Distributed Constrained Optimization over Constrained Communication Topologies

Julius Pfrommer

Vision and Fusion Laboratory
Institute for Anthropomatics
Karlsruhe Institute of Technology (KIT), Germany
julius.pfrommer@kit.edu

Technical Report IES-2014-07

Abstract: The Max-Sum algorithm, an instance of the Generalized Distributive Law family, is known to solve Distributed Constraint Optimization Problems (DCOP) where the summed utility functions of interacting agents are maximized. However, Max-Sum relies on available communication channels between all agents that partake in a utility function. We present a generalization of Max-Sum that solves DCOP exactly in situations where the communication network layout does not match the agents' utility inter-dependencies.

1 Introduction

In Distributed Constraint Optimization (DCOP), a set of agents (each represented by a variable that represents his action, choosen from a finite domain) coordinates their actions in order to maximize the summed utility the agents experience. In this work we built upon the well-known Max-Sum algorithm, a member of the Generalized Distributive Law (GDL) family of message passing schemes [AM00] [KFL01]. Max-Sum is widely used for DCOP [PF05] [KV06], however, message-passing with Max-Sum is only guaranteed to converge to a maximum assignment if the utility inter-dependencies of the agents form a tree-graph. Here, we present a generalization of Max-Sum that can infer the exact max-marginals in deterministic time on DCOP instances where

1. not all (inter-dependent) agents can exchange messages, but the communication graph is still connected
2. agent inter-dependencies form loops.

Existing DCOP algorithms can be roughly classified into search based [MSTY05], local-search based [MTB+04] and inference based [PF05] methods. Our approach falls into the inference based class of algorithms. It is different from DCOP in settings where communication is limited/expensive [FRPJ08] [PGCMRA11] as we constrain the the availability of communication channels and not only the throughput. It also differs from exact and approximate inference on junction trees for DCOP [BM10] [VRAC11]. Our method requires no graph triangulation and grouping of agents into a tree-like hypergraph. Instead, we cut communication links between interacting agents until the remaining graph forms a tree. The messages exchanged between agents are adapted, so that local utility information is propagated within the relevant portions of the graph only. This leads to a very natural handling of situations where the communication topology ist constrained.

The paper is organized as follows. We give an overview on the original Max-Sum algorithm in Section 2. In Section 3, we introduce Max-Sum with Remote Neighbours. The performance of our approach is evaluated on an example scenario in Section 4. The paper concludes in Section 5 with a discussion of the results and some pointers for future research.

2 The Max-Sum Algorithm

Let V be a set of variables $i \in V$, each defined on a finite domain $\mathcal{X}_i$. The goal is to maximize some function f where $\mathcal{X} = \prod_{i \in V} \mathcal{X}_i,\ f : \mathcal{X} \to \mathbb{R}_{-\infty}$. The codomain $\mathbb{R}_{-\infty} = \mathbb{R} \cup \{-\infty\}$ makes complex constraints easier to handle algorithmically. That is, if x is constrained to lie in some set $\tilde{\mathcal{X}} \subset \mathcal{X}$, we define $\forall x \notin \tilde{\mathcal{X}}, f(x) = -\infty$ and keep the cartesian product of the variable domains $\mathcal{X}_i$ as the domain of f. This leads to the same maximum solution, provided that $\sup_{x \in \tilde{\mathcal{X}}} f(x) > -\infty$. The trivial approach to enumerate all possible solutions

$$x^* = \arg\max_{x \in \mathcal{X}} f(x)$$

obviously scales exponentially in the number of variables and must fail even for modest problem sizes. Instead, we exploit the structure of the specific f at hand. Assume that f is a sum of functions ψ_α, called *factors*, each of which depends only on a subset of the variables $\alpha \in 2^V$. The set of all factors is $A \subset 2^V$:

$$f(x) = \sum_{\alpha \in A} \psi_\alpha(x_\alpha), \quad x_\alpha \in \prod_{i \in \alpha} \mathcal{X}_i,\ \forall i \in \alpha, (x_\alpha)_i = x_i$$

