Constraint Composite Graph-Based Lifted Message Passing for Distributed Constraint Optimization Problems

Ferdinando Fioretto* **Hong Xu†** **Sven Koenig†** **T. K. Satish Kumar†**
*University of Michigan, Ann Arbor, Michigan 48109, United States of America
†University of Southern California, Los Angeles, California 90089, United States of America
fioretto@umich.edu, {hongx,skoenig}@usc.edu, tkskwork@gmail.com

Abstract

The *Distributed Constraint Optimization Problem* (DCOP) offers a powerful approach for the description and resolution of cooperative multi-agent problems. In this model, a group of agents coordinates their actions to optimize a global objective function, taking into account their local preferences. In the majority of DCOP algorithms, agents operate on three main graphical representations of the problem: (a) the *constraint graph*, (b) the *pseudo-tree*, or (c) the *factor graph*. In this paper, we introduce the *Constraint Composite Graph* (CCG) for DCOPs, an alternative graphical representation on which agents can coordinate their assignments to solve the distributed problem suboptimally. By leveraging this representation, agents are able to reduce the size of the problem. We propose a novel variant of Max-Sum–a popular DCOP incomplete algorithm–called *CCG-Max-Sum*, which is applied to CCGs. We also demonstrate the efficiency and effectiveness of CCG-Max-Sum on DCOP benchmarks based on several network topologies.

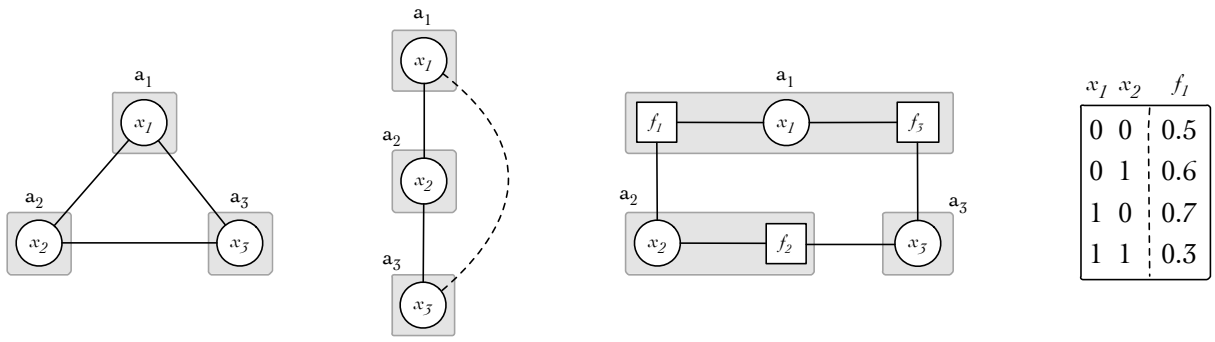
Introduction

In a cooperative *multi-agent system* multiple autonomous agents interact to pursue personal goals and to achieve shared objectives. The *Distributed Constraint Optimization Problem (DCOP)* model (Modi et al. 2005; Yeoh and Yokoo 2012) is an elegant formalism to describe cooperative multi-agent problems that are distributed in nature. In this model, a collection of agents coordinate a value assignment to the problem variables with the goal of optimizing a global objective within the confines of localized communication. DCOPs have been used to solve a variety of problems in the context of coordination and resource allocation (Léauté and Faltings 2011; Zivan et al. 2015; Miller, Ramchurn, and Rogers 2012), sensor networks (Farinelli et al. 2008), and device coordination in smart homes (Rust, Picard, and Ramparany 2016; Fioretto, Yeoh, and Pontelli 2017).

DCOP algorithms are either *complete* or *incomplete*. Complete algorithms find an optimal solution to the problem employing one of two broad modus operandi: distributed search-based techniques (Modi et al. 2005; Yeoh, Felner, and Koenig 2010; Netzer, Grubshtein, and Meisels 2012) or distributed inference-based techniques (Petcu and Faltings 2005; Vinyals, Rodríguez-Aguilar, and Cerquides 2011). In search-based techniques, agents traverse the search space by selecting value assignments and communicating them to other agents. Inference-based techniques rely instead on the notion of agent belief, describing the best cost an agent can achieve for each value assignment to its variables. These beliefs drive the value-selection process of the agents to find an optimal solution to the problem.

Since finding an optimal DCOP solution is NP-hard (Modi et al. 2005), optimally solving a DCOP requires exponential time or space in the worst case. Thus, there is growing interest in the development of incomplete algorithms, which trade off solution quality for better runtimes. Similar to complete algorithms, incomplete algorithms can be classified as local search-based (Maheswaran, Pearce, and Tambe 2004; Zhang et al. 2005) and inference-based (Petcu, Faltings, and Mailler 2007; Farinelli et al. 2008). Some incomplete algorithms have been used in several multi-agent applications. For instance, Max-Sum (Farinelli et al. 2008; Stranders et al. 2009) is an inference-based incomplete algorithm which has been successfully used to solve sensor networks problems (Farinelli et al. 2008), multi-agent task allocation for rescue teams in disaster areas (Ramchurn et al. 2010), and smart home coordination problems (Rust, Picard, and Ramparany 2016).

In both complete and incomplete DCOP algorithms, the problem resolution process is characterized by the graphical representation of the problem. The three most important problem representations are the *constraint graph*, the *pseudo-tree*, and the *factor graph*. The first represents a problem as a graph whose nodes describe the variables and whose edges describe the constraints. The second is a rearrangement of the constraint graph, where a subset of edges forms a rooted tree and where two variables in the scope of the same constraint appear in the same branch of the tree. The third represents the problem as a bipartite graph where nodes represent both variables and constraints, and edges link the constraint nodes to the variables in their scope. In many local search algorithms, such as MGM (Maheswaran, Pearce, and Tambe 2004), DSA (Zhang et al. 2005), or the region-optimal algorithm family (Pearce and Tambe 2007), agents operate directly on the constraint graph and perform distributed local searches by exchanging information with their neighbors in the constraint graph. In the main inference-based algorithms, the agents operate either on a pseudo-tree (e.g., P-DCOP (Petcu, Faltings, and



(a) Constraint Graph (b) Pseudo-Tree (c) Factor Graph (d) Constraint

Figure 1: Example DCOP: Constraint graph (a), pseudo-tree (b), factor graph (c), and a constraint (d).

