem instances, where n is the number of agents. However, quite surprisingly, we show that an approximation algorithm, whose computational cost is about the same as finding the value of the grand coalition (the coalition of all agents), can find a CS with quality guarantees. More specifically, we develop an algorithm with parameter k that can find a CS whose social surplus is at least $\max(k/(w^*+1), k/\lfloor n/2 \rfloor)$ of the optimal CS, where w^* is the tree width of a constraint graph. When $k = 1$, the complexity of this algorithm is about the same as solving just one DCOP. More specifically, assuming the size of a variable domain (the number of possible values) is d, the search space size of a DCOP is d^n, while the search space size of the approximation algorithm is $(d+1)^n$ for $k = 1$. Also, we develop an anytime algorithm that repeatedly applies the approximation algorithm and experimentally show that the average approximation ratio of the approximation algorithm is by far superior to the theoretical worst-case bound.

The contribution of this work is two-fold: (1) for CSG research, this work introduces a novel representation scheme of a characteristic function and efficient approximation algorithms with worst-case guarantees, and (2) for DCOP research, this work introduces a promising new, but computationally challenging application domain that requires an extension of traditional DCOP formalization. We believe this paper introduces a new exciting research domain, which is built on the success of existing CSG and DCOP research.

Model

Let $T = \{1, 2, \dots, n\}$ be the set of all agents. We assume that the value of coalition S is given by a characteristic function v. A characteristic function $v : 2^T \rightarrow \mathbb{R}$ assigns a value to each set of agents (coalition) $S \subseteq T$.

We assume that the value of a coalition S, $v(S)$, is defined as an optimal solution of a DCOP among the agents of coalition S. Agent i has variable x_i, which represents the choice of its action. Agent i chooses the value of x_i from a finite, discrete domain D_i. The rewards on the values of these variables are determined as below. For notation simplicity, we assume that reward functions are either unary or binary, which is a standard convention used in DCOP literature. The results presented in this paper hold when there exist k-ary reward functions for $k \geq 3$.

There exists unary reward $r_i : D_i \rightarrow \mathbb{R}$ on the value $d_i \in D_i$ of variable x_i, which represents reward $r_i(d_i)$ when agent i chooses value (action) d_i. We assume that there exists at least one value $d_i \in D_i$ such that $r_i(d_i) \geq 0$. Also, there exists binary reward $r_{i,j} : D_i \times D_j \rightarrow \mathbb{R}$ on variables x_i and x_j, which represents reward $r_{i,j}(d_i, d_j)$ when agents i and j are in the same coalition and chooses values (actions) d_i and d_j, respectively. Let us denote that $A \in \Pi_{i \in S} D_i$ is a value assignment to all variables. Thus, $v(S)$ is defined as:

$$\max_A \{\sum_{i \in S} r_i(d_i) + \sum_{i,j \in S} r_{i,j}(d_i, d_j)\}.$$

A coalition structure CS is a partition of T, into disjoint, exhaustive coalitions. More precisely, $CS = \{S_1, S_2, \dots\}$ satisfies the following conditions:

$$\forall i, j (i \neq j),\ S_i \cap S_j = \phi,\ \bigcup_{S_i \in CS} S_i = T.$$

In other words, in a CS, each agent belongs to exactly one coalition, and some agents may be alone in their coalitions.

For example, if there exist three agents a, b, and c, then there are seven possible coalitions: $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{b, c\}$, $\{a, c\}$, $\{a, b, c\}$, and five possible coalition structures: $\{\{a\}, \{b\}, \{c\}\}$, $\{\{a, b\}, \{c\}\}$, $\{\{a\}, \{b, c\}\}$, $\{\{b\}, \{a, c\}\}$, $\{\{a, b, c\}\}$.

