A Traffic Management System for Real-Time Traffic Optimisation in Railways

Maura Mazzarello, Ennio Ottaviani
On AIR s.r.l.
Piazza Campetto, 2 - 16123 Genova, Italy
e-mail: maura.mazzarello@onairweb.com, ennio.ottaviani@onairweb.com

Abstract
The aim of this paper is to introduce the architecture, the approach and the current implementation of an advanced Traffic Management System (TMS) able to optimise traffic fluency in large railway networks equipped with either fixed or moving block signalling systems. The focus of the paper is on the core functions of the TMS, responsible for real-time conflict management and speed regulation.

Keywords
Conflict Detection & Resolution, Scheduling, Speed Regulation.

1 Introduction

With the increase in rail traffic, conflicts can arise between trains, even with small delays. If trains are forced to stop or slow down at a conflict point, in many cases, the result is a remarkable loss of time and energy,

The TMS here described addresses the problem of real-time traffic regulation and optimisation in railway networks equipped with different signalling systems (fixed or moving block) and consisting of multiple adjacent and interconnected local control areas. The TMS was developed under two subsequent EU projects, named COMBINE (e.g. [3] and [8]) and COMBINE 2 ([2]), and it was enhanced during application studies and a pilot project by ProRail. The first COMBINE project focused on real-time optimisation of rail traffic in areas equipped with moving block safety system compliant with ERTMS/ETCS Level 3. As a natural consequence, COMBINE 2 project expanded the TMS architecture and capabilities in order to optimise the rail traffic in large railway networks equipped with mixed signalling systems.

This paper concentrates on the core modules of the TMS architecture, which are responsible for automatic local traffic optimisation and control, respectively named Conflict Detection & Resolution (CDR) and Speed Profile Generator (SPG). The CDR is responsible for automatic real-time train scheduling and routing. It applies a model based on the so-called alternative graph formulation ([7]). The alternative graph is a powerful discrete optimisation tool, especially designed to deal with scheduling problems where the response time is a critical factor for the evaluation of the goodness of the approach. The SPG is responsible for plan execution. Operating strictly connected to CDR, SPG computes an optimal speed profile for each train, in order to make the CDR plan being executed in a safe and energy saving manner

We next introduce the rationale of the research, then sketch the TMS architecture, and eventually describe the conflict resolution model and the regulation model.

2 Research Framework

The scenario devised for the present TMS is a next future complex railway network with heterogeneous signalling systems. According to such scenario, it is reasonable that only a subset of existing or new main corridors will be equipped with moving block signalling systems (e.g. high speed lines). It is also envisaged that low costs positioning/communication technologies (GPS/GSM) will be adopted for enhancing traffic management effectiveness, while relying on fixed block signalling as far as safety issues are concerned (train separation). Probably, a number of secondary lines will be still managed through "traditional" fixed block signalling systems.

Given the different types of traffic (e.g. passengers vs. freight, international vs. regional or local, …), the different traffic regulation objectives (e.g. minimum travel time vs. punctuality with respect to the timetable or energy consumption reduction) and the different management approaches characterising such different types of lines, it is reasonable that responsibilities will be distributed among local control areas. Of course, as long as interactions among local control areas are possible, a co-ordination for the global network has to be granted by a network controller. This was the research framework of the COMBINE 2 project.

The previous COMBINE project produced a Demonstrator of TMS for local areas equipped with moving block signalling. An important test campaign carried out with the Demonstrator in a simulated environment enabled to identify and analyse the most significant technical and operational parameters, which affect the TMS feasibility and performance in a single (local) control area.

The main objective of the next COMBINE 2 project was to exploit the results and the insight derived from the previous experience, in order to address a more complex scenario, i.e. traffic management for railway networks equipped with different signalling systems (fixed or moving block) and consisting of multiple adjacent and interconnected local control areas.

For the developed TMS, primary constraints are the safety and operational rules. In particular, the TMS is compliant both with the ERTMS-level 3 safety system framework, and with the NS54 fixed block safety system (the safety system implemented by the Dutch railways).

3 TMS Architecture

The TMS here described is based on layered system architecture, in order to make it potentially suitable for any ERTMS/ETCS compliant system and for railway networks of any size. The complete TMS architecture is obtained by combining two basic concepts: decomposition of large networks in local areas, and modularity, in order to use a single approach for managing different signalling systems. In COMBINE 2 architecture each area is controlled by a local TMS, and several local TMSs are co-ordinated at a higher hierarchical level. Information exchange and collaboration in taking decisions between adjacent TMSs are critical issues when managing trains crossing the border between adjacent areas. To reach this objective, COMBINE 2 devised new algorithms for traffic optimisation and new methodologies for managing the interaction among TMSs controlling adjacent railway areas.