Mailler 2007)) or a factor graph (e.g., Max-Sum). In the former, agents exchange messages following the structure of the pseudo-tree, typically alternating between a phase in which messages are propagated up from the leaf agents to the root agent of the pseudo-tree, and one in which information is propagated down. In the latter case, there are two types of entities, namely, variable nodes (representing variables) and function nodes (representing constraints). Both these entities participate in the message exchange process to solve the problem.

All these representations allow agents to exploit the graphical structure of the problem. However, they hide the numerical structure of the problem's constraints. Thus, in this paper, we introduce the *Constraint Composite Graph (CCG)* for DCOPs, a lifted graphical representation that provides a framework for exploiting simultaneously the graphical structure of the agent-coordination process as well as the numerical structure of the constraints involving the variables controlled by the agents. CCGs have been recently introduced in the context of Weighted Constraint Satisfaction Problems (WCSPs) (Kumar 2008a; 2008b; 2016), and shown to be highly effective in solving a wide range of problems (Xu, Kumar, and Koenig 2017; Xu, Koenig, and Kumar 2017). We contribute to the development of inference-based DCOP algorithms by investigating the CCG representation for DCOPs and developing a variant of Max-Sum which can be used directly on CCGs.

Contributions: This paper makes the following contributions: **(1)** We adapt the recently introduced CCG representation for *Weighted Constraint Satisfaction Problems (WCSPs)* to DCOPs. **(2)** We present a novel framework for solving DCOPs sub-optimally whose agent interactions are driven by the structure of the CCG representation. **(3)** By leveraging this representation, agents are able to exploit techniques that are effective, in general, in reducing the size of the original problem. **(4)** We analyze the behavior of the proposed framework on different graph topologies and show its efficiency and effectiveness on several important classes of graphs, including grid networks and scale-free networks, which are used to describe many applications in distributed settings.

To the best of our knowledge, we are the first in proposing a distributed message-passing algorithm based on the CCG representation. We refer to our algorithm as a "lifted" message passing algorithm to refer to that it works on the CCG representation of a DCOP.

Background

We now review the distributed constraint optimization framework, the graphical models commonly adopted to represent a DCOP, and the CCG model.

Distributed Constraint Optimization

A *Distributed Constraint Optimization Problem (DCOP)* is a tuple $P = \langle \mathbf{X}, \mathbf{D}, \mathbf{F}, \mathbf{A}, \alpha \rangle$, where: $\mathbf{X} = \{x_1, \ldots, x_n\}$ is a set of *variables*; $\mathbf{D} = \{D_{x_1}, \ldots, D_{x_n}\}$ is a set of finite *domains* for the variables in $\mathbf{X}$; $\mathbf{F} = \{f_1, \ldots, f_e\}$ is a set of *constraints* (also called *cost functions*), where $f : \prod_{x \in \mathbf{x}^f} D_x \rightarrow \mathbb{R}_+ \cup \{\infty\}$ and $\mathbf{x}^f \subseteq \mathbf{X}$ is the set of the variables (also called the *scope*) of f; $\mathbf{A} = \{a_1, \ldots, a_p\}$ is a set of *agents*; and $\alpha : \mathbf{X} \rightarrow \mathbf{A}$ is a function that maps each variable to one agent. Figure 1(d) shows an example constraint. It specifies the costs of all combinations of values for the variables x_1, x_2 in the scope of the constraint. For a variable $x \in \mathbf{X}$, we use $\mathbf{f}^x$ to denote the set of constraints that involve x in their scopes.

A *partial assignment* σ_X is an assignment of values to a set of variables $X \subseteq \mathbf{X}$ that is consistent with the domains of the variables; i.e., it is a partial function $\theta : \mathbf{X} \rightarrow \cup_{i=1}^{n} D_{x_i}$ such that, for each $x_j \in \mathbf{X}$, if $\theta(x_j)$ is defined (i.e., $x_j \in X$), then $\theta(x_j) \in D_{x_j}$. For a set of variables $V = \{x_{i_1}, \ldots, x_{i_h}\} \subseteq X$, $\pi_V(\sigma_X) = \langle \theta(x_{i_1}), \ldots, \theta(x_{i_h}) \rangle$ is the *projection* of σ_X to the variables in V, where $i_1 < \ldots < i_h$. When $V = \{x_i\}$ is a singleton, we write $\pi_{x_i}(\sigma_X)$ to denote the projection of σ_X to x_i. The *cost* $\mathcal{F}(\sigma_X) = \sum_{f \in \mathbf{F}: \mathbf{x}^f \subseteq X} f(\pi_{\mathbf{x}^f}(\sigma_X))$ of an assignment σ_X is the sum of the evaluation of the constraints involving all the variables in X. A *solution* is a partial assignment σ_X (written σ for shorthand) for all the variables of the problem, i.e., with $X = \mathbf{X}$, whose cost is finite (i.e., $\mathcal{F}(\sigma) \neq \infty$).

The goal is to find an optimal solution $\sigma^* = \operatorname{argmin}_\sigma \mathcal{F}(\sigma)$. In this paper, we restrict our attention to Boolean DCOPs (i.e., DCOPs where all domains are $\{0, 1\}$).

Despite our focus on Boolean DCOPs, the concepts introduced in the next sections are easily generalizable, as discussed in the Conclusions.

