Distributed constraint satisfaction problems to model railway scheduling problems

P. Tormos[2], M. Abril[1], M. A. Salido[1], F. Barber[1], L. Ingolotti[1] & A. Lova[2]
[1] DSIC, Universidad Politécnica de Valencia, Spain
[2] DEIOAC, Universidad Politécnica de Valencia, Spain

Abstract

Railway Scheduling is considered to be a difficult and time-consuming task. Despite real railway networks being modelled as Constraint Satisfaction Problems (CSPs), 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 independently 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.

1 Introduction

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 [2]. 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

WIT Transactions on The Built Environment, Vol 88, © 2006 WIT Press
www.witpress.com, ISSN 1743-3509 (on-line)
doi:10.2495/CR060291

complexity of the problem depending on its size. Schrijver and Steenbeek [3] 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 [4] 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 [5], Walker and Ryan [6]), 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. [7] 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.

In parallel computing, many researchers are working on graph partitioning [8], [9]. 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. 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.

2 Problem modelling: a constraint satisfaction problem

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), and 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. 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,).

A distributed CSP (DCSP) is a CSP in which the variables and constraints are distributed among automated agents [10]. 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.

2.1 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, 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.
    - *Overtaking constraint*: Two trains going at different speeds in the same direction can only overtake each other at stations.

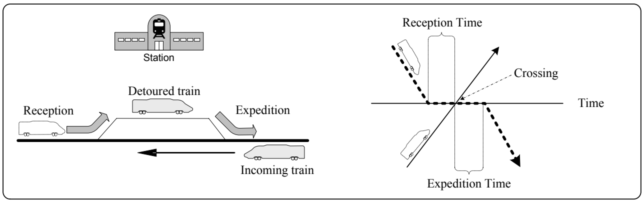


Figure 1: Constraints related to crossing and overtaking in stations.

    - *Expedition (Reception) time constraint*. There exists a given time to put (detour) a train back on (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.

- *Departure interval* for the departure of the first trains going in both the up and down directions.

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).
   - *Time constraints*, between each two contiguous stations,

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. The complete CSOP is presented in Figure 2. Let us 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.

3 Partition proposals

3.1 Partition proposal 1: graph partitioning

The first way to distribute the problem is carried out by means of a graph partitioning software called METIS [11], for the purpose of this distribution, the model constraints are converted into binary constraints. METIS provides two programs *pmetis* and *kmetis* for partitioning an unstructured graph into k equal size parts. 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.

3.2 Partition proposal 2: by trains

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 or more trains. Figure 3a 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.

In this way, the partition model gives us the possibility of partition the problem such as each agent is committed to a operator.

This model allow us to manage efficiently priorities between different types of trains (regional trains, high speed trains, freight trains).

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

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

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

/**Objective Function**

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

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

/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) $TA_{j,k} + ET_j \leq TD_{i,k} \vee TA_{i,k} + ET_i \leq TD_{j,k}$;

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

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

Figure 2: Formal model of the railway scheduling problem.

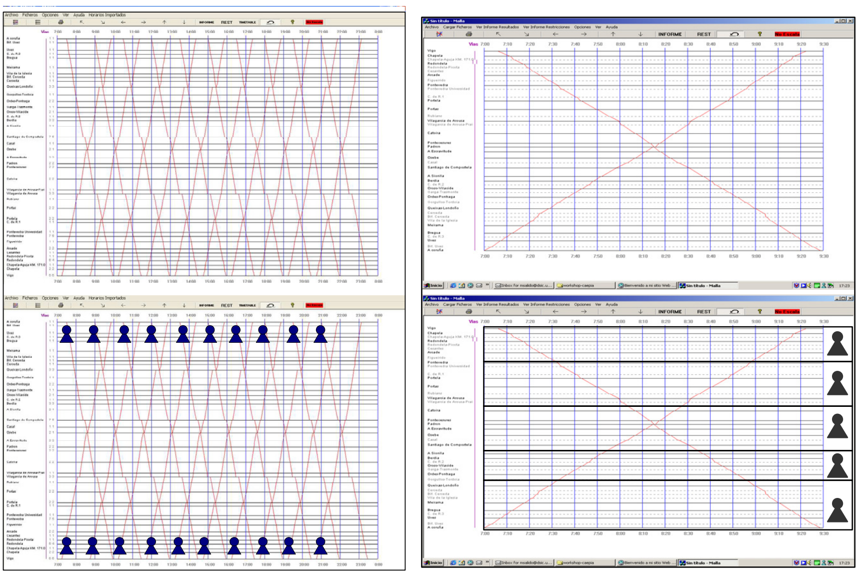
3.3 Partition proposal 3: by stations

The third model is based on distributing the original railway problem by means of contiguous stations. 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 railway infrastructure and detect restricted regions (bottlenecks). To balance the problem, each agent is committed to a different number of stations.

