Algorithms for Self-Organizing Railway Operations

Grant Agreement N°:	875022
Project Acronym:	**SORTEDMOBILITY**
Project Title:	Self-Organized Rail Traffic for the Evolution of Decentralized MOBILITY
Funding scheme:	Horizon 2020 ERA-NET Cofund
Project start:	1 June 2021
Project duration:	3 Years
Work package no.:	3
Deliverable no.:	2
Status/date of document:	Final, 30/11/2023
Due date of document:	30/11/2023
Lead partner for this document:	univEiffel
Project website:	www.sortedmobility.eu

Dissemination Level		
PU	Public	**X**
RE	Restricted to a group specified by the consortium and funding agencies	
CO	Confidential, only for members of the consortium and funding agencies	

Revision control / involved partners

Following table gives an overview on elaboration and processed changes of the document:

Revision	Date	Name / Company short name	Changes
1	10/11/23	Federico Naldini, Bianca Pascariu, Paola Pellegrini - univEiffel	Chapter 4
2	10/11/23	Leo D'Amato, Fabio Oddi, Vito Trianni, ISTC-CNR	Chapters 2, 3, 5, 6, 7
3	20/11/23	Federico Naldini, Bianca Pascariu, Paola Pellegrini - univEiffel	Complement to Chapter 7, Chapters 1, 8
4	30/11/23	Paola Pellegrini	Finalization

Following project partners have been involved in the elaboration of this document:

Partner No.	Company short name	Involved experts
1	univEiffel	Federico Naldini, Bianca Pascariu, Paola Pellegrini
2	SNCF	Rémy Chevrier
3	DTU	Carlos Lima Azevedo
4	BDK	
5	ISTC-CNR	Leo D'Amato, Fabio Oddi, Vito Trianni
6	RFI	
7	TUDelft	Egidio Quaglietta

Executive Summary

The objective of D3.2 "Algorithms for Self-Organizing Railway Operations" is to detail the algorithm design and implementation produced in the SORTEDMOBILITY project for achieving railway traffic self-organization.

Chapter 1 introduces the deliverable.

Chapter 2 summarizes the process designed to achieve self-organization. The various modules composing this process are described in the following chapters.

Chapter 3 focuses on train neighborhood identification.

Chapter 4 details the hypothesis generation performed by each train.

Chapter 5 describes the hypothesis compatibility check carried out by each train, considering the hypotheses of its neighbors.

Chapter 6 proposes the consensus approach trains apply to agree on traffic management evolution.

Chapter 7 presents how the selected hypotheses are merged into a valid RTTP.

Chapter 8 concludes the deliverable.

Table of contents

Table of figures

Table of abbreviations

rtRTMP	real-time Railway Traffic Management Problem
RTTP	Real Time Traffic Plan
TDS	Track Detection Section
TSP	Traffic State Prediction

1 INTRODUCTION

In this deliverable, we detail the algorithms we developed within SORTEDMOBILITY to implement railway traffic self-organization.

Traffic self-organization is achieved through the application of a modular process, designed within the project and presented in Deliverable 3.1 "Design Choices for Self-Organizing Railway Operations". This process is summarized in Section 2, where all the modules involved are introduced. In particular, the various modules are described specifying their objective, role and working principles.

The following sections include the description of the specific algorithms we designed for each module, including the specifications of the current implementations. In particular, Section 3 pertains to neighborhood selection, Section 4 and 5 to the hypothesis generation and compatibility, Section 6 to the consensus achievement and Section 7 to the merge operation among the resulting hypotheses. Finally, Section 8 reports some conclusions.

2 SUMMARY OF THE SELF-ORGANIZATION PROCESS DESIGN

In this section, we describe the principles we set in SORTEDMOBILITY for designing the traffic controller to achieve self-organized decisions making. The controller we propose may be deployed in a totally unsupervised manner, with the automatic implementation of agreed decisions. Otherwise, it may be deployed with human interaction. For example, a dispatcher may be in charge of validating the decisions before implementation. We think this would somehow go against the nature of self-organization, but it may be pertinent from an industrial perspective. Another option, which may be a good compromise between the two extremes, may see dispatchers taking charge of particularly critical situations, and automatic implementation in normal operations. The overall process design is out of the scope of the project, but we think the proposed design is compatible with most envisageable options.

We formalize the controller decision making as a modular process, in which four sub-processes are to be carried out by single trains and a final one is performed by the traffic control center. Figure 1 depicts this process.

