Distributed Constraint Satisfaction Problems to Model Railway Scheduling Problems

M. Abril, M. A. Salido, F. Barber, L. Ingolotti, P. Tormos, A. Lova
Dpto. Sistemas Informáticos y Computación
Camino de Vera s/n
46071, Valencia, Spain
{mabril, msalido, fbarber, lingolotti}@dsic.upv.es
{ptormos, allova}@deio.upv.es

Abstract

Railway Scheduling is considered to be a difficult and time-consuming task. This is due to real railway networks can be modelled as Constraint Satisfaction Problems (CSPs), but they require a huge number of variables and constraints. The general CSP is known to be NP-complete; however, distributed models may reduce the exponential complexity by dividing the problem into a set of subproblems. In this work, we present several proposals to distribute the railway scheduling problem into a set of sub-problems as independent as possible. The first technique carries out a partition over the constraint network, meanwhile the second distributes the problem by trains and the third technique divides the problem by means of contiguous stations.

Introduction

Train timetabling is a difficult and time-consuming task, particularly in the case of real networks, where the number of constraints and the 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. (Cordeau, Toth, & Vigo 1998) 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 (Serafini & Ukovich 1989). 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 (Schrijver & Steenbeek 1994) 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. Nachtigall and Voget (Nachtigall & Voget 1997) also use PESP constraints to model the cyclic behavior of timetables and to consider the minimization of passenger waiting times as the objective function. Their solving procedure starts with a solution that is obtained in a way similar to the one that timetable designers in railway companies use. This initial timetable is then improved by using a genetic algorithm. In our problem, the waiting time for connections is not taken into account because we only consider the timetabling optimization for a single railway line. The train scheduling problem can also be modeled as a special case of the job-shop scheduling problem (Silva de Oliveira (Silva de Oliveira 2001), Walker et al. (Walker & Ryan 2005)), where train trips are considered as jobs that are scheduled on tracks that are regarded as resources. The majority of these works consider the scheduling of new trains on an empty network. However, railway companies usually also require the optimization of new trains on a line where many trains are already in circulation (that is, trains that have a fixed timetable). With this main objective, Lova et al. (Lova *et al.* 2006) propose a scheduling method based on reference stations where the priority of trains, in the case of conflict, changes from one iteration to another during the solving process.

Our goal is to model the railway scheduling problem as a Constraint Satisfaction Problems (CSPs) 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 such as the solution can be found efficiently.

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.

In parallel computing, many researchers are working on graph partitioning (Schloegel, Karypis, & Kumar 2003), (Karypis & Kumar 1998). The main objective of these techniques 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 (Karypis & Kumar 1995). Graph partitioning can also be applied to constraint satisfaction problem. Thus, we can use ideas about graph partitioning, when dealing with railway scheduling problem, to distribute the problem into a set of sub-problems.

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

In the following section, we summarize some definitions. In section 3, we study three models to distribute the railway scheduling problem. In section 4, we present the distributed model to be solved by the DCSP. An evaluation among different models is carried out in section 5. Finally we summarizes the conclusions and future work in section 6.

Definitions

This section presents CSPs in a slightly non-standard form, which will be convenient for our purposes, and will unify works from constraint satisfaction communities.

Definition 1: 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 is *satisfiable* or *consistent* when it has a solution at least.

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.

A ***running map*** : contains information regarding railway topology (stations, tracks, distances between stations, traffic control features, etc.) and the schedules of the trains that use this topology (arrival and departure times of trains at each station, frequency, stops, crossings, etc,).

Distributed CSP: A distributed CSP (DCSP) is a CSP in which the variables and constraints are distributed among automated agents (Yokoo & Hirayama 2000).

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 2: A *block agent* a_j is a virtual entity that essentially has the following properties: autonomy, social ability, reactivity and pro-activity (Wooldridge & Jennings 1995).