The value of a coalition structure CS, denoted as $V(CS)$, is the sum of the value of coalitions, $V(CS) = \sum_{S_i \in CS} v(S_i)$, or equivalently,

$$V(CS) = \max_A \{\sum_{i \in S \in CS} r_i(d_i) + \sum_{i,j \in S \in CS} r_{i,j}(d_i, d_j)\}.$$

An optimal coalition structure CS^* maximizes the social surplus, i.e., CS^* satisfies the following condition: $\forall CS, V(CS) \leq V(CS^*)$, or equivalently,

$$CS^* = \arg\max_{CS} \max_A \{\sum_{i \in S \in CS} r_i(d_i) + \sum_{i,j \in S \in CS} r_{i,j}(d_i, d_j)\}.$$

In a standard DCOP, the goal is to find an optimal value assignment such that $v(T) = \{\sum_{i \in T} r_i(d_i) + \sum_{i,j \in T} r_{i,j}(d_i, d_j)\}$ is maximized. On the other hand, in our CSG formalization, if two agents work in different coalitions, there exists no positive/negative interaction between them. The goal is to partition agents into groups so that the obtained social surplus is maximized. In this formalization, there exists no positive/negative interaction among coalitions. This corresponds to a commonly used assumption in coalitional game theory called *no externality*, i.e., the gain of a coalition is independent from other coalitions.

We say a characteristic function is super-additive, if for any disjoint sets $S_i, S_j, v(S_i \cup S_j) \geq v(S_i) + v(S_j)$ holds. In super-additive cases, solving CSG becomes trivial, i.e., the grand coalition (the coalition of all agents) is optimal. In our DCOP formalization, we can represent a non-super-additive characteristic function. Furthermore, a characteristic function is not restricted to *monotone*, i.e., in our DCOP formalization, there is a chance that for two coalitions, S and S', where $S \supset S'$, $v(S)$ can be smaller than $v(S')$.

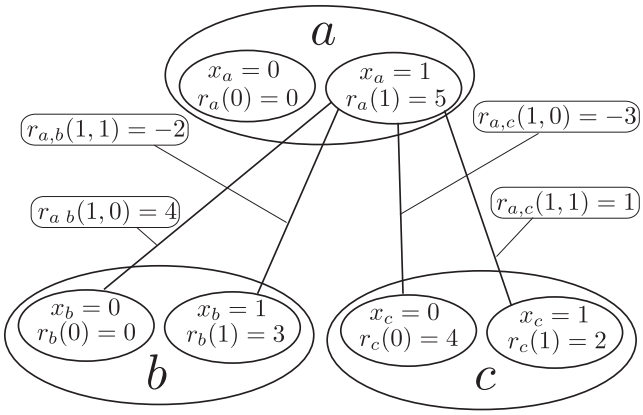


Figure 1: Reward Functions in Example 1

Example 1 *Let there be three agents a, b, and c. We assume that each agent has two actions. Thus, we assume that there are three variables x_a, x_b, and x_c, and the domain of each variable is $\{0, 1\}$.*

The unary/binary rewards are as follows. We assume that all reward values that are not explicitly described are 0 (Figure 1).

$$r_a(1) = 5, \quad r_b(1) = 3, \quad r_c(0) = 4,$$
$$r_c(1) = 2, \quad r_{a,b}(1,0) = 4, \quad r_{a,b}(1,1) = -2,$$
$$r_{a,c}(1,0) = -3, \quad r_{a,c}(1,1) = 1.$$

The value of $v(\{a, b, c\})$ is obtained by solving a DCOP among three agents. The optimal value assignment is $x_a = 1, x_b = 0, x_c = 1$, and $v(\{a, b, c\})$ is calculated as $r_a(1) + r_b(0) + r_c(1) + r_{a,b}(1,0) + r_{b,c}(0,1) + r_{a,c}(1,1) = 12$. Also, the value of $v(\{a, b\})$ is obtained by solving a DCOP between x_a and x_b. The optimal assignment is $x_a = 1, x_b = 0$, and $v(\{a, b\})$ is calculated as $r_a(1) + r_b(0) + r_{a,b}(1,0) = 9$. In this case, if agent c acts alone, it can obtain unary reward 4 by setting its value to 0. Thus, in this case, the grand coalition is not optimal. The optimal coalition structure is $\{\{a, b\}, \{c\}\}$ and the optimal value assignment is $x_a = 1, x_b = 0, x_c = 0$.

Representation Size of DCOP Formalization

To explicitly represent a characteristic function, we need to specify the value of a coalition for all 2^n possible coalitions. If the number of reward functions is small, say, there exist only unary/binary rewards, the number of reward functions is only $O(n^2)$. Thus, we can represent a characteristic function much more concisely. Obviously, using k-ary reward functions, we can represent any characteristic function as a DCOP. However, the motivation for using our DCOP formalization is not to represent an arbitrary characteristic function, but to directly represent the underlying optimization problem among agents.