Given a DCOP P, its *constraint graph* is $G_P = (\mathbf{X}, E_C)$, where an undirected edge $\{x, y\} \in E_C$ exists if and only if there exists an $f \in \mathbf{F}$ such that $\{x, y\} \subseteq \mathbf{x}^f$. The constraint graph provides a standard representation of a DCOP instance. It highlights the locality of interactions among agents and therefore is commonly adopted by DCOP resolution algorithms. Figure 1(a) shows an example constraint graph of a DCOP with three agents a_1, a_2, and a_3, each controlling one variable with domain {0,1}. There are three constraints: f_1 with scope $\mathbf{x}^{f_1} = \{x_1, x_2\}$, f_2 with scope $\mathbf{x}^{f_2} = \{x_2, x_3\}$, and f_3 with scope $\mathbf{x}^{f_3} = \{x_1, x_3\}$.

The *pseudo-tree* for P is a subgraph $T_P = \langle \mathbf{X}, E_T \rangle$ of G_P such that T_P is a spanning tree of G_P, i.e., a connected subgraph of G_P that contains all nodes and is a rooted tree, with the following additional condition: for each $x, y \in \mathbf{X}$, if $\{x, y\} \subseteq \mathbf{x}^f$ for some $f \in \mathbf{F}$, then x and y appear in the same branch of T_P (i.e., x is an ancestor of y in T_P or vice versa). Figure 1(b) shows one possible pseudo-tree of our example DCOP, where the solid lines represent tree edges and the dotted line represents a *backedge* that connects an agent with one of its ancestors.

A factor graph (Kschischang, Frey, and Loeliger 2001) is a bipartite graph used to represent the factorization of a function. Given a DCOP P, the corresponding factor graph $F_P = \langle \mathbf{X}, \mathbf{F}, E_F \rangle$ is composed of variable nodes $x \in \mathbf{X}$, function nodes $f \in \mathbf{F}$, and edges E_F such that there is an undirected edge between function node f and variable node x if and only if $x \in \mathbf{x}^f$. Figure 1(c) illustrates the factor graph of our example DCOP, where each agent a_i controls its variable x_i and, in addition, a_1 controls the constraints f_1 and f_3, and a_2 controls the constraint f_2.

Max-Sum

Max-Sum (Farinelli et al. 2008) is a popular incomplete DCOP algorithm. Max-Sum agents operate on a factor graph F_P through a synchronous iterative process. Albeit the logic of each variable node and each function node is executed within an agent, to ease exposition, in what follows, we treat them as entities that are able to send and receive messages.

In each iteration, each function node f exchanges messages with the nodes of variables in its scope $\mathbf{x}^f$, and each variable node x exchanges messages with the nodes of constraints which involve x in their scopes $\mathbf{f}^x$. Thus, each node exchanges messages with its neighbors in the factor graph.

The content of the messages sent by each function (variable) node is based exclusively on the information received from neighboring variable (function) nodes. The message $q^i_{x\to f}$ sent by a variable node x to a function node f in $\mathbf{f}^x$ at iteration i contains, for each value $d \in D_x$, the aggregated costs for d received from all neighboring function nodes in iteration $i-1$, excluding f. It is defined as a function $q^i_{x\to f} : D_x \to \mathbb{R}_+ \cup \{\infty\}$, whose value is 0 for all $d \in D_x$ when $i = 0$ and

$$q^i_{x\to f}(d) = \alpha^i_{xf} + \sum_{f' \in \mathbf{f}^x \setminus \{f\}} r^{i-1}_{f'\to x}(d) \qquad (1)$$

when $i > 0$, where $r^{i-1}_{f'\to x}$ is the message received by variable node x from function node f' in iteration $i-1$ and α^i_{xf} is a *normalizing* constant used to prevent the values of the transmitted messages from growing arbitrarily. It is chosen such that

$$\sum_{d \in D_x} q^i_{x\to f}(d) = 0$$

holds. The message $r^i_{f\to x}$ sent by a function node f to a variable node x in $\mathbf{x}^f$ in iteration i contains, for each value $d \in D_x$, the minimum cost of any assignments of values to the variables in $\mathbf{x}^f$ in which x takes value d. It is defined as a function $r^i_{f\to x} : D_x \to \mathbb{R}_+ \cup \{\infty\}$, whose value is 0 when $i = 0$ and

$$r^i_{f\to x}(d) = \min_{\sigma_{\mathbf{x}^f} :\, \pi_x(\sigma_{\mathbf{x}^f}) = d} f(\sigma_{\mathbf{x}^f}) + \sum_{x' \in \mathbf{x}^f \setminus \{x\}} q^i_{x'\to f}(\pi_{x'}(\sigma_{\mathbf{x}^f})) \qquad (2)$$

when $i > 0$. Here, $\sigma_{\mathbf{x}^f}$ represents a possible value assignment to all variables involved in the scope $\mathbf{x}^f$ of the constraint f, under the constraint that variable $x \in \mathbf{x}^f$ takes value d.

The agent controlling a variable node x decides its value assignment at the end of each iteration by computing its associated belief $b^i_x(d)$ for each $d \in D_x$:

$$b^i_x(d) = \sum_{f \in \mathbf{f}^x} r^{i-1}_{f\to x}(d)$$

and choosing the assignment d^{*i} such that,

$$d^{*i} = \operatorname*{argmin}_{d \in D_x} b^i_x(d). \qquad (3)$$

This form of message passing allows an inference-based method: Max-Sum agents initialize all their messages to 0 and, in each iteration $i > 1$, retain only the most recent messages, overwriting the messages received in previous iterations.

Max-Sum is an incomplete DCOP algorithm. However, on acyclic problems, it is guaranteed to converge to an optimal solution (Farinelli et al. 2008).

