Proceedings of the International Multiconference on
Computer Science and Information Technology pp. 679–685
ISBN 978-83-60810-14-9
ISSN 1896-7094

Distributed Scheduling for Real-Time Railway Traffic Control

Tiberiu Letia, Mihai Hulea and Radu Miron
Dept. of Automation
Technical University of Cluj-Napoca
C. Daicoviciu St., 15,
40020, Cluj-Napoca, Romania
Email: Tiberiu.Letia, Mihai.Hulea, Radu.Miron@aut.utcluj.ro

***Abstract*—The increase of railway traffic efficiency and flexibility requires new real-time scheduling and control methods. New charter trains have to be added continuously without disturbing the other (periodic) train moves or decreasing the safety conditions. A distributed method to schedule new trains such that their real-time constraints are fulfilled is presented. The trains have timelines to meet and hence the deadlines are extracted taking into account the included laxity. The trains have pre-established routes specifying the stations and required arrival times. The paths containing the block sections from one station to another are dynamically allocated without leading to deadlocks.**

I. Introduction

A. Justification of the Problem

RAILWAY networks can solve many of the transportation problems raised by modern society. Railway traffic improvements involve higher flexibility, speed and density. Besides the regular trains (with fixed routes and timetables), new charter trains have to be dynamically accepted without losing the safety and disturbing the other train schedules.

Early conventional railway traffic control has been focused on safety and scheduling train arrival times such that they can be met. Train traffic was of very low density and its efficiency was based on long trains. To avoid train delays, their rates were low and the train speeds were much under their capacities so that the timetable could be fulfilled. Such systems were inflexible and the railway resources were underused.

Railway traffic is described as an emerging network embedded real-time application with long and short reaction time magnitudes. The long durations of event reactions allow the usage of expensive scheduling algorithms that are not accepted in many distributed real-time control applications. The short reaction times involve decisions to be taken under corresponding time constraints.

The railway traffic control system is a dynamic one that operates in an environment with dynamically uncertain properties that include transient and resource overloads, arbitrary arrivals, arbitrary failures and decrease of traffic parameters.

Unlike classical real-time control applications that usually concern only the response times to meet the deadlines, railway traffic involves the reasoning about end-to-end timelines and the reaction to events such that the global traffic system fulfills the time requirements. Despite many uncertainties, the control system is expected to guaranty that all the trains behave according to timelines.

Due to the large dimension of railway networks, the centralized control is not appropriate in the current circumstances because of the need of safety, the communication delays, the complexity of the system and the difficulties to get the right control decisions in time. These are the main reasons for developing autonomous decentralized control systems for railway traffic.

Global train traffic planning is a possible approach of the current problems. A set of trains with their routes and initial departure times is given. The feasible solution provides the train arrival and departure times at the railway stations contained in the given routes. The solution is usually obtained through large system simulation and use of the minimization of different criteria. At this level, train traffic control refers to sending signals such that all the train timetables are fulfilled in all the railway stations.

A train traveling from one point to another involves some dependent real-time activities. The train crossing an interlocking performs an activity directly controlled by the control system. The traveling from one interlocking to another is usually free movement. Some traffic lights can be added to split the long track lines into smaller block sections to increase the track utilization. In this case, a safe policy requires that each block section contains only one train at a time and between each pair of trains a non occupied block section is compulsory. Using Global Positioning System (GPS) and the wireless communication some moving block sections can be implemented. This can lead to higher track utilization, but traffic safety is based on GPS and wireless communication system reliability.

Traffic system goals are:

- to minimize traffic cost;
- to maximize traffic system throughput
- to fulfill train timing requirements;
- to guaranty system safety;
- to minimize fault effects on train schedules and
- to sustain railway maintenance.

B. Related Work

The basic principles of railway traffic control are given in [1]. These include the interlocking usage, resource management and dividing the railway network into different parts. The assessment of scheduling is performed by the capability of the schedule to meet the needs of customers and the capabilities of the trains to recover the delays according to their timetables.

The train deviations from the scheduled timetable should be removed during the operation [2].

New trends of train traffic control and management started since 1997 [3]. An autonomous decentralized train control and management system is proposed to attain both the real-time properties for train control such as the real-time traffic and non-real-time properties for train management.