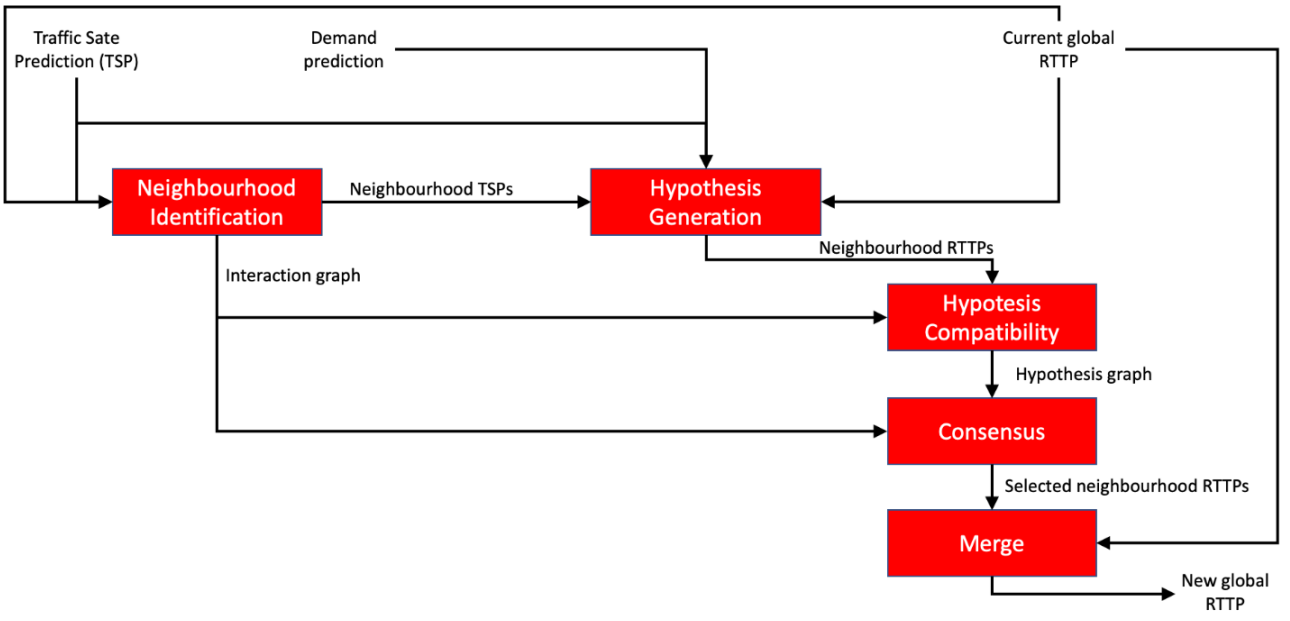

Figure 1. Pipeline representing the self-organization process.

As detailed in Deliverable 4.1 "Data Exchange Format and Software Interfaces", the process starts with the reception of the current Real-Time Traffic Plan (RTTP) and Traffic State Prediction (TSP), along with the prediction of the passenger demand. The timings in the RTTP are corrected to be consistent with the current TSP: if an unexpected perturbation occurs, while train routes and passing orders of the RTTP are deployed, the specific passing times will be different from the expected ones. We do this correction with a variant of RECIFE-MILP, an algorithm originally proposed for the real-time railway traffic management problem (rtRTMP) (Pellegrini et al., 2014, 2015). This algorithm identifies good - optimal if enough time is available - train routes, timings and passing orders given a control area and a traffic situation. The control area infrastructure is represented microscopically, considering detailed track and interlocking system characteristics. Train length is also considered when making decisions. The classic RECIFE-MILP formulation manages traffic considering fixed-block systems, but a variant of the formulation has been produced to consider moving-block ones (Versluis et al., 2023). RECIFE-MILP exploits a

commercial mixed-integer linear programming (MILP) solver, typically CPLEX, to make traffic management decisions in two steps. First, a single route is considered for each train, and only timings and passing orders are optimized. This route can be the one planned in the timetable or one chosen in a previous optimization run. Second, routing decisions are added to the panel of possible actions, considering the solution of the first step as a starting point. The algorithm stops when either a solution is proven to be optimal, or the available computational time has elapsed. To correct the RTTP timings, only the first step of the algorithm is performed, and additional constraints are set to ensure the passing orders of the RTTP are preserved. This procedure can be carried out either at the control center, or by each train independently.