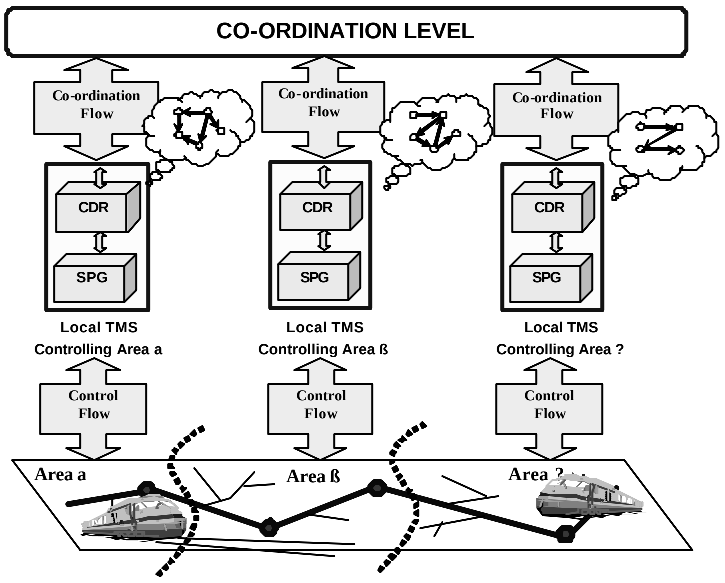

Figure 1: A network of local TMSs co-ordinated at a higher level

A hierarchy of modules compounds the TMS architecture (see Figure 1). The CDR module is responsible for automatic real-time train scheduling and routing. Given the current timetable, a set of constraints and the position and speed of each train in the area, the CDR automatically detects conflicts and creates a schedule of earliest/latest possible arrival times, departure times and speed for the trains at a set of key points. Key points for the optimal scheduling can be: possible conflict points, train target points, and other constraint points (e.g. speed restriction limits, slopes). The triple (arrival times, departure times, speed) for each train and for each key point is called a goal. The SPG system, which is at the lowest layer of the TMS, must then compute a speed profile for each train that will achieve all the goals in a safe and fuel-efficient manner. A Co-ordination Level is then necessary for large areas controlled by more than one single automated TMS. Actually, for wide and complex areas, the conflict resolution problem can be too large to be solved within the tight computing time requirements. In such cases, a decomposition strategy has to be used to detect and solve several well-structured sub-problems, of reasonable size, and weakly connected with each other. Then, efficient algorithms take care of solving the sub-problems, and co-ordination criteria ensure that the solutions found by the local solvers are globally feasible. The Co-ordination Level is based on an aggregate description of the interactions between trains in each local area. This aggregate information is passed from the local TMS to the Co-ordinator. This compact representation of precedence constraints in each area allows the co-ordination level to compute in a fast way whether the local solutions are globally feasible or not.

In this paper we mainly deal with local scheduling and regulation issues, thus providing a detailed description of CDR and SPG models, their current implementation and tuning deriving from real applications.

Thanks to the modularity of TMS architecture, CDR can operate as stand-alone module, as well as linked to SPG and other communication tools. As stand-alone tool, CDR generates optimal schedules, given different traffic scenarios. It was applied in studies for capacity estimation and operational control strategies at bottlenecks.

Connected to SPG and external modules, CDR exploits its capability to promptly react to traffic disruptions, making it possible a real-time flexible traffic handling. It was used and tested in a real-world pilot, shortly introduced in Section 6.

3.1 TMS Real-Time Operation

This section provides a description of how the research prototype COMBINE TMS was importantly adapted and improved, to be used in a real environment. Considering the adherence of TMS model to real environment, actually TMS takes account on the one hand of the position and speed of each train in the area, and on the other hand of the characteristics of the train such as traction, weight and length and of the characteristics of the infrastructure such as gradients, admissible speeds, signal positions and signal patterns. Other main improvements concerned the interface with the real world instead of simulators, and new conditions that the TMS should be able to deal with. Particularly, the following issues addressed crucial improvements in CDR-SPG:

- The process plan can change during a session (trains may be added or cancelled or existing information as e.g. scheduled times and local routes may be changed on-line).
- Route bookings have to be communicated to the dispatchers who may not accept them.
- Sudden infra or train degradations may occur.
- Both equipped (GPS + integrated GSM/PocketPC) and not equipped trains can drive through the area. It can also happen that a train is intended to be controlled, but for some reason the equipment does not arrive on time or is broken. Consequently, the train will be uncontrolled after all.
- TMS must be able to start-up in an 'active' environment, that is to say an environment in which trains are already in service.
- TMS must be able to deal with the fact that its plan can be disregarded:
  - Advisory speeds are not obeyed exactly by train drivers and sometimes are not obeyed at all
  - Changes in train precedence or route bookings are not properly implemented by dispatcher (also if previously accepted)
  - Communication delays between the TMS and the trains vary and messages may get lost occasionally.
  - Position and speed information from the trains are not exact.
  - TMS must be able to predict the driving characteristics of the trains in a very accurate way. Besides, the actual train characteristics may differ from the expected ones
  - Actual stopping times can be lengthened, with respect to the plan. Stop extensions can cause goal failures and request of an up-dated schedule

The main improvements introduced in CDR-SPG, in order to cope with the above requirements and to make the TMS more robust and efficient, are listed here.

Schedule windows. TMS performance should not be affected by the number of trains running through the controlled area. CDR guarantees good performance by dividing the global scheduling problem into manageable control windows, and then updating the solutions. A schedule window includes all the trains already present in the area at that moment, plus all the trains expected to enter the area within the *schedule-window-time*. The *schedule-window-time* depends on network and traffic complexity and it is connected to the update rate. It is set by a configuration parameter (typical values are in the range15-30 minutes). Schedule windows are partly overlapped, as each train is included into the schedule window for its whole running inside the area.

Updates. Updates are events triggering a new schedule from CDR. Both periodical (for instance every 15 minutes) updates and random updates are managed. By means of periodical updates new trains entering the current schedule window are taken into account in the next schedule calculation. Random updates are event-driven updates, triggered by perturbations to the schedule execution. Such perturbations derive from schedule failures, for instance due to unexpected events.

Route changes. CDR acts both on precedence relations and on routings. The re-routing problem is complex from a computational point of view. We apply a set of early pruning rules to reduce the search space. Besides, different routing alternatives are chosen by the expected gain in punctuality.

Co-operation between CDR and SPG. CDR and SPG are strictly connected in real-time operation. The co-operation between them was improved to make the reaction of TMS faster and more efficient as soon as unexpected events occur. SPG continuously evaluate the impact of changes in the current situation (new train position and speed, new goals, new constraints) and triggers the generation of a new schedule by CDR, when the current schedule is not still applicable.

Prediction SPG predicts the position and speed of the running trains, taking into account that, during the processing time trains keep running with the current advisory speeds, and that communication takes some time too. The prediction uses the last information received by SPG about train status and it is based on a detailed knowledge of the dynamic behaviour of the actual train configuration. The prediction is updated on a cyclic basis (about every 10 seconds). It copes with control loop delays, as well as missing or uncertain data (more details in Section 5).

Uncontrolled trains. Uncontrolled trains are trains that are not equipped at all, or trains not co-operative with TMS suggestions. Uncontrolled trains do not follow TMS speed advices. On the other hand, uncontrolled trains cannot be ignored by TMS. TMS predicts the behaviour of such trains simulating the normal train driver behaviour (drive at planned speed, if too late accelerate to maximum speed, if on time again go back to planned speed). TMS handles the impact of uncontrolled trains on controlled trains.

Driving tables. The TMS uses a number of detailed driving tables that describe the acceleration, deceleration or coasting curves of a certain type of train on a certain slope, and for each combination of relevant train types, slopes, driving modes. Three driving modes are considered: maximum acceleration, deceleration (applying service brake) or coasting (i.e. driving without application of traction force or braking force). Regarding freight trains, the exact characteristics of the trains in the controlled area cannot be described very well, so TMS predicts the behavior of the actual freight trains on the basis of the driving tables for a number of characteristic freight train compositions.