A single delayed train can cause a domino effect of secondary delays over the entire network, which is the main concern of planners and dispatchers [4].

Train scheduling implementations are:

- *off-line scheduling* when all the train arrival times and departure times are calculated before the train starts. The trains behave exactly as they were planned. No unexpected event happens and no new train can appear.
- *on-line scheduling* when the scheduling is performed during the train traffic operation. Some trains have variable delays, unexpected events happen, and new train scheduling requests are required and accepted during the operation.

Some train scheduling approaches are based on:

- distributed artificial intelligence (using trackside intelligent controllers [5]). This kind of allocation of function can optimize the use of resources, reduce complexity and enhance the reliability and availability of the traffic system.
- heuristics methods (as genetic algorithms [6] or ant colony systems [7]). The NP-hard problem complexity with respect to the number of conflicts in the schedule is avoided by generating random solutions and guiding the search.
- auction-based [8]. Each train is represented by an agent that bids for right to travel through a network from its source to destination.
- interactive scheduling [9]. Interactive applications are used to assist planners in adding new trains on a complex railway network. It includes many trains whose timetables cannot be modified because they are already in circulation.

An improvement can be obtained using the GPS and wireless communication between train engine and local control center [10]. Some distributed signal control systems based on the Internet technology are also used [11].

Formal development and verification of a distributed railway control system are performed applying a series of refinement and verification steps [12].

The distributed train scheduling problem has some similarities with distributed software job scheduling [13], [14].

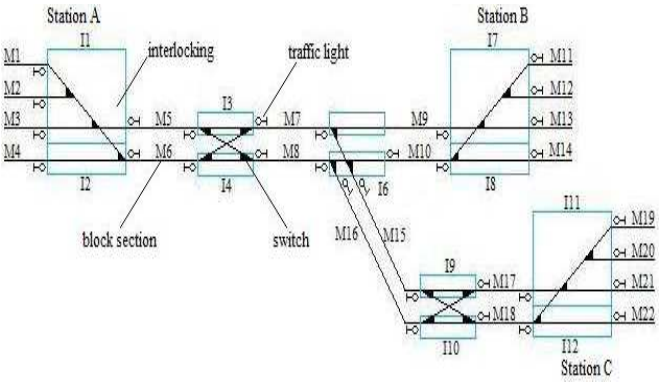

Fig. 1. The railway network structure

Both have to fulfill real-time constraints relative to finishing time, communication requests and resource management. The concept of collaborative scheduling is also applicable. The problem of the railway interlocking scheduling has some common features with independent scheduling of each node of a distributed software system. Each node constructs its local schedule using only local information. The lack of global information makes it impossible for a node to make a globally optimal decision. Thus it is possible for a node to make a scheduling decision that is locally optimal in terms of the utility that can be accrued to the node, but compromises global optimality. The collaborative scheduling is a paradigm for systems that can withstand its large overhead.

II. Structure and Objectives

A. Railway Structure

Figure 1 represents a railway network between three stations. On the graph are represented the traffic lights and the switch points. The interlockings are marked by I1, , I12. M1,, M22 denote the block sections controlled by managers. The train movements are controlled by traffic lights and switch points.

An *interlocking* is an arrangement of neighbor interconnected sets of (switch) points and (traffic light) signals such the train movements through them is performed in a proper and safe sequence.

Generally, a *train schedule* is a designation of train description, day, route, speed, arrival and departure times of a train. The train schedule also contains the station dwell times. Some other train stops are required if the necessary track lines are not available when the trains reach the interlocking.

Figure 2 shows a train trajectory with a variable laxity. The notations are:

- *ea* for earliest arrival time;
- *la* for latest arrival time;
- *er* for earliest release (exit) time and
- *lr* for latest release (exit) time.

The objective is to schedule the train move such that it arrives at the next (destination) point between earliest and latest arrival time.