The Constraint Composite Graph

We now describe the *constraint composite graph (CCG)*, a graphical structure that can be used to represent DCOPs. Its goal is to exploit simultaneously the graphical structure of the agent interactions as well as the numerical structure of the cost functions. It is a node-weighted tripartite graph $G_{\text{CCG}} = \langle V = \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}, E, w \rangle$, where $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ are the three partitions of the nodes V: $\mathbf{X}$ contains nodes that correspond to decision variables, whereas $\mathbf{Y}$ and $\mathbf{Z}$ contain nodes that correspond to auxiliary variables. We use $G_{\text{CCG}_i} = \langle V_i = X_i \cup Y_i \cup Z_i, E_i, w_i \rangle$ to denote the portion of the CCG decomposed from constraint f_i. The concept of a CCG was first proposed by Kumar (2008a) as a combinatorial structure associated with a *Weighted Constraint Satisfaction Problem (WCSP)*. WCSPs are similar to DCOPs, except that all computations are centralized. In this proposal,

Algorithm 1: CCG-MAX-SUM

```
   // CCG Construction Phase
1  foreach f_i ∈ F_i do
2      p_i ← construct-polynomial(f_i);
3      G_CCG_i = ⟨V_i = X_i ∪ Y_i ∪ Z_i, E_i, w_i⟩ ←
         decompose-polynomial(p_i) ;
4  foreach f ∈ F_CCG_i involving variable v_j with α(v_j) ≠ a_i do
5      a_i sends f to a_α(v_j);
6  When agent a_i receives f involving v_i ∈ X_i from
     neighboring agent a_j:   f_v_i(1) ← f_v_i(1) + f(1) ;
   // Message Passing Phase
7  μ_v_i→v_j ← 0   (∀v_i ∈ V_i, ∀v_j ∈ N(v_i));
8  while termination condition is not met do
9      Wait for all messages μ_v_j→v_i from
         v_j ∈ N(v_i) (∀v_i ∈ V_i);
10     foreach v_i ∈ V_i do
11         Update μ_v_i→v_j according to Equation (6);
12 for v_i ∈ X_i do
13     if w_v_i < Σ_{v_j ∈ N(v_i)} μ_v_j→v_i then v_i ← 1 else v_i ← 0;
```

it was shown that the task of solving a WCSP can be reformulated as the task of finding a *Minimum Weighted Vertex Cover (MWVC)* on its associated CCG (Kumar 2008a; 2008b; 2016).

A desirable property of the CCG is that it can be constructed in polynomial time and is always tripartite (Kumar 2008a; 2008b; 2016). CCGs also enable the use of *kernelization methods* for solving WCSPs (Xu, Kumar, and Koenig 2017), which are polynomial-time procedures that can simplify a problem to a smaller one, called the kernel. The *Nemhauser-Trotter reduction (NT reduction)* (Nemhauser and Trotter 1975; Chlebík and Chlebíková 2008) is one such kernelization method. It makes use of a maxflow procedure to find the kernel and can be extended in a distributed way (Homayounnejad and Bagheri 2015).

In the next section, we introduce an extension of the Max-Sum algorithm, called CCG-Max-Sum, which can be used directly on CCGs.

CCG-Max-Sum

CCG-Max-Sum is an incomplete, iterative DCOP algorithm which works in two phases, namely, the *CCG construction phase* and the *message passing phase*, which are executed sequentially and summarized in Algorithm 1. In the CCG construction phase, the agents coordinate in the construction of a CCG and take ownership of the auxiliary variables and constraints introduced by this lifted graphical representation. Afterwards, in the message passing phase, the agents execute the iterative synchronous process which extends the Max-Sum algorithm.

In what follows, we use $G_i = \langle \mathbf{X}_i, \mathbf{F}_i \rangle$ to denote the subgraph of the constraint graph controlled by agent a_i, where the sets $\mathbf{X}_i \subseteq \mathbf{X}$ form a partition of the set of variables $\mathbf{X}$, and the sets $\mathbf{F}_i \subseteq \mathbf{F}$ form a partition for the constraint set $\mathbf{F}$.

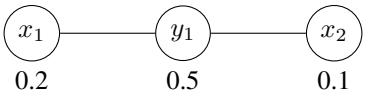
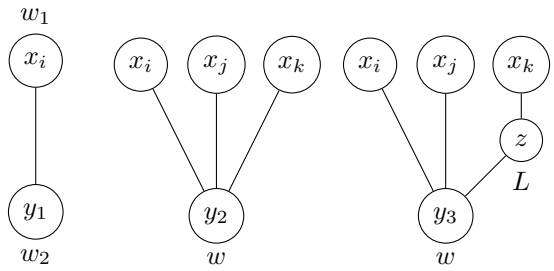


Figure 2: The projection of an MWVC on the IS $\{x_1, x_2\}$ of this node-weighted undirected graph leads to Figure 1(d). The weights on x_1, x_2, and y_1 are 0.2, 0.1, and 0.5, respectively. The entry 0.6 in cell $(x_1 = 0, x_2 = 1)$ in Figure 1(d), for example, indicates that, when x_1 is necessarily excluded from the MWVC but x_2 is necessarily included in it, then the weight of the MWVC—$\{x_2, y_1\}$—is 0.6.



(a) $w \cdot x_i$ (b) $-w \cdot (x_i \cdot x_j \cdot x_k)$ (c) $w \cdot (x_i \cdot x_j \cdot x_k)$

Figure 3: The lifted graphical representation of terms in a polynomial for linear (a), negative nonlinear (b), and positive nonlinear (c) terms. We assume that $w > 0$ in (b) and (c) (but no such assumption in (a)). A node has a zero weight if no weight is shown. In (a), w_1 and w_2 satisfy $w_1 - w_2 = w$.