4 Model for Conflict Resolution

The CDR system is based on the alternative graph representation of the train movements. The alternative graph formulation, first introduced in Mascis and Pacciarelli ([6]), is an effective model for studying complex scheduling problems, arising in manufacturing, as well as in railway traffic control ([7]). With this formulation the variables of the problem are the starting times of the operations, i.e. the time at which a given train enters a given track element or block section. We denote by t_i the starting time of operation *i*, *i=1,…,n*.

Associating a node to each operation, the problem can be usefully represented by the triple *G=(N,F,A)* that is called *alternative graph*. The alternative graph is as follows. There is a set of nodes *N*, a set of directed arcs *F* and a set of pairs of directed arcs *A*. Arcs in the set *F* are *fixed* and f_{ij} is the length of arc *(i,j)* *Î* *F*. Arcs in the set *A* are *alternative*. If *((i,j),(h,k))* *Î* *A*, then *(i,j)* and *(h,k)* are said to be *paired* and *(i,j)* is *the alternative* of *(h,k)*. Let $\boldsymbol{a}_{ij}$ be the length of the alternative arc *(i,j)*. In this model the arc length can be positive, null or negative.

A *selection S* is a set of arcs obtained from *A* by choosing at most one arc from each pair. The selection is *complete* if exactly one arc from each pair is chosen. Given a pair of alternative arcs *((i,j),(h,k))* *Î* *A*, *(i,j)* is said to be *selected* in *S* if *(i,j)* *Î* *S*, whereas *(i,j)* is *forbidden* in *S* if *(h,k)* *Î* *S*. Finally, the pair is *unselected* if neither *(i,j)* nor *(h,k)* is selected in *S*.

Given a selection *S*, let *G*(*S*) indicates the graph *(N,FÈS)*. A selection *S* is *consistent* if the graph *G*(*S*) has no positive length cycles. Given a consistent selection *S*, an *extension* of *S* is a complete consistent selection *S'* such that *SÍS'*, if it exists. An important difference between the disjunctive graph and the alternative graph is that, given a consistent selection *S*, in the former graph always exists an extension of *S*, whereas in the latter graph may exist no extension of *S*.

With this notation each schedule is associated with a complete consistent selection on the corresponding alternative graph.

By definition, the *makespan* of a consistent selection *S* is the length of the longest path from node 0 to node *n* in *G*(*S*). The makespan of a schedule is therefore the makespan of the associated complete consistent selection.

4.1 Formulation of the Train Scheduling Model

The alternative graph is a powerful discrete optimisation tool, especially designed to deal with scheduling problems where the response time is a critical factor for the evaluation of the goodness of the approach. This method is fast and detailed at the same time, and it is able to include in the optimisation model all the relevant features and constraints needed to produce efficient and realistic train scheduling solutions.

In what follows a brief description of the alternative graph model for a rail network is given. The case of fixed block signalling system will be first addressed. The results will be then extended to deal with the moving block case at the end of this section. In order to define this model the railway network is modelled as a set of block section and signals.

A block section takes a given time to be covered, which is known in advance for each train. Clearly, besides the cover time, a delay may occur at the end of a block section if the signal is red or yellow.

Hence, a node in the alternative graph corresponds to the time at which a given train enter a given block section. In what follows we enumerate, from 1 to *n*, all the pairs (train, block section), and indicate with *B(i)* the block section associated with node *i*. With this notation, the variables of the problem are the times t_i at which the associated train enters the corresponding block section *B(i)*. Each t_i is computed as the longest path connecting node *0* to node *i*.

Special Nodes
Basically, the nodes of the graph are located at the entry (exit) of each block section. However, in order to include in the graph all the information needed to get a proper schedule, we have to add other nodes located where the infrastructure presents any kind of modification constraining the train behavior. So we put nodes even at switches and speed limitation borders.

Moreover, let us define a few nodes, which have a special meaning in the graph-building context.

Entry Node: this node represents the entrance of a train in the controlled area at a given block section (entry section). All the entry nodes are linked to a common Init Node (denoted by *0*) by a fixed arc carrying the expected entry time.

Exit Node: this node represents the exit of a train from the controlled area. All the exit nodes are linked to a common End Node (denoted by *n*) by a fixed arc carrying the planned exit time (with the sign changed). Doing so, the End Node receives a computation of the maximum expected delay.

Position Node: this node represents the current train position. It is inserted directly onto the train path and it is connected to the common Init Node by a fixed arc, carrying the current time of the train position measurement.

The Blocking Constraint
As stated above, we represent the blocking constraint with a pair of alternative arcs. Let us consider two operations (nodes) *i* and *j*, belonging to different trains, such that *B(i)=B(j)*. Since *i* and *j* cannot be processed at the same time, we associate with them a pair of alternative arcs. Each arc represents the fact that one operation must be processed before the other one. If *i* is processed before *j*, *B(i)* can host *j* only after the starting time of the subsequent operation *s(i)*, when *i* leaves *B(i)*.

Hence, we represent this situation with the alternative arc *(s(i),j)* with a suitable length *q*. Similarly, if *j* is processed before *i*, *B(j)* can host *i* only after the starting time of *s(j)*.

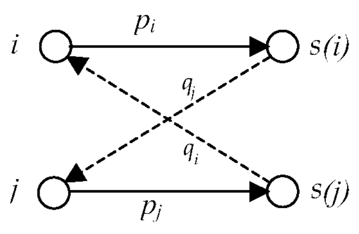

Figure 2: Blocking constraint

The length of an alternative arc can be easily evaluated in order to cope with practical safety issues. In our TMS implementation, this length is composed by a fixed positive term T_0 (for instance *30 sec*) plus an extra variable term T_1 that takes into account the length of the first passing train and its speed. This in order to model the real situation in which the head of a train may enter a block section only a given time after that the tail of the preceding train left it.

An example of a simple graph for two trains is reported in Figure 3.

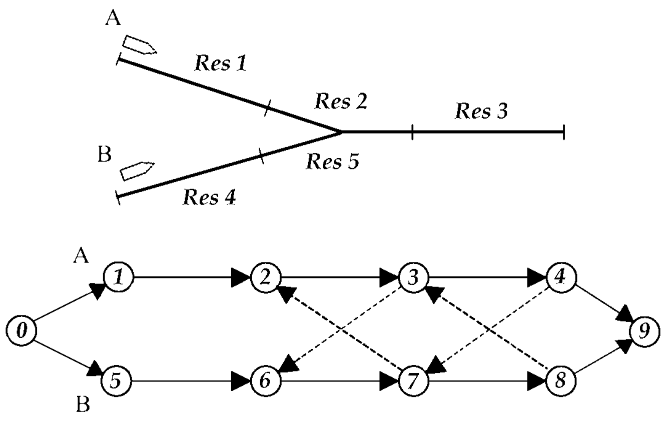

Figure 3: – A simple network and the associated alternative graph

Figure 3 shows two trains A and B running in a little network (a simple confluence). We introduce RA={Res1, Res2, Res3} and RB={Res4, Res5, Res3} as the ordered set of resources for train A and B. Note that Res2 and Res5 are special resources used to model the routings. They are different, but partially overlapped. Res3 is completely shared by both trains.