The RTTP and the TSP are received by each train, which operates a neighborhood identification to select the set of trains with which it may have an interaction. Various types of interaction may be taken into account here, depending on the modeling choices made for the real-time Railway Traffic Management Problem (rtRTMP) (e.g., including or excluding rerouting possibilities) and the process design (e.g., frequency of traffic monitoring).

Once each train has identified its neighborhood, it performs the hypothesis generation: it generates a set of traffic management strategies - hereafter, hypotheses - in which changes are defined for both the path of the train itself and the ones of the trains in its neighborhood. The characteristics of these hypotheses depend on the modeling choices made for the rtRTMP: hypotheses may differ in train routing, ordering, timing, speed profile, passenger transfer possibilities, etc. To produce and associate a value to these hypotheses, the train may rely on the prediction of the actual demand related to the different traffic management strategies: in particular, the train may take into account the impact of each strategy on passenger utility, in terms of, e.g., waiting and travel times.

On the basis of the generated hypotheses, the interactive consensus process takes place. Each train shares its hypotheses with the other trains, possibly keeping their values as private information, and receives the hypotheses of its neighbors. It carries out a *hypothesis compatibility* check to identify which hypotheses are consistent and could hence be implemented concurrently. In

principle, various permissive definitions of consistency may be considered for hypothesis compatibility, depending on rtRTMP modeling and decision process design. Finally, each train engages with its neighbors for the hypotheses selection, trying to identify its best hypothesis while preserving consistency with the hypotheses selected by its neighbors.

After a consensus is reached or after a maximum time available for decision making has elapsed, the trains return the currently selected hypotheses. The traffic control center receives all hypotheses and applies a merge operation, aggregating them into a complete RTTP and possibly solving any remaining incompatibility. In particular, incompatibilities may arise in the long term, between paths of trains not currently identified as belonging to the same neighborhood, or even in the short term if permissive hypothesis compatibility principles are defined. In this case, a reparation strategy must be applied to obtain a feasible RTTP, as close as possible to the one obtained by merging hypotheses.

3 NEIGHBOURHOOD IDENTIFICATION

The first step for the decentralized self-organization of rail traffic is the identification of the set of neighbors relative to each train in the network. We define a neighbor of a train t as any train t' that may be in conflict with t within a time horizon h, starting at the current time.

The algorithm we propose takes as inputs the current RTTP and TSP, the former being updated with newly discovered perturbations as discussed in Section 2.

The produced output consists of a set of modified TSPs, one per train, in which neighboring trains are identified through an additional XML attribute named inNeighborhood, whose value can be false or true.

In order to identify its neighbors, each train exploits a representation of the rail network consisting in a graph of connected TDSs extracted from the infrastructure description (see Deliverable 4.1 "Data Exchange Format and Software Interfaces"). Each TDS is tagged with the ID of the trains that may occupy it, along with the expected occupation times and the route information. To calculate the expected occupation times, we add up the running time of each

TDS along the considered route on top of the current time. If a train has multiple routes that include the same TDS, all occupation times are considered.

Once the mapping of trains to TDSs is complete, each train checks all the TDSs it may use within the time horizon h (i.e., the occupation time is included between the current time T and $T+h$. It identifies as neighbors all the other trains that may use one or more of these TDS within the same time horizon.

The identification of the neighborhoods leads to the generation of the so called *interaction graph* $G_T=(v_T,\varepsilon_T)$. Here, v_T is the set of nodes, each representing a train, and ε_T is the set of edges: an edge connects two nodes corresponding to trains that are in the neighborhood of each other.

4 HYPOTHESIS GENERATION

The hypothesis generation module is based on the RECIFE-MILP formulation. Within SORTEDMOBILITY, traffic management solutions are assessed both from the perspective of railway operations and from the point of view of passengers and freight operators, if the case study involves mixed traffic.

To do so, we propose a variant of RECIFE-MILP, in which the demand prediction module detailed in D2.2 "Data-Driven Operational Prediction Model" and a passenger assignment module are integrated into the solution process. In particular, as in classical applications and as mentioned in Section 2, the future traffic state is considered (supplied by a simulator or a prediction module in the form of a TSP, as described in D4.1 "Data Exchange Format and Software Interfaces"). Differently from what is typically done in railway traffic management, a demand prediction is also produced, consisting of a distribution of passengers per origin-destination (OD) pair for the near future. These passengers are assigned to trains, assuming the latter follow the route and passing order decisions of the RTTP being considered. This can either be the one representing the original timetable, or the one representing the last traffic management decisions made.