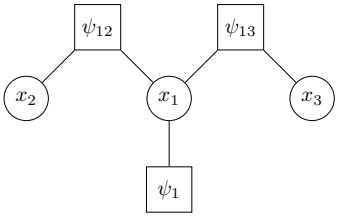


Figure 2.1: Example factor graph corresponding to $f(x_1, x_2, x_3) = \psi_1(x_1) + \psi_{12}(x_1, x_2) + \psi_{13}(x_1, x_3)$. By convention, variable nodes are represented as circles and factor nodes as squares.

This decomposition of f can be represented as a bipartite undirected *factor-graph* $G = (V, A, E)$. See [KFL01] and Fig. 2.1 for an example of a factor graph. Edges (i, α) indicate that the factor ψ_α depends on the variable i. Even though G is undirected, let E contain a pair of directed edges $e = (\underline{e}, \overline{e})$ for every factor-variable relationship to denote directed communication when it occurs. W.l.o.g., assume G to be connected. If that is not the case, G is made up of independent subgraphs $G_k = (V_k, A_k, E_k)$. Since there are no edges (i, α) between subgraphs with $i \in V_k,\ \alpha \in A_{k'}$ for $k \neq k'$, optimizing f reduces to solving

$$x_k^* = \arg\max_{x_k \in \mathcal{X}_k} \sum_{\alpha \in A_k} \psi_\alpha(x_\alpha), \quad \mathcal{X}_k = \prod_{i \in V_k} \mathcal{X}_i$$

independently for each subgraph k, to which the techniques for connected graphs apply.

The Max-Sum algorithm is an instance of the Generalized Distributive Law (GDL) family of algorithms used for solving inference problems that can be stated in terms of a factor graph [KFL01]. GDL is defined on commutative semirings. Max-Sum, according to the problem statement in the beginning of this section, assumes the specific ring $(+, 0, \max, -\infty)$, taking $+$ as the operator for combining *factorisations* and $\max$ as the operator for the *marginalisation* of variables with their respective identity element. Other commutative semirings are widely used as well, e.g. for probabilistic reasoning, but are not discussed here.

Max-Sum relies on communication channels between the variable and factor nodes over which they exchange message-functions $m : \mathcal{X}_i \to \mathbb{R}_{-\infty}$ where the variable i is either the sending or the receiving variable node[1]. Variable nodes i send messages to the factor nodes in their neighborhoud $N(i) = \{\alpha : (i, \alpha) \in E\}$ and

[1]Here, they represent a mapping from a discrete domain to a scalar value which can be trivially encoded as a table for communication.

factor nodes send messages to the $i \in \alpha$. Further, assume discrete time periods t in which every node exchanges messages with all of its neighbours. Often times the messages are initialized to zero in t_0. Other initializations may have better convergence properties, but are not discussed here.

$$
\begin{aligned}
m^{t_0}_{i\to\alpha}(x_i) &= m^{t_0}_{\alpha\to i}(x_i) = 0 \\
m^{t}_{i\to\alpha}(x_i) &= \kappa + \sum_{\beta\in N(i)\setminus\{\alpha\}} m^{t-1}_{\beta\to i}(x_i) \\
m^{t}_{\alpha\to i}(x_i) &= \kappa + \max_{x_{\alpha\setminus\{i\}}} \Big[\psi_\alpha(x_\alpha) + \sum_{j\in\alpha\setminus\{i\}} m^{t-1}_{j\to\alpha}(x_j)\Big]
\end{aligned}
\tag{2.1a}
$$

Maximizing over $x_{\alpha\setminus\{i\}}$ in (2.1a) should be read as maximizing over $x_\alpha \in \prod_{l\in\alpha} \mathcal{X}_l$, $(x_\alpha)_i = x_i$ where the component of x_α related to variable i is fixed to x_i. When summing over $j \in \alpha \setminus \{i\}$, we denote the component of x_α related to variable j as x_j. The normalisation constant κ is selected for every message so that the message-function is zero for the first element in the domain of the message. The normalisation is required for convergence in loopy graphs, even though it does not guarantee convergence. Otherwise, the values of the exchanged messages can grow or diminish undefinitely. Note that the normalisation does not change the assignments selected during maximization since all choices are over- or underestimated by the same amount. Normalisation does however prevent the computation of the true marginals at the nodes with local information only.