Let us show the meaning of all the nodes and arcs included in the graph:

- Node 0 is the common entry node, holding the zero time reference.
- Node 9 is the common exit node, where the maximum delay is computed.
- Arcs (0,1) and (0,4) defines the entry time of the trains.
- Arcs (4,9) and (8,9) defines the exit time (with the sign changed) of the trains.
- All the other fixed arcs define the minimum travel time of the trains over the corresponding resources.
- The alternative arcs (3,6) and (7,2) are paired, and so are the arcs (4,7) and (8,3).

So, by these definitions it is clear that:

- Nodes 1-2-3-4 represent the path of train A (node 1/4 store the entry/exit time)
- Nodes 5-6-7-8 represent the path of train B (node 5/8 stores the entry/exit time)

A conflict may arise if train A and train B try to use the shared resources violating the blocking constraint. A proper selection of one of the two alternatives ensures conflict resolution.

Constraints in Train Scheduling

In this section, we illustrate some examples of alternative graphs associated with typical constraints arising in train scheduling.

Minimum Speed Constraint: the constraint that a train must travel at a speed not lower than a minimum speed within a block section, corresponds to a maximum travel time d_i for the train within a given block section, i.e. to a maximum time allowed for completing the associated operation *i* and starting the subsequent operation *j*. Let us call p_i the processing time of operation *i*, i.e. the travel time associated to the *i-th* pair (train, block section), represented by a fixed arc. So the Minimum Speed Constraint is represented by another fixed arc in the opposite direction, with length $-d_i$. If $p_i > d_i$ a positive cycle occurs, so in order to guarantee feasibility of the solution the constraint must be satisfied.

Such constraint is commonly used when a railway line slopes up over a certain gradient. In such cases some freight trains should not decrease their speed under a certain limit, otherwise they would not be able to reach the top.

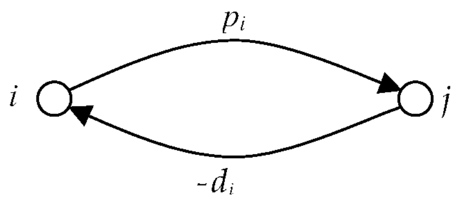

Figure 4: Minimum speed constraint

Passing Constraint: this constraint imposes that a train must pass through a node i only after a given time d_i. This is modeled by inserting a fixed arc joining the node i with the Init Node and carrying the passing time d_i as length. So t_i will be the maximum between d_i and the value computed by traveling along the train path.

This kind of inserted fixed arc is called a *target arc*, because it represents a requirement to be filled by the scheduling algorithm.

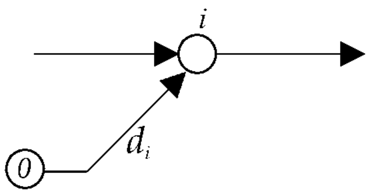

Figure 5: Passing constraint

Stop and Departure Constraint: stops are modeled with a pair of nodes i and j linked by a fixed arc.. The first node defines the arrival of the train at the stopping location, and the second node defines the departure of the same train from the stop. The arc joining the stopping nodes has a length p_{ij} defined by the *minimum stopping time* of the train at this location. Such time may be externally provided or set by default at a fixed value. Moreover, the departure node j is normally affected by a passing constraint, as the current departure time cannot be lower than the planned one.

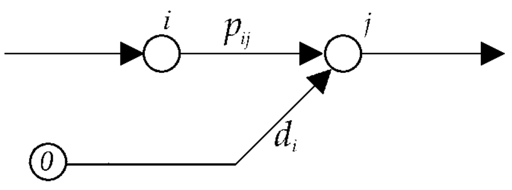

Figure 6: Stop and departure constraint

Connection Constraint: the connection constraint associated to a train departure can be handled very similarly to the previous one. The only difference is that the constrained arc now joins the first stopping node of another train. So train A can leave node j only a fixed time m_j (*minimum connection time*) after the arrival of train B at node k.

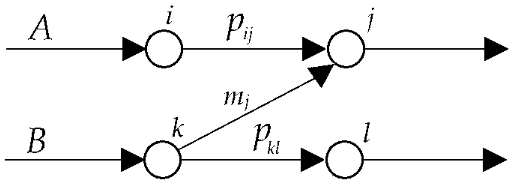

Figure 7: Connection constraint

Out of Order Constraint: if a block section is unavailable for trains in a given time interval, we can model this situation with a pair of alternative arcs. Let us suppose that the out of order section is delimited by nodes i and j. Therefore, a plan is feasible only if every train using that section satisfies one of these two possibilities:

- the train exits the section (node j) before the starting of the unavailable period
- the train arrives in the section (node i) after the end of the unavailable period

The alternative is represented by two arcs joining nodes i and j respectively and the common init node. A cycle occurs if passing time in node i and node j are not compliant with the chosen alternative. The same arc structure depicted in Figure 8 is repeated for all the trains using the unavailable block section.

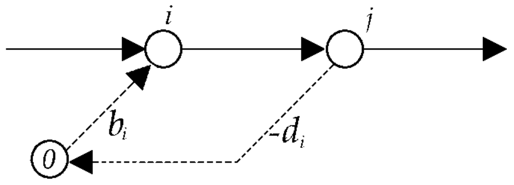

Figure 8: Out of order constraint

Precedence Constraint: The precedence constraint between train A and B (A must pass before B) is simply handled by changing the state of every pair of unselected alternative arcs connecting graph nodes belonging to A and B. We select all arcs directed from A to B and forbid all the paired ones. In this way the alternative is solved directly in the graph build phase.

Moving Block Signalling

The case of a moving block signalling system is now shortly addressed. A moving block section can be represented as a resource with multiple capacities in which two consecutive trains cannot enter simultaneously, but rather with a minimum time lag, whose value can be set in advance. Since the overtaking is not allowed within a resource, the model must represent this constraint.

The resulting model for a block section is then reported in Figure 9.

