Engineering Applications of Artificial Intelligence 21 (2008) 723–732

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Feasible distributed CSP models for scheduling problems

Miguel A. Salido *, Adriana Giret

Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022 - Valencia, Spain

ARTICLE INFO

Article history:
Received 29 January 2008
Accepted 1 March 2008
Available online 22 April 2008

Keywords:
Distributed CSP
Constraint satisfaction
Holonic system
Multi-agent system
Scheduling

ABSTRACT

A distributed constraint satisfaction problem (DisCSP) is a CSP in which variables and constraints are distributed among multiple automated agents. Many researchers have developed techniques for solving DisCSPs. They assume for simplicity that each agent has exactly one variable. For real planning and scheduling problems, these techniques require a large number of messages passing among agents, so these problems are very difficult to solve. In this paper, we present a general distributed model for solving real-life scheduling problems. This distributed model is based on the idea of holonic systems. Furthermore, we propose some guidelines for distributing large-scale problems. Finally, we present two case studies in which two scheduling problems are distributed by using our model.

1. Introduction

In recent years, we have seen increasing interest in distributed constraint satisfaction problem (DisCSP) formulations to model combinatorial problems (see Faltings and Yokoo, 2005). There is a rich set of real-world distributed applications, such as network systems, planning, scheduling, resource allocation, etc., for which the DisCSP paradigm is particularly useful. In such distributed applications, privacy issues, knowledge transfer costs, robustness against failure, etc., preclude the adoption of a centralized approach (Faltings and Yokoo, 2005).

Briefly, a constraint satisfaction problem (CSP) consists of the following:

- a set of variables $X = \{x_1, x_2, \ldots, x_n\}$;
- each variable $x_i \in X$ has a set D_i of possible values (its domain);
- a finite collection of constraints $C = \{c_1, c_2, \ldots, c_p\}$ restricts the values that the variables can simultaneously take.

A solution to a CSP is an assignment of values to all the variables so that all constraints are satisfied. A problem with a solution is termed *satisfiable* or *consistent*.

A typical example of a CSP is the graph coloring problem. The objective is to color each region so that adjacent regions have different colors (Fig. 1). A problem of this kind is called a CSP since the objective is to find a configuration that satisfies the given conditions (constraints). Even though the definition of a CSP is very simple, a surprisingly wide variety of Artificial Intelligence problems can be formalized as CSPs. Therefore, the research on CSP has a long and distinguished history in Artificial Intelligence (Mackworth, 1992).

For example, in the graph coloring problem presented in Fig. 1, it is obvious that region x must be colored with a different color than region y, and so on. Therefore, we can formalize this problem as a CSP, in which there are four variables x, y, z, w each of which corresponds to a regions. The domain of each variable is the set of available colors: red, blue, green. The constraints that force that the adjacent regions must be colored with different colors are $x \neq y$, $x \neq z$, $x \neq w$, $y \neq w$.

CSP (graph coloring problem):

Variables: $\{x, y, z, w\}$;
Domain of each variable: $\{\text{red}, \text{blue}, \text{green}\}$;
Constraints: $x \neq y$, $x \neq z$, $x \neq w$, $y \neq w$.

A solution is a valid combination of values for these variables: $x = \text{red}$, $y = \text{blue}$, $z = \text{blue}$ and $w = \text{green}$.

A DisCSP is a CSP in which the variables and constraints are distributed among automated agents (Yokoo and Hirayama, 2000). Finding a value assignment to variables that satisfies inter-agent constraints can be viewed as achieving coherence or consistency among agents.

The above example of the graph coloring problem can be modelled as a DisCSP, where each variable in the resulting DisCSP is owned by one particular agent who ensures that the variable has a value assigned to it. Thus, agent x owns variable x and so on (see Fig. 2). The actual search for solving the DisCSP can be carried out by a central agent, or in a distributed manner through message exchange among the agents. This approach, which is

* Corresponding author. Tel.: +34 963877007; fax: +34 963877359.
E-mail addresses: msalido@dsic.upv.es (M.A. Salido), agiret@dsic.upv.es (A. Giret).

0952-1976/$ - see front matter
doi:10.1016/j.engappai.2008.03.006

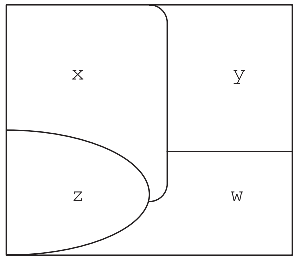

Fig. 1. Example of a constraint satisfaction problem (graph coloring).