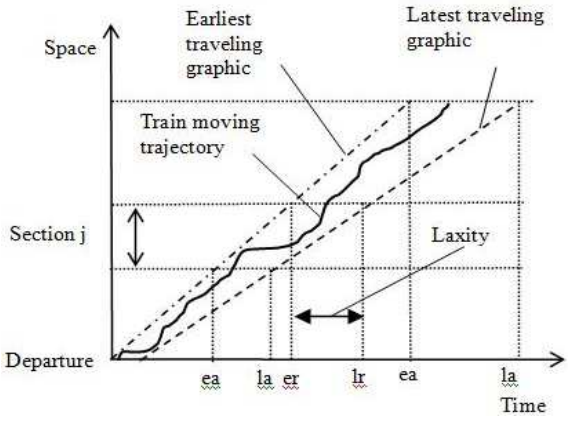

Fig. 2. Train traveling diagram with variable laxity time

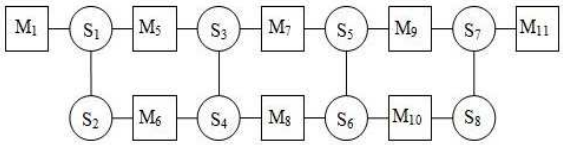

Fig. 3. Distributed scheduling structure

The control decisions take as a target to lead the train such that it reaches the earliest arrival time. The performance evaluation of the system behavior considers that the control fulfills the requirements if all the trains arrive before their specified latest arrival times. If the laxity time is not consumed during a block section or in front of a switch it is added and available to the next sections.

A similar case can be constructed such that the laxity time remains fixed on each section. The train move on a section has to recover the laxity time consumed on the previous one.

In both cases of policy decisions, the laxity time provides the deadline of train behavior together with the earliest arrival time.

B. Traffic Control Objectives

The current study refers to finding a train path starting at a given time, the schedule and the control between two neighbor stations such that the global planned times are fulfilled. This means to find a path from one railway station platform (or block section) to the neighbor railway station platform and the necessary resources (block sections and interlockings). The train traffic control between the neighbor stations means to apply the schedule on-line. The traffic light and switch (point) signals are applied according to the trains current positions and schedules.

III. Distributed Scheduling

A. Architecture of Distributed Scheduler

Figure 3 represents the software components involved by a train traveling path scheduling. The section managers are denoted with $M_1, \ldots, M_{11}$. $S_1, \ldots, S_8$ represent the interlocking schedulers.

The distributed scheduling is performed by collaboration of train agents, interlocking schedulers and block section managers. The agents have information about the possible paths and expected timetables. For some (charter) trains an acceptable solution is such that the mentioned trains reach the destination as soon as possible.

The interlocking schedulers allocate their resources on-line when the trains approach the interlocking; meanwhile the block section managers allocate the resources off-line (before the trains start their travel from one station to another). An agent asks a scheduler to reserve its controlled interlocking for a specified duration during a given time interval. The scheduler grants it only if the requested task does not delay unacceptably the already scheduled trains, such that the last ones miss their deadlines.

B. Agent Behavior

The train agent's goal is to get a path that fulfills the timing requirements from a railway station platform or block section to another neighbor station.

The agent has to solve a local problem defined by the current train position, departure time, next station block section and arrival time. The planned duration has included, besides the necessary moving time, a laxity time used to compensate the waiting (delay) times involved by crossing of interlockings.

There are two train agent behaviors (approaches):

- In the first one the train agent asks all the schedulers and managers of the possible paths (*listOfPaths*) from the departure station to the neighbor station to accept the train moves. The train agent chooses the best schedule analyzing the schedulers and managers responses.
- In the second one the train agent demands the move specifying the train parameters only to the first scheduler. This is responsible further on to get all the possible paths from departure to neighbor destination. The train agent gets the possible schedules and chooses the best of them. It announces the neighbor scheduler about the chosen path. The neighbor scheduler announces further its neighbor involved components in the path about the firm reservation.

The first approach involves a transaction where the train agent (algorithm) takes a list of possible paths, the start time and the laxity. Using the schedulers and the managers it fills the list of paths with times. Finally, it chooses the best path and announces the scheduling participants about that.

The following notations are added:

- *is* for train input speed at the arrival time at the entrance in the interlocking;
- *os* for train output speed at the exit from the interlocking;
- *st* for train start time;
- *Lx* for laxity time;
- *C* for train worst case crossing time of the interlocking;
- *dd* for absolute deadline.