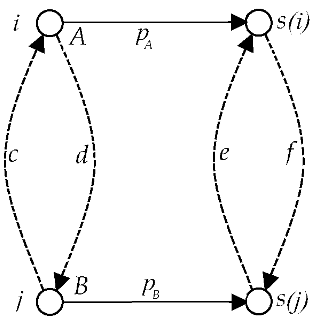

Figure 9**:** The alternative graph model for a moving block signalling system

In this case for each pair of trains, (A and B in the Figure) and for each resource two pairs of alternative arcs must be inserted, namely the pair (c,f) and the pair (e,d). The length of such arcs is precisely the time lag between the trains at the entrance/exit of the resource. For example, if train A precedes train B at the entrance of the resource, then B must enter at least d time unit after A, and therefore must exit from the resource at least f time units after A. Note that, in a feasible solution cannot be selected the arcs c and d or e and f, since otherwise there would be a positive length cycle in the resulting graph. Therefore, in a feasible selection must be selected either d and f OR c and e. Such way, the no overtaking constraint is satisfied.

4.2 The Scheduling Algorithm

The scheduling algorithm minimises a suitable function of exit delays, acting both on train precedence relations at conflict points and on train routings. The first high level heuristic consists in using sequence and routing hierarchically: firstly, precedence relations are analysed in order to minimise the objective function, then re-routing actions are applied, and finally new precedence relations are evaluated. It is a sub-optimal procedure, but it allows obtaining feasible and good schedules in a fast way.

In the following, the main tasks of the scheduling algorithm are introduced.

Create Plan

Given the timetable, possible constraints and the positions of all the trains inside the *schedule window*, CDR creates an optimised conflict-free schedule, by means of the alternative graph model. In order to create the schedule, the following steps are performed:

1. Minimal Schedule Generation. For each train a chain of nodes and arcs is generated. The chain represents the sequence of actions to be performed by the train (e.g. perform route x, enter track y, enter track z). A duration time is associated with each action: this time is evaluated assuming the train running at a constant speed (possibly the planned speed or the maximum speed allowed by the infrastructure) and without

taking into account any conflict. This computation takes also in account the dynamic characteristics of the train, the speed limits over the single tracks and the time targets at stations (considering the minimum stopping time), in order to provide a travel time estimates, which is as accurate as possible. The obtained schedule is called a *minimal schedule*.

2. Graph Pre-Processing. The following pre-processing task is performed: for each train its current position is considered in order to find out which set of precedence relations is already implied and cannot be changed. In other terms, we have to select all alternative arcs implicitly selected by the current train positions and to forbid all the paired arcs. All the other alternative arcs remain still unselected.
3. Graph Check. The set of constraints received with the plan is represented by means of a set of arcs that is added to the graph. A check is performed to verify if the graph contains negative length cycles. If so, a warning message has to be sent to the Dispatcher, stating that plan constraints do not match the current train positions.
4. Conflict Detection. A conflict occurs according to the following rule: IF train *A* enters resource *x* before train *B* AND train *B* exits from resource x before train *A*, THEN this is a conflict. IF no conflict is found, then EXIT (a feasible solution is found).
5. Conflict Resolution. The conflict is solved by processing in the graph a pair of arcs representing one of the following two constraints:
   - If *B* wins the conflict:
     - train *A* entry time in *x* is higher than train *B* exit time in *x*
   - If *A* wins the conflict:
     - train *B* entry time in *x* is higher than train *A* exit time in *x*

   That is to say, one of the alternatives for the resource *x* and trains *A* and *B* is selected. As a consequence, the other one is forbidden. Such decision has to be propagated in the graph, in order to ensure that a selection on resource *x* does not create a conflict with a subsequent selection on resource *y*. This is done in a similar way as in step 2 for the processing of the implicit precedence relations.
6. Termination/Continuation. IF the graph is cyclic THEN exit (found an unfeasible solution) ELSE goto 4.

Heuristics

How to solve the conflict (how to decide whether A or B wins) depends on local rules (e.g. which train is more delayed, which train has greater priority etc.). There are many examples of simple rules for solving conflicts (e.g. [6]). Solving a conflict corresponds to select one arc of an unselected pair of alternative arcs and to forbid the paired arc. Among all the possible methods to decide which arc has to be selected in order to optimize the makespan, we selected the so-called AMCC (Avoid Maximum Current C_{max}).

AMCC looks for the an alternative arc $(h,k) \in A$ such that:

$$l(0,h) + a_{hk} + l(k,n) = \max_{\{(u,v)\in A\}}\{l(0,u) + a_{uv} + l(u,n)\}.$$

In other words, (h,k) is an arc that, if selected, would improve most the length of the longest path in $G(S)$. Hence, AMCC select its alternative (i,j) as the best choice. A second order rule may be applied when the selection is not unique. In our test this heuristics outperforms the other listed in [6] and so it was selected as the most promising rule in the current implementation of conflict resolution in our TMS..

Other priorities are clearly possible, based on the importance of the train (e.g. high speed passenger trains first, then other passenger trains, then freight trains), even if such kind of static rules cannot use the information on the current status of the network, and therefore are not optimised.

Re-Routing

If the algorithm ends with a feasible solution, depending on the quality of this solution it is still possible to decide whether to keep the solution (if the quality is good enough) or to try to find a better solution by re-routing some trains.

Different routing choices are extracted from all the allowed routing alternatives, and are evaluated by the optimisation process. Of course in this phase it is not possible to explore exhaustively all the possibilities (i.e. all the combinations of routings), so a proper heuristic must be used.

The adopted heuristic consists in selecting carefully a single train at each turn, and running the standard optimisation for all the available alternatives of this train, and then storing the best solution. Then, if a timeout is not still reached, we explore another train and perform another optimisation, with the best routing for the previous train frozen. If the new solution is better than the previous one, the new one is stored as the best.

The approach is clearly sub-optimal, but it is surely feasible and effective with respect to time responses. The key aspect is to make a good choice for the trains with possible alternative routes, having in mind that only a few alternatives will be fully evaluated, and applying then a suitable heuristics. We look for a couple of trains joined by the most critical arc (i.e. an arc causing the highest increasing in train delay), and try to reroute these train firstly.

Post-Processing

When a final feasible solution is found, a post-processing task is applied on it, by using a time distribution algorithm allowing the evaluation of a time/speed pair (i.e. a goal) for each node of the graph. The aim of the post-processing task is to roughly estimate the train future behaviour in order to pass to SPG a feasible goal list.