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

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, where variable $TA_{i,k}$ represents that train i arrives at station k and the variable $TD_{i,k}$ means that train i departs from station k:

1. **Traffic rules** guarantee crossing and overtaking operations. The main constraints to take into account are:
   - ***Crossing constraint***: Any two trains going in opposite directions must not simultaneously use the same one-way track.
     $$TA_{i,A} < TD_{j,A} \text{ or } TA_{j,B} < TD_{i,B}$$
     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 1).
   - ***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.
     $$TD_{i,A} < TD_{j,A} \rightarrow TA_{i,B} < TA_{j,B}$$
     The train being passed is detoured form the main track so that the faster train can pass the slower one (see Figure 1).
   - ***Expedition time constraint***. There exists a given time to put a detoured train back on the main track and exit from a station.

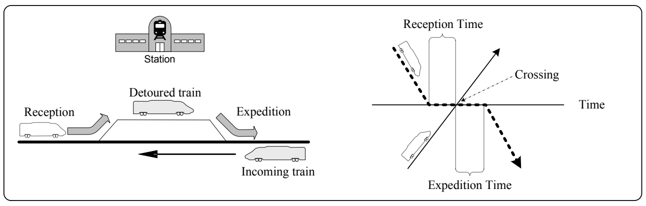


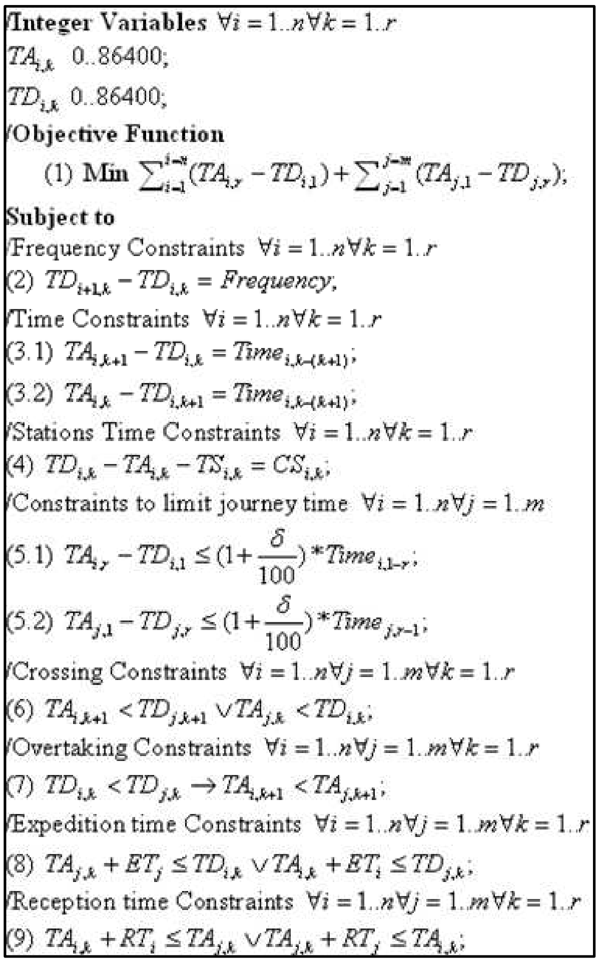
Figure 1: Constraints related to crossing and overtaking in stations

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

2. **User Requirements**: The main 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***. Train departure 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 and arrival and departure times of trains at stations. Constraints are composed by user requirements, traffic rules, and topological constraints.

The complete CSOP is presented in Figure 2. 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. $Time_{i,k-(k+1)}$ is the journey time of train i to travel from station k to $k+1$; $TS_{i,k}$ and $CS_{i,k}$ 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.



/**Integer Variables** $\forall i = 1..n \forall k = 1..r$

$TA_{i,k}$ 0..86400;

$TD_{i,k}$ 0..86400;

/**Objective Function**

$$(1)\ \textbf{Min} \sum_{i=1}^{i=n}(TA_{i,r} - TD_{i,1}) + \sum_{j=1}^{j=m}(TA_{j,1} - TD_{j,r});$$

