Domain-Dependent Distributed Models for Railway Scheduling*

M.A. Salido[1], M. Abril[1], F. Barber[1], L. Ingolotti[1], P. Tormos[2], A. Lova[2]

[1] *DSIC, Universidad Politécnica de Valencia,Spain*

[2] *DEIOAC, Universidad Politécnica de Valencia,Spain*

{*msalido, mabril, fbarber, lingolotti*}*@dsic.upv.es*

{*ptormos, allova*}*@eio.upv.es*

Abstract

Many combinatorial problems can be modelled as Constraint Satisfaction Problems (CSPs). Solving a general CSP is known to be NP-complete, so closure and heuristic search are usually used. However, many problems are inherently distributed and the problem complexity can be reduced by dividing the problem into a set of subproblems. Nevertheless, general distributed techniques are not always appropriate to distribute real-life problems. In this work, we model the railway scheduling problem by means of domain-dependent distributed constraint models, and we show that these models maintained better behaviors than general distributed models based on graph partitioning. The evaluation is focused on the railway scheduling problem, where domain-dependent models carry out a problem distribution by means of trains and contiguous sets of stations.

Key words: Constraint Satisfaction Problems, Railway Scheduling.

1 Introduction.

Many real problems in Artificial Intelligence (AI) as well as in other areas of computer science and engineering can be efficiently modelled as Constraint

* This work has been partially supported by the research projects TIN2004-06354-C02-01 (Min. de Educacin y Ciencia, Spain-FEDER), FOM-70022/T05 (Min. de Fomento, Spain) and by the Future and Emerging Technologies Unit of EC (IST priority - 6th FP), under contract no. FP6-021235-2 (project ARRIVAL).

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.

Most of the work is focused on general methods for solving CSPs. They include backtracking-based search algorithms. While the worst-case complexity of backtrack search is exponential, several heuristics to reduce its average-case complexity have been proposed in the literature [2]. However, many of the problems solved by using centralized algorithms are inherently distributed. Some works are currently based on distributed CSPs (see the special issue of Artificial Intelligence, Volume 161).

Furthermore, many researchers are working on graph partitioning [3], [5]. The main objective of graph partitioning is to divide the graph into a set of regions so that each region has roughly the same number of nodes and the sum of all edges connecting different regions is minimized. Graph partitioning can be applied to telephone network design, sparse gaussian elimination, data mining, clustering and physical mapping of DNA. Fortunately, there are many heuristics that can solve this problem efficiently. For instance, graphs with over 14000 nodes and 410000 edges can be partitioned in under 2 seconds [4].

The partition of a CSP can be performed in a general (domain-independent) way by using graph partitioning techniques (see Figure 1 (2)). For instance, a CSP can be divided into several subCSPs so that the constraints among variables of each subCSP are minimized. Otherwise, a domain-dependent partition can be used (see Figure 1 (3)). This requires a deeper analysis of the problem to be solved. This paper shows that in several problems, a domain-dependent partition obtains a more adequate distribution so that greater efficiency is obtained. Moreover, cooperation among the distributed CSP solvers can also be improved.

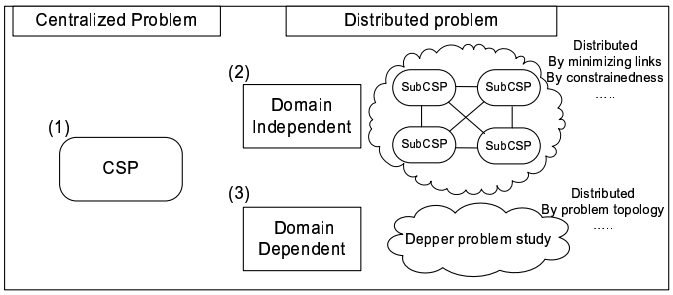

Fig. 1. Centralized&Distributed Problem.

A general graph partitioning software called METIS [4] can be applied to compare its behavior in the CSP environment with a domain-dependent distributed model.

Our aim is to model the railway scheduling problem as a Constraint Satisfaction Problem (CSP) and solve it by using different models of distributed CSPs. Due to the deregulation of European railway operators, long transnational journeys must be scheduled. These journeys involve many stations in different countries with different railway policies. For instance, to travel from London to Madrid, the train must cross three different countries, each of which has different characteristics and different policies. Furthermore, each railway operator maintains strategic/private information that should not be revealed to competitors. These circumstances have motivated us to distribute the railway scheduling problem.

