Towards Improving the Expressivity and Scalability of Distributed Constraint Optimization Problems

William Yeoh
Department of Computer Science and Engineering
Washington University in St. Louis
wyeoh@wustl.edu

Abstract
Constraints have long been studied in centralized systems and have proven to be practical and efficient for modeling and solving resource allocation and scheduling problems. Slightly more than a decade ago, researchers proposed the distributed constraint optimization problem (DCOP) formulation, which is well suited for modeling distributed multi-agent coordination problems. In this paper, we highlight some of our recent contributions that are aiming towards improved expressivity of the DCOP model as well as improved scalability of the accompanying algorithms.

1 Introduction
In a distributed constraint optimization problem (DCOP), agents need to coordinate the assignments of values to their variables in such a way that maximizes their aggregated utilities [Modi et al., 2005; Petcu and Faltings, 2005a]. They are well suited for modeling multi-agent coordination problems where the primary interactions are between local subsets of agents, such as in meeting scheduling [Maheswaran et al., 2004b], sensor networks [Farinelli et al., 2014], multi-robot coordination [Zivan et al., 2015], coalition formation [Ueda et al., 2010], smart grid [Kumar et al., 2009; Fioretto et al., 2017b], and smart home automation [Rust et al., 2016; Fioretto et al., 2017a] problems.

When DCOPs were introduced slightly more than a decade ago, research efforts were initially focused on the investigation of different algorithmic paradigms to solve the problem, including exact search-based methods [Modi et al., 2005; Gershman et al., 2009; Yeoh et al., 2010; Gutierrez et al., 2011], exact inference-based methods [Petcu and Faltings, 2005a; Vinyals et al., 2011], approximate search-based methods [Maheswaran et al., 2004a; Zhang et al., 2005; Zivan et al., 2014], approximate inference-based methods [Farinelli et al., 2014; Zivan and Peled, 2012], and approximate sampling-based methods [Ottens et al., 2017].

Since then, the field has substantially matured and researchers have begun to propose generalizations to the model and algorithms to better capture and exploit characteristics of more complex applications. The goal of this paper is to briefly highlight our recent contributions in this area. We refer readers to our overview and survey articles [Yeoh and Yokoo, 2012; Fioretto et al., 2018a] for a more comprehensive perspective on the state of the art of DCOPs.

2 DCOP Model
A distributed constraint optimization problem (DCOP) [Modi et al., 2005] is a tuple \((A, X, D, F, \alpha)\), where:
- \(A = \{a_i\}_{i=1}^p\) is a set of agents;
- \(X = \{x_i\}_{i=1}^n\) is a set of decision variables;
- \(D = \{D_x\}_{x \in X}\) is a set of finite domains and each variable \(x \in X\) takes values from the set \(D_x \in D\);
- \(F = \{f_i\}_{i=1}^k\) is a set of constraints, each defined over a set of decision variables: \(f_i : \prod_{x \in X} D_x \rightarrow \mathbb{R}_+^+ \cup \{-\infty\}\), where infeasible configurations have \(-\infty\) utility, \(x_i^f \subseteq X\) is the scope of \(f_i\); and
- \(\alpha : X \rightarrow A\) is a function that associates each decision variable to one agent.

The goal is to find an optimal solution \(x^* = \arg\max_x F(x)\), where \(F(x)\) is the sum of utilities across all constraints.

3 Our Recent Contributions
We now briefly highlight several key contributions that we have made in this area.

3.1 Hierarchical Decomposition for DCOPs
In most complex distributed multi-agent applications, each agent typically needs to solve complex local subproblems [Kim and Lesser, 2013; Giuliani et al., 2014; Amigoni et al., 2015]. While these complex local structures can be captured by the regular DCOP model, through allowing an agent to control multiple variables, unfortunately, many DCOP algorithms commonly assume that each agent controls only one variable. To cope with such restrictions, reformulation techniques are commonly used to transform a regular DCOP into one where each agent controls exclusively one variable. There are two commonly used reformulation techniques [Burke and Brown, 2006; Yokoo, 2001]: (i) Compilation, where each agent creates a new \textit{pseudo-variable}, whose domain is the Cartesian product of the domains of all variables of the agent; and (ii) Decomposition, where each agent creates a \textit{pseudo-agent} for each of its variables. While both techniques are relatively simple, they can be inefficient. In
compilation, the memory requirement for each agent grows exponentially with the number of variables that it controls. In decomposition, the DCOP algorithms will treat two pseudo-agents as independent entities, resulting in unnecessary computation and communication costs.

Therefore, we proposed a novel decomposition method, called multi-variable agent (MVA) decomposition, to overcome these limitations [Fioretto et al., 2016b]. Our MVA decomposition enables a separation between the agents’ local subproblems, which can be solved independently using centralized solvers, and the DCOP global problem, which requires coordination between the agents. Thus, it enables the use of different centralized and distributed solvers in a hierarchical and parallel way, where different agents can use different centralized solvers that exploit their local subproblem structures for efficiency gains. Not surprisingly, our experimental results show that DCOP algorithms using our MVA decomposition have reduced computation and communication costs compared to using the compilation and decomposition methods.