If G is a tree-like factor graph without loops, the exchanged messages converge after completing a forwards/backwards schedule. This schedule says that nodes can only send messages to a neighbour if they have received messages from all other neighbours. Consequently, the schedule starts at the leaf nodes, propagates throughout the tree and ends when all leaf nodes have received a message themselves. Then, the sum of the messages that a variable node i has received is the max-marginal of i on the original function f (up to normalization hat will not change the max assignment computed via the max-marginal). For a proof of this, see Proposition 2 for the proposed Max-Sum with Remote Neighbours, of which, by Proposition 3, the original Max-Sum algorithm is a special case. In loopy graphs, a forward/backwards schedule obviously cannot work. Instead, nodes can send messages asynchronously at any time. For example, every node sends in every period t a message to every neighbour. Randomized schedules are also commonly used in distributed settings. There are no guarantuees for convergence on loopy graph. But if Max-Sum converges, then its solution is a local minima of the Bethe free energy [YFW05] and therefore often useful in practice.

3 Max-Sum with Remote Neighbours

Let $G = (V, A, E)$ be a loopy factor graph. We remove edges until $G' = (V, A, E')$ with $E' \subset E$ forms a tree-graph but is still connected. Now, some factors in A depend on a variable to which they have no edge in the graph. We call these the *remote neighbours* of the factor. The set of both direct and remote neighbours of a factor node is $\tilde{N}(\alpha)$ and $\tilde{N}(i)$ for a variable node. In settings where variables represent agents and their utilities, it is natural to have factors that depend on the choices of several agents, but with a utility that is "local" to a single agent. In that case, we split the relevant factor $\alpha \in A$ into α^i, such that $\psi_\alpha(x_\alpha) = \sum_{i \in \alpha} \psi_{\alpha^i}(x_\alpha)$. The superscript denotes the variable to which the split factor has a direct connection in the graph. See Fig. 3.1 for example transformations.

Recall that there is a pair of directed edges $e \in E'$ for all neighborhood relations in G'. Since G' is a tree-graph, removing any edge would divide the tree into two independent subgraphs. We denote the subgraph that is implicitly "on the side" of the sending node $\underline{e}$ as G'_e. Since remote neighbours can only be reached by relaying messages over multiple hops, the rules by which variables are marginalized out in the messages need to be adapted. For this, we introduce *extended messages*

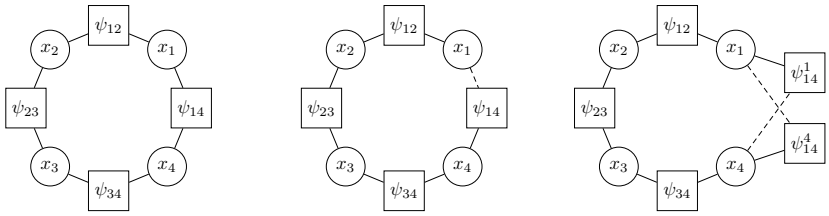
$$\tilde{m}_e = \langle \tilde{V}_e, (c_{e,i}, |\tilde{N}(i)|)_{i \in \tilde{V}_e}, \bar{m}_e \rangle \,.$$

The time-index is omitted for extended messages since Max-Sum with Remote Neighbours is intended only for a forwards/backwards pass schedule on tree-graphs. Messages are only sent once a message has been received from neighbours but the target-node in questions (see also Sec. 2). The set $\tilde{V}_e$ is the union of (a) the variables in the subgraph G'_e with a (remote or direct) neighbour not in G'_e and (b) the variables not in G'_e that are in the domain of a factor in G'_e. We denote the variable nodes not contained in V_e with $\overline{V_e} = V \setminus V_e$ and similarly for factor nodes.