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 in the planning of train scheduling to obtain conclusions about the maximum capacity of the network, to identify bottlenecks, etc.

In this work, we compare several ways to distribute the railway scheduling problem. It is partitioned into a set of subproblems by graph partitioning, types of trains and contiguous stations.

In the following section, we summarize some definitions about distributed CSPs and briefly describe our distributed model. Section 3 summarizes the railway scheduling problem. Section 4 presents different partition proposals. Section 5 presents an evaluation among different models. Finally, section 6 presents our conclusions.

2 Distributed CSPs.

This section presents numeric CSPs in a slightly non-standard form, which will help to unify works from different constraint satisfaction communities. This section also presents our distributed model.

2.1 Definitions.

A ***CSP*** consists of: a set of variables $X = \{x_1, x_2, ..., x_n\}$; each variable $x_i \in X$ has a set D_i of possible values (its domain); a finite collection of constraints $C = \{c_1, c_2, ..., c_p\}$ restricting 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*.

State: one possible assignment of all variables.

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 (DCSP) is a CSP in which the variables and constraints are distributed among automated agents [12].

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.

A block agent a_j is a virtual entity that essentially has the following properties: autonomy, social ability, reactivity and pro-activity [11].

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. For example, new partial consistent states can be extended by *block agents* to more complete consistent partial states.

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, $A = \{a_1, a_2, ..., a_k\}$.

2.2 The Distributed Model.

In the specialized literature, there are many works about distributed CSPs. In [12], 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 [12].

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 [6], but it partitions the problem into k subproblems as independent as possible, classifies the subproblems into the appropriate order, and solves them concurrently.

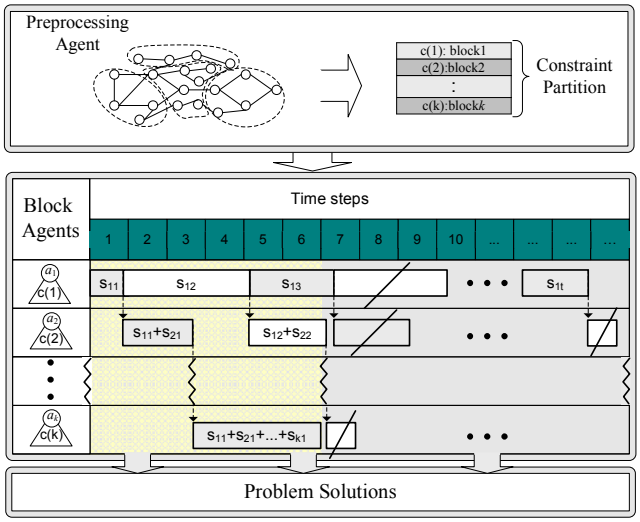

Fig. 2. Multi-agent model.

In all our proposals, the problem is partitioned into k blocks or clusters in order to be studied by agents called *block agents*. Furthermore, a partition agent is committed to classifying the subproblems in the appropriate order depending on the selected proposal. For instance, if Metis is selected to partition the problem, the partition agent must classify the subproblems so that the most interrelated problem is studied first.

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 algorithm, 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 of 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 they both work concurrently to solve their specific subproblems; *block agent* 1 tries to find another solution and *block agent* 2 tries to solve its subproblem knowing that its *common variables* have been assigned by *block agent* 1. Thus, using the variable assignments generated by the previous *block agents*, *block agent* j works concurrently with the previous *block agents* and tries to find a more complete consistent state using a search 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 2 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 2 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*. A solution is found when the last *block agent* assigns values to its *new variables* satisfying its block of constraints.

3 The Railway Scheduling Problem.

Train timetabling is a difficult and time-consuming task, particularly in the case of real networks, where the number and complexity of constraints grow drastically. A feasible train timetable should specify the departure and arrival time of each train to each location of its journey in such a way that the line capacity and other operational constraints are taken into account. Traditionally, train timetables are generated manually by drawing trains on the time-distance graph. The train schedule is generated from a given starting time and is manually adjusted so that all constraints are met. High priority trains are usually placed first followed by lower priority trains. It can take many days to develop train timetables for a line, and the process usually stops once a feasible timetable has been found. The resulting plan of this procedure may be far from optimal.