Subject to

/Frequency Constraints $\forall i = 1..n \forall k = 1..r$

(2) $TD_{i+1,k} - TD_{i,k} = Frequency;$

/Time Constraints $\forall i = 1..n \forall k = 1..r$

(3.1) $TA_{i,k+1} - TD_{i,k} = Time_{i,k-(k+1)};$

(3.2) $TA_{i,k} - TD_{i,k+1} = Time_{i,k-(k+1)};$

/Stations Time Constraints $\forall i = 1..n \forall k = 1..r$

(4) $TD_{i,k} - TA_{i,k} - TS_{i,k} = CS_{i,k};$

/Constraints to limit journey time $\forall i = 1..n \forall j = 1..m$

(5.1) $TA_{i,r} - TD_{i,1} \le (1 + \frac{\delta}{100}) * Time_{i,1-r};$

(5.2) $TA_{j,1} - TD_{j,r} \le (1 + \frac{\delta}{100}) * Time_{j,r-1};$

/Crossing Constraints $\forall i = 1..n \forall j = 1..m \forall k = 1..r$

(6) $TA_{i,k+1} < TD_{j,k+1} \vee TA_{j,k} < TD_{i,k};$

/Overtaking Constraints $\forall i = 1..n \forall j = 1..m \forall k = 1..r$

(7) $TD_{i,k} < TD_{j,k} \rightarrow TA_{i,k+1} < TA_{j,k+1};$

/Expedition time Constraints $\forall i = 1..n \forall j = 1..m \forall k = 1..r$

(8) $TA_{j,k} + ET_j \le TD_{i,k} \vee TA_{i,k} + ET_i \le TD_{j,k};$

/Reception time Constraints $\forall i = 1..n \forall j = 1..m \forall k = 1..r$

(9) $TA_{i,k} + RT_i \le TA_{j,k} \vee TA_{j,k} + RT_j \le TA_{i,k};$

Figure 2: Formal Model of the Railway Scheduling Problem.

Partition Proposals

Due to specific properties in the railway scheduling problem, several models can be adopted to distribute the problem.

Partition Proposal 1

The first way to distribute the problem is carried out by means of a graph partitioning software called METIS (METIS), for the purpose of this distribution, the model constraints are converted into binary constraints, this is trivial and the result is a binary CSP. METIS provides two programs *pmetis* and *kmetis* for partitioning an unstructured graph into *k* equal size parts. In this way, the railway

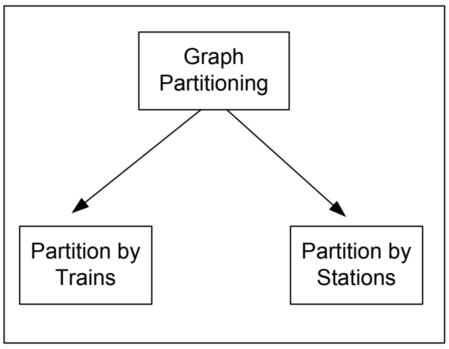


Figure 3: Distributed models: From Graph Partitioning to Train Partitioning and Station Partitioning.

scheduling problem can be modelled as a constraint network. This network can be partitioned in 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 the results are not appropriate. To improve the partition procedure, we extract additional information from the railway topology to obtain better partitions such as partition proposal 2 and 3.

Partition Proposal 2

The second model is based on distributing the original railway problem by means of train type. Each agent is committed to assign values to variables regarding a train or trains to minimize the journey travel. Depending on the selected number of partitions, each agent will manage one o more trains. Figure 4 shows a running map with 20 partition, each agent manages one train. This partition model has two important advantages: Firstly, this model allow us to improve privacy. Currently, due to the policy of deregulation in the European railways, trains from different operators work in the same railway infrastructure. In this way, the partition model gives us the possibility of partition the problem such as each agent is committed to a operator. Thus, different operators maintain privacy about strategic data. Secondly, this model allow us to manage efficiently 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 firstly carry out value assignment to variables, in order to achieve better journey travels

Partition Proposal 3

The third model is based on distributing the original railway problem by means of contiguous stations. Due to deregulation of European railways operators, long journeys may be scheduled. However, long journeys involve large number of stations at different countries with different railway policies. Therefore, a logical partition of the railway network can be carried out by means on regions (contiguous stations). To carry out this type of partition, it is important to analyze the