$$\begin{aligned} \tilde{V}_e^a &= \{i \in V_e : \tilde{N}(i) \cap \overline{A_e} \neq \varnothing\} \\ \tilde{V}_e^b &= \{i \in \overline{V_e} : \tilde{N}(i) \cap A_e \neq \varnothing\} \\ \tilde{V}_e &= \tilde{V}_e^a \cup \tilde{V}_e^b \end{aligned} \tag{3.1}$$

For messages in the inverse direction, $\tilde{V}_{\underline{e} \to \overline{e}} = \tilde{V}_{\overline{e} \to \underline{e}}$. This follows directly from (3.1) by replacing the sets V_e and A_e with their complement.

The integer $c_{e,i}$ counts how many nodes related to i (neighbours of i and i itself) are contained in the sending subgraph G'_e. This value is updated locally before



(a) (b) (c)

Figure 3.1: The loopy factor graph in (a) transforms into the factor tree-graph with remote neighbours (b) where x_1 and ψ_{14} are their respective remote neighbour. The transformed factor graph in (c) contains a split factor. The transformed graph gives results for variable assignments equivalent to the graph in (a) if $\psi_{14}(x_{14}) = \psi_{14^1}(x_{14}) + \psi_{14^4}(x_{14})$.

sending a message m_e.

$$c_{e,i} = \mathbb{1}_{i \in \tilde{N}(\underline{e})} + \mathbb{1}_{\underline{e}=i} + \sum_{k \in N(\underline{e}) \setminus \{\overline{e}\}} c_{k \to \underline{e},i} \tag{3.2}$$

Implicitly, $c_{e,i} = 0$ if it is not defined in the message m_e. The indicator function $\mathbb{1}$ evaluates to one if the supplied condition is true and zero otherwise. Note that (3.2) is formulated both for sending variable nodes and sending factor nodes.

Proposition 1 *A variable i is contained in $\tilde{V}_e$ if and only if $0 < c_{e,i} < |\tilde{N}(i)| + 1$.*

PROOF. Since G'_e is a tree-graph and only factors in $\tilde{N}(i)$ and the variable node i itself can add to (3.2), $c_{e,i}$ is bounded with $c_{e,i} \in \{0, \dots, |\tilde{N}(i)| + 1\}$. Firstly, if $c_{e,i} = 0$, then neither is $i \in V_e$, nor is there a factor α, such that $\alpha \in A_e \cap \tilde{N}(i)$ as any of these conditions would have increased $c_{e,i}$ in (3.2). It follows from (3.1) that $i \notin \tilde{V}_e$. The inverse argument proceeds analogously. Secondly, assume that $i \in \tilde{V}_e$ and $c_{e,i} = |\tilde{N}(i)| + 1$. If $i \in \tilde{V}_e^a$, then $c_{e,i} < |\tilde{N}(i)| + 1$ since at least one neighbour factor $\alpha \in \tilde{N}(i)$ is not contained in A_e and has not contributed to (3.2). If $i \in \tilde{V}_e^b$, then $c_{e,i} < |\tilde{N}(i)| + 1$ with a similar argument. This contradicts the assumption. Lastly, let $c_{e,i}$ such that $0 < c_{e,i} < |\tilde{N}(i)| + 1$. If i is contained in V_e, then $\tilde{N}(i) \cap \overline{A_e}$ is nonempty as $c_{e,i}$ would equal $|\tilde{N}(i)| + 1$ otherwise. If i is not contained in V_e, then $\tilde{N}(i) \cap A_e$ is nonempty as $c_{e,i}$ would equal zero otherwise. It follows from the exhaustion of cases that $i \in \tilde{V}_e \Leftrightarrow 0 < c_{e,i} < |\tilde{N}(i)| + 1$. □