This is necessary, as the conflict resolution method solves the conflicts by modifying travel times, but this may lead to unsatisfactory train behaviours. So the post-processing algorithm apply a simple dynamic programming scheme to each train in order to define a list of goals, which is compatible with the schedule and with the train manoeuvring capabilities (acceleration/deceleration curves), and moreover reduce as much as possible the number of speed changes. This implies for each node i the definition of how much the time t_i associated to this node may raise without creating a new conflict. If this time is r_i, the dynamic programming engine has to find a set of values for t'_i and v_i (corrected time and speed of the train at node i), such that $t_i<t'_i<r_i$ and v_i is reachable from t'_{i-1},v_{i-1}, where we intend the node $i-1$ as the preceding node of i for this train.

However the time/speed goals are intended only as an indication for the SPG, and they are always associated with definite time/speed tolerances, in order to let the SPG enough space for its own optimisation task.

5 Model for Speed Regulation

Most authors (see, for example, Kraay [5]) observe that, in practice, the energy consumption is minimised when all trains are operated at the lowest speeds consistent with their required performance levels. This simple argument is adopted by the SPG in order to generate the best speed profile enabling the train to reach the next goal in time. This task is quite simple, as the SPG computes train speed profiles after the delivery by CDR of a conflict-free plan. Such way, all meet-pass decisions have already been taken, and SPG can deal with all the trains in an independent way. The optimal speed profile must ensure the CDR plan being executed in a safe and energy saving manner.

5.1 The Control Loop

Aim of the speed regulation task is to control the train traffic in order to realise the current plan as well as possible and to save energy at the same time, while considering the constraints given by the safety system and the operational procedures. The main parameter affecting the speed regulation performance is related to the different communication delays in the control loop. The control loop delays, for a moving block case, are presented in
Figure 10.

When the TMS receives from the RBC speed and position data for the circulating trains, it calculates an advisory speed for each train. The computation time is a parameter t_1 depending on the TMS internal algorithms efficiency.

A delay t_2 is introduced by the RBC to communicate this advisory speed to the GSM-R system and another delay t_3 is the communication time needed by the GSM-R system to send this advisory speed to each driver. An important delay t_4 is associated with the driver reaction time. This parameter includes both the time needed to recognise that the advisory speed is changed, and the time needed to perform a reaction (like braking or accelerating). Other delays are generated by the train on-board systems responsible for estimating the train position and speed (parameter t_5) and by the GSM-R to communicate this information to the RBC (parameter t_6). Finally, a delay t_7 takes into account the communication time from RBC to TMS.

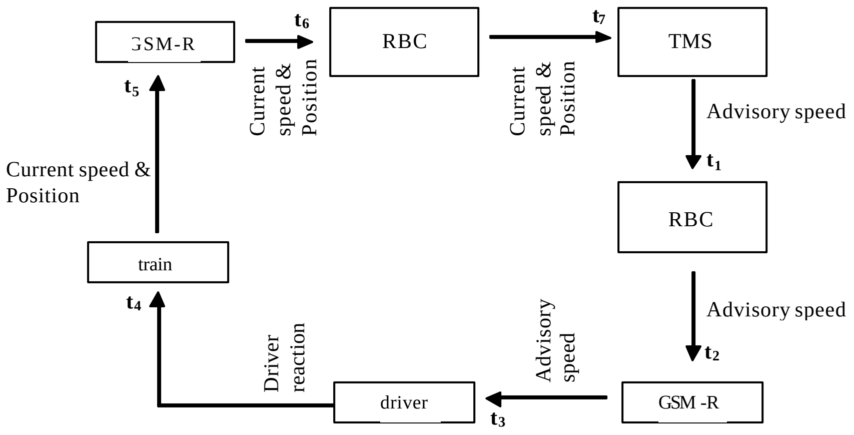

Figure 10: The control loop in ERTMS Level 3

An equivalent communication delay loop is present in the enhanced fixed block case (i.e. traditional fixed block + GPS/GSM).

Let t be the total delay of the control loop, defined by:

$$t = \sum_{i=1}^{7} t_i \; .$$

The parameter t is composed by a set of parameters that are all independent of the TMS, but parameter t_1.

From the viewpoint of the TMS performance, the only relevant parameter is the total delay t. In fact, let assume that at time t the positions of all trains in the rail network are detected and sent to the TMS. This information reaches the TMS at time $t+t_5+t_6+t_7$. Then, after a computing time t_1 by TMS, all the control actions (e.g., advisory speeds) are sent to the trains. The advisory speeds *relative* to the measures detected at time t finally reach the trains at time $t+t$. Note that different distributions of delays $t_1, \dots , t_7$ giving the same sum t will produce the same results since, in any case, the TMS computes the advisory speeds based on the measure detected at time t, while these advisory speeds reach the trains only at time $t+t$. The effect of the delay control loop could cause a reduced capacity of the TMS to control the trains. As a consequence, to reach a proper control of the trains, the TMS must *estimate* position and speed of the trains at time $t+t$,, on the basis of the knowledge of the position and speed of the trains at time t, and on an estimation of the parameter t. This computation introduces a further error in the detection of the real position and speed of the trains at time $t+t$. In other words, the effect of parameter t is similar to the effects of the other two parameters. More important, due to the control loop delay, an error in the estimation of train positions and speeds cannot be avoided, even if the real positions and speeds is detected with complete precision.

The next example shows that the control loop delay is in fact the most important parameter to be taken into account. Consider a train travelling at *180 km/h*, and assume that the position and speed of the train are known with complete accuracy, while the parameter t is known (for instance *20 sec*) with a reasonable error of ±*3 sec*. Then, the TMS must estimate the position and speed of the train at time $t+t$. Clearly, the train can vary its speed during time t, thus introducing an error in the estimate of the train speed at time $t+t$. This error equals the maximum speed variation for the train during the time t. Moreover, even if the train travels at constant speed *180 km/h* during time t, the error ±*3 sec* causes a position error equal to *150 m*. Note that the parameter t is usually known only stochastically, and therefore significant inaccuracy in the estimation of t easily dominates the position and speed errors caused by all the other technological parameters.

In COMBINE project many experiments were carried out to evaluate the performance of the TMS under different values of the technological parameters. The aim of the experiments was to analyse how technological parameters affect the TMS ability in controlling and optimising the traffic. As a conclusion of the experiments, the absolute control loop delay is very important and its acceptable value depends on the traffic patterns and network layout. However, the TMS operates in an efficient way even under severe traffic conditions and large loop delays (1 minute).