The literature of the 1960s, 1970s, and 1980s relating to rail optimization was relatively limited. Compared to the airline and bus industries, optimization was generally overlooked in favor of simulation or heuristic-based methods. However, Cordeau et al. [1] point out greater competition, privatization, deregulation, and increasing computer speed as reasons for the more prevalent use of optimization techniques in the railway industry. Our review of the methods and models that have been published indicates that the majority of authors use models that are based on the Periodic Event Scheduling Problem (PESP) introduced by Serafini and Ukovich [8]. The PESP considers the problem of scheduling as a set of periodically recurring events under periodic time-window constraints. The model generates disjunctive constraints that may cause the exponential growth of the computational complexity of

the problem depending on its size. Schrijver and Steenbeek [7] have developed CADANS, a constraint programming- based algorithm to find a feasible timetable for a set of PESP constraints. The scenario considered by this tool is different from the scenario that we used; therefore, the results are not easily comparable. The train scheduling problem can also be modeled as a special case of the job-shop scheduling problem (Silva de Oliveira [9], Walker et al. [10]), where train trips are considered jobs that are scheduled on tracks that are regarded as resources.

Our goal is to model the railway scheduling problem as a Constraint Satisfaction Problem (CSP) and solve it using constraint programming techniques. However, due to the huge number of variables and constraints that this problem generates, a distributed model is developed to distribute the resultant CSP into a semi-independent subproblems so that the solution can be found efficiently.

3.1 Constraints in the Railway Scheduling Problem

There are three groups of scheduling rules in our railway scheduling problem: traffic rules, user requirements rules and topological rules. A valid running map must satisfy the above rules. These scheduling rules can be modelled using the following constraints:

(1) **Traffic rules** guarantee crossing and overtaking operations. The main constraints to take into account are:

- ***Crossing constraint***: Any two trains (T_i and T_j) going in opposite directions must not simultaneously use the same one-way track. That is, train i arrives to station A before train j departs from station A ($T_iA_A < T_jD_A$), or train j arrives to station B before train i departs from station B ($T_jA_B < T_iD_B$).

$$T_iA_A < T_jD_A \text{ or } T_jA_B < T_iD_B$$

  The crossing of two trains can be performed only on two-way tracks and at stations where one of the two trains has been detoured from the main track (Figure 3).

- ***Overtaking constraint***: Any two trains (T_i and T_j) going at different speeds in the same direction can only overtake each other at stations.

$$T_iD_A < T_jD_A \rightarrow T_iA_B < T_jA_B$$

  The train being passed is detoured from the main track so that the faster train can pass the slower one (see Figure 3).

- ***Expedition time constraint***. There exists a given time to put a detoured train back on the main track and exit from a station (see Figure 3 right).

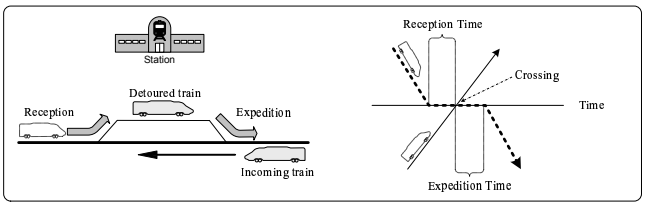

Fig. 3. Constraints related to crossing and overtaking in stations

- ***Reception time constraint***. There exists a given time to detour a train from the main track so that crossing or overtaking can be performed (see Figure 3 right).

(2) **User Requirements**: The constraints due to user requirements are:
- ***Type and Number of trains*** going in each direction to be scheduled.
- ***Path of trains***: Locations used and ***Stop time*** for commercial purposes in each direction.
- ***Scheduling frequency***. The departure of trains must satisfy frequency requirements in both directions. This constraint is very restrictive because, when crossings are performed, trains must wait for a certain time interval at stations. This interval must be propagated to all trains going in the same direction in order to maintain the established scheduling frequency. The user can require a fixed frequency, a frequency within a minimum and maximum interval, or multiple frequencies.
- ***Departure interval*** for the departure of the first trains going in both the up and down directions.
- ***Maximum slack***. This is the maximum percentage δ that a train may delay with respect to the minimum journey time.