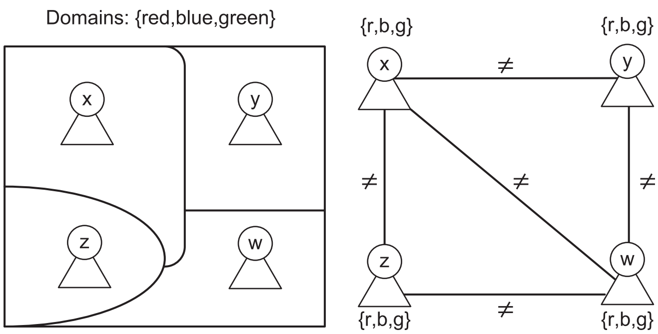

Fig. 2. Example of a distributed constraint satisfaction problem (graph coloring). Variables are nodes and constraints are arcs.

illustrated in Fig. 2, was pioneered by Yokoo et al. (1992, 1998b) in their asynchronous backtracking algorithm. In asynchronous backtracking, the DisCSP is solved by asynchronous message exchange. It assumes a priority ordering among agents (e.g., a unique serial number assigned to each agent) and an agent is responsible for enforcing all constraints between itself and all variables owned by higher agents in this ordering. The problem is solved through an exchange of messages that does not need to be synchronized among agents.

The advantage of distributed search is that an agent only needs to know about agents that own a variable that it has a constraint with, but not about the entire problem. Thus, in the example of Fig. 2, the agent that owns variable *z* only has to communicate with the agents that owns *x* and *w*, but not with the agent that owns *y*. In certain algorithms, it can happen that new connections are created dynamically during search; thus, *z* might become connected to *y*. The approach is most suitable when the overall problem is large, but not very densely connected. However, when the problem is densely connected or the number of variables is very high, the cost of communication during the solving process is likely to become very high, too.

The most cited papers related to DisCSP make the following assumptions for simplicity in describing their algorithms:

(i) Each agent has exactly one variable.
(ii) All constraints are binary.
(iii) Each agent knows all constraint predicates that are relevant to its variable.

Although the great majority of real problems are naturally modelled as non-binary CSPs, the second assumption is comprehensible due to the fact that there exist some techniques that translate any non-binary CSP into an equivalent binary one (Bacchus and van Beek, 1998).

However, the first assumption is too restrictive, and the main basic research focuses on small instances. Also, little work has been done to solve real-life problems.

2. Main features in DisCSPs

In this section, we present the main features that make the use of DisCSPs appropriate. It is well known that if all knowledge about the problem could be gathered into one agent, this agent could solve the problem alone using traditional centralized constraint satisfaction algorithms. However, such a centralized solution is often inadequate or even impossible. Faltings and Yokoo (2005) present some reasons why distributed methods may be desirable:

- The cost of creating a central authority: A CSP may be naturally distributed among a set of agents. In such cases, a central authority for solving the problem would require adding an additional element that was not present in the architecture. Examples of such systems are sensor networks or meeting scheduling.
- The knowledge transfer costs: In many cases, constraints arise from complex decision processes that are internal to an agent and cannot be articulated to a central authority. Examples of this range from simple meeting scheduling, where each participant has complex preferences that are hard to articulate, to coordination decisions in virtual enterprises that result from complex internal planning. A centralized solver would require such constraints to be completely articulated for all possible situations. This would entail prohibitive costs.
- Privacy/security concerns: Agents involve constraints that may represent strategic information that should not be revealed to competitors, or even to a central authority. This situation often arises in many enterprises. Privacy is easier to maintain in distributed solvers.
- Robustness against failure: The failure of the centralized server can be fatal. In a distributed method, a failure of one agent can be less critical and other agents might be able to find a solution without the failed agent. Such concerns arise, for example, in sensor networks, but also in web-based applications where participants may leave while a constraint solving process is ongoing.

These reasons have motivated significant research activity in distributed constraint satisfaction. Up to now, the field has reached a certain maturity and has developed a range of different techniques. Nevertheless, most of the works are focused on developing new techniques which are evaluated using toy problems and random benchmarks.

3. Open issues in DisCSPs

In spite of significant progress, there are many important open issues in DisCSPs. The six main open issues for using DisCSPs are the following:

- While distributed algorithms eliminate the need for a central authority, the currently known algorithms pay a high price in efficiency. In general, the message traffic even for a single agent can be higher than what would be required to communicate the entire problem to a leader agent that could solve it centrally. More research is required to significantly reduce the communication requirements, possibly with radically different algorithms that are better suited for distribution.
- Many DisCSP algorithms assume an agent has enough knowledge to evaluate constraints that are related to its variables. If this is not true, some constraints may still have to be communicated or additional communication may be needed.