5.2 The SPG operating cycle

The main output of SPG is an advisory speed sent to each train whenever a change of speed is required. It is the result of an evaluation of the current plan goals and the current state of running trains and infrastructure. The SPG algorithm was implemented as a very fast numerical procedure, able to control the train speed in real time. It performs basically a sequence of the following steps: position/speed forecast - safety check - speed evaluation - route booking. Let us provide a brief description of all of them, before concentrating on the core of the speed profile generator algorithm.

The position/speed forecast task is necessary because of the total delay *t* existing in the control loop, and precisely the delay between the position measurement time and the advisory speed application time. If *t* is large, it is possible that a speed suggestion is no more optimal when the train receives it. Therefore, a forecasting step is needed to preserve optimality even with delayed control loops. Forecasting is performed simply propagating the current motion of the train for *t* seconds. If a speed change is currently in execution, we suppose it will continue in the next future until completion. Whenever a change of speed has already been sent to the train but not still applied, we suppose it will be applied as soon as it will be received. In other words, the forecasting step assumes the driver is co-operative and reacts after a given amount of time to speed suggestions.

The safety check ensures that each train has a movement authority (defined as the maximum space the train has already reserved for its motion) according to the specifications. This step is performed keeping in account the positions of possible interfering trains and, if required, the booking operations. If the safety check fails, a warning message is sent and a default (possibly zero) speed is suggested to the train. In normal operations, this is a very remote case, as goals provided by CDR are conflict-free.

The speed evaluation aims at finding the optimal speed profile between the current train position and the next goal position. This step will be described later in detail.

The route booking reserves the infrastructure in a timely way. Booking policy is strongly dependent on specifications, but in general it is preferable that a train books its route as late as possible, in order to let the infrastructure free for other trains. The latest time to book is determined by the time needed to stop the train at the current speed. Given a fixed SPG cycle time (say, *10sec*), we choose to book a route a few cycles before this limit. At the end of each cycle, SPG warns and activates CDR whenever a train goal could not be reached, providing also an estimate of the arrival time. This triggers the production of a new schedule by the CDR.

5.3 The Speed Profile Generator Algorithm

As anticipated, the algorithm for the SPG was conceived as a simple and efficient task, since the number of constraints on the routing and scheduling of trains, coming from the CDR, and the strict time constraints, did not allow complex computations. Hence, the adopted approach separately solves a problem for each train sequentially, starting from an *unconstrained* train, i.e. a train such that there are no other trains between it and the target. The key idea is that the energy consumption is minimised when the train runs at the lowest speed compatible with its target. Here, the target is the next relevant point (i.e. the next goal) for the train, given the current position. The target may be the first encountered target along the line, or a successive one if the first is too close to perform an efficient speed regulation.

Safety Check

A simple and very fast safety check is performed to verify if the train is able to stop at the end of its current Movement Authority. This check is very important in order to avoid that the RBC takes the control of the train with a safety braking. Clearly, this phase is performed differently in moving and fixed block technology, because the concept of Movement Authority changes between the two signalling modes.

With moving block technology the Movement Authority Length *(MAL)* is computed by looking at the preceding train and comparing the speed of the two trains. With fixed block technology, the MAL is computed by verifying the completion of booking operations on the next route to be used by the train.

Hence, the check is performed by comparing the needed space to stop the train at the current speed v, $S(v)$, with the current distance from the end of the Movement Authority. In practice the Safety Check consist in checking the following inequality:

$$S(v) < MAL + SP.$$

Here *SP* is a parameter (Safety Parameter) that depends on several technological parameters, such as the uncertainty in the train position and speed, the delay of the communication system in communicating the train position to the SPG, and the delay in implementing the actions required by the SPG.

The space $S(v)$ is simply computed using the deceleration curve in the dynamic characteristics of the train, so this check is very fast. If the inequality is TRUE, then the train can continue following its current speed (that can be updated by the next step). If the inequality is FALSE then the train has two possibilities:

1. if the MAL is limited, due to a preceding train (having smaller speed), then the current train will adapt its speed to the speed of the preceding train, i.e. the speed of the current train will become a bit smaller than the speed of the preceding train.
2. if the MAL is limited, due to a route that has not been booked/set yet, then the current train will start stopping.

Speed Evaluation

The speed evaluation task is the core function of the SPG. As stated before, the key idea is that the optimum speed is the lowest possible speed satisfying the constraints. With reference to the next goal to be reached, the objectives can be summarized as follows:

- an arrival time T_{fin} such that $T_{min}<T_{fin}<T_{max}$
- an arrival speed V_{fin} such that $V_{min} < V_{fin} < V_{max}$
- a speed profile in order to save energy as possible

So it is possible to define an optimisation problem like the following: "given a train with a current state (T_{cur},V_{cur}) and a goal defined by $(T_{min},T_{max}, V_{min},V_{max})$, at a known distance from current train position, find a feasible speed profile such that a suitable cost function C is minimum".

The feasibility of the speed profile implies not only a compatible arrival time/speed, but also a speed profile compatible with train dynamic characteristics (acceleration / deceleration curves).

In principle we can optimally solve this general problem by standard constrained dynamic programming techniques (e.g. [1]). Here we follow a sub-optimal approach in order to strongly reduce the involved computational complexity and to ensure real-time control. This must be guaranteed even when many trains are running simultaneously and a high sampling rate of train positions is required.

We reduce the class of available speed profiles to a class in which the distance between the current train position and the next goal is covered mostly at a constant speed V_{opt} (the advisory speed suggested by SPG). We admit at most two speed changes at the beginning and at the end of the time interval, in order to match the boundary conditions on speed. A sketch of a typical speed profile in shown in Figure 11.

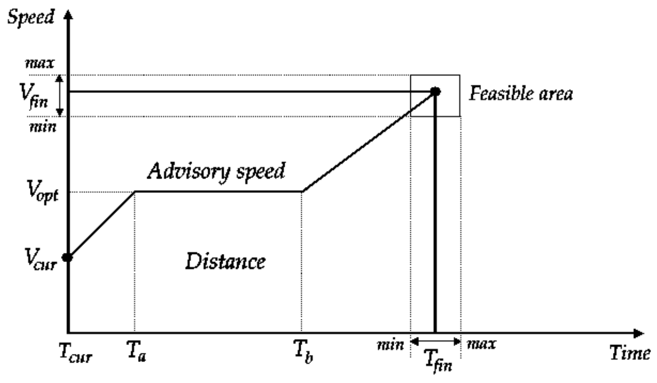

Figure 11: SPG Driving Model

In this approach the only degree of freedom is V_{opt}, so we can minimize C by acting on a single variable (one-dimensional problem). In fact, the two transient intervals are completely determined when V_{opt} is defined. The first transient is directly defined by the speed change V_{cur}- V_{opt}, whose duration is known. The second transient is determined by a simple argument. After the first transient, we compute the expected arrival time T_{exp} when the train moves at constant speed V_{opt}. There are three cases to consider:

- The couple (T_{exp}, V_{opt}) is inside the feasible area, so the second speed change is unnecessary.
- The train arrives too early, so deceleration is necessary in order to reach the goal. Among all the possible V_{fin}, starting from V_{opt}, we select the first one (the highest) ensuring feasibility of the goal (if any).
- The train arrives too late, so acceleration is necessary in order to reach the goal. Among all the possible V_{fin}, starting form V_{opt}, we select the first one (the smallest) ensuring feasibility of the goal (if any).