CCG Construction Phase

The CCG construction proceeds in 3 stages:

1. Expressing Constraints as Polynomials In this stage, each agent a_i transforms the constraints $f_i \in \mathbf{F}_i$ it controls into polynomials p_i (line 2 of Algorithm 1) using standard Gaussian Elimination. Consider the example constraint f_1 in Figure 1(d), which involves the variables x_1 and x_2. It can be written as a polynomial $p_1(x_1, x_2)$ in x_1 and x_2 of degree 1 each:

$$p_1(x_1, x_2) = c_{00} + c_{01}x_1 + c_{10}x_2 + c_{11}x_1x_2.$$

The coefficients c_{00}, c_{01}, c_{10}, and c_{11} of the polynomial can be computed by solving a system of linear equations, where each equation corresponds to an entry in the constraint table, using standard Gaussian Elimination. In our example:

$$p_1(0,0) = 0.5 \qquad p_1(0,1) = 0.6$$
$$p_1(1,0) = 0.7 \qquad p_1(1,1) = 0.3.$$

2. Decomposing the Terms of the Polynomials In this stage, for each $f_i \in \mathbf{F}_i$, the agent that controls it constructs a subgraph G_{CCG_i} of the CCG (line 3 of Algorithm 1). At the end of this stage, each agent introduces new sets of auxiliary variables Y_i and Z_i and replaces its constraints with a new set $\mathbf{F}_{\text{CCG}_i}$ of constraints that involve the decision variables

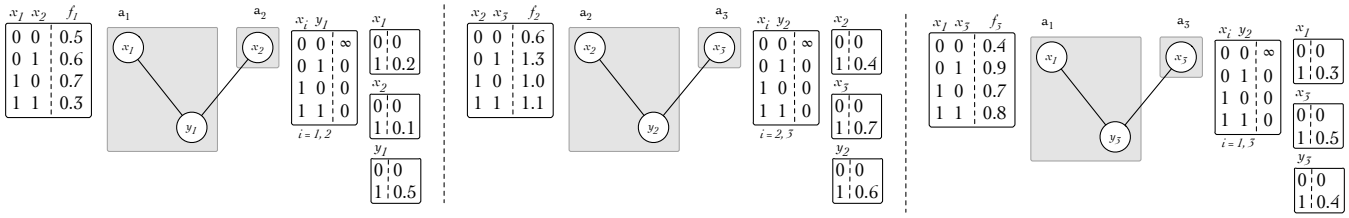


Figure 4: CCG gadget graph construction in the "Decomposing the Terms of Polynomials" stage for the example DCOP of Figure 1. The original constraint is shown on the left of each panel, the associated CCG gadget graph is shown in the middle, and, the new constraints are shown on the right of each panel.

and its newly introduced auxiliary variables. Before describing this procedure, we review the concept of the MWVC, a cornerstone concept for the notion of the CCG.

A *minimum vertex cover* of $G = \langle V, E \rangle$ is the smallest set of nodes $S \subseteq V$ such that every edge in E has at least one of its endpoint nodes in S. When G is node-weighted, (i.e., each node $v_i \in V$ has a non-negative weight w_i associated with it), its MWVC is defined as a vertex cover of minimum total weight of its nodes.

For a given graph G, one can project MWVCs on a given independent set (IS) $U \subseteq V$. (An IS is a set of nodes in which no two nodes are connected by an edge.) The input to such a projection is the graph G as well as an IS $U = \{u_1, u_2, \ldots, u_k\}$ on G. The output is a table of 2^k numbers. Each entry in this table corresponds to a k-bit vector. We say that a k-bit vector t imposes the following restrictions: **(a)** If the i^{th} bit t_i is 0, then node u_i has to be excluded from the MWVC; and **(b)** if the i^{th} bit t_i is 1, then the node u_i has to be included in the MWVC. The projection of an MWVC on the IS U is then defined to be a table with entries corresponding to each of the 2^k possible k-bit vectors $t^{(1)}, t^{(2)}, \ldots, t^{(2^k)}$. The value of the entry that corresponds to $t^{(j)}$ is the weight of the MWVC conditioned on the restrictions imposed by $t^{(j)}$.

Figure 2 illustrates this projection for the subgraph of our example DCOP problem of Figure 1(a) that involves variables x_1 and x_2 and constraint f_1, whose costs are shown in Figure 1(d).

The table produced by projecting an MWVC on the IS U can be viewed as a constraint over $|U|$ Boolean variables. Conversely, given a (Boolean) constraint, we design a lifted representation for it so as to be able to view it as the projection of an MWVC on an IS for some intelligently constructed node-weighted undirected graph (Kumar 2008a; 2008b). The lifted graphical representation of a constraint depends on the nature of the terms in the polynomial that describes the constraint. We distinguish three classes of terms: *linear terms*, *negative nonlinear terms*, and *positive nonlinear terms*. We can construct a lifted graphical representation, i.e., a *gadget graph*, for each term in the polynomial of each constraint as follows.

- **A linear term** can be represented with the two-node graph shown in Figure 3(a) by connecting the variable node with an auxiliary node.
- **A negative nonlinear term** can be represented with the "flower" structure in Figure 3(b). Consider the term $-w \cdot (x_i \cdot x_j \cdot x_k)$ where $w > 0$. The projection of an MWVC on the "flower" structure on the variable nodes represents $w - w \cdot (x_i \cdot x_j \cdot x_k)$. The constant term w does not affect the optimality of the solution.
- **A positive nonlinear term** can be represented using the "flower+thorn" structure shown in Figure 3(c). Consider the term $w \cdot (x_i \cdot x_j \cdot x_k)$ where $w > 0$. The projection of an MWVC on the "flower+thorn" structure on the variable nodes represents $L \cdot (1 - x_k) + w - w \cdot (x_i \cdot x_j \cdot (1 - x_k))$, where $L > w + 1$ is a large real number. By constructing gadget graphs that cancel out the lower order terms as shown before, we arrive at a lifted graphical representation of the positive nonlinear term.