Agent algorithm (first approach):

1: **input**: *trainParameters, trainRoute, Lx*;
2: **input**: *currentStation, listOfPaths*;
3: **input**: *listOfSchedules*;
4: **output**: *trainSchedule*;
5: *Initialization*: *listOfSchedules = extend(listOfPaths)*;
6: *currentSection=getCurrentSection()*;
7: *ea=st; la=st+Lx*;
8: *is=0*;
9: **for** all paths from *listOfPaths* **do**
10: choose an unvisited path;
11: **while** (*next*) choose the next component as *next*; **do**
12: *(er,lr,os)=next.request(trainParameters,ea,la,Lx, is)*;
13: **if** *(er==0)* **then break**;
14: **else** *ea= er; la=lr; is=os*; fill in the *listOfSchedule*;
15: **end while**;
16: **end for**;
17: choose the *bestPath* from the *listOfSchedule*;
18: notify all the participants;
19: **return** *bestPath* as the train schedule;

The laxity of traveling from one station to the neighbor station is distributed uniformly to all the paths sections. During traveling from one section to another, if the laxity was not consumed it can be added to the next section.

C. Scheduler Behavior

Each interlocking is controlled by a scheduler. This can respond to another software component if a new train T_k can be scheduled during a given time interval $[T_k.ea, T_k.la]$ (where ea is the earliest arrival time and la is the latest arrival time of the train T_k at the entrance of the interlocking) without unacceptably delaying the already scheduled trains. If any train is scheduled such that the finishing time of crossing the interlocking is longer than the deadline, then the schedule of the train set is not feasible. A new train can be added to be moved through interlocking only if the schedule of the all train sets with the arrival time intervals overlapping is feasible.

The train $T_k.dd$ deadline of crossing through interlocking is given by:

$$T_k.dd = T_k.ea + T_k.Lx \tag{1}$$

1) Feasibility analysis: The worst case for the feasibility analysis is when all the trains of a set arrive simultaneously as close as possible to their deadlines.

Let t_x be the latest time when the trains of a given set can arrive at the same time.

$$t_x = \sup_t \bigcap_k T_k.AI \tag{2}$$

where $T_k.AI = [T_k.ea, T_k.la]$ is the arrival time interval of the train T_k at the entrance of an interlocking. The time t_x is the latest arrival time of any train contained in the intersection of arrival intervals of the considered set.

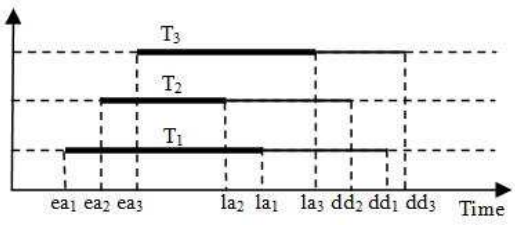

Fig. 4. Train arrival intervals

Figure 4 shows the arrival intervals of three trains (T_1, T_2, T_3) with overlapping arrival time intervals. For this example $t_x = la_2$.

Jackson's rule states: "Given a set of n independent tasks, any algorithm that executes the tasks in order of nondecreasing deadlines is optimal with respect to minimizing of the maximum lateness [15]."

The feasibility test is:

$$\forall i = 1, ..., n; \sum_{k=1}^{n} T_k.C \leq T_i.dd \tag{3}$$

The previous formula is used off-line to feasibility test analysis. This test has to be applied for all train sets that overlap with the new train added to schedule.

The scheduler uses for on-line traffic control the earliest deadline first (EDF) algorithm. That means if more than one train arrives at the same time, the train with the earliest deadline gets first the right to cross the interlocking. Taking into account that the train deadlines are fixed, the algorithm can be applied on-line using fixed priorities.

The list *scheduledQueue* contains elements with the attributes: *trainIdentifier* $T_i, T_i.ea, T_i.la, T_i.dd, T_i.C$.

For evaluation of the worst loading of an interlocking by a train set with the overlapping arrival intervals, the following formula can be used:

$$load = \frac{\sum_{k=1}^{n} T_k.C}{\sum_{k=1}^{n} (T_k.dd - t_x)} \tag{4}$$

For the reason of robustness a path with smaller load factors of the contained interlockings is preferred. On the other side, if the load is small, there is a greater possibility to obtain a feasible scheduling if a new train agent demands the move through interlocking.

2) Scheduller Algorithm: The following notations are added:

- *Cmin* for the minimum crossing time of the interlocking;
- *Cmax* for the maximum crossing time.

