Distributed Constraints for Large-Scale Scheduling Problems*

Student name: Montserrat Abril
Supervisor name: Miguel A. Salido, Federico Barber

Dpto. Sistemas Informáticos y Computación, Universidad Politécnica de Valencia
Camino de Vera s/n, 46022, Valencia, Spain
Student:mabril@dsic.upv.es, Supervisors:{msalido, fbarber}@dsic.upv.es

Abstract. Many problems of theoretical and practical interest can be formulated as Constraint Satisfaction Problems (CSPs). The general CSP is known to be NP-complete; however, distributed models may reduce the exponential complexity by partitioning the problem into a set of subproblems. In this work, we present a distributed model for solving large-scale CSPs in which agents are committed to sets of constraints. Our technique carries out a partition over the constraint network by a graph partitioning software, such as each subproblem is as independent as possible and, it can be solved in a reasonable time. We have focused our research to railway scheduling problem where the resultant CSP maintains thousand of variables and constraints.

keywords: Constraint Satisfaction Problems, graph partitioning, railway scheduling problem.

1 Introduction

Many real problems in Artificial Intelligence (AI) as well as in other areas of computer science and engineering can be efficiently modeled as Constraint Satisfaction Problems (CSPs) and solved using constraint programming techniques. Some examples of such problems include: spatial and temporal planning, qualitative and symbolic reasoning, diagnosis, decision support, scheduling, hardware design and verification, real-time systems and robot planning.

Furthermore, many researchers are working on graph partitioning [5], [2]. The main objective of graph partitioning is to divide the graph into a set of regions such that each region has roughly the same number of nodes and the sum of all edges connecting different regions is minimized. Fortunately, many heuristics may solve this problem efficiently. For instance, graphs with over 14000 nodes and 410000 edges can be partitioned in under 2 seconds [1]. Graph partitioning can also be applied to constraint satisfaction problem. Thus, we can use graph

* This work has been partially supported by the grant TIN2004-06354-C02- 01 (MEC, Spain - FEDER) and GV04B/516 (Generalidad Valenciana, Spain).

partitioning, when dealing with large-scale CSPs, to distribute the problem into a set of sub-CSPs.

Our research is also focused on the railway scheduling problem. Railway traffic has increased considerably, which has created the need to optimize the use of railway infrastructures. This is, however, a hard and difficult task. The scheduling problem can be modeled by a CSP which is composed by thousand of variables and constraints. The overall goal of a long-term collaboration between our group at the Polytechnic University of Valencia (UPV) and the National Network of Spanish Railways (RENFE) is to offer assistance to help in the planning of train scheduling, to obtain conclusions about the maximum capacity of the network, to identify bottlenecks, etc. Due to topological properties of the railway scheduling problem, the resultant CSP can be distributed in semi-independent subproblems such as the solution can be solved easier.

In this work, we propose a distributed model in which the railway scheduling problem is partitioned into a set of subproblems and solved by different search algorithms. The partition is carried out by means of a graph partitioning software called METIS [3]. METIS provides two programs *pmetis* and *kmetis* for partitioning an unstructured graph into k equal size parts. In this way, the constraint partition is carried out in a preprocessing step in which an agent called *partition agent* carries out a partition of the original problem in semi-independent subproblems. Thus, the problem is partitioned in k blocks in order to be studied by agents called *block agents*. Furthermore, the partition agent is committed to classify the subproblems such as the most interrelated problem is studied first.

In the following section, we summarize some definitions. In section 3, we present distributed model. In section 4, we apply this model to railway scheduling problem. A preliminary evaluation is carried out in section 5. Finally we summarizes the conclusions and future work in section 6.

2 Definitions

In this section, we review some basic definitions as well as basic heuristics for CSPs.

State: one possible assignment of all variables; the number of states is equal to the product of the domain size.

Partition : A partition of a set C is a set of disjoint subsets of C whose union is C. The subsets are called the blocks of the partition.

Distributed CSP: A distributed CSP is a CSP in which the variables and constraints are distributed among automated agents [7].

Each agent has some variables and attempts to determine their values. However, there are interagent constraints and the value assignment must satisfy these interagent constraints. In our model, there are k agents $1, 2, ..., k$. Each agent knows a set of constraints and the domains of variables involved in these constraints.

Definition 1: A *block agent* a_j is a virtual entity that essentially has the following properties: autonomy, social ability, reactivity and pro-activity [6].

Block agents are autonomous agents. They operate their subproblems without the direct intervention of any other agent or human. *Block agents* interact with each other by sending messages to communicate consistent partial states. They perceive their environment and changes in it, such as new partial consistent states, and react, if possible, with more complete consistent partial states.