Figure 3b (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 3b (down)). Thus, we can obtain important results such as railway capacity, consistent timetable, etc.

4 Evaluation

In this section, we carry out an evaluation between our distributed models and a centralized model. 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.



(a) Proposal 2: Trains. (b) Proposal 3: Stations.

Figure 3: Distributed railway scheduling problem.

Figure 4a shows the running time when the number of trains increase, meanwhile Figure 4b shows the running time when the number of stations increase. 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 4a) and when the number of stations increased (Figure 4b). However, in both cases, the distributed model maintained better behavior than the centralized model.

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 the problem. 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

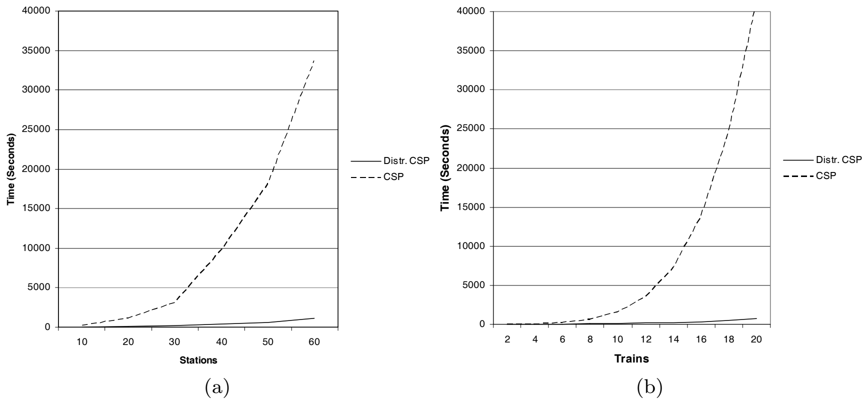


Figure 4: Running Time when the number of trains (a)/stations (b) increased.

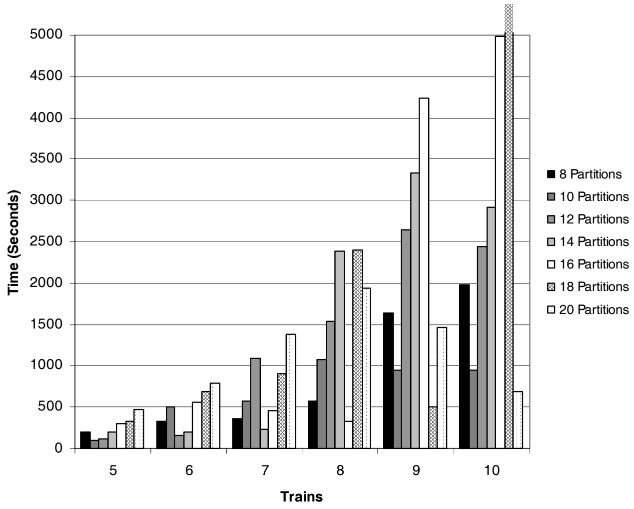


Figure 5: Running Time with different partitions.

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.

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 5 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 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.

5 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. The evaluation section shows the railway scheduling problem can be solved more efficiently in a distributed way.

Acknowledgements

This work has been partially supported by the research projects TIN2004-06354-C02- 01 (Min. de Educación y Ciencia, Spain-FEDER), FOM-70022/T05 (Min. de Fomento, Spain) and FP6-021235-2, IST-STREP (UE). We appreciate the assistance of Jose Estrada (Spanish Administration of Railway Infrastructure - ADIF).

References

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

[2] Serafini, P. & Ukovich, W., A mathematical model for periodic scheduling problems. *SIAM Journal on Discrete Mathematics*, pp. 550–581, 1989.

[3] Schrijver, A. & Steenbeek, A., Timetable construction for railned. *Technical Report, CWI, Amsterdam, The Netherlands*, 1994.

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

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

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

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

[8] Schloegel, K., Karypis, G. & Kumar, V., Graph partitioning for high-performance scientific simulations. *Sourcebook of parallel computing*, pp. 491–541, 2003.

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

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

[11] METIS, http://www-users.cs.umn.edu/ karypis/metis/ index.html.