3.2 Linear-space Sampling-based Algorithm

Distributed UCT (DUCT) [Ottens et al., 2017] is the first sampling-based DCOP algorithm that was introduced. Unfortunately, its memory requirement per agent is exponential in the number of agents in the problem, which prohibits it from some multi-agent applications like sensor networks, where agents may have a very limited amount of memory. In response to this limitation, we proposed the Distributed Gibbs algorithm [Nguyen et al., 2013], which extends upon the (centralized) Gibbs algorithm [Geman and Geman, 1984], which is used to solve maximum a posteriori (MAP) estimation problems in graphical models.

To do this, we first showed that one can map DCOPs to MAP estimation problems [Kumar et al., 2011], where DCOP variables and constraints correspond to random variables and potential functions of MAP estimation problems. Then, any algorithm that solves MAP estimation problems can theoretically be used to solve DCOPs as well. However, as DCOP algorithms must be distributed and agent oriented, MAP estimation algorithms, which are typically centralized, cannot be directly applied to solve DCOPs. Thus, we extended the well-known Gibbs algorithm to a distributed, agent-oriented version to solve DCOPs. A key property of this algorithm is that its memory requirement per agent is linear in the number of agents in the problem in contrast to DUCT’s exponential memory requirement.

3.3 Exploiting Parallelism using GPUs

Sampling-based DCOP algorithms, such as Distributed Gibbs, perform a significant number of sampling operations that are conditionally-independent operations. As such, instead of performing these sampling operations sequentially, they can be done in parallel and sped up through the use of graphical processing units (GPUs) [Fioretto et al., 2016a]. We further show that when Distributed Gibbs is used in conjunction with the MVA decomposition, we can achieve further speedups as the local subproblems within each agent can also be sampled in parallel [Fioretto et al., 2016b]. Finally, we also show that other inference-based DCOP algorithms like DPOP can also be sped up through the use of GPUs, as the computation of utility tables that are propagated between agents can be decomposed into conditionally-independent operations [Fioretto et al., 2018b].

The main challenge in this line of work is to parallelize the algorithms in such a way that optimizes the speedup from using GPUs. While it is relatively simple to develop correct programs (e.g., by incrementally modifying a sequential program), it is nevertheless challenging to design an efficient solution. Several factors are critical in gaining performance. Memory levels have significantly different sizes and access times, and various optimization techniques are available (e.g., accesses to consecutive global memory locations by contiguous threads can be coalesced into a single memory transaction). Thus, optimization of these programs requires a thorough understanding of GPU hardware characteristics.

3.4 Constraint Propagation for DCOPs

Constraint propagation [Mohr and Henderson, 1986] is a commonly used technique, especially in the constraint programming community, to speed up the search for solutions in constraint-based models. The key idea is that one can prune portions of the search space if they do not satisfy some subset of hard constraints (i.e., constraints that must be satisfied) in the problem. Motivated by their success in centralized constraint-based models, we proposed a new constraint propagation technique that is tailored for DCOP inference algorithms that use pseudo-trees. Our new branch consistency approach [Fioretto et al., 2014] can be viewed as a generalization of the more traditional arc consistency and a weaker version of path consistency [Mohr and Henderson, 1986], where it is customized for distributed operations of agents that can communicate only with neighboring agents.

Instead of designing specialized constraint propagation techniques, one can also leverage the advances made by other communities to automatically propagate constraints. Towards this end, we proposed the use of answer set programming (ASP), developed by the logic programming community, in solving DCOPs [Le et al., 2015; 2017]. When constraint utilities are described in functional form, they can be compactly represented as ASP rules and propagated between agents efficiently, resulting in improved scalability through reduced runtimes and reduced communication overheads. This line of work is especially promising as continuous advancements made by the logic programming community can be automatically adopted to solve DCOPs with little effort.

3.5 Dynamic and Uncertain DCOPs

All our contributions described above are in the context of solving regular DCOPs. However, many multi-agent coordination problems change dynamically over time and with uncertainty. Early efforts in modeling such dynamic DCOPs have been to represent them as sequences of regular DCOPs, with changes between subsequent problems, and the goal is to solve each individual DCOP optimally [Petcu and Faltinings, 2005b]. As changes between subsequent DCOPs can be small, a large portion of the problem will remain unchanged.
and, consequently, a large portion of the solution to the previous problem can be reused. With this observation in mind, we proposed incremental search-based approaches that identify and reuse such reusable portions, and focus their search efforts on solving the portion of the problem that changed [Yeoh et al., 2015]. Results show that the speedup obtain is, not surprisingly, correlated with the size of the reusable portion of the previous solution.