(3) **Railway infrastructure Topology and type of trains** to be scheduled give rise to other constraints to be taken into account. Some of them are:
- Number of ***tracks in stations*** (to perform technical and/or commercial operations) and the number of tracks between two locations (one-way or two-way). No crossing or overtaking is allowed on a one-way track,
- ***Time constraints***, between each two contiguous stations,
- Added ***Station time constraints*** for technical and/or commercial purposes.

The complete set of constraints, including an objective function, transform the CSP into a constraint satisfaction and optimization problem (CSOP), where the main objective function is to minimize the journey time of all trains. Variables are frequencies, arrival and departure times of trains at stations, and binary auxiliary variables generated for modelling disjunctive constraints. Constraints are composed of user requirements, traffic rules, and topological constraints.

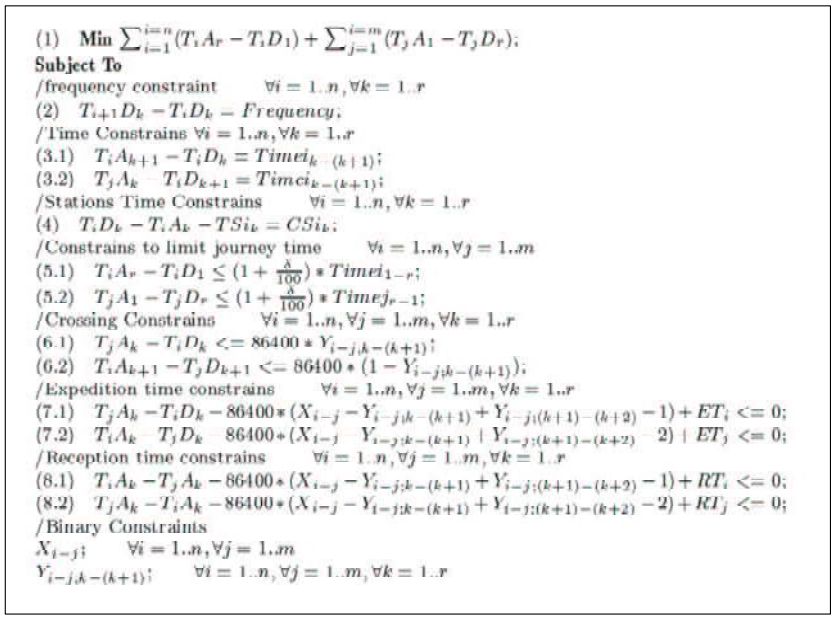

Fig. 4. Formal Mathematical Model of the Railway Scheduling Problem..

The complete CSOP is presented in Figure 4. Let's suppose a railway network with r stations, n trains running in the down direction, and m trains running in the up direction. We assume that two connected stations have only one line connecting them. T_iA_k represents that train i arrives at station k; T_iD_k means that train i departs from station k; $Timei_{k-(k+1)}$ is the journey time of train i to travel from station k to $k+1$; TSi_k and CSi_k represent the technical and commercial stop times of train i in station k, respectively; and ET_i and RT_i are the expedition and reception time of train i, respectively.

4 Partition Proposals

4.1 Domain-Independent: Partition Proposal 1.

A natural way to distribute the problem is carried out by means of a graph partitioning software called METIS [4]. METIS provides two programs, *pmetis* and *kmetis*, for partitioning an unstructured graph into k equal size parts. In this way, the railway scheduling problem can be modelled as a constraint network. This network can be partitioned into semi-independent subproblems by means of METIS. However, this software does not take into account additional information about the railway infrastructure or the type of trains to guide the partition, so the generated clusters may not be the most appropriate and therefore, the results are not appropriate either. Figure 5 shows the distribution carried out by METIS. Red agent is committed to assigning variables

to train 0 and train 1 at the beginning and ending of their journeys. Green agent studies two trains in disjoint part of their journeys. METIS carries out a partition of the constraint network generated by the corresponding CSP. However, as can observed visually, the best partition generated by a well-known software (METIS) is not the most appropriate for this problem. To improve the partition procedure, we extract additional information from the railway topology to obtain better partitions such as the partition proposals 2 and 3.