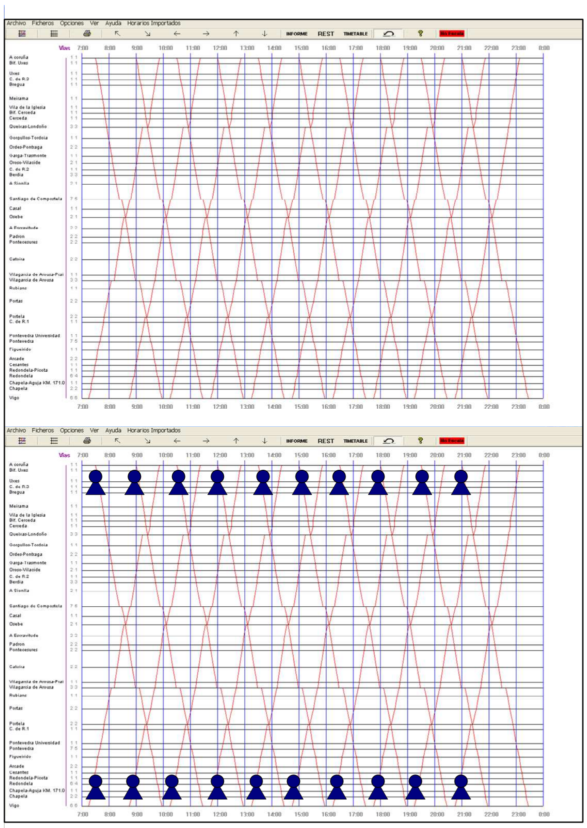


Figure 4: Distributed Railway Scheduling Problem. Proposal 2.

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 few stations if they are bottlenecks. Furthermore, the agents committed with bottleneck have preferences to assign values to variables due to their domains are reduced (variable ordering).

Thus, the running map to be scheduled between two cities is decomposed in several and shorter running maps. Figure 5 (up) 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 5 (down)). Thus, we can obtain important results such as railway capacity, consistent timetable, etc.

The Distributed Model

In the specialized literature, there are many works about distributed CSPs. In (Yokoo & Hirayama 2000), Yokoo et al. present a formalization and algorithms for solving distributed CSPs. These algorithms can be classified as either distributed stochastic search methods, synchronous backtracking or asynchronous backtracking (Yokoo & Hirayama 2000).

Our model can be considered as a synchronous model. It is meant to be a framework for interacting agents to achieve

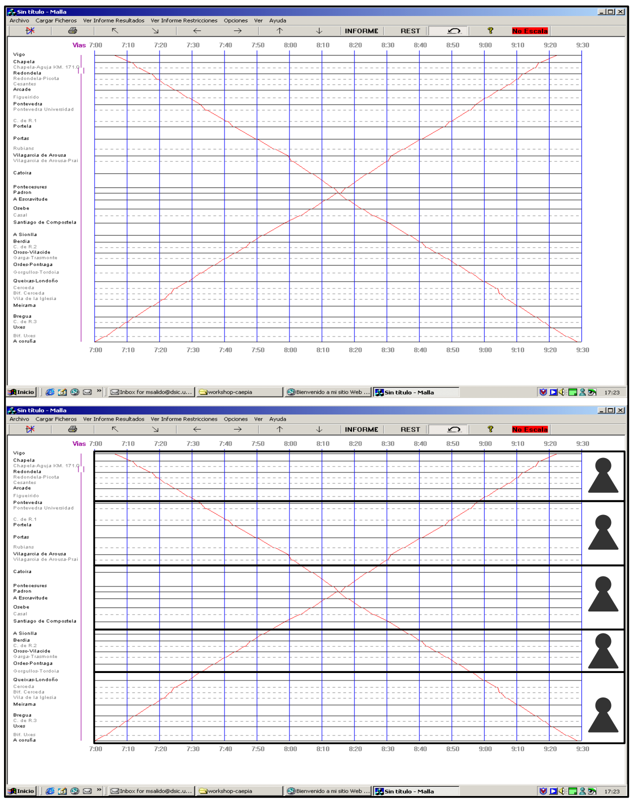


Figure 5: Distributed Railway Scheduling Problem. Proposal 3.

a consistent state. The main idea of our multi-agent model is based on (Salido, Giret, & Barber 2003) but partitioning the problem in k subproblems as independent as possible, classifying the subproblems in the appropriate order and solving them concurrently.