Our scheme can compactly represent any characteristic function that can be compactly represented by other schemes, such as *Marginal contribution networks* (MC-nets) (Ieong and Shoham 2005) and *Synergy coalition group* (SCG) (Conitzer and Sandholm 2006). On the other hand, there exists a characteristic function that can be compactly represented by our scheme, but other schemes need exponentially more space, i.e., the following theorems hold.

Theorem 1 *Any characteristic function that is compactly represented by MC-nets and SCG can also be compactly represented using a DCOP formalization, i.e., the number of required rules/constraints increases at most in $O(n)$.*

Theorem 2 *There exists a characteristic function that can be represented compactly using a DCOP formalization, while MC-nets and SCG require exponentially more space to represent this characteristic function.*

We omit proofs due to space limitations. These results are intuitively natural since characteristic functions represented by DCOP formalization would be more complex than those of MC-nets and SCG.

Complexity of CSG with DCOP

One of the central research questions in compact representation schemes of a characteristic function is the computational complexity of finding or proving the existence of solution concepts (e.g., core, Shapley value). Our DCOP formalization is not good for this purpose. Actually, even a very simple problem, i.e., checking the feasibility of an imputation (a value distribution among agents of a coalition), is NP-complete since we need to solve a DCOP.

Theorem 3 *Checking whether an imputation is feasible is NP-complete.*

Proof *When the value assignments of variables are given, calculating the value of a coalition and checking the value is more than or equal to the sum of the values in the imputation can be done in polynomial time. Thus, this problem is in class NP.*

Next, we reduce a decision version of a standard COP problem, i.e., checking whether there exists a solution of a COP better than a given threshold τ, to this feasibility checking problem. For the original COP, we create a feasibility checking problem instance with exactly the same variables and reward functions. Also, we create an imputation where the sum of the values in the imputation is equal to τ. If the answer to the obtained feasibility checking problem instance is "yes," then the answer to the original problem is also "yes," and vice versa. Since the decision problem of a COP is NP-complete, this feasibility check problem must be NP-hard. Also, since it is in class NP, it is NP-complete. □

Next, we examine the computational complexity of CSG using our DCOP formalization. We prove that a decision problem of CSG using a DCOP formalization is NP-complete.

Theorem 4 *A decision problem of CSG using a DCOP formalization, i.e., checking whether there exists a CS where $V(CS)$ is greater than a given threshold is NP-complete.*

Proof *For a given CS and value assignments, checking whether $V(CS)$ is greater than or equal to a given threshold can be solved in polynomial time. Thus, this problem is in class NP. Next, we reduce a standard COP problem, where all reward values are non-negative, to CSG. For the original COP, we create a CSG problem instance with exactly the same variables and reward functions. Since all reward values are non-negative, a CS that contains only the grand coalition is optimal. More specifically, if CS^* contains multiple coalitions, then we can create another CS', which uses the same value assignments as CS^* but all coalitions are merged into a single coalition. Since all binary reward values are non-negative, $v(CS^*) \leq v(CS')$ holds. Thus, an optimal solution of the original COP must be identical to the solution in the CSG problem instance. Since the decision problem of a COP is NP-complete, a CSG with DCOP is NP-hard. Since it is also in class NP, it is NP-complete.* □

Approximation Algorithm

Basic Ideas

The main idea of our approximation algorithm is to only search for a restricted subset of all coalition structures without explicitly calculating the value of coalitions. More specifically, we search for coalition structures, each of which contains only k coalitions with multiple agents (referred to as multi-agent coalitions). Except for the multi-agent coalitions, every other coalition contains only a single agent (referred to as single-agent coalition). To search for a restricted subset of coalition structures, we slightly modify the original DCOP instance.

Details of Approximation Algorithm

We create a DCOP problem instance by slightly modifying the original DCOP problem as follows, so that we can search for coalition structures that contain at most k multi-agent coalitions.