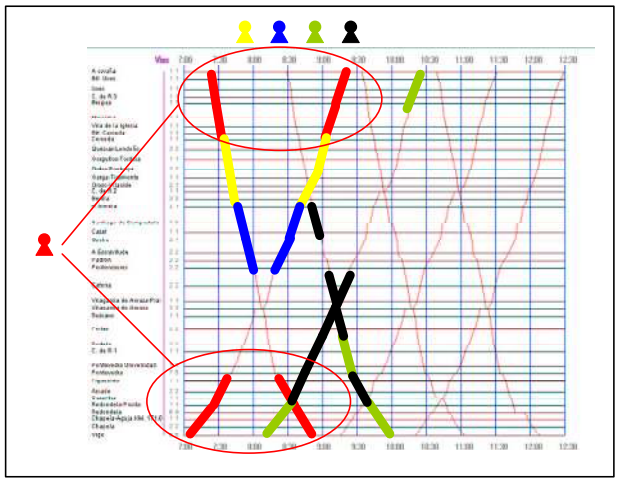

Fig. 5. Distributed Railway Scheduling Problem: Proposal 1.

4.2 Domain-Dependent: Partition Proposal 2.

This model extracts additional information from the railway topology to obtain a better partition by distributing the original problem by means of train type. The problem can be considered as a scheduling problem where trains are tasks, so the partition is carried out by trains (task-driven). Each agent is committed to assigning values to variables related to a train or trains in order to minimize the journey. Depending on the selected number of partitions, each agent will manage one o more trains. Figure 6 left, shows a running map with 20 partition, where each agent manages one train. This partition model has two important advantages. First, this model allow us to improve privacy. Currently, due to the policy of deregulation in the European railways, trains from different operators work on the same railway infrastructure. Thus, the partition model gives us the possibility of partitioning the problem so that each agent is committed to an operator. Different operators can maintain privacy about strategic data. Secondly, this model allows us to efficiently manage priorities between different types of trains (regional trains, high-speed trains, freight trains). In this way, agents committed to priority trains (high-speed trains) will first carry out value assignments to variables in order to achieve better journeys.

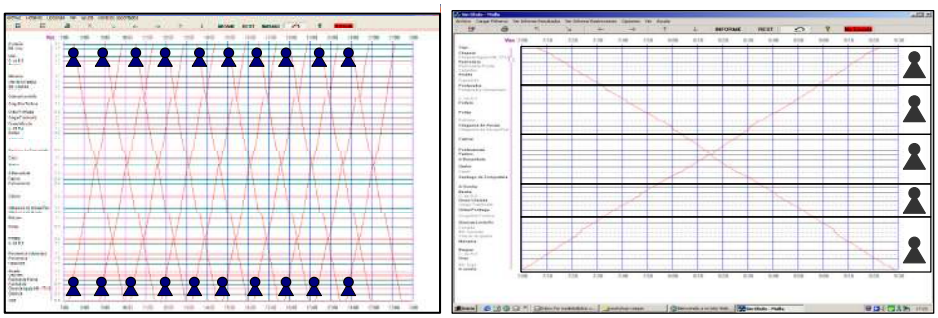

Fig. 6. Distributed Railway Scheduling Problem. Proposal 2 (left) and proposal 3 (right).

4.3 Domain-Dependent: Partition Proposal 3.

Partition proposal 3 is based on distributing the original railway problem by means of contiguous stations. The problem can also be considered as a scheduling problem where stations are resources and the partition is carried out by stations (resource-driven). Due to the deregulation of European railways operators, long journeys can be scheduled. However, long journeys involve a large number of stations in different countries with different railway policies. Therefore, a logical partition of the railway network could be carried out by regions (contiguous stations). To carry out this type of partition, it is important to analyze the railway infrastructure and detect restricted regions (bottlenecks). To balance the problem, each agent is committed to a different number of stations. An agent can manage many stations if they are not restricted stations, whereas an agent can manage only a few stations if they are bottlenecks. Furthermore, the agents with bottlenecks have preference to assign values to variables due to the fact that their domains are reduced (variable ordering).

Thus, the running map to be scheduled between two cities is decomposed into many shorter running maps. The set of stations is partitioned into blocks of contiguous stations, and the agents cooperate with each other to achieve a global solution (Figure 6 (right)). Thus, important results can be obtained such as railway capacity, consistent timetable, etc.

5 Evaluation.

In this section, we present an evaluation between our distributed model and a centralized model. Furthermore, we evaluate the behavior of a distributed model generated by a general software program called METIS and two proposed partition models. To this end, we have used a well-known CSP solver

called Forward Checking (FC) [1] .