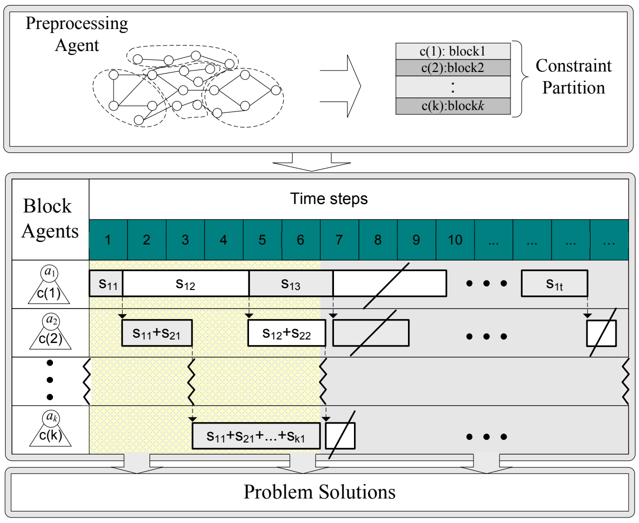


Figure 6: Multi-agent model.

In all our proposals, the problem is partitioned in k blocks or clusters in order to be studied by agents called *block agents*. Furthermore, a partition agent is committed to classify 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 such as 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, 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 simultaneously 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 simultaneously with *block agents* $1, 2, ...(k-1)$, tries to find a consistent state in order to find a problem solution.

Figure 6 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 6 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.

Evaluation

In this section, we carry out an evaluation between our distributed model and a centralized model. Furthermore, we evaluate the behavior of three proposed partition models. To this end, we have used a well-known CSP solver called CON'FLEX[1] which uses Forward Checking (FC) algorithm.

[1]It can be found in: http://www-bia.inra.fr/T/conflex/ Logiciels/adressesConflex.html.

Table 1: $< n, 20, 120 >$: 20 stations, 120 minutes frequency.

Trains (n)	*Variables*	*Constraints*
1	80	107
2	160	234
3	240	379
4	320	550
5	400	742
6	480	953
7	560	1186
8	640	1442
9	720	1717
10	800	2010

Table 2: $< 5, s, 120 >$: 5 trains, 120 minutes frequency.

Stations (s)	*Variables*	*Constraints*
10	200	527
20	400	742
30	600	1178
40	800	1608
50	1000	2073
60	1200	2555

This empirical evaluation was carried out over a real railway infrastructure that joins two important Spanish cities (La Coruna 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.

Tables 1 and 2 show the parameters used to evaluate the behavior of the centralized model and the distributed model with the proposal partitions. We can observe that the complexity increased when the number of trains and stations increased. All instances maintain a frequency $f = 120$ minutes.

Table 1 shows the number of variables and the number of constraints generated when the number of trains in each direction increased from 1 to 10 in a railway infrastructure with 20 stations and a frequency of 120 minutes $< n, 20, 120 >$.

Table 2 shows the number of variables and the number of constraints generated when the number of stations increased from 10 to 60 in a running map with 5 train in each direction and a frequency of 120 minutes $< 5, s, 120 >$. Because the real railway infrastructure maintains 40 stations, we have virtually eliminated and added stations to carried out this evaluation.

General graph partitioning applications work well in general graphs. However, in the railway scheduling problem, we did not obtain good results using these softwares. We evaluate the partition proposal 1 by using METIS in several instances of Table 1. However, the obtained results were even worse in the distributed model than in the centralized model. We studied the partitions generated by METIS and we observed that the journey of a train is partitioned in several clusters, and each cluster was composed by tracks of trains in opposite directions. This cluster is easy to solve but very difficult to propagate to other agents. Furthermore, the following partition proposals make the contrary, that is, they never join tracks of trains in opposite directions.

So, we can conclude that the problem is very dependent of the partition that we carry out, and a general partition based on low connectivity is not always the best solution.