- We add one new value for each variable called "independent," which means that the agent acts independently. The unary reward of "independent" equals the maximal unary reward for other values.
- Also, agent i has k copies of its domain D_i. For example, consider the case where i has $D_i = \{d_1, d_2\}$ and $k = 2$. Then, agent i has a new domain $D'_i = \{d_{1,1}, d_{2,1}, d_{1,2}, d_{2,2}, independent\}$.
- Each value in a copied domain indicates which action the agent chooses and to which coalition it belongs.
- All binary rewards related to at least one "independent" value are 0, since the agent who chooses "independent" forms its own coalition.
- The unary reward value of a copied value is identical with the original value. In the above example, $r_i(d_{1,1}) = r_i(d_{1,2}) = r_i(d_1)$ and $r_i(d_{2,1}) = r_i(d_{2,2}) = r_i(d_2)$.
- A binary reward value between copied values belonging to the same coalition equals the original binary reward values, otherwise, the reward value is 0.

We can solve this new DCOP using any existing algorithms that can obtain an optimal solution, such as ADOPT, DPOP, etc. (or any centralized COP algorithms, e.g., (Dechter 1999), if we can collect all information to a centralized server)[1]. By using the obtained solution, we create a CS where an agent who chooses "independent" forms its own coalition, and an agent who chooses a value within j-th copy joins j-th coalition[2].

The search space of this DCOP is $(kd+1)^n$, where d is the domain size of the original DCOP. When $k = 1$, the search space is $(d+1)^n$. Thus, if $k = 1$, the computational cost of this algorithm is about the same as solving the original DCOP, whose search space is d^n.

Let us show a simple example where $k = 1$.

Example 2 *Consider the agents and reward-functions of example 1. We add one new value for each variable called "independent." Since $k = 1$, we don't add copied values for each variable. Thus, the domain of each variable is $\{0, 1, independent\}$. The optimal value assignment of this new DCOP problem instance is $x_a = 1, x_b = 0, x_c = independent$. In this case, this assignment means that agents a and b form coalition $\{a, b\}$ and agent c forms its own coalition $\{c\}$. Thus, $V(\{\{a, b\}, \{c\}\})$ is calculated as $r_a(1) + r_b(0) + r_c(independent) + r_{a,b}(1, 0) + r_{b,c}(0, independent) + r_{a,c}(1, independent) = 13$. In this way, we can obtain a CS by solving just one DCOP instance. In this case, the obtained coalition structure is optimal.*

Worst-case Bound based on Number of Agents

We first show that a bound can be given based on the number of multi-agent coalitions in CS^*.

Theorem 5 *The worst-case ratio of the approximate algorithm is k/l, i.e., for obtained coalition structure CS^{ap}_k, $V(CS^*) \leq l \cdot V(CS^{ap}_k)/k$ holds, where l is the number of multi-agent coalitions in CS^*.*

Proof *We assume CS^* contains l multi-agent coalitions. Consider another coalition structure CS'_k, which is obtained as follows. First, we set $CS'_k = CS^*$. Then, we continue to choose multi-agent coalition S, where $v(S)$ is the smallest in CS'_k, and divide it into single-agent coalitions, until k multi-agent coalitions remain in CS'_k. In other words, we divide $l - k$ multi-agent coalitions in CS^*. Let us assume S_{min} is a multi-agent coalition in CS'_k, where $v(S_{min})$ is the smallest. By dividing a multi-agent coalition into single-agent coalitions, the maximal loss is less than or equal to $v(S_{min})$, since we divide only coalition S', where $v(S') \leq v(S_{min})$ holds. Note that we assume there exists at least one*

[1] To be more precise, some existing algorithms including ADOPT are originally designed for cost-minimizing problems. Also, the existence of negative rewards can be problematic for pruning. However, we can directly apply DP-based algorithms such as DPOP. Furthermore, we can apply a simple modification of ADOPT by assuming the possible value of each reward function is bounded.

[2] In $(kd+1)^n$ search space, there are $k!$ duplicated solutions, since the names of groups/coalitions are "indistinguishable." We can reduce such duplicated solutions by adding symmetry breaking constraints (Gent 2001).

value $d_i \in D_i$ for x_i where $r_i(d_i) \geq 0$. Thus, the value of a single-agent coalition is non-negative.