Together with the $c_{e,i}$, the number of i's (direct and remote) neighbours $|\tilde{N}(i)|$ is contained in $\tilde{m}_e$. Thus, extended messages can be constructed with information

received via extended messages from neighbour nodes and locally available information. The difference to the original Max-Sum is that the domain $\mathcal{X}_e = \prod_{i \in \tilde{V}_e} \mathcal{X}_i$ of the message function $\bar{m}_e$ may be comprised of multiple variables. Let the *extended domain* of any (factor or variable) node ν be $\mathcal{X}_{\nu^+} = \prod_{i \in \nu^+} \mathcal{X}_i,\ \nu^+ = \{i : \exists k \in N(\nu), i \in \tilde{V}_{k \to \nu}\}$. The value of every $\bar{m}_e(x_e)$ is computed by taking the maximum over all $y \in \mathcal{X}_{\underline{e}^+} : \forall i \in \tilde{V}_e, y_i = (x_e)_i$ by taking the sum of the relevant messages (from all direct neighbours but the target $\overline{e}$) and the local factor $\psi_{\underline{e}}$ on the variable assignment y. If $\underline{e}$ is a variable node, of course $\psi_{\underline{e}}$ evaluates to zero.

$$\bar{m}_e(x_e) = \max_{x_{\underline{e}^+ \setminus \tilde{V}_e}} \left[\psi_{\underline{e}}(x_{\underline{e}}) + \sum_{\substack{g=(k \to \underline{e}),\\ k \in N(\underline{e}) \setminus \overline{e}}} \bar{m}_g(x_g) \right] \tag{3.3}$$

Proposition 2 *Let G' be a factor tree-graph with remote neighbours representing a function f. After completing a forwards/backwards schedule, the extended messages exchanged on G' have converged and the max-marginal of the variable $i \in V$ on f can be computed as*

$$f_i^*(x_i) = \max_{\substack{y \in \mathcal{X},\\ y_i = x_i}} f(y) = \max_{x_{i^+ \setminus \{i\}}} \sum_{\substack{g=(k \to i),\\ k \in N(i)}} \bar{m}_g(x_g)\,.$$

PROOF. Messages $\tilde{m}_e$ on a tree-graph do not depend on any information from messages sent by the (currently) receiving node $\overline{e}$. Therefore, after the forwards/backwards schedule has completed, every updated message $\tilde{m}_e$ according to (3.3) is identical to the message that has last been sent over e.

Next, we show that the value of a message function $\bar{m}_e(x_e)$ for a given x_e equals the summed factor-values in the sending subgraph V_e where all variables in $\tilde{V}_e$ are assigned according to x_e and the variables in V_e without (remote) neighbours outside of G_e are assigned to maximize the sum of factors in G_e.

$$\bar{m}_e(x_e) = \max_{\substack{x \in \mathcal{X}_{V_e \cup \tilde{V}_e},\\ \forall i \in \tilde{V}_e, x_i = (x_e)_i}} \sum_{\alpha \in A_e} \psi_\alpha(x_\alpha)$$

Note that $V_e \cup \tilde{V}_e = \bigcup_{\alpha \in A_e} \tilde{N}(\alpha)$ contains all variables that are the (remote) neighbour of any factor in A_e. Recall that variables i are max-marginalized out during the message construction (3.3) only if $i \in V_e \setminus \tilde{V}_e$, i.e. if no factor outside of G_e is a (remote) neighbour of i. That means the distributive law on the Max-Sum commutative semiring can be applied [AM00]. For example:

$$\max_{x_1, x_2} \left[\psi_1(x_1) + \psi_{12}(x_1, x_2) \right] = \max_{x_1} \left[\psi(x_1) + \max_{x_2} \psi(x_1, x_2) \right]$$

The argument is applied recursively until the leaf nodes are reached, for which it holds trivially.