This empirical evaluation was carried out with two different types of problems: random problems and benchmark problems.

Random problems.

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.

Table 1 compares the running time of the model distributed by METIS with the centralized model. In Table 1 left, the arity of binary constraints and the size of the partition were fixed, and the number of variables was increased from 100 to 500. The table shows that the running time for small problems was worse for the distributed model than for the centralized model. However, when the number of variables increased, the behavior of the distributed problem was better. In Table 1 right, the number of variables and the arity of binary constraints were fixed, and the size of the partition was increased from 3 to 20. The table shows that the size of the partition is important for the distributed model. For small problems, the number of partitions must be low. However, for large CSPs (railway Scheduling Problems) the size of the partition must be higher. In this case, the appropriate number of partitions was 7.

As can be observed, the distributed model using METIS works well for random instances. However, in real scheduling problems, domain-dependent distributed models are necessary to optimize running times.

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

Problem	*Distributed Model (sc.)*	*Centralized Model (sc.)*	*Problem*	*Distributed Model (sc.)*	*Centralized Model (sc.)*
$< 100, 25, 10 >$	12	14	$< 200, 25, 3 >$	26	75
$< 200, 25, 10 >$	16	75	$< 200, 25, 5 >$	19	75
$< 300, 25, 10 >$	19	140	$< 200, 25, 7 >$	14	75
$< 400, 25, 10 >$	30	327	$< 200, 25, 9 >$	16	75
$< 500, 25, 10 >$	42	532	$< 200, 25, 20 >$	22	75

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

Benchmark problems.

This empirical evaluation was carried out over a real railway infrastructure that joins two important Spanish cities (La Coruña and Vigo). The journey between these two cities is currently divided by 40 stations. In our empirical evaluation, each set of random instances was defined by the 3-tuple $< n, s, f >$, where n was the number of periodic trains in each direction, s the number of stations, and f the frequency. The problems were randomly generated by modifying these parameters.

General graph partitioning software programs work well in general graphs. However, in the railway scheduling problem, we did not obtain good results using these programs. We evaluated partition proposal 1 by using METIS in several instances $< n, 20, 120 >$. Figure 8 (left), shows that the obtained results were even worse in the distributed model than in the centralized model. For a low number of trains, the behavior was better than the complete model. However, with more than 8 trains, the distributed model was unable to solve the problem in 1,000,000 seconds, so the program was aborted. We studied the partitions generated by METIS and we observed that the journey of a train was partitioned in several clusters, and that each cluster was composed by tracks of trains in opposite directions. This cluster was easy to solve but it was very difficult to propagate to other agents. In contrast, partition proposals 2 and 3 never join tracks of trains in opposite directions.

Figure 7 shows the behavior of partition proposal 2, where the number of partition/agents was equal to the number of trains. In both graphs, it can be observed that the running time increased when the number of trains increased (Figure 7 right) and when the number of stations increased (Figure 7 left). However, in both graphs, the distributed model maintained better behavior than the centralized model.

Partition proposal 2 (distributed by trains) was similar to partition proposal 3 (distributed by stations) but had better behavior, mainly when the number of trains increased (see Figure 8 right). However, partition proposal 3 maintained a uniform behavior.

6 Conclusions.

Distributed CSPs are a promising area of research, where graph partitioning techniques can be applied. However, our study shows that domain-dependent distributed constraint models maintain better behaviors than general distributed models. This study focuses on the railway scheduling problem, where

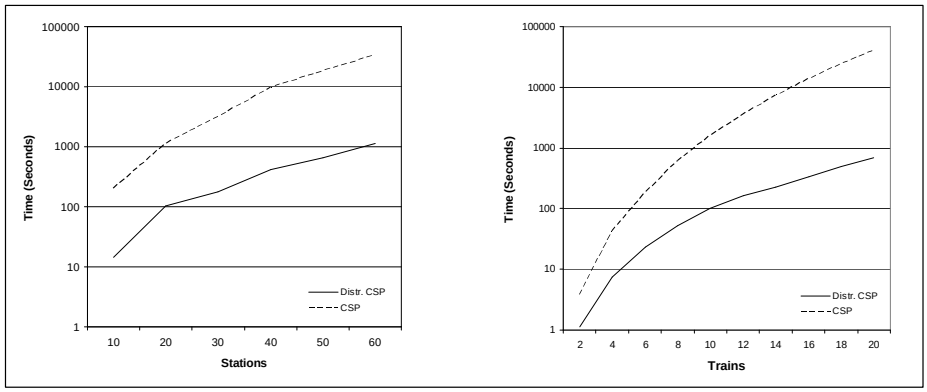

Fig. 7. Running Time when the number of trains and stations increased (proposal 2).