Since CS'_k contains k multi-agent coalitions, $v(S_{min}) \leq V(CS'_k)/k$ holds. The solver maximizes the DCOP, i.e., $V(CS'_k) \leq V(CS^{ap}_k)$ holds. Thus, $v(S_{min}) \leq V(CS^{ap}_k)/k$ holds and the following conditions hold:

$$\begin{aligned} V(CS^*) &\leq (l-k)v(S_{min}) + V(CS'_k) \\ &\leq (l-k)v(S_{min}) + V(CS^{ap}_k) \\ &\leq \tfrac{l}{k} \cdot V(CS^{ap}_k). \end{aligned}$$

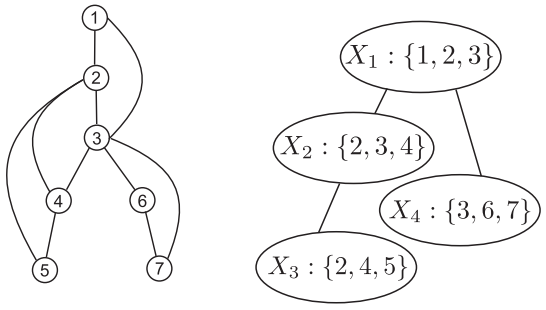
□

Theorem 6 *The worst-case ratio of the approximate algorithm is $k/\lfloor n/2 \rfloor$, i.e., for obtained coalition structure CS^{ap}_k, $V(CS^*) \leq \lfloor n/2 \rfloor \cdot V(CS^{ap}_k)/k$ holds. Also, this bound is tight.*

Proof *The proof is obvious from Theorem 5 and the fact that there are at most $\lfloor n/2 \rfloor$ multi-agent coalitions in CS^*. When the values of single-agent coalitions are zero and the multi-agent coalitions are 1, this bound is tight.* □

Worst-case Bound based on Tree Width

Instead of $\lfloor n/2 \rfloor$, we can use $w^* + 1$, where w^* is the *tree width* of a constraint graph. It is well-known that the tree width characterizes the complexity of various graph-based optimization algorithms, including CSP/COP (Diestel 2000; Dechter 2003).



(a) Original Graph (b) Tree Decomposition

Figure 2: Example of Tree Decomposition

For graph $G = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is a set of nodes and $\boldsymbol{E}$ is a set of edges, we denote the tree width of G as $tw(G)$. If original graph G is a tree, its tree width is 1. A pseudo-tree is often used in DCOP algorithms (Modi et al. 2003; Petcu and Faltings 2005). The tree width is related to the *induced width* of a pseudo-tree (Dechter 2003), i.e., if the tree width is w^*, then the induced width of any pseudo-tree is at least w^*.

We show a formal definition of the tree width below. It is rather complicated and difficult to understand. Please refer to graph theory textbooks, such as (Diestel 2000), for details. Formally, for graph $G = (\boldsymbol{V}, \boldsymbol{E})$, a tree decomposition of G is a pair $(\boldsymbol{X}, \boldsymbol{T})$, such that $\boldsymbol{X} = \{X_1, X_2, \ldots\}$, where each $X_s \in \boldsymbol{X}$ is a subset of $\boldsymbol{V}$ and $\boldsymbol{T}$ is a tree whose nodes are $\boldsymbol{X}$. $(\boldsymbol{X}, \boldsymbol{T})$ must satisfy the following conditions:

(node coverage): $\bigcup_{X_s \in \boldsymbol{X}} X_s = \boldsymbol{V}$,

(edge coverage): $\forall (i,j) \in \boldsymbol{E}, \exists X_s \in \boldsymbol{X}$, such that $i \in X_s$ and $j \in X_s$ hold.

(coherence): If $i \in X_s$ and $i \in X_t$, then $\forall X_z$, where X_z is a tree node and it is along the unique path between X_s and X_t, $i \in X_z$ holds.

The width of tree decomposition $(\boldsymbol{X}, \boldsymbol{T})$ is given as $\max_{X_s \in \boldsymbol{X}} |X_s| - 1$, and the tree width of G, which is denoted as $tw(G)$, is the minimum width among all possible tree decompositions of G. Figure 2(b) shows an example of the tree decomposition of the graph illustrated in Figure 2(a). Since for all X_s, $|X_s| \leq 3$ holds, the width of this tree decomposition is 2. This happens to be the minimum width for all decompositions, thus the tree width of this graph is 2.

