

arXiv:1705.03916v1 [cs.LG] 10 May 2017

Under consideration for publication in Theory and Practice of Logic Programming

Solving Distributed Constraint Optimization Problems Using Logic Programming

Tiep Le, Tran Cao Son, Enrico Pontelli, William Yeoh

Computer Science Department
New Mexico State University Las Cruces, NM, 88001, USA
E-mail: {tle, tson, epontelli, wyeh}@cs.nmsu.edu

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

This paper explores the use of Answer Set Programming (ASP) in solving Distributed Constraint Optimization Problems (DCOPs). The paper provides the following novel contributions: (1) It shows how one can formulate DCOPs as logic programs; (2) It introduces ASP-DPOP, the first DCOP algorithm that is based on logic programming; (3) It experimentally shows that ASP-DPOP can be up to two orders of magnitude faster than DPOP (its imperative programming counterpart) as well as solve some problems that DPOP fails to solve, due to memory limitations; and (4) It demonstrates the applicability of ASP in a wide array of multi-agent problems currently modeled as DCOPs.

1 Introduction

Distributed Constraint Optimization Problems (DCOPs) are optimization problems where agents need to coordinate the assignment of values to their local variables to maximize the overall sum of resulting constraint utilities (Modi et al. 2005; Petcu and Faltings 2005a; Mailler and Lesser 2004; Yeoh and Yokoo 2012). The process is subject to limitations on the communication capabilities of the agents; in particular, each agent can only exchange information with neighboring agents within a given topology. DCOPs are well-suited for modeling multi-agent coordination and resource allocation problems, where the primary interactions are between local subsets of agents. Researchers have used DCOPs to model various problems, such as the distributed scheduling of meetings (Maheswaran et al. 2004; Zivan et al. 2014), distributed allocation of targets to sensors in a network (Farinelli et al. 2008), distributed allocation of resources in disaster evacuation scenarios (Lass et al. 2008), the distributed management of power distribution networks (Kumar et al. 2009; Jain et al. 2012), the distributed generation of coalition structures (Ueda et al. 2010) and the distributed coordination of logistics operations (Leaute and Faltings 2011).

This article extends our previous conference paper (Le et al. 2015) in the following manner: (1) It provides a more thorough description of the ASP-DPOP algorithm; (2) It elaborates on the algorithm's theoretical properties with complete proofs; and (3) It includes additional experimental results.