Figure 7 shows the running time of the instances presented in Table 1 meanwhile Figure 8 shows the running time of the instances presented in Table 2. In both Figures, the partition model selected was partition proposal 2, where the number of partition/agents was equal to the number of trains. In both figures, we can observe that the running time increased when the number of trains increased (Figure 7) and when the number of stations increased (Figure 8). However, in both cases, the distributed model maintained better behavior than the centralized model.

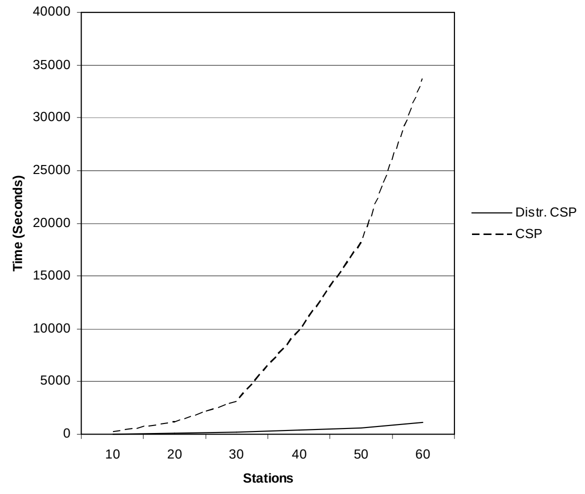


Figure 7: Running Time when the number of trains increased.

The partition proposal 2 was the best of the partition proposals, where we can schedule many trains in large railway infrastructure. However, how many partitions must divide the railway problem? If we select a large number of partitions, each subproblem is very easy, but the efficiency decreased due to communication messages. If we select a low number of partitions, each subproblem may be also difficult to solve. So, an appropriate number of partitions must be studied to solve the problem efficiently.

Figure 9 shows the running time with different partitions in problems where we fix the frequency (120 minutes), the number of stations (20) and the number of trains in each direction was increased from 5 to 10. Each instance was solved by the distributed model with different number of partitions (8,10,12,14,16,18,20 partitions). We can observe

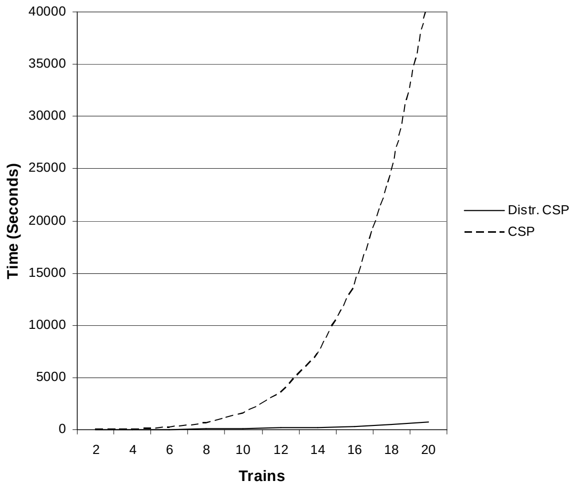


Figure 8: Running Time when the number of station increased.

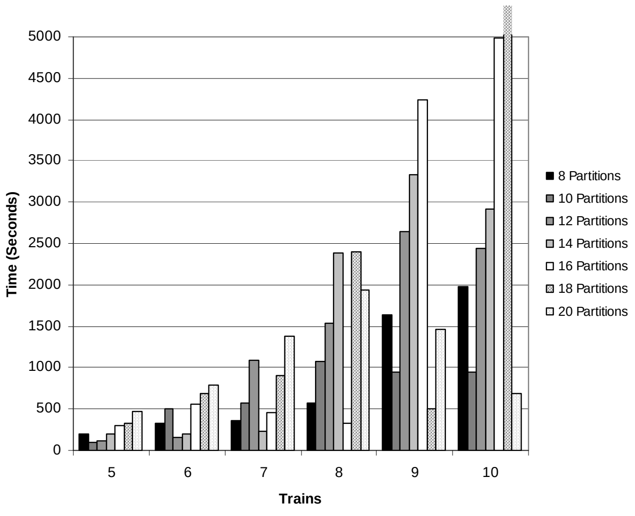


Figure 9: Running Time with different partitions.

the direct relation between the number of trains and the number of partitions. Thus, a agent was committed to schedule a train. If a agent was committed to schedule several trains, the efficiency decreased. Similar results happened when a train was scheduled by several agents.