Procedure *decompose-polynomial* on line 3 of Algorithm 1 takes the input polynomial p_i associated with a constraint f_i, constructed in stage 1, and returns its lifted representation G_{CCG_i}, where $X_i = \mathbf{x}^{f_i}$, and Y_i, Z_i are the set of auxiliary variables introduced by the procedure, E_i is the set of edges between the G_{CCG_i} graph nodes, and w_i is the set of weights associated with the variables in X_i, Y_i, and Z_i. For a variable $v_i \in X_i \cup Y_i \cup Z_i$, a unary constraint f_{v_i} in $\mathbf{F}_{\text{CCG}_i}$ is defined as

$$f_{v_i}(v_i) = \begin{cases} w_i, & \text{if } v_i = 1, \\ 0, & \text{if } v_i = 0. \end{cases} \tag{4}$$

For each edge $\{v_i, v_j\}$ in E_i, a constraint $f_{\{v_i,v_j\}}$ in $\mathbf{F}_{\text{CCG}_i}$ is defined as

$$f_{\{v_i,v_j\}}(v_i, v_j) = \begin{cases} \infty, & \text{if } v_i = v_j = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

For a CCG gadget graph G_{CCG_i}, X_i contains nodes that correspond to decision variables, Z_i contains the nodes with weight L (if any), and Y_i contains the other nodes. At the end of this stage each agent $a_i \in \mathbf{A}$ controls the set of decision variables in $\mathbf{X}_i$ and the set of auxiliary variables $\cup_{f_j \in \mathbf{F}_i} Y_j \cup Z_j$, for all constraints $f_j \in \mathbf{F}_i$ controlled by agent a_i.

3. Merging Gadget Graphs into a CCG Finally, the CCG-Max-Sum agents construct the CCG by merging their gadget graphs G_{CCG_i}. This stage is done incrementally. Every time an agent builds a new gadget graph, it (1) updates its internal graphical representation to include the auxiliary

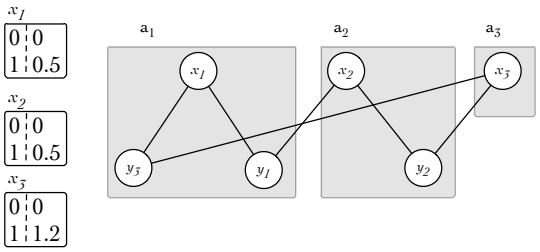


Figure 5: The CCG of the example DCOP of Figure 1, obtained by merging the CCG gadget graphs of Figure 4.

variables introduced by the construction, and (2) increases the weight associated with the agent's variables. Each agent a_i sends to its neighbor a_j all unary constraints in $\mathbf{F}_{\text{CCG}_i}$ involving variable v_j controlled by agent a_j (i.e., $\alpha(v_j) = a_j$) (lines 4–5). When an agent receives a new unary constraint f which involves one of its decision variables v_i, it increases the weight associated with the constraint ($f_{v_i}(v_i)$) for the value $f_{v_i}(1)$ (line 6).

The communication structure of the underlying DCOP does not vary after the CCG construction. If an agent a_i is a neighbor of an agent a_j in the constraint graph of the original DCOP, then a_i is also a neighbor of a_j in the lifted DCOP representation.

Figure 4 shows the construction of the CCG associated with our example DCOP of Figure 1. There are three unary and three binary constraints. Their lifted graphical representations are shown next to them. Every node in the CCG is given a weight equal to the sum of the individual weights of the nodes in the merged CCG gadget graphs.

Computing the MWVC for the CCG yields a solution for the DCOP: If variable $x_i \in \mathbf{X}$ is in the MWVC, then it is assigned the value 1 in the DCOP, otherwise it is assigned the value 0.

Message Passing Phase

Once the CCG has been constructed, the agents start the message passing phase to find a vertex cover with a small total weight. The message passing scheme is similar to that of Max-Sum: During each iteration, each agent waits to receive all messages from its neighbors, updates the current values (beliefs) for the variables it controls, computes the messages to send to its neighbors based on its new beliefs, and sends these to all its neighbors. Here, we adapt the algorithm presented in (Xu, Kumar, and Koenig 2017) (see Algorithm 1). Differently from Max-Sum, where each function node exchanges messages with its neighboring variable nodes, and each variable node exchanges messages with its neighboring function nodes, in CCG-Max-Sum, the messages are exchanged between (decision and auxiliary) variables nodes in the CCG. The message $\mu_{u\to v}$ sent by a variable u to a variable v in iteration i is:

$$\mu^i_{u\to v} = \max\left\{ w_u - \sum_{t\in N(u)\setminus\{v\}} \mu^{i-1}_{t\to u}, 0 \right\}, \qquad (6)$$

where w_u is the weight associated with variable u, and $N(u)$ is the set of neighboring variables of variable u in the CCG. Equation (6) is derived from Equations (1) and (2) using an approach similar to that in (Xu, Kumar, and Koenig 2017). These steps are shown on lines 7–11 of Algorithm 1. When the algorithm terminates, for a node v, if $w_v < \sum_{u\in N(v)} \mu_{u\to v}$, then v is selected into the MWVC; otherwise it is not. A variable is assigned value 1 if its corresponding decision variable node in the CCG is selected into the MWVC; otherwise it is assigned value 0 (lines 12–13).

Experimental Evaluation

In this section, we compare the solution costs of CCG-Max-Sum, Max-Sum, which is executed on the factor graph, and DSA (Zhang et al. 2005), a local search DCOP algorithm. We also analyze the effect of using the NT reduction (Nemhauser and Trotter 1975) in conjunction with CCG-Max-Sum (denoted CCG-Max-Sum-k). The NT reduction is executed as a preprocessing centralized step. We evaluate these algorithms on random minimization Boolean DCOPs over three classical networks topologies (Kiekintveld et al. 2010): *grid networks*, *scale-free networks*, and *random networks*, to cover both structured and unstructured problems. The costs of each joint assignment to the variables involved in a constraint are generated by sampling from the discrete uniform distribution $U(1, 100)$. We generate 30 different problem instances, run the algorithms for 5000 iterations, and report the average of those runs.