Now, assume that variable i has received messages $\tilde{m}_{\alpha \to i}$ from all neighbours $\alpha \in N(i)$. For every selected $x_i \in \mathcal{X}_i$ we then have locally available information on the maximum summed factor-values that can be achieved in the subgraphs behind all outgoing edges, i.e. the entire graph. □

Proposition 3 *On a tree-like factor graph G without remote neighbours, the message functions $\bar{m}$ exchanged in Max-Sum with Remote Factors are equal to the corresponding messages m of the original Max-Sum algorithm up to normalization.*

PROOF. For this, we show that the domain of the extended messages exchanged between any factor α and variable i is $\mathcal{X}_i$ and therefore $\tilde{V}_{i \to \alpha} = \tilde{V}_{\alpha \to i} = \{i\}$. If the sending node is the variable node i, then $\tilde{V}^a_{i \to \alpha} = \{i\}$ since no other variable in $V_{i \to \alpha}$ has a neighbour in $\overline{V_{i \to \alpha}}$. Also, $\tilde{V}^b_{i \to \alpha} = \varnothing$ since no variable node in $\overline{V_{i \to \alpha}}$ has a neighbouring factor in $V_{i \to \alpha}$. This follows directly from G being a tree-graph. Similar arguments hold for messages from α to i. Since the domain of the exchanged messages on G is identical, it is easy to see that (2.1) and (3.3) are equivalent when $\kappa = 0$ (normalization is not required for convergence on trees). Since the message functions sent from the leaf nodes do not depend on received messages, they are identical for Max-Sum and Max-Sum with Remote Neighbours and consequently all $\bar{m}_e = m_e$. □

4 Evaluation

This section presents the results of a simple scenario that compares the proposed algorithm in comparison with loopy messsage passing. The algorithm implementation and the scenario example can be accessed online at `https://github.com/jpfr/pygmalion`.

The scenario consists of eight variables, each with a finite, nominal scale of size 5, and eight factors linking the variables to form two connected circles according to Fig. 4.1. The factor functions ψ_α map each element in their domain to a scalar that was randomly sampled from the uniform distribution on $[0, 1]$ during instantiation. The goal is then to find the variable assignment that maximizes the sum of all factors. For Max-Sum with Remote Factors, we removed two edges so that the transformed graph forms a connected tree as can be seen in Fig. 4.1. The size of the scenario is such that brute-force search (in 5^8 possible solutions) can still be applied to verify the results.

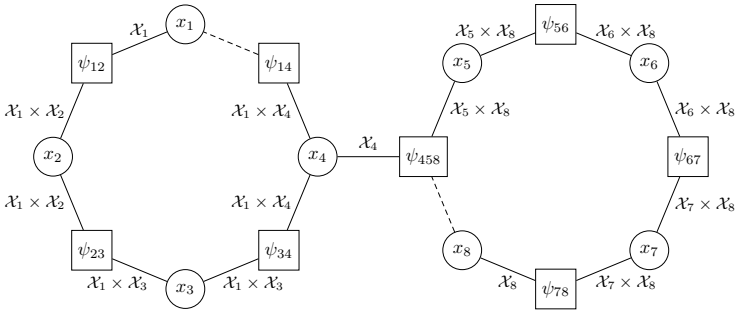


Figure 4.1: The factor graph with remote factors in the example scenario. Dashed edges have been removed for Max-Sum with Remote Factors. The remaining edges are annotated with the domain of the message-functions $\bar{m}_e$ that are passed over it.

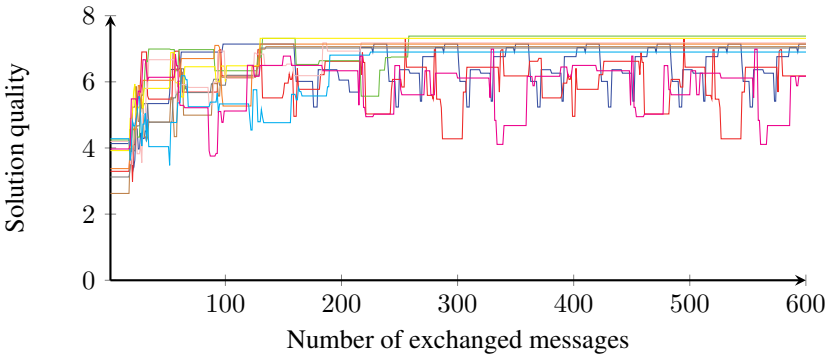


Figure 4.2: Performance of loopy Max-Sum message passing on 10 instances of the scenario. The instances that have not converged after 600 messages show a recurring pattern that is repeated indefinitely.