Let us consider constraint network $G = (\boldsymbol{V}, \boldsymbol{E})$, where $\boldsymbol{V}$ is a set of agents and $(i,j) \in \boldsymbol{E}$ exists for each binary reward between x_i and x_j. The following theorems hold.

Theorem 7 *When $tw(G) = w^*$, then there exists an optimal coalition structure CS^* that contains at most $w^* + 1$ multi-agent coalitions.*

Proof *Assume that there exists an optimal coalition structure $CS^{*'}$ that contains more than $w^* + 1$ multi-agent coalitions. Without loss of generality, we can assume that within each $S \in CS^{*'}$, any two agents $i, j \in S$ are connected directly or indirectly by agents in S. If this is not the case for some S, we can divide S into multiple coalitions without reducing the value of the CS. Then, consider a graph $G' = (\boldsymbol{V}', \boldsymbol{E}')$, where $\boldsymbol{V}' = CS^{*'}$, i.e., each node is a coalition in $CS^{*'}$, and there exists an edge between two coalitions S and S', if there exists $x_i \in S$ and $x_j \in S'$, such that a binary reward exists between x_i and x_j. G' is obtained from G by* contracting *the edges between two agents that belong to the same coalition in $S \in CS^{*'}$. More precisely, contracting edge (i,j) means removing edge (i,j) and node j, and connecting all neighbors of j to i. The tree width does not increase by contraction (Diestel 2000). This is because, for tree decomposition $(\boldsymbol{X}, \boldsymbol{T})$, if we replace j to i in each $X_s \in \boldsymbol{X}$ (and remove duplications of i), then, we still have a valid tree decomposition (i.e., all three conditions still hold). Thus, $tw(G') \leq tw(G) = w^*$ holds.*

Also, it is well-known that if the tree width is w^*, the graph can be colored using $w^* + 1$ different colors, so that nodes connected by an edge have different colors (Diestel 2000). This is because if we color each node in $X_s \in \boldsymbol{X}$ differently, we obtain a valid coloring. Thus, G' can be colored using $w^* + 1$ different colors. Then, if S and S' have the same color, we can merge them without reducing the value of the coalition structure (since there exists no positive/negative rewards between S and S'). By merging coalitions with the same color, we obtain a coalition structure CS^* that contains at most $w^* + 1$ multi-agent coalitions and $V(CS^*) = V(CS^{*'})$ holds. □

Theorem 8 *If $tw(G) = w^*$, the worst-case ratio of the approximate algorithm is $k/(w^* + 1)$, i.e., for obtained coali-*

tion structure CS_k^{ap}, $V(CS^*) \leq (w^*+1) \cdot V(CS_k^{ap})/k$ holds. Also, this bound is tight.

We omit the proof due to space limitations. It is basically similar to the proof of Theorem 6.

Furthermore, w^* characterizes the complexity of the COP/DCOP algorithms. The following theorem holds.

Theorem 9 *If $tw(G) = w^*$, then we can develop a centralized algorithm that obtains optimal CS^* with time and space complexity $O(n \cdot ((w^*+1)d+1)^{w^*+1})$.*

Proof *If we create w^*+1 copies of the domains and solve the obtained DCOP instance using a centralized algorithm, we can obtain an optimal solution. Then, the domain size becomes $(w^*+1)d+1$. By creating copies of the domains, the structure of a constraint network does not change, and thus the tree width does not change. We can use any centralized algorithm (e.g., DP based algorithm, bucket elimination (Dechter 1999)) whose time and space complexities are $O(n \cdot {d_s}^{w^*+1})$, where d_s is the domain size.* □

Also, we can use any DCOP algorithm that is bounded by tree width, such as DPOP (Petcu and Faltings 2005), to develop a distributed algorithm with a complexity bound.

Although finding w^* is NP-hard (Arnborg 1985), we can easily obtain an upper-bound of w^*. For example, we can use arbitrary total ordering among agents (or use a heuristic method to find a good ordering (Dechter 2003)) and create a pseudo-tree. The induced width of the pseudo-tree gives an upper-bound of w^*, which can be used as a pessimistic worst-case bound.