2 Tiep Le, Tran Cao Son, Enrico Pontelli, and William Yeoh

The field has matured considerably over the past decade, since the seminal ADOP T paper (Modi et al. 2005), as researchers continue to develop more sophisticated solving algorithms. The majority of the DCOP resolution algorithms can be classified in one of three classes: (1) Search-based algorithms, like ADOP T (Modi et al. 2005) and its variants (Yeoh et al. 2009; Yeoh et al. 2010; Gutierrez et al. 2011; Gutierrez et al. 2013), AFB (Gershman et al. 2009), and MGM (Maheswaran et al. 2004), where the agents enumerate combinations of value assignments in a decentralized manner; (2) Inference-based algorithms, like DPOP (Petcu and Faltings 2005a) and its variants (Petcu and Faltings 2005b; Petcu and Faltings 2007; Petcu et al. 2007; Petcu et al. 2008), max-sum (Farinelli et al. 2008), and Action GDL (Vinyals et al. 2011), where the agents use dynamic programming techniques to propagate aggregated information to other agents; and (3) Sampling-based algorithms, like DUCT (Ottens et al. 2012) and D-Gibbs (Nguyen et al. 2013; Fioretto et al. 2014), where the agents sample the search space in a decentralized manner. The existing algorithms have been designed and developed almost exclusively using imperative programming techniques, where the algorithms define a control flow, that is, a sequence of commands to be executed. In addition, the local solver employed by each agent is an ad-hoc implementation. In this paper, we are interested in investigating the benefits of using declarative programming techniques to solve DCOPs, along with the use of a general constraint solver, used as a black box, as each agent's local constraint solver. Specifically, we propose an integration of Distributed Pseudo-tree Optimization Procedure (DPOP) (Petcu and Faltings 2005a), a popular DCOP algorithm, with Answer Set Programming (ASP) (Niemela 1999; Marek and Truszczyński 1999) as the local constraint solver of each agent. This paper provides the first step in bridging the areas of DCOPs and ASP; in the process, we offer novel contributions to both the DCOP field as well as the ASP field. For the DCOP community, we demonstrate that the use of ASP as a local constraint solver provides a number of benefits, including the ability to capitalize on (i) the highly expressive ASP language to more concisely define input instances (e.g., by representing constraint utilities as implicit functions instead of explicitly enumerating their extensions) and (ii) the highly optimized ASP solvers to exploit problem structure (e.g., propagating hard constraints to ensure consistency). For the ASP community, the paper makes the equally important contribution of increasing the applicability of ASP to model and solve a wide array of multi-agent coordination and resource allocation problems, currently modeled as DCOPs. Furthermore, it also demonstrates that general, off-the-shelf ASP solvers, which are continuously honed and improved, can be coupled with distributed message passing protocols to outperform specialized imperative solvers. The paper is organized as follows. In Section 2, we review the basic definitions of DCOPs, the DPOP algorithm, and ASP. In Section 3, we describe in detail the structure of the novel ASP-based DCOP solver, called ASP-DPOP, and its implementation. Section 4 provides an analysis of the properties of ASP-DPOP, including proofs of soundness and completeness of ASP-DPOP. Section 5 provides some experimental results, while Section 6 reviews related work. Finally, Section 7 provides conclusions and indications for future work.

Solving Distributed Constraint Optimization Problems Using Logic Programming

3.2 Background

In this section, we present an overview of DCOPs, we describe DPOP, a complete distributed algorithm to solve DCOPs, and provide some fundamental definitions of ASP.

3.1 Distributed Constraint Optimization Problems

A Distributed Constraint Optimization Problem (DCOP) (Modi et al. 2005; Petcu and Faltings 2005a; Mailler and Lesser 2004; Yeoh and Yokoo 2012) can be described as a tuple $M = \langle X, D, F, A, \pm \rangle$ where: $X = \{x_1, \dots, x_n\}$ is a finite set of (decision) variables; $D = \{D_1, \dots, D_n\}$ is a set of finite domains, where D_i is the domain of the variable $x_i \in X$, for $1 \leq i \leq n$; $F = \{f_1, \dots, f_m\}$ is a finite set of constraints, where f_j is a k_j -ary function $f_j: D_{j_1} \times D_{j_2} \times \dots \times D_{j_{k_j}} \rightarrow \mathbb{R}$ that specifies the utility of each combination of values of variables in its scope; the scope is denoted by $sc_p(f_j) = \{x_{j_1}, \dots, x_{j_{k_j}}\}$; $A = \{a_1, \dots, a_p\}$ is a finite set of agents; and $\pm: X \rightarrow A$ maps each variable to an agent. We say that a variable x is owned by an agent a if $\pm(x) = a$. We denote with \pm_i the set of all variables that are owned by an agent a_i , i.e., $\pm_i = \{x \in X \mid \pm(x) = a_i\}$. Each constraint in F can be either hard, indicating that some value combinations result in a utility of $-\infty$ and must be avoided, or soft, indicating that all value combinations result in a finite utility and need not be avoided. A value assignment is a (partial or complete) function x that maps variables of X to values in D such that, if $x(x_i)$ is defined, then $x(x_i) \in D_i$ for $i = 1, \dots, n$. For the sake of simplicity, and with a slight abuse of notation, we will often denote $x(x_i)$ simply with x_i . Given a constraint f_j and a complete value assignment x for all decision variables, we denote with x_{f_j} the projection of x to the variables in $sc_p(f_j)$; we refer to this as a partial value assignment for f_j . For a DCOP M , we denote with