In problems where only the constraint utilities change (the agents, variables, domains, scope of constraints, etc. remain unchanged), one can model the changes in utilities as a function of an underlying state that changes over time. For example, in a sensor network application, a dynamically changing phenomena that the network is trying to observe can be represented as an underlying state that changes over time. To solve these problems, called Markovian dynamic DCOPs, we proposed distributed reinforcement learning algorithms that learn the underlying state transition function and, consequently, the resulting constraint utilities, thereby allowing them to find good solutions over time [Nguyen et al., 2014]. Finally, when the state transitions functions are known or can be estimated a priori, we proposed proactive DCOP algorithms that take them into account when searching for solutions [Hoang et al., 2016; 2017].

3.6 Preference Elicitation for DCOPs
Existing DCOP algorithms typically assume that all constraint utilities are fully specified a priori as they are provided as part of the problem definition. However, this assumption does not hold in some applications such as smart home scheduling problems [Fioretto et al., 2017a], where the utilities of some constraints represent user preferences and should thus be elicited from users. With this motivation in mind, we proposed the preference elicitation problem for DCOPs, which partitions the constraints into constraints with known utilities and constraints with unknown utilities that must either be elicited or approximated [Tabakh et al., 2017]. We then proposed several heuristics, including one based on minimax regret, to solve this problem.

Following up on this initial work, we also investigated the use of matrix completion algorithms to identify which subset of utilities to elicit in order to best approximate the remaining unelicited utilities [Le et al., 2018]. Further, we model the cognitive bother cost of the user associated with providing the elicited utilities, and optimize the sequence of questions to ask users such that the total bother cost is within some pre-defined threshold. While both approaches are described for general models, they were evaluated in the context of smart home automation problems, where we show that they perform better than random baseline methods.

3.7 Variable-to-Agent Mappings for DCOPs
Finally, the conventional DCOP model assumes that the mapping of variables to agents is part of the model description and is thus given as an input. This assumption is reasonable in many applications where there are obvious and intuitive mappings. For example, in a smart home scheduling problem, agents correspond to the different smart homes, and variables correspond to the different smart devices within each home. In this case, the agent controls all the variables that map to the devices in its home. However, in other applications, there may be more flexibility in the mapping of variables to agents. For example, imagine an application where a team of unmanned aerial vehicles (UAVs) need to coordinate with each other to effectively survey an area. In this application, agents correspond to UAVs and variables correspond to the different zones in the area to be surveyed. The domain for each variable may correspond to the different types of sensors to be used and/or the different times to survey the zone. Since a UAV can survey any zone, there are multiple possible assignments of zones to UAVs (i.e., there are multiple possible mappings of variables to agents).

While choosing a good mapping is important as it can have a significant impact on an algorithm’s runtime, choosing an optimal mapping may be prohibitively time consuming as this is an NP-hard problem [Rust et al., 2016]. Considering these issues, coupled with the fact that this step is only a preprocessing step prior to the execution of the actual DCOP algorithm, we developed a generic heuristic-based algorithm that can be executed in a centralized or decentralized manner [Khan et al., 2018]. At a high level, the algorithm uses Fortune’s algorithm [Fortune, 1987] to partition the DCOP graph into p partitions, where p is the number of agents in the problem, in such a way where the number of high-degree nodes (i.e., variables that are in the scopes of large numbers of constraints) is balanced across all partitions. Surprisingly, our experimental results show that our heuristic-based approach finds mappings that result in runtimes that are only within 10% of the runtimes found with optimal mappings for GDL-based DCOP algorithms on random and scale-free graphs.

4 Conclusions and Future Directions
While the theoretical foundations and algorithmic improvements for conventional DCOPs have matured significantly over the past decade, their deployment to realistic applications is unfortunately lagging. To make this transition, we hypothesize that researchers will need to make further progress in generalizing the conventional DCOP model, adapting it to specific applications, and, most importantly, developing efficient algorithms for the corresponding extensions. Our efforts described in this paper are motivated by this belief. Other promising directions include investigations in the intersection of DCOPs and game theory, such as the work by Chapman et al. [2008], and the use of machine learning techniques for DCOPs, such as the work by Kumar and Zilberstein [2011] and Ghosh et al. [2015]. Additionally, orthogonal to any-time [Zivan et al., 2014] and any-space algorithms [Petcu and Faltings, 2007; Yeoh et al., 2009], any-communication algorithms (i.e., algorithms that adapt the number, size, and content of messages sent between agents based on the degree of congestion in the network) should also be of interest, especially in applications with potentially degraded communication channels.

Acknowledgments
The contributions highlighted in this paper are a result of collaborations with many other researchers. Special thanks go to...
Ferdinando Fioretto, Khoi Hoang, Md. Mosaddek Khan, Tiep Le, Duc Thien Nguyen, and Atena M. Tabakhi, the student researchers who led their respective research thrusts. The author also gratefully acknowledges support from the National Science Foundation through grants 1345232, 1540168, and 1550662.

References