Scheduler grant algorithm of an interlocking:

1: **input**: *trainParameters*;
2: **input**: *ea, la, Lx, is*;
3: **input**: *scheduledQueue*;
4: **input**: *interlockingParameters*;

5: **output**: *er, lr, os*;
6: **output**: *feasibility, load*;
7: *Initialization: load=0; lr=0*;
8: calculate the best case crossing time *Cmin* using *trainParameters* and *interlockingParameters*;
9: *er = ea + Cmin*;
10: calculate the worst case crossing time *Cmax* and *os* as the maximum speed at the exit of the interlocking using *trainParameters* and *interlockingParameters*;
11: find all the train sets with arrival intervals that overlap with the arrival interval of the new train;
12: **for** all train sets **do**
13: calculate the worst case arrival time t_x of the *trainSet* using formula (2);
14: *dl* =add the worst case crossing time *Cmax* of all trains from *trainSet*;
15: *temp = la + Cmax + dl*;
16: **if** the formulae (3) are fulfilled with *C = Cmax* **then** *feasibility = true*;
17: **else** *feasibility = false*; **break**;
18: *ld* = $(\Sigma_i T_i.C)/(\Sigma_i(T_i.dd - t_x)$;//formula (4)
19: **if** *(load* $\prec$ *ld)* **then** *load=ld*;
20: **if** *(lr* $\prec$ *temp)* **then** *lr = temp*;
21: **end for**;
22: **return** *er, lr, os, feasibility, load*;

D. The Scheduling Improvement

In the presented scheduling algorithm, if a train with lower priority arrives with a very short duration earlier than a train with higher priority, the first one gets the right of crossing. This is inconvenient if the train global priorities express the operator's desires that some trains have to use the interlocking before the others when they arrive almost in the same time.

An algorithm improvement can be: if a lower priority train arrives before a higher priority train, and the first train cannot cross the interlocking before the higher priority train arrival, but the first one can be delayed without missing the timing constraint, the interlocking has to be blocked until the higher priority train arrives and then the EDF algorithm is applied.

An oracle construction can be performed based on GPS or installing detectors on the block sections and estimating the arrival time at the interlocking based on the train current speed. That leads to know in advance the train arrival times during a specified period of time.

Let $T_i.at$ be the arrival time of the train T_i and *B* the blocking time of the interlocking until the higher priority train arrives. The test of scheduling feasibility is:

$$\forall i = 1, ..., n; T_i.at + B + \sum_{k=1}^{n} T_k.C \leq T_i.dd \qquad (5)$$

The trains can accept different blocking times given by the formula:

$$B_i = T_i.at - T_i.at - \sum_{k=1}^{n} T_k.C \qquad (6)$$

TABLE I
BLOCK SECTION - STATE TABLE

Time	Solicitor	State
0	$Train_x$	occupied
1	$Train_y$	reserved
2	$Train_y$	reserved
3	-	free
4	$Train_z$	requested
5	$Train_z$	requested
...	...	...

This acceptable blocking time depends on the train arrival time and its deadline.

Trains with higher priorities usually have higher speed and the proposed improvement involves that a higher priority train can cross the interlocking without waiting. That makes possible that a higher priority train needs shorter laxity time such that the feasibility scheduling test to be fulfilled. This improvement can be used to diminish the unexpected delay of a train due to some faults.

E. Manager Behavior

The resource manager has the task to reserve on the train agent's request the block section and to maintain the current state of the resource. A block section could have the following state: free, requested, reserved and occupied in every minute. The manager gets information from sensors about occupancy and clearance of the section. The section state is updated at every minute.

The resource manager keeps the Block Section - State Table with reserved periods of the resources for each train.

The train agent asks the reservation calling the method:
request(trainParameters, ea, la, Lx, is)

The manager has information about section length and maximum accepted speed. It calculates the necessary time to move from one end to another and reserves an extra *Lx* time. If it is not able to perform the reservation, the manager reserves zero length time intervals.