Definition 2: A *multi-agent system* is a system that contains the following elements:

1. An environment in which the agents live (variables, domains, constraints and consistent partial states).
2. A set of reactive rules, governing the interaction between the agents and their environment (agent exchange rules, communication rules, etc).
3. A set of agents, $A = \{a_1, a_2, ..., a_k\}$.

3 The Distributed Model

In the specialized literature, there are many works about distributed CSPs. In [7], Yokoo et al. present a formalization and algorithms for solving distributed CSPs. These algorithms can be classified as either centralized methods, synchronous backtracking or asynchronous backtracking [7].

Our model can be considered as a synchronous model. It is meant to be a framework for interacting agents to achieve a consistent state. The main idea of our multi-agent model is based on [4] but partitioning the problem in k subproblems as independent as possible, classifying the subproblem in the appropriate order and solving them concurrently.

Once the constraints are divided into k blocks by a *preprocessing agent*, a group of *block agents* concurrently manages each block of constraints. Each *block agent* is in charge of solving its own subproblem by means of a search algorithm. Each *block agent* is free to select any algorithm to find a consistent partial state. It can select a local search algorithm, a backtracking-based algorithm, or any other, depending on the problem topology. In any case, each *block agent* is committed to finding a solution to its particular subproblem. This subproblem is composed by its CSP subject to the variable assignment generated by the previous *block agents*. Thus, *block agent* 1 works on its group of constraints. If *block agent* 1 finds a solution to its subproblem, then it sends the consistent partial state to *block agent* 2, and both they work concurrently to solve their specific subproblems; *block agent* 1 tries to find other solution and *block agent* 2 tries to solve its subproblem knowing that its *common variables* have been assigned by *block agent* 1. Thus, *block agent* j, with the variable assignments generated by the previous *block agents*, works concurrently with the previous *block agents*, and tries to find a more complete consistent state using a search

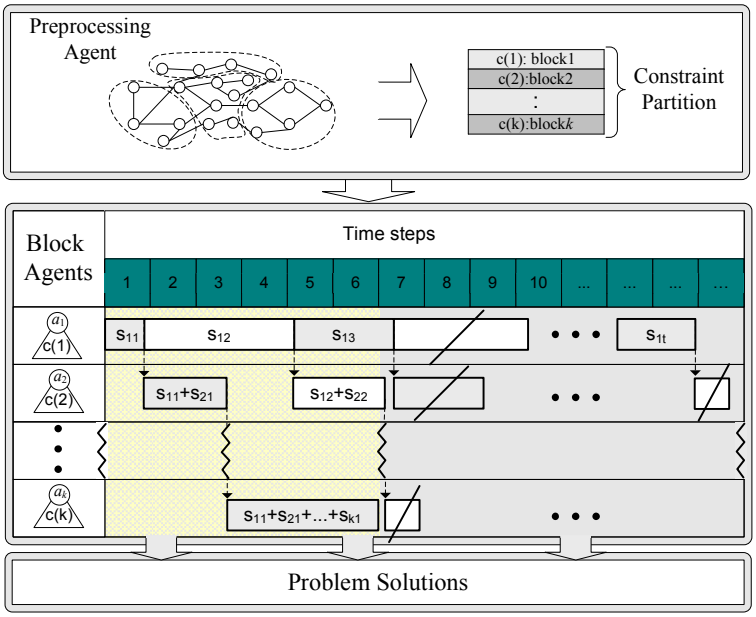

Fig. 1. Multi-agent model.

algorithm. Finally, the last *block agent* k, working concurrently with *block agents* $1, 2, ...(k-1)$, tries to find a consistent state in order to find a problem solution.

Figure 1 shows the multi-agent model, in which the *preprocessing agent* carries out the network partition and the *block agents* (a_i) are committed to concurrently finding partial problem solutions (s_{ij}). Each *block agent* sends the partial problem solutions to the following *block agent* until a problem solution is found (by the last *block agent*). For example, state: $s_{11} + s_{21} + ... + s_{k1}$ is a problem solution. The concurrence can be seen in Figure 1 in *Time step* 6 in which all *block agents* are concurrently working. Each *block agent* maintains the corresponding domains for its *new variables*. The *block agent* must assign values to its *new variables* so that the block of constraints is satisfied. When a *block agent* finds a value for each *new variable*, it then sends the consistent partial state to the next *block agent*. When the last *block agent* assigns values to its *new variables* satisfying its block of constraints, then a solution is found.

4 Application to Railway Scheduling Problem

Our distributed model is being evaluated to the railway scheduling problem. This scheduling problem can be modeled by a CSP which is composed by thousand of variables and thousand of constraints. It is very hard to solve as an entire CSP. However, if the scheduling problem is distributed, it can be solved easier. To this end, and depending on the desired solution, the preprocessing agent carries out

the partition by means of the software called METIS, or it divides the running map in clusters composed by contiguous stations.