The partition proposal 3, based on distributing the railway problem by means of contiguous stations was evaluated using the instances presented in Table 3.

This table shows the number of variables and the number of constraints generated when the number of trains in each direction increased from 1 to 10 in a railway infrastructure with 10 stations and a frequency of 120 minutes $< n, 10, 120 >$.

Figure 10 shows the running time of the centralized model and the distributed model, of the instances presented in Table 3 with a fixed number of partitions (6 partitions). It can be observed that the distributed model maintained a better behavior than the centralized model in all instances even with

Table 3: $< n, 10, 120 >$: 10 stations, 120 minutes frequency.

Trains (n)	*Variables*	*Constraints*
1	40	78
2	80	157
3	120	241
4	160	378
5	200	527
6	240	675
7	280	859
8	320	1078
9	360	1288
10	400	1515

a fixed number of partitions. However, in this type of distribution proposal, determining the appropriate number of partitions is difficult. It depends on the number of stations, the distance between them, the inclination of tracks, the number of tracks between stations, etc.

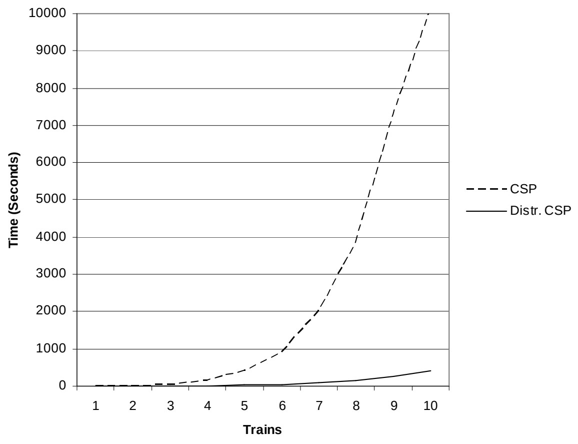


Figure 10: Running Time in problem with 6 partitions.

Conclusion and Future work

In this paper, we present a distributed model for solving the railway scheduling problem, in which several proposals are developed to distribute the railway scheduling into a set of sub-problems as independent as possible. Then, a set of *block agents* are incrementally and concurrently committed to building partial solutions until a global solution is found. The evaluation section shows the railway scheduling problem can be solved more efficiently in a distributed way. We are working on developing new heuristics to solve the distributed model in a more efficient way. Furthermore, it is necessary to built up a formal relation between the railway topology and the appropriate number of partitions.

References

Cordeau, J.; Toth, P.; and Vigo, D. 1998. A survey of optimization models for train routing and scheduling. *Transportation Science* 32:380–446.

Karypis, G., and Kumar, V. 1995. Using METIS and parMETIS.

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

Lova, A.; Tormos, P.; Barber, F.; Ingolotti, L.; Salido, M.; and Abril, M. 2006. Intelligent train scheduling on a high-loaded railway network. *Lecture Notes in Computer Science, (to appear).*

METIS. http://www-users.cs.umn.edu/ karypis/metis/ index.html.

Nachtigall, K., and Voget, S. 1997. Minimizing waiting times in integrated fixed interval timetables by upgrading railway tracks. *European Journal of Operational Research* 103:610–627.

Salido, M.; Giret, A.; and Barber, F. 2003. Distributing Constraints by Sampling in Non-Binary CSPs. *In IJCAI Workshop on Distributing Constraint Reasoning* 79–87.

Schloegel, K.; Karypis, G.; and Kumar, V. 2003. Graph partitioning for high-performance scientific simulations. *Sourcebook of parallel computing* 491–541.

Schrijver, A., and Steenbeek, A. 1994. Timetable construction for railned. *Technical Report, CWI, Amsterdam, The Netherlands*.

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

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

Walker, C., S. J., and Ryan, D. 2005. Simultaneous disruption recovery of a train timetable and crew roster in real time. *Comput. Oper. Res* 2077–2094.

Wooldridge, M., and Jennings, R. 1995. Agent theories, arquitectures, and lenguajes: a survey. *Intelligent Agents* 1–22.

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