Distributed Constraint Reasoning

Marius C. Silaghi
Florida Institute of Technology, USA

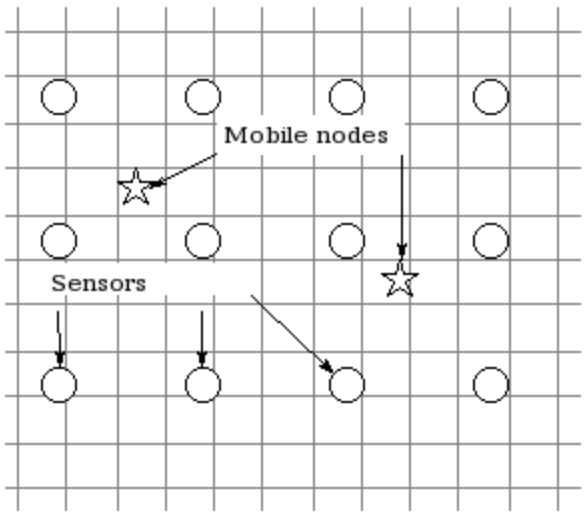
Makoto Yokoo
Kyushu University, Japan

INTRODUCTION

Distributed constraint reasoning is concerned with modeling and solving naturally distributed problems. It has application to the coordination and negotiation between semi-cooperative agents, namely agents that want to achieve a common goal but would not give up private information over secret constraints. When compared to centralized constraint satisfaction (CSP) and constraint optimization (COP), one of the most expensive operations is communication. Other differences stem from new coherence and privacy needs. We review approaches based on asynchronous backtracking and depth-first search spanning trees.

Distributed constraint reasoning started as an outgrowth of research in constraints and multi-agent systems. Take the sensors network problem in Figure 1, defined by a set of geographically distributed sensors that have to track a set of mobile nodes. Each sensor can watch only a subset of its neighborhood at a given time. Three sensors need to simultaneously focus on the same mobile node in order to locate it. Approaches modeling and solving this problem with distributed constraint reasoning are described in (Bejar, Domshlak, Fernandez, Gomes, Krishnamachari, Selman, &Valls, 2005).

Figure 1. Sensor network



There are two large classes of distributed constraint problems. The first class is described by a set of *Boolean relations* (aka *constraints*) on possible assignments of variables, where the relations are distributed among agents. They are called distributed constraint satisfaction problems (DisCSPs). The challenge is to find assignments of variables to values such that all these relations are satisfied. However, the reasoning process has to be performed by collaboration among the agents. There exist several solutions to a problem, and ties have to be broken by some priority scheme. Such priorities may be imposed from the problem description where some agents, such as government agencies, are more important than others. In other problems it is important to ensure that different solutions or participants have equal chances, and this property is called *uniformity*. When no solution exists, one may still want to find an assignment of the variables that conflict as few constraints as possible. The second class of problems refers to numerical optimization described by a set of functions (*weighted constraints*) defined on assignments of variables and returning positive numerical values. The goal is to find assignments that minimize the objective function defined by the sum of these functions. The problems obtained in this way are called distributed constraint optimization problems (DisCOPs). Some problems require a fair distribution of the amount of dissatisfaction among agents, minimizing the dissatisfaction of the most unsatisfied agent.

There are also two different ways of distributing a problem. The first way consists of distributing the data associated with it. It is defined in terms of which agents know which constraints. It can be shown that any

such problem can be translated into problems where all non-shared constraints are *unary* (constraints involving only one variable), also called *domain constraints*. Here one can assume that there exists a single unary constraint for each variable. It is due to the fact that any second unary constraint can be reformulated on a new variable, required to be equal to the original variable. The agent holding the unique domain constraint of a variable is called the *owner of that variable*. Due to the availability of this transformation many solutions focus on the case where only the unary constraints are not shared by everybody (also said to be private to the agents that know them). Another common simplification consists in assuming that each agent has a single unary constraint (i.e., a single variable). This simplification does not reduce the generality of the addressable problems since an agent can participate in a computation under several names, e.g., one instance for each unary constraint of the original agent. Such false identities for an agent are called pseudo-agents (Modi, Shen, Tambe, & Yokoo, 2005), or abstract agents (Silaghi & Faltings, 2005).

The second way of distributing a problem is in terms of who may propose instantiations of a variable. In such an approach each variable may be assigned a value solely by a subset of the agents while the other agents are only allowed to reject the proposed assignment. This distribution is similar to restrictions seen in some societies where only the parliament may propose a referendum while the rest of the citizens can only approve or reject it. Approaches often assume the simultaneous presence of both ways of distributing the problem. They commonly assume that the only agent that can make a proposal on a variable is the agent holding the sole unary constraint on that variable, namely its owner (Yokoo, Durfee, Ishida, & Kuwabara, 1998). When several agents are allowed to propose assignments of a variable, these authorized agents are called *modifiers* of that variable. An example is where each holder of a constraint on a variable is a legitimate modifier of that variable (Silaghi & Faltings, 2005).

BACKGROUND

The first challenge addressed was the development of *asynchronous algorithms* for solving distributed problems. Synchronization forces distributed processes to run at the speed of the slowest link. Algorithms that do not use synchronizations, namely where participants are at no point aware of the current state of other participants, are flexible but more difficult to design. With the exception of a few *solution detection* techniques (Yokoo & Hirayama, 2005), (Silaghi & Faltings, 2005), most approaches gather the answer to the problem by reading the state of agents after the system becomes idle and reaches the so called *quiescence* state (Yokoo et al., 1998). Algorithms that eventually reach quiescence are also called *self-stabilizing* (Collin, Dechter, & Katz, 1991). A *complete* algorithm is an algorithm that guarantees not to miss any existing solution. A *sound* algorithm is a technique that never *terminates* in a suboptimal state.

Another challenge picked by distributed constraint reasoning research consists of providing privacy for the sub-problems known by agents (Yokoo et al., 1998). The object of privacy can be of different types. The *existence of a constraint* between two variables may be secret as well as the *existence of a variable* itself. Many approaches only try to ensure the *secrecy of the constraints*, i.e., the hiding of the identity of the *valuations* that are penalized by that constraint. For optimization problems one also assumes a need to keep secret the amount of the penalty induced by the constraint. As mentioned previously, it is possible to model such problems in a way where all secret constraints are unary (approach known as having *private domains*). Some problems may have both secret and public constraints. Such public constraints may be used for an efficient preprocessing prior to the expensive negotiation implied by secret constraints. Solvers that support guarantees of privacy at any cost employ *cryptographic multi-party computations* (Yao 1982). There exist several cryptographic technologies for such computations, and some of them can be used interchangeably by distributed problem solvers. However, some of them offer information theoretical security guarantees (Shamir, 1979) being resistant to any amount of computation, while others offer only cryptographic security (Cramer, Damgaard, & Nielsen, 2000) and can be broken using large amounts of computation or quantum computers. The result of a computation may reveal secrets itself and its damages can be reduced by being careful in formulating the query to the solver. For example, less information is lost by requesting the solution to be picked randomly than by requesting the first solution. The computations can be done cryptographically by a group of *semi-trusted servers*,