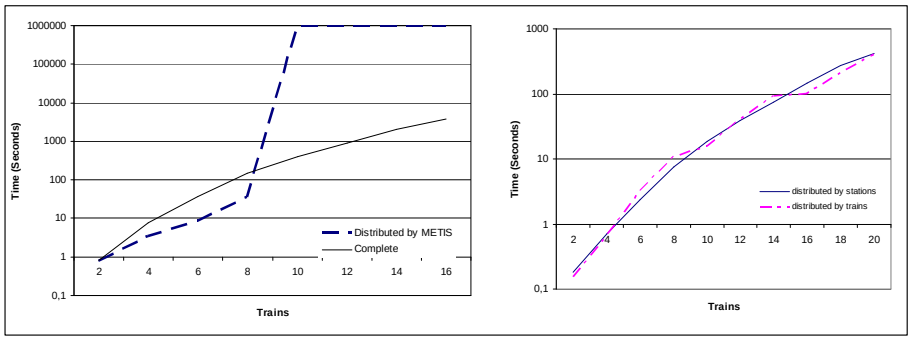

Fig. 8. Running Time using Metis and Complete (left). Comparison of proposals 2 and 3 (right).

specific railway characteristics make the problem domain-dependent. Each instance depends on the number of trains in each direction, the number of stations, the distance between stations, etc. Our evaluation shows that general distributed models had a better behavior than the centralized model, but it also shows that domain-dependent distributed models are more efficient than general ones. Our domain-dependent model makes a distribution that is based on trains (which can be assimilated to the tasks to be done) and on stations (which can be assimilated to resources). Research on models for domain-based distribution is a promising area of research in distributed CSPs, which can also be applied to a distributed architecture where each subCSP collaborated to find a global solution.

References

[1] J. Cordeau, P. Toth, and D. Vigo, 'A survey of optimization models for train routing and scheduling', *Transportation Science*, **32**, 380–446, (1998).

[2] R. Dechter and J. Pearl, 'Network-based heuristics for constraint satisfaction problems', *Artificial Intelligence*, **34**, 1–38, (1988).

[3] B. Hendrickson and R.W. Leland, 'A multi-level algorithm for partitioning graphs', in *Supercomputing*, (1995).

[4] G. Karypis and V. Kumar, 'Using METIS and parMETIS', (1995).

[5] G. Karypis and V. Kumar, 'A parallel algorithm for multilevel graph partitioning and sparse matrix ordering', *Journal of Parallel and Distributed Computing*, 71–95, (1998).

[6] M.A. Salido, A. Giret, and F. Barber, 'Distributing Constraints by Sampling in Non-Binary CSPs', *Distributed Constraint Problem Solving and Reasoning in Multi-agent Systems. Frontiers in Artificial Intelligence and Applications*, **112**, 77–91, (2004).

[7] A. Schrijver and A. Steenbeek, 'Timetable construction for railned', *Technical Report, CWI, Amsterdam, The Netherlands*, (1994).

[8] P. Serafini and W. Ukovich, 'A mathematical model for periodic scheduling problems', *SIAM Journal on Discrete Mathematics*, 550–581, (1989).

[9] E. Silva de Oliveira, 'Solving single-track railway scheduling problem using constraint programming', *Phd Thesis. Univ. of Leeds, School of Computing*, (2001).

[10] C. Walker, J. Snowdon, and D. Ryan, 'Simultaneous disruption recovery of a train timetable and crew roster in real time', *Comput. Oper. Res*, 2077–2094, (2005).

[11] M. Wooldridge and R. Jennings, 'Agent theories, arquitectures, and lenguages: a survey', *Intelligent Agents*, 1–22, (1995).

[12] M. Yokoo and K. Hirayama, 'Algorithms for distributed constraint satisfaction: A review', *Autonomous Agents and Multi-Agent Systems*, **3**, 185–207, (2000).