Manager request algorithm:

1: **input**: *trainParameters*;
2: **input**: *ea, la, Lx, is*;
3: **input**: *sectionSpeed, sectionLength*;
4: **output**: *er, lr, os*;
5: determine the train speed *sp*;
6: calculate the moving time *mt*;
7: *er = ea + mt*;
8: *lr = mt + Lx*; // calculate the later release of the resource
9: **if**(the resource is free between *ea* and *lr*) **then**
10: mark on the Block Section State Table the *attempt* of reservation for *trainID*;
11: **return** *er, lr, os=sp;*// respond with the latest // release time and the output speed;
12: **else return** *er=lr=0, os=is*;

Agent confirmation call is performed by the method:
confirm(trainID, ea, lr)

That reserves firmly the necessary resources and releases the resources attempted to be acquired, but not necessary for the chosen path.

IV. Implementation and Tests

Two approaches were used to test the scheduling algorithms. One uses the implementation of the proposed algorithms (based on real-time scheduler) and the other uses an implementation based on genetic algorithm. Both approaches use the same railway network model and have the same set of trains already scheduled. A new train schedule is required. Meanwhile the real-time scheduler uses the earliest arrival time as a target and the deadlines only for scheduling feasibility test; the evolutionary system has the goal to obtain the shortest traveling times having the arrival times between earliest arrival times and deadlines.

A. Genetic Algorithm for Train Traffic Scheduling

A *path* from a platform (or a block section) of station A to a platform (or a block section) of station B consists of a sequence of linked elements (interlockings and block sections) used by a train for moving from departure to destination.

The solving of the *scheduling problem* using a *genetic algorithm* has the goal to find the best path and the train speeds on the contained (path) elements. As a consequence, a solution is a pair *(path, set of speeds)*.

A *train schedule* (departure time, path, set of speeds) is *viable* if the train reaches the destination and its trajectory does not overlap any time and any element of the trajectory of any other train from the given train set.

A *train schedule is better* than any other if, starting at departure time and following the solution (path, set of speeds), the train reaches the destination at a time closer to arrival time (if there is given an arrival time), or earlier (if no arrival time is specified) than the time obtained with other solutions.

Between two stations there are a limited number of paths.

1) Individual coding: An individual codifies all the paths from departure to destination and the train average speeds on all involved elements. This codification is implemented on a matrix with the number of lines equal with the number of possible paths, and the number of columns equal with the maximum number of elements of any of the paths from departure to destination. As a consequence, each matrix line corresponds to a path. The elements of the line describe the train average speeds on the path elements. Due to the possibility that the path element numbers differ from one path to another, some elements on the right-side of the matrix could not correspond to real train speeds.

2) Individual evaluation: Using the train departure time, departure block section and individual codification, the train *traffic simulator* determines for each path of an individual the arrival times. The railway net traffic could contain other trains already scheduled and their schedules are not acceptable to be modified. If the trajectory of an already scheduled train overlaps at the same time the trajectory of the train attempted to be scheduled, the value of the fitness function corresponding to this path is drastically penalized. The evaluation of an individual is given by the *fitness function* that in this case is the weighted sum of the schedules (path, set of speeds) evaluations.

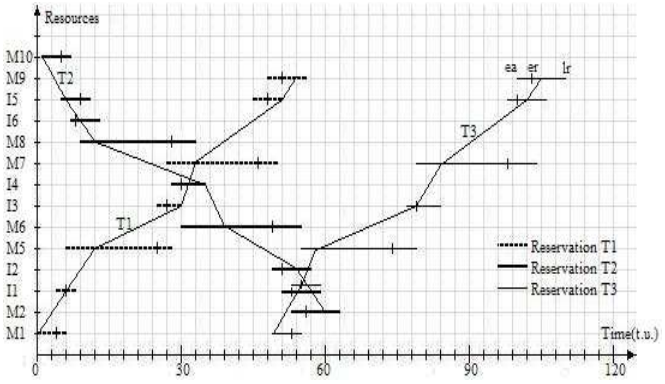

Fig. 5. Example of scheduling

3) New individual creation: The solution search using the genetic algorithms is performed by individual creation and evaluation. A new individual creation is obtained by:

- *Mutation.* An individual line (i.e. a path) and an element (i.e. the train speed on an element) of it are randomly chosen. The value of the element is randomly modified taking into account the specified speed limits.
- *Crossover.* Two individuals are chosen. A randomly chosen matrix column is used to cut the individuals' matrices in two parts. Two new individuals are constructed using parts from different matrices.