For grid networks, we generate two-dimensional 10×10 grids and connect each node with its nearest neighbors. For scale-free networks, we create an n-node network based on the Barabasi-Albert model (Barabási and Albert 1999). Starting from a connected 2-node network, we repeatedly add a new node, randomly connecting it to two existing nodes. In turn, these two nodes are selected with probabilities that are proportional to the numbers of their connected edges. Finally, for random networks, we create an n-node network, whose density p_1 produces $\lfloor n(n-1)p_1 \rfloor$ edges. We report experiments on low density problems ($p_1 = 0.4$) and high density problems ($p_1 = 0.8$), and fix the maximum constraint arity to 4. Constraints of arity 4 and 3, respectively, are generated by merging first all cliques of size 4 and then those of size 3. The other edges are used to generate binary constraints. In each configuration, we verify that the resulting constraint graph is connected. For all problems, we set the number of agents to 100. In order to emphasize the solution costs returned by the algorithms, we implement them within an anytime framework, as proposed in (Zivan, Okamoto, and Peled 2014). Such a framework is used by the agents to memorize the best solution found up to the current iteration.

We first analyze the anytime behavior of the algorithms.

Figures 6(a)–(d) show the solution costs reported by all DCOP algorithms in each iteration, on two-dimensional grid networks (a), scale-free networks (b), and random networks with low density ($p_1 = 0.4$) (c) and high density ($p_1 = 0.8$) (d). The figures illustrate the anytime behavior of the algorithms. The shaded region around each line describes the

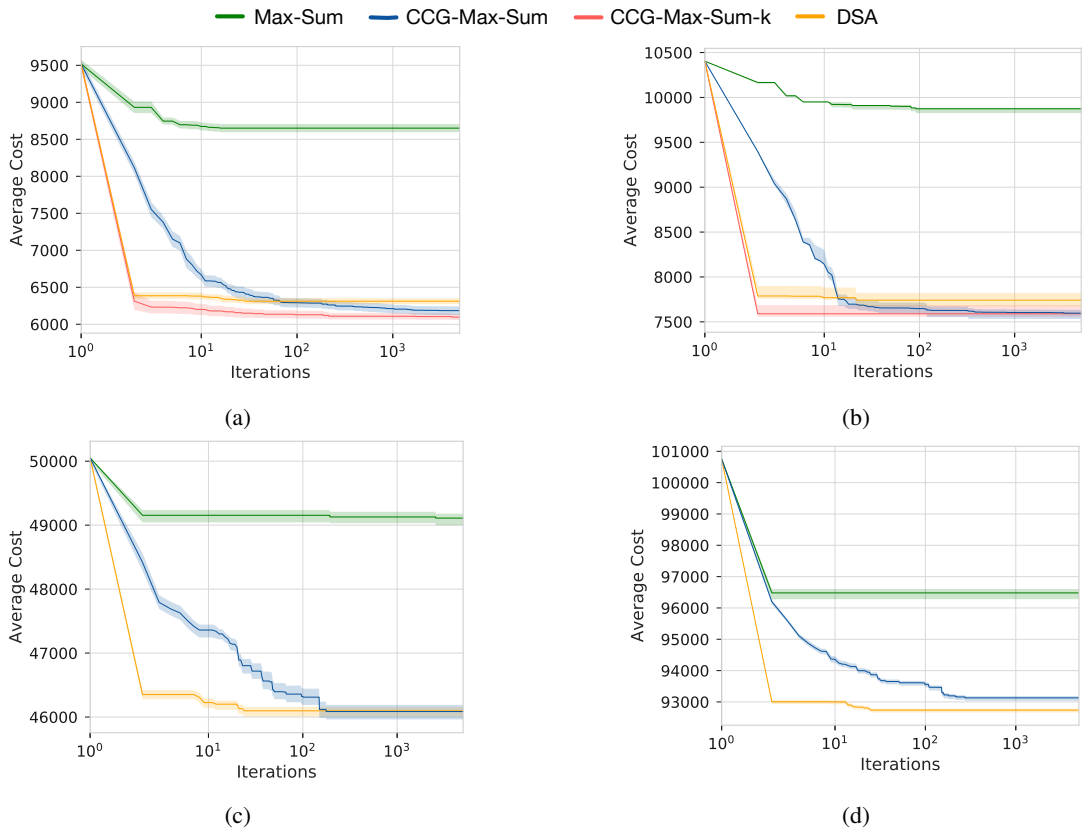


Figure 6: Solution costs for DCOPs with 100 agents on two-dimensional grid networks (a), scale-free networks (b), low density random networks ($p_1 = 0.4$) (c), and high-density random networks ($p_1 = 0.8$) (d). The blue and red curves overlap in (c) and (d).

confidence error interval of the solution costs reported by each algorithm. The plots use a log-10 scale for the x-axis. We observe that Max-Sum reports solutions with the highest costs among the costs of the solutions reported by all other algorithms. DSA agents quickly find local minima, outperforming Max-Sum agents. For structured networks (Figures 6(a) and (b)), the costs of the solutions reported by CCG-Max-Sum are smaller than those reported by both Max-Sum and DSA, after an average of 95 iterations, for grid networks, and 12 iterations, for scale-free networks. Additionally, CCG-Max-Sum-k, which exploits the kernelization preprocessing, reports solutions with the smallest costs from as early as the first iteration.

On random network benchmarks (Figures 6(c) and (d)), the effect of the kernelization is negligible and the costs of the solutions reported by CCG-Max-Sum-k are identical to those of CCG-Max-Sum (thus, the former is omitted). On low density problems, CCG-Max-Sum and DSA report similar solution costs, albeit DSA converges faster than CCG-Max-Sum. On high density problems, CCG-Max-Sum reports solutions with slightly higher costs than those reported by DSA.

Finally, we consider the CCG construction phase as preprocessing step. Its construction time affects only marginally the first iteration of the algorithms. In our experiments, the average CCG construction time is $0.24t$, with t being the time of one iteration.

Thus, our experiments suggest that CCG-Max-Sum can bring decisive advantages on grid and scale-free network instances, which are important for a large variety of DCOP applications ((Farinelli et al. 2008; Fioretto, Yeoh, and Pontelli 2017; Rust, Picard, and Ramparany 2016)).