The passenger assignment is based on two main elements: passenger incidence times and train-based path choices. The former is the arrival time of a

passenger at their origin station. We predict the incidence times by first predicting OD matrices for short time intervals (e.g., five minutes), and then distributing the predicted number of passengers using waiting time distributions from the literature. In particular, following the study of Eltved et al. (2019) for the Copenhagen suburban railway network, we infer the incidence time distribution considering the waiting time distribution known for each station, and knowing the number of people arriving at a station in a time interval.

For each OD pair and incidence time, a set of available train-based paths can be identified by looking at the RTTP. This set includes all train combinations that the passenger may want to use, looking at historical data. Historical data also say with what probability a passenger will use each train-based path. We use this probability to compute the expected number of passengers on each train and the used train connections.

All passengers with common train-based path choice and incidence time compose a passenger group.

For each passenger group, we estimate the desired arrival time as the time at which they would reach their destination if all trains were traveling on time.

Traffic management decisions are made considering the so obtained passenger assignment to trains. In particular, additional variables and constraints are added in RECIFE-MILP to impose the used train connections as soft constraints: it is possible to drop a connection if the feeder train is late, but the quality of the solution takes it into account. In particular, the objective function minimizes the sum of train delay at destination (representing the railway operations perspective) and of passenger delay at destination, with respect to their desired arrival time. A penalty is added to this sum if connections are dropped, and this penalty is a function of the frequency of the trains on the lines the passengers who miss the connection are traveling on, to represent the delay they will suffer.

With these additional variables, RECIFE-MILP generates a number of solutions. New passenger assignments are computed for each of these solutions, and they are used in a final assessment to rank the different solutions.

The formal details of the formulation and algorithm accounting for passenger demand are available in Pascariu et al. (2023).

To use RECIFE-MILP for hypothesis generation, we emulate the fact that it is executed by a specific train to find alternative traffic management options for itself and its neighbors. The specific train running RECIFE-MILP is the focal train. Before starting the optimization, we set as constraints all routes and passing orders of trains not included in the neighborhood of the focal train, as they appear in the input RTTP. Hence, the focal train will only be able to propose traffic management decisions modifying the previously planned paths of either itself or its neighbors. In the objective function, depending on the case study, we either weigh the delay of all trains equally, or we allow each focal train to use private information to weight its own delay differently from the one of the others.

In each run, to generate a set of hypotheses, not only the optimal solution is searched, but also a panel of different ones. To do so, in the second step of the algorithm, we run CPLEX using either '.populate()' or `.solve()' with the 'MIP::Pool::RelGap' parameter set to p: the best solution found in the available computational time is returned, together with all the other ones found which are less than p% worse than the former, p being an input parameter. These solutions are screened to eliminate equivalent ones, in which timings change but routing and passing orders are the same. For each set of equivalent solutions, only the one with the earliest train trips is kept. As the maximum number of hypotheses returned for each train (n) is an input parameter, only the RTTPs corresponding to the best n-1 solutions become actual hypotheses. In addition, the input RTTP is returned as a hypothesis, considering that, in the worst case, preserving the traffic management decisions currently being implemented is a feasible option.

The cost associated with each hypothesis corresponds to the value of the objective function optimized by the focal train. If the latter uses private information to assess solutions, then it is possible for two trains with the same neighborhood to independently generate one or more identical hypotheses but attribute different values to them.

5 HYPOTHESIS COMPATIBILITY

The hypothesis compatibility module checks if two neighboring trains t_1 and t_2 have got compatible hypotheses. If this is not the case, there is no solution to the consensus problem. This can happen when the individual neighborhoods of t_1 and t_2 are largely different, so that the two trains may optimize two different traffic situations. To reduce this effect, we allow for *hypothesis sharing*. During this phase, each train delivers its own hypotheses to its neighbors. For example, consider the case of three trains: t_1 with the hypothesis h_1, t_2 with the hypothesis h_2 and t_3 with h_3, where $(t_1,t_2)\in\epsilon_T$ and $(t_2,t_3)\in\epsilon_T$. After the hypothesis sharing, the train t_1 will have the hypotheses $h_1$and h_2, the train t_2 will have the hypotheses h_1,h_2 and h_3 and the train t_3 will have the hypotheses $h_2$and h_3. Sharing is performed only once to ensure that the hypothesis graph is connected, while avoiding that the same hypothesis diffuses throughout the whole network. In the example above, this means that h_1 will not reach t_3, because t_2 only shares its own hypotheses and not those received from other trains.