4) Individual's selections: Genetic algorithms work with populations of individuals. The selection of individuals that survive from one generation to another is obtained using the fitness function. The individuals with higher values of fitness functions have higher chances to survive.

The *solution of the scheduling problem* is chosen by taking from all the individuals the best value of the pair (path, set of speeds).

B. Solution Comparison

The solutions obtained using the distributed scheduling algorithms and the genetic algorithm are represented in Figure 5.

The solution given by the genetic algorithm for the traveling of three trains is represented by continuous lines. On the figure are also drawn the block section reservations provided by the real-time scheduler. Each horizontal line describes the mentioned values of *ea*, *er*, and *lr*. The interlocking *I1* is concurrently demanded by two trains (T2 and T3). The genetic algorithm solution avoids the simultaneous use of the interlocking by delaying the T2 train.

V. Conclusion

The proposed scheduling method does not lead to deadlock due to advance resource reservation. Comparing the perfor-

mance of the proposed real-time scheduling algorithms with the genetic algorithm performance, the first is lower but needs much smaller computation power (memory and time). The proposed method can provide deterministic time to get the solutions. It also has the advantage to be finally applied on-line and so it is able to diminish the variations of the train arrival times. The proposed method can be used to design the railway networks such that to be capable of providing a specified throughput with real-time features.

REFERENCES

[1] J. Pachl, "Railway Operation and Control," VTD Rail Publishing, Mountlake Terrace WA 98043 USA, 2004.

[2] J. Tornquist and J. A. Persson, "Train traffic deviation handling using tabu search and simulated annealing," *Proceeding of the 38th Annual Hawaii International Conference on System Science*, 2005, p. 73a.

[3] S. Shoji, A. Igarashi, "New trends of train control and management systems with real-time and non-real-time properties," *Proceedings of the 3rd International Symposium on Autonomous Decentralized Systems (ISADS'97)*, 1997, pp. 319–326.

[4] R. M. P. Goverde, "A delay propagation algorithm for large-scale scheduled rail traffic," *Preprints of 11th IFAC Symposium on Control in Transportation Systems*, Delft, Netherlands, 2006, pp. 169–175.

[5] T. Tao, "A train control system for low-density lines based on intelligent autonomous decentralized system (IADS)," *Proceedings of the Sixth International Symposium on Autonomous Decentralized Systems (ISADS'03)*, 2003.

[6] A. Higgins and E. Kozan, "Heuristics techniques for single line train scheduling," Journal of Heuristics, 3, Kluwer Academic Publishers, 1997, pp. 43–62.

[7] K. Ghoseri and F. Merscedsolouk, "ACT-Ts: Train scheduling using ant colony system," Journal of Applied Mathematics and Decision Science, 2006, pp. 1–28.

[8] D. C. Parkes and L. H. Ungar, "An auction-based method for decentralized train scheduling," Proceedings of the Fifth International Conference on Autonomous Agents, Montreal, Canada, 2000, pp. 43–50.

[9] A. Lova, P. Tormas, F. Barber, L. Ingolotti, M. A. Salido and M. Abril, "Intelligent train scheduling on high-loaded railway network," Algorithmic Methods for Railway Optimization, Springer, Berlin, 2007.

[10] A. Zimmermann and G. Hommel, "A train control system case study in model-based real-time system design," *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03)*, Nice, France, 2003, p118b.

[11] Y. Fukuta, G. Kogure, T. Kunifuji, H. Sugahara, R. Ishima and M. Matsumoto, "Novel railway signal control system based on the internet technology and its distributed control architecture," *Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems (ISADS'07)*, 2007.

[12] A. Hauxthausen and J. Peleska, "Formal development and verification of a distributed railway control system," *IEEE Trans. on Software Engineering*, Vol. 26, No. 8, 2000, pp. 687–701.

[13] S. F. Fahmy, B. Ravindran and E. D. Jensen, "On collaborative scheduling of distributable real-time threads in dynamic, Networked Embedded Systems," *Proceedings of the 11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*, 2008, pp. 485–491.

[14] S. F. Fahmy, B. Ravindran, and E. D. Jensen, "Scheduling distributable real-time threads in the presence of crash failures and message losses," ACM SAC, Track on Real-Time Systems, 2008.

[15] G. C. Buttazzo, "Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications," Second edition, Berlin: Springer-Verlag, 2005.