Conclusions

In this paper we adapted the *Constraint Composite Graph (CCG)* graphical representation encoding for Distributed Constraint Optimization Problems (DCOPs). The CCG provides a framework for exploiting simultaneously the graphical structure of the agent interaction process as well as the numerical structure of the constraints of a DCOP instance. We use this representation to introduce CCG-Max-Sum, a novel incomplete DCOP algorithm which extends Max-Sum by executing the distributed message passing phase on the CCG.

Compared to a version of Max-Sum which is executed on factor graphs and other incomplete DCOP algorithms, CCG-Max-Sum finds solutions of better quality within fewer iterations on several DCOP benchmarks.

While this paper introduced an inference-based algorithm

that operates on the CCG of a DCOP, we believe that the CCG can also be exploited with other classes of DCOP algorithms. Additionally, the ideas presented in this paper are extendable to DCOPs with non-Boolean variables, as shown in (Kumar 2008b). We expect CCG-Max-Sum to be efficient for large domain sizes since the size of the CCG increases only polynomially with respect to domain sizes.

Future directions include applying CCG-Max-Sum to problems with hard constraints: Many types of hard constraints may be simplified during the construction of the CCG, and therefore resulting in smaller CCGs. Another direction is to investigate the application of the *Crown Reduction* (Chlebík and Chlebíková 2008) to CCG-Max-Sum, a kernelization method that is not widely known in the AI community.

Acknowledgments

The research at the University of Southern California was supported by the National Science Foundation (NSF) under grant numbers 1724392, 1409987, and 1319966. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies or the U.S. government.

References

Barabási, A.-L., and Albert, R. 1999. Emergence of scaling in random networks. *Science* 286(5439):509–512.

Chlebík, M., and Chlebíková, J. 2008. Crown reductions for the minimum weighted vertex cover problem. *Discrete Applied Mathematics* 156(3):292–312.

Farinelli, A.; Rogers, A.; Petcu, A.; and Jennings, N. 2008. Decentralised coordination of low-power embedded devices using the Max-Sum algorithm. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 639–646.

Fioretto, F.; Yeoh, W.; and Pontelli, E. 2017. A multiagent system approach to scheduling devices in smart homes. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 981–989.

Homayounnejad, S., and Bagheri, A. 2015. An efficient distributed max-flow algorithm for wireless sensor networks. *Journal of Network and Computer Applications* 54:20–32.

Kiekintveld, C.; Yin, Z.; Kumar, A.; and Tambe, M. 2010. Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 133–140.

Kschischang, F. R.; Frey, B. J.; and Loeliger, H.-A. 2001. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* 47(2):498–519.

Kumar, T. K. S. 2008a. A framework for hybrid tractability results in Boolean weighted constraint satisfaction problems. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 282–297.

Kumar, T. K. S. 2008b. Lifting techniques for weighted constraint satisfaction problems. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics (ISAIM)*.

Kumar, T. K. S. 2016. Kernelization, generation of bounds, and the scope of incremental computation for weighted constraint satisfaction problems. In *Proceedings of the International Symposium on Artificial Intelligence and Mathematics (ISAIM)*.

Léauté, T., and Faltings, B. 2011. Distributed constraint optimization under stochastic uncertainty. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 68–73.

Maheswaran, R.; Pearce, J.; and Tambe, M. 2004. Distributed algorithms for DCOP: A graphical game-based approach. In *Proceedings of the Conference on Parallel and Distributed Computing Systems (PDCS)*, 432–439.

Miller, S.; Ramchurn, S. D.; and Rogers, A. 2012. Optimal decentralised dispatch of embedded generation in the smart grid. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 281–288.

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

Nemhauser, G. L., and Trotter, L. E. 1975. Vertex packings: Structural properties and algorithms. *Mathematical Programming* 8(1):232–248.

Netzer, A.; Grubshtein, A.; and Meisels, A. 2012. Concurrent forward bounding for distributed constraint optimization problems. *Artificial Intelligence* 193:186–216.

Pearce, J., and Tambe, M. 2007. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1446–1451.

Petcu, A., and Faltings, B. 2005. A scalable method for multiagent constraint optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1413–1420.

Petcu, A.; Faltings, B.; and Mailler, R. 2007. PC-DPOP: A new partial centralization algorithm for distributed optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 167–172.

Ramchurn, S. D.; Farinelli, A.; Macarthur, K. S.; and Jennings, N. R. 2010. Decentralized coordination in robocup rescue. *The Computer Journal* 53(9):1447–1461.

Rust, P.; Picard, G.; and Ramparany, F. 2016. Using message-passing DCOP algorithms to solve energy-efficient smart environment configuration problems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 468–474.

Stranders, R.; Farinelli, A.; Rogers, A.; and Jennings, N. R. 2009. Decentralised coordination of continuously valued control parameters using the Max-Sum algorithm. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 601–608.

Vinyals, M.; Rodríguez-Aguilar, J.; and Cerquides, J. 2011. Constructing a unifying theory of dynamic programming DCOP algorithms via the generalized distributive law. *Autonomous Agents and Multi-Agent Systems* 22(3):439–464.

Xu, H.; Koenig, S.; and Kumar, T. K. S. 2017. A constraint composite graph-based ILP encoding of the Boolean weighted

CSP. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP)*, 630–638.

Xu, H.; Kumar, T. K. S.; and Koenig, S. 2017. The Nemhauser-Trotter reduction and lifted message passing for the weighted CSP. In *Proceedings of the International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming (CPAIOR)*, 387–402.

Yeoh, W., and Yokoo, M. 2012. Distributed problem solving. *AI Magazine* 33(3):53–65.

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

Zhang, W.; Wang, G.; Xing, Z.; and Wittenberg, L. 2005. Distributed stochastic search and distributed breakout: Properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence* 161(1–2):55–87.

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

Zivan, R.; Okamoto, S.; and Peled, H. 2014. Explorative anytime local search for distributed constraint optimization. *Artificial Intelligence* 212:1–26.