Note that it is quite rare to have a quality bound for DCOP/CSG approximation algorithms. A notable exception is (Sandholm et al. 1999). Our algorithm is inspired by (Sandholm et al. 1999), which also searches for a restricted subset of all coalition structures. However, since the subsets searched are different, the obtained bounds are different. We can expect w^* would be small when agents are loosely coupled.

Anytime Algorithm

We can construct an anytime algorithm by incrementing k one by one. Alternatively, we can repeatedly apply the approximation algorithm with $k=1$ as follows.

1. Apply the approximation algorithm with $k=1$.
2. If the obtained CS only contains single-agent coalitions, then finish this algorithm.
3. Otherwise, remove the agents in multi-agent coalition. If no agent remains, then finish this algorithm. Otherwise, go to (1).

For each application of the approximate algorithm, one multi-agent coalition is found and removed (or the algorithm terminates). Thus, the approximation algorithm is applied at most $\lfloor n/2 \rfloor$ times. This anytime algorithm is much more efficient than an algorithm with a large k. However, it is a greedy algorithm and its worst-case bound is still $\max(1/(w^*+1), 1/\lfloor n/2 \rfloor)$.

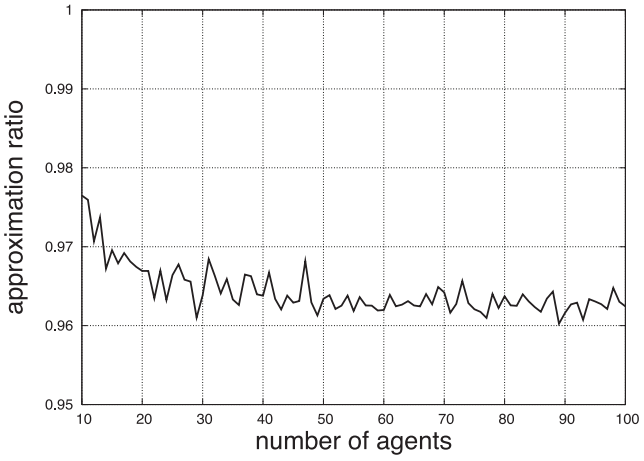


Figure 3: Approximation Ratio of the Algorithm ($n \in [10,100]$, $|D|=2$, $w^*=1$, $k=1$)

Table 1: Approximation Ratio of the Algorithm ($n=10$, $|D|=2$, $w^* \in [1,5]$, $k=1$)

induced width	1	2	3	4	5
average ratio	0.965	0.957	0.962	0.928	0.941

Experiments

We experimentally examined the solution quality obtained by our proposed algorithm. We compared the optimal solution with the solution obtained by our algorithm with $k=1$ in following two cases.

case (1): $n \in [10,100]$, $w^*=1$.

case (2): $n=10$, $w^* \in [1,5]$.

n is the number of agents and w^* is the tree width of a constraint graph. In these experiments, we set the domain size of each variable $|D|$ to 2 and chose a unary reward value and a binary reward value from the uniform distribution $[-10,10]$.

In case (1), we restrict a graph structure to a tree (i.e., $w^*=1$). Thus, the theoretical worst-case bound is $1/(w^*+1)=0.5$. In this case, we can obtain the optimal solution in polynomial time. We generated 100 problem instances for each number of agents. Figure 3 illustrates the average approximation ratio. The x-axis indicates the number of agents, while the y-axis shows the average approximation ratio. We can see that the average ratio is more than 96% of the optimal solutions. This is by far superior to the theoretical worst-case bound, i.e., 0.5.

In case (2), we examine the case where the graph structure is not restricted to a tree, i.e., $w^*>1$. Since finding w^* is NP-hard, we use an upper-bound of w^*, i.e., the induced width of the pseudo-tree obtained by the greedy algorithm described in (Dechter 2003). We set $n=10$, and generate 10 problem instances for each induced width. Table 1 shows the average approximation ratio. We can see that the average ratio is more than 92% of the optimal solutions even if we increase the induced width. This is also by far superior to

the theoretical worst-case bound. For example, when the induced width is 5, the pessimistic estimation of the worst-case bound obtained by the induced width is $1/(5+1) = 0.17$.