Afterwards, the compatibility check between hypotheses takes place. The compatibility is evaluated for all pairs of trains connected in the interaction graph, and among all their hypotheses. In practice, given two trains t_1 and t_2, where $(t_1,t_2)\in\epsilon_T$, for any hypothesis of t_1 the compatibility is checked with any hypothesis of t_2. Clearly, a shared hypothesis is by default compatible with the original one.

The compatibility check is performed in the following way. Given two different hypotheses h_1 and h_2, the route of each train $t_{i,2}\in h_2$ is replaced with the route of the same train $t_{i,1}\in h_1$. If the utilization times of the replaced route overlap with at least another one in h_1, then h_1 and h_2 are declared incompatible. In case of no overlap, the original route of h_2 is restored and the next train is checked. If all the trains' routes are tested and no incompatibility is detected, then the roles of h_1 and h_2 are swapped and another check is performed. After the second round of compatibility check, if no incompatibility is detected, h_1 and h_2 are declared compatible. As an alternative implementation, we perform the same procedure but considering only the route of the focal train, to be checked within the hypothesis of the neighbor train.

Following the compatibility check, the hypothesis graph $G_H=(v_H,\varepsilon_H)$ is created. Here, v_H is the set of all hypotheses from all trains, while ε_H represents the set of edges, each connecting two compatible hypotheses. Finally, a cost c_h is associated to each vertex $h \in v_H$. This is the cost of the hypothesis as returned by the hypothesis generation. If the hypothesis has been shared by a neighbor train, its cost is computed from the point of view of the receiving train.

6 CONSENSUS

Self-organization is achieved through a consensus process, i.e. an iterative procedure whose goal is to converge to a good, possibly optimal, solution to a given problem. In SORTEDMOBILITY, during the consensus process, each train aims at selecting the best possible hypothesis that is compatible with the one chosen by each of its neighbors.

The state of a train t_i is represented by the hypothesis $h_\square \in H_i$ currently selected. At start, each train selects its "best" hypothesis, i.e. the one with the minimum cost, and the consensus process starts. At each iteration, one single train, the decision maker t_i, makes a decision about the hypothesis to promote while all the other trains remain in their respective states. To this end, the decision maker t_i makes an observation and consequently makes an action. The observation consists in observing the state $h'_\square \in H_j$ of one randomly selected neighbor $t_j \in N_i$, while the available actions for t_i are either keeping its hypothesis h or switching to another hypothesis in H_i. On the basis of this observation,t_i applies the following default policy:

- If h and h' are compatible OR t_i has only one hypothesis, then t_i keeps hypothesis h.
- If h and h' are not compatible ANDt_i has other hypotheses that are compatible with h', then t_i switches to the hypothesis with minimum cost among these compatible hypotheses.
- If h and h' are not compatible ANDt_i has no other compatible hypotheses with h', thent_i switches to its hypothesis with minimum cost.

The process described above is repeated at each time step and stops either when all the trains have selected a hypothesis that is compatible with the

hypotheses of all their respective neighbors or when a maximum number of time steps is reached.

The above basic policy is designed to promote consensus as much as possible, because trains do tend to align to the hypotheses chosen by their neighbors, even disregarding the cost of the chosen hypothesis. In other words, trains are cooperative, as they would select even costly hypotheses in the attempt to reach an agreement with neighbors. Such cooperativeness does not support the choice of optimal solutions, however, because any consensus state would be acceptable. We have therefore introduced the probability of cooperation to reduce the compliance of trains in changing their state in favor of neighbors. The decision makert_i determines whether or not to change its state following a probability of cooperation, which depends on the normalized cost $C_h \in [0,1]$ of the hypothesis h currently selected by t_i (where $C_h=0$ for the best hypothesis in H_i and $C_h=1$ for the worst one) and on its level of competitiveness α:

$$p(cooperate \vee C_h, \alpha) = max$$

The idea behind this definition is that the higher the cost C_h, the more likely for the train to take part in the consensus process and possibly replace the selected hypothesis h with a better one. Similarly, the higher the level of competitiveness α, the lower the probability of cooperation of the train as long as it holds a good hypothesis. We define three types of trains according to the value of α:

- base train: fully cooperative train ($\alpha=0$) that always tries to adapt its hypothesis to the neighbors.
- fixed train: it has $\alpha>0$ and when it cooperates it implements the default policy.
- adaptive train: it starts with $\alpha>0$ but then α converges to zero, iteration by iteration.

With these three setups, we can obtain different system behaviors, trading-off consensus time with optimality of the selected solution.