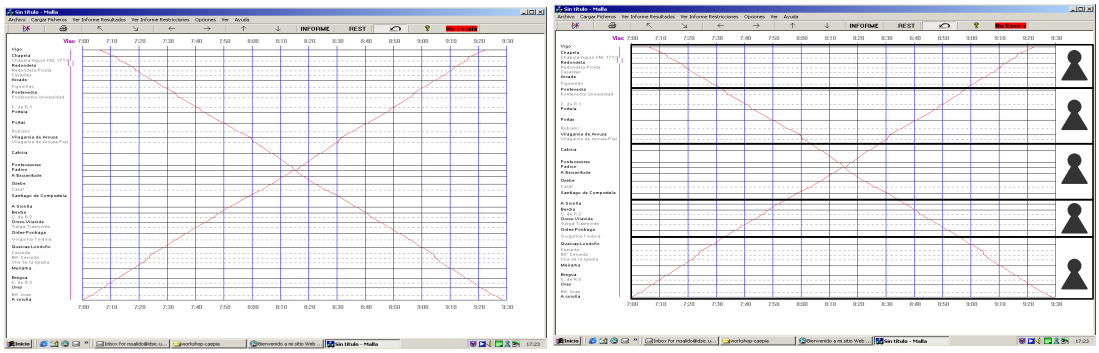

Fig. 2. Distributed Railway Scheduling Problem.

Thus, the running map to be scheduled between two cities is decomposed in several and shorter running maps. Figure 2 (left) shows a running map to be scheduled. The set of stations will be partitioned in block of contiguous stations and a set of agents will coordinate to achieve a global solution (Figure 2 (right)). Thus, we can obtain important results such as railway capacity, consistent timetable, etc.

5 Preliminary Results

In this section, we carry out an evaluation between our distributed model and a centralized model. To this end, we have used a well-known CSP solver called Forward Checking (FC)[1].

In our evaluation, each set of random CSPs was defined by the 3-tuple $< n, a, p >$, where n was the number of variables, a the arity of binary constraints and p the number of partitions. The problems were randomly generated by modifying these parameters.

In Table 1 we compare the running time of the distributed model with the centralized problem. We fixed the arity of binary constraints and the size of the partition, and the number of variables was increased from 50 to 500. We can observe that the running time for small problems was worse by the distributed model than the centralized problem. However, when the number of variables increased, the behavior of the distributed problem was better. It also depends on the size of the partition. For small problems, the number of partition must be low. However, for large CSPs (railway Scheduling Problems) the size of the partition must be higher.

[1] Forward Checking were obtained from CON'FLEX. It can be found in: http://www-bia.inra.fr/T/conflex/ Logiciels/adressesConflex.html.

Table 1. Random instances $< n, a, p >$, n:variables, a:arity and p:partition size.

Variables	*Arity*	*Partition Size*	*Distributed Model*	*Centralized Model*
50	25	10	12	3
100	25	10	12	14
150	25	10	15	37
200	25	10	16	75
250	25	10	17	98
300	25	10	19	140
350	25	10	23	217
400	25	10	30	327
450	25	10	32	440
500	25	10	42	532

6 Conclusion and Future work

In this paper, we present a distributed model for solving large-scale CSPs, in which a *preprocessing agent* is committed to partitioning the constraint network in semi-independent sub-CSPs. Then, a set of *block agents* are incrementally and concurrently committed to building partial solutions until a global solution is found. Thus, hard problems can be solved more efficiently. We are working on applying this model to railway scheduling problem, where there exist thousand of variables and constraints.

References

1. G. Karypis and V. Kumar, 'Using METIS and parMETIS', Technical report, (1995).
2. G. Karypis and V. Kumar, 'A parallel algorithm for multilevel graph partitioning and sparse matrix ordering', *Journal of Parallel and Distributed Computing*, **48**(1), 71–95, (1998).
3. METIS, *http://www-users.cs.umn.edu/ karypis/metis/index.html.*
4. M.A. Salido, A. Giret, and F. Barber, 'Distributing Constraints by Sampling in Non-Binary CSPs', *In IJCAI Workshop on Distributing Constraint Reasoning*, 79–87, (2003).
5. K. Schloegel, G. Karypis, and V. Kumar, 'Graph partitioning for high-performance scientific simulations', *Sourcebook of parallel computing*, 491–541, (2003).
6. M. Wooldridge and R. Jennings, 'Agent theories, arquitectures, and lenguajes: a survey', *Intelligent Agents*, 1–22, (1995).
7. M. Yokoo and K. Hirayama, 'Algorithms for distributed constraint satisfaction: A review', *Autonomous Agents and Multi-Agent Systems*, **3**, 185–207, (2000).