Conclusion

We proposed a novel formalization of CSG problems, i.e., the value of a characteristic function is given as an optimal solution of a DCOP among the agents of a coalition. We proposed an approximation algorithm with parameter k, whose social surplus is at least $\max(k/(w^*+1), k/\lfloor n/2 \rfloor)$ of the optimal CS. When $k = 1$, its computational cost is about the same as finding the value of the grand coalition. These results illustrate that our DCOP formalization is effective for CSG, since it can successfully model the locality of interactions among agents. We also proposed an anytime algorithm and experimentally showed that the average approximation ratio of the approximation algorithm is by far superior to the theoretical worst-case bound.

Our future works include developing more efficient algorithms, applying the ideas developed in this paper to real-world application domains, and extending our formalization to handle the case *with externality*, i.e., there can be positive/negative interactions among coalitions.

References

Arnborg, S. 1985. Efficient algorithms for combinatorial problems on graphs with bounded decomposability—a survey. *BIT* 25(1):2–23.

Chechetka, A., and Sycara, K. 2006. No-commitment branch and bound search for distributed constraint optimization. In *the Proceedings of the 5th International joint Conference on Autonomous Agents and Multi-agent Systems*, 1427–1429.

Conitzer, V., and Sandholm, T. 2006. Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence* 170(6):607–619.

Dang, V. D.; Dash, R. K.; Rogers, A.; and Jennings, N. R. 2006. Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *the Proceedings of the 21st National Conference on Artificial Intelligence*, 635–640.

Dechter, R. 1999. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence* 113(1-2):41–85.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Deng, X.; Ibaraki, T.; and Nagamochi, H. 1997. Algorithms and complexity in combinatorial optimization games. In *the Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms)*, 720–729.

Diestel, R. 2000. *Graph Theory*. Springer.

Gent, I. P. 2001. A symmetry breaking constraint for indistinguishable values. In *the Proceedings of the 1st International Workshop on Symmetry in Constraint Satisfaction Problems*, 469–473.

Ieong, S., and Shoham, Y. 2005. Marginal contribution nets: a compact representation scheme for coalitional games. In *the Proceedings of the 6th ACM Conference on Electronic Commerce*, 193–202.

Mailler, R., and Lesser, V. 2004. Solving distributed constraint optimization problems using cooperative mediation. In *the Proceedings of the 3rd International joint Conference on Autonomous Agents and Multi-agent Systems*, 438–445.

Modi, P. J.; Shen, W.-M.; Tambe, M.; and Yokoo, M. 2003. An asynchronous complete method for distributed constraint optimization. In *the Proceedings of the 2nd International joint Conference on Autonomous Agents*, 161–168.

Ohta, N.; Conitzer, V.; Ichimura, R.; Sakurai, Y.; Iwasaki, A.; and Yokoo, M. 2009. Coalition structure generation utilizing compact characteristic function representations. In *the Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*, 623–638.

Petcu, A., and Faltings, B. 2005. A scalable method for multiagent constraint optimization. In *the Proceedings of the 19th International Joint Conference on Artificial Intelligence*, 266–271.

Rahwan, T., and Jennings, N. R. 2008. Coalition structure generation: dynamic programming meets anytime optimisation. In *the Proceedings of the 23rd Conference on Artificial Intelligence*, 156–161.

Rahwan, T.; Ramchurn, S. D.; Dang, V. D.; Giovannucci, A.; and Jennings, N. R. 2007. Anytime optimal coalition structure generation. In *the Proceedings of the 22nd Conference on Artificial Intelligence*, 1184–1190.

Rothkopf, M. H.; Pekeč, A.; and Harstad, R. M. 1998. Computationally manageable combinatorial auctions. *Management Science* 44(8):1131–1147.

Sandholm, T., and Lesser, V. R. 1997. Coalitions among computationally bounded agents. *Artificial Intelligence* 94(1-2):99–137.

Sandholm, T.; Larson, K.; Andersson, M.; Shehory, O.; and Tohmé, F. 1999. Coalition structure generation with worst case guarantees. *Artificial Intelligence* 111(1-2):209–238.

Sandholm, T. 1993. An implementation of the contract net protocol based on marginal cost calculations. In *the Proceedings of the 11th National Conference on Artificial Intelligence*, 295–308.

Yeh, D. Y. 1986. A dynamic programming approach to the complete set partitioning problem. *BIT* 26(4):467–474.