7 MERGE

After the consensus module, the RTTP merging phase begins. Here, the hypothesis selected by each train is used to build up the new global RTTP

starting from the currently deployed one. In particular, for each train that belongs to a connected component in which the consensus is reached, its path is taken from its selected RTTP (hypothesis). This path is used to replace the one corresponding to the same train in the current RTTP. For the trains belonging to a connected component that has not reached consensus, no path change is applied.

In the end, consistency checks are performed over the merged RTTP, searching for TDSs with overlapping utilization times. Indeed, after the merge of various RTTPs, it is possible that the route of different trains partially overlap. In particular, this may happen for trains which use common TDSs in the future, at a time farther than the length of the time horizon considered for determining neighborhoods. If this happens, the RTTP must be repaired before being deployed. To do so, we use a variant of RECIFE-MILP, in which we set as objective the minimization of the number of passing order changes. Specifically, the merged RTTP including stairway overlaps is supplied as input to the new variant of RECIFE-MILP. Here, routes cannot be changed, and a binary variable is associated to each existing one, representing a passing order. Additional constraints are introduced to set this variable to zero if the passing order of the input RTTP is preserved, and one if it is changed. The objective is finding an operationally feasible solution minimizing the sum of these new variables. After this, if any change was necessary, new timings are found with the RECIFE-MILP variant described in Section 2, given the new passing orders.

8 CONCLUSIONS

In this deliverable, we reported the details of the algorithm design and implementation produced in the SORTEDMOBILITY project for achieving traffic self-organization.

The implementations proposed are initial proposals of each module, which will make the object of development in future studies.

These initial proposals are used in the experimental analysis of SORTEDMOBILITY, allowing the first implementation of traffic self-organization combined with a thorough simulation assessment.

9 REFERENCES

M. Adnan, F. Pereira, C. Lima Azevedo, K. Basak, M. Lovric, S. Raveau, Y. Zhu, J. Ferreira, C. Zegras, and M. Ben-Akiva. (2016) SimMobility: A multi-scale integrated agent-based simulation platform.

M. Eltved, O.A. Nielsen, and T.K. Rasmussen. (2019) An assignment model for public transport networks with5

both schedule- and frequency-based services, EURO Journal on Transportation and Logistics, 8(5), 769-79.

ERA. (2023) Getting rail freight on the right track. URL: https://www.era.europa.eu/content/getting-rail-freight-right-track. Accessed October 18th, 2023.

ERJU. (2022) Europe's rail. URL: https://rail-research.europa.eu/. Accessed October 20th, 2022.

IEA. (2019) The future of rail. URL: https://www.iea.org/reports/the-future-of-rail. Accessed October 18th, 2023.

H. Hamann and A. Reina. (2022) Scalability in Computing and Robotics. IEEE Transactions on Computers, 71(6):1453–1465. doi: 10.1109/tc.2021.3089044

ONTIME. (2011) Optimal networks for train integration management across Europe. URL: https://cordis.europa.eu/project/id/285243. Accessed October 20th, 2022.

B. Pascariu, J.V. Flensburg, P. Pellegrini, C. Lima Azevedo. (2023) Formulation and solution framework for real-time railway traffic management with demand prediction. Submitted. Please contact the authors for details.

P. Pellegrini, G. Marlière, R. Pesenti, and J. Rodriguez. (2015) RECIFE-MILP: an effective MILP-based heuristic for the real-time railway traffic management

problem. Intelligent Transportation Systems, IEEE Transactions on, 16(5):2609–2619.

P. Pellegrini, G. Marlière, and J. Rodriguez. (2014) Optimal train routing and scheduling for managing traffic perturbations in complex junctions. Transportation Research Part B: Methodological, 59C:58–80.

E. Quaglietta. (2013) A simulation-based approach for the optimal design of signalling block layout in railway networks. Simulation Modelling Practice and Theory, 46. doi: 10.1016/j.simpat.2013.11.006

G. D. M. Serugendo, M.-P. Gleizes, and A. Karageorgos. (2006) Self-Organisation and Emergence in MAS: An Overview. Informatica, 30(1):45–54.

S2R. Shift2rail, 2022. URL: https://rail-research.europa.eu/about-shift2rail/. Accessed October 20th, 2023.

N.D. Versluis, P. Pellegrini, E. Quaglietta, R.M.P. Goverde, and J. Rodriguez. (2023) An approximate conflict detection and resolution model for moving-block signalling by enhancing RECIFE-MILP. In 10th International Conference on Railway Operations Modelling and Analysis - RailBelgrade 2023, Belgrade, Serbia.