This simple argument avoids carrying on a full optimisation even for V_{fin}, leading to a cost function, which is a function of V_{opt} only. If a feasible value of V_{fin} does not exits, the current value of V_{opt} in skipped.

The range of available V_{opt} is bounded by the maximum speed of the train and the maximum speed allowed by the line. In addition, the values too close to V_{cur} should be skipped in order to avoid the suggestion of speed changes smaller than a given threshold (typically *10Km/h*). This does not imply any loss of control because of the high frequency of the SPG regulation cycle (about *10sec*).

Moreover, a quantization of the range of possible speed reduces the computation of C to a limited number of cases. Typically speed can be quantized in *5 Km/h* steps, without any loss of regulation capabilities.

The cost function C is composed by three terms, C_P, C_S, C_E. Each one reflects one of the three constraints listed above.

Punctuality term C_P – Given V_{opt} we compute the arrival time T_{arr} and then the delay $D=max(0,T_{arr}-T_{min})$. So we define

$$C_P = a\,D.$$

If $T_{arr}>T_{max}$ or $T_{arr}<T_{min}$, the value of V_{opt} is unfeasible.

Speed term C_S – Given V_{opt} we compute the arrival speed V_{fin}. So we define

$$C_S = 0,$$

if V_{fin} is inside the range (V_{min},V_{max}), otherwise

$$C_S = b\ max[(V_{fin}-V_{max}),(V_{min}-V_{fin})].$$

In this way the speed constraint may be slightly violated, depending on the parameter b. This is done in order to enlarge the feasibility region.

Energy term C_E – Given V_{opt} and V_{fin} we compute an empirical quantity mixing the amount of required changes of speed (a constant speed is preferred), and the speed itself (a lower speed is preferred). So we define

$$C_E = c\ [abs(V_{cur}-V_{opt})+abs(V_{fin}-V_{opt})]+d\ V_{opt}.$$

The weights used in all terms have been estimated by analysing typical requirements of the applications and they have been fine tuned during a simulation phase. However, the meaning of the terms is physically simple, and so adaptation to specific requirements is not a complex issue.

6 TMS in Action

The TMS here described was applied to the pilot “The Green Wave”, actually called “de Groene Golf” (dGG). The pilot was a first attempt by the Strategy and Innovation department of ProRail to test Dynamic Traffic Management (DTM) in practice.

One of the objectives of the pilot was determining whether the TMS actually generates the results (increased punctuality and energy savings) predicted on the basis of calculations and simulations. The pilot was carried out in June 2004. The pilot area involved the routes Roosendaal-Dordrecht and Breda–Dordrecht, including the junction at Lage Zwaluwe, particularly susceptible to delays.

For the pilot, the TMS communicated with a Tracking & Tracing system (T&T) and the Procesleiding system (VPT-PRL).

The effects of the TMS were demonstrated using a number of cases occurring during the pilot and analyzing the precise course of events. In some cases, it was verified that major problems caused by trains running a few minutes late, would in fact have been prevented by taking up the TMS plan and speed advices.

7 Conclusions and Future Work

In this paper we introduced the model and the current implementation of the core functions of an advanced TMS. The proposed systems can be used both separately, as scheduling and regulation layers, and integrated in the complete TMS.

Currently we are working on further improvements both in TMS core functions and in

TMS interfaces. Particularly, it is under investigation the use of abstract interfaces and standard formats for efficient data exchange among applications (e.g. RailML® standard in [4]). The aim is to make the TMS much more independent of the chosen implementation of the surrounding systems and more focused on the core functionalities.

References

[1] Askey S., Sheridan, T., *Safety of High Speed Ground Transportation Systems - Human Factors Phase II: Design and Evaluation of Decision Aids for Control of High-Speed Trains: Experiments and Model,* National Technical Information Service, Springfield, VA 22161, 1996

[2] Giannettoni M., Savio S., "The European project COMBINE 2 to improve knowledge on future rail traffic management systems", In: Allen, J., Hill, R.J., Brebbia, C.A., Sciutto, G., Sone, S. (eds.), *Computers in Railways IX*, pp. 695-704, WIT Press, Southampton, 2004

[3] Giannettoni M., Savio S., "Traffic management in moving block railway systems: the results of the EU project COMBINE", In: Allen, J., Hill, R.J., Brebbia, C.A., Sciutto, G, Sone, S. (eds.), *Computers in Railways VIII*, pp. 953-962, WIT Press, Southampton, 2002

[4] Huerlimann D., Nash A, Krauß V.P., Schütte J., "RailML – "A standard data interface for railroad applications", In: Allen, J., Hill, R.J., Brebbia, C.A., Sciutto, G., Sone, S. (eds.), *Computers in Railways IX*, pp. 233-240, WIT Press, Southampton,.2004

[5] Kraay D. R., Harker, P. T., *Real-Time Scheduling of Freight Railroads,* Transportation Research Part B, 29: pp. 213-229, 1995

[6] Mascis A., Pacciarelli D., *Machine Scheduling via Alternative Graphs*, Report DIA-46-2000, Dipartimento di Informatica e Automazione, Università Roma Tre, Roma, 2000

[7] Mascis A., Pacciarelli,D., Pranzo, M., *Models and Algorithms for traffic management of rail networks*, Report DIA-74-2002, Dipartimento di Informatica e Automazione, Università Roma Tre, Roma, 2002

[8] Pellegrini F., Savio S., "Moving block and traffic management in railway applications: the EC project COMBINE", In: Allen, J., Hill, R.J., Brebbia, C.A., Sciutto, G., Sone, S. (eds.), *Computers in Railways VII*, pp. 11-20, WIT Press, Southampton, 2000