Figure 4.2 shows the result of Max-Sum on the loopy graph. At every time t all nodes send message to all of their neighbours if these messages are different from the preceding ones. The variable assignments after every message exchange were computed by taking the (running) max-marginal of each variable nodes and maximizing it locally. It can be seen, that inference with Max-Sum can be quite irratic and does not converge in all scenario instances.

By contrast, Max-Sum with Remote Neighbours takes only 30 messages (one forwards/backwards schedule) to infer the maximum solution for all scenario instances. The average message size increase compared to Max-Sum is less than 5, i.e. the domain size of the variables that were made remote neighbours. Figure 4.2 also shows the domain of the message-functions $\bar{m}_e$ passed over the edges. Note how the domain-size increase due to a missing edge is *loop-local* and does not spill over into the adjacent loop. This is an indicator that Max-Sum with Remote Neighbours will perform well for many important applications. Extensive benchmarks and comparison with other DCOP approaches are currently being developed.

5 Conclusion

In this paper, we introduced Max-Sum with Remote Neighbours, a method for the exact inference of maximum utility solutions in distributed constraint optimization settings. This method generalizes the original Max-Sum, that is widely used for DCOP applications, and makes it applicable to settings where the agent communication is constraints and the communication graph does not match the agents utility inter-dependencies. The only requirement for our method is that the communication graph is still connected. The performance of Max-Sum with Remote Neighbours was evaluated on an example scenario. We also applied our method on loopy graphs where some communication links were cut in order to form a tree-graph. Here, it showed superior performance compared to applying the original Max-Sum on loopy graphs, as is often done in practice. It is an open question which links to remove to minimize the inference complexity. Here, we suspect a rich connection to the large body of work dealing with the decomposition of graphs into junction trees with a minimized tree-width.

Bibliography

[AM00] Srinivas M Aji and Robert J McEliece. The generalized distributive law. *Information Theory, IEEE Transactions on*, 46(2):325–343, 2000.

[BM10] Ismel Brito and Pedro Meseguer. Cluster tree elimination for distributed constraint optimization with quality guarantees. *Fundamenta Informaticae*, 102(3):263–286, 2010.

[FRPJ08] Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas R Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 639–646, 2008.

[KFL01] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

[KV06] Jelle R. Kok and Nikos Vlassis. Using the max-plus algorithm for multiagent decision making in coordination graphs. In *Robot Soccer World Cup IX*, pages 1–12. Springer, 2006.

[MSTY05] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1):149–180, 2005.

[MTB+04] Rajiv T Maheswaran, Milind Tambe, Emma Bowring, Jonathan P Pearce, and Pradeep Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed multi-event scheduling. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 310–317, 2004.

[PF05] Adrian Petcu and Boi Faltings. DPOP: A Scalable Method for Multiagent Constraint Optimization. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, pages 266–271, 2005.

[PGCMRA11] Marc Pujol-Gonzalez, Jesus Cerquides, Pedro Meseguer, and Juan A Rodriguez-Aguilar. Communication-constrained DCOPs: Message approximation in GDL with function filtering. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 379–386. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[VRAC11] Meritxell Vinyals, Juan A Rodriguez-Aguilar, and Jesús Cerquides. Constructing a unifying theory of dynamic programming DCOP algorithms via the generalized distributive law. *Autonomous Agents and Multi-Agent Systems*, 22(3):439–464, 2011.

[YFW05] Jonathan S Yedidia, William T Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory, IEEE Transactions on*, 51(7):2282–2312, 2005.