Journal of Rail Transport Planning & Management 29 (2024) 100427

Contents lists available at ScienceDirect

Journal of Rail Transport Planning & Management

journal homepage: www.elsevier.com/locate/jrtpm

FULL LENGTH ARTICLE

Towards self-organizing railway traffic management: concept and framework

Leo D'Amato [a,b], Federico Naldini [c,*], Valentina Tibaldo [b], Vito Trianni [b], Paola Pellegrini [c]

[a] *Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10124 Torino, Italy*
[b] *Institute for Cognitive Sciences and Technologies, CNR, Via San Martino della Battaglia 44, 00185 Rome, Italy*
[c] *Univ Gustave Eiffel, COSYS-ESTAS, F-59650 Villeneuve d'Ascq, France*

ARTICLE INFO

Keywords:
Railway traffic management
Optimization algorithms
Self-organization
Consensus

ABSTRACT

Railway traffic management requires a timely and accurate redefinition of routes and schedules in response to detected perturbations of the original timetable. To date, most of the (automated) solutions to this problem require a central authority to make decisions for all the trains in a given control area. An appealing alternative is to consider trains as intelligent agents able to self-organize and determine the best traffic management strategy. This could lead to more scalable and resilient approaches, that can also take into account the real-time mobility demand. In this paper, we formalize the concept of railway traffic self-organization and we present an original design that enables its real-world deployment. We detail the principles at the basis of the sub-processes brought forth by the trains in a decentralized way, explaining their sequence and interaction. Moreover, we propose a preliminary proof of concept based on a realistic setting representing traffic in a French control area. The results allow conjecturing that self-organizing railway traffic management may be a viable option, and foster further research in this direction.

1. Introduction

Train timetables are defined so as to serve expected demand as much as possible, while respecting a number of constraints. These constraints are linked to the available railway infrastructure, fleet and crew, as well as to commercial considerations. Timetables are designed on the one hand to maximize the rail network capacity, and on the other hand to be robust to daily perturbations. To this end, they include margin or buffer times to allow the recovery of small delays. These are often insufficient to cope with perturbations, which can have a significant impact on operations and on the performance of the railway system.

To cope with perturbations, dispatchers decide on train routes and schedules, with the aim of minimizing some measure of delay. Dispatchers are in charge of traffic in specific control areas. In current practice, they dispose of very little decision support tools and base their decisions on personal experience. The academic literature proposes a large number of optimization algorithms to support dispatchers in their decisions, which are however almost never deployed in practice. These algorithms tackle the real-time Railway Traffic Management Problem (rtRTMP), which is a scheduling and routing problem. To enable practical deployment, current studies consider the enhancement of rtRTMP algorithms to deal with models mimicking reality as much as possible. Moreover, large effort is devoted to the solution of technical issues which still make actual deployment impossible. For example, the collection of all relevant information and the definition of the formats to be used to achieve interoperability are tasks still to be completed by the railway

* Corresponding author.
E-mail addresses: leo.damato@polito.it (L. D'Amato), federico.naldini@univ-eiffel.fr (F. Naldini), valentina.tibaldo@istc.cnr.it (V. Tibaldo), vito.trianni@istc.cnr.it (V. Trianni), paola.pellegrini@univ-eiffel.fr (P. Pellegrini).

https://doi.org/10.1016/j.jrtpm.2023.100427
Received 9 May 2023; Received in revised form 28 November 2023; Accepted 1 December 2023
Available online 4 January 2024

industry. The work performed in the framework of the European Joint Undertaking Shift2Rail (S2R, 2022) and the one planned for its successor Europe's Rail (ERJU, 2022) well represent these trends.

The European project SORTEDMOBILITY (2021) proposes a complete paradigm shift for traffic management. Here, trains are seen as intelligent agents. Based on the observation of the current traffic situation, the planned timetable and the expected demand, they autonomously decide how traffic shall evolve. They do so by exploiting distributed intelligence techniques known under the name of consensus protocols. Trains are hence capable of self-organization and, as such, they can be very flexible and reactive to perturbations. To preserve the possible implementation of self-organization with the current centralized interlocking system functioning, the decisions agreed on by trains are collected and merged at the traffic control center. On the one hand, this guarantees a final feasibility check of the decisions made. On the other hand, it allows operating the interlocking for the implementation of train routes and schedules. In SORTEDMOBILITY, the potential of self-organization is assessed in simulation. In particular, the project exploits the microscopic railway simulator EGtrain Quaglietta (2013) and the multi-scale integrated agent-based simulation platform SimMobility (Adnan et al., 2016). The former allows the precise consideration of railway technical constraints, while the latter guarantees the realistic evaluation of the impact of traffic management decisions on passenger mobility patterns. In this way, train traffic management can be optimized with respect to routing and scheduling aspects as well as to their impact on the mobility demand.

Inspired by research in distributed, multi-agent systems, we believe that self-organization may allow the system to effectively deal with three major challenges faced by classic traffic management approaches (Hamann and Reina, 2022; Serugendo et al., 2006). First, it could efficiently scale up to large networks, which is a major issue for most existing traffic management approaches. Failures and disturbances would only have an impact on the immediate vicinity of the affected area rather than in a whole (sub-)network. Reaction times could be very fast thanks to the short-circuit of long decision chains, resulting in lower peaks of delays and shorter recovery time. Second, it could satisfy the need for transport customization: it could leave aside the very concept of rigid timetabling and exploit the flexibility of self-organization to respond to multi-modality needs in terms of synchronization, accessibility discrepancies across heterogeneous travelers, or modal substitution in case of service performance changes due to, e.g., disruptions. By satisfying this need, it may help in accommodating the increasing passenger demand envisaged if the politically aimed modal shift to railway is achieved, with a possible doubling of demand between 2017 and 2030 (IEA, 2019). Third, it could simplify and encourage cooperation and local competition in a dynamic context. For example, on the one hand, trains may cooperate in optimizing the amount of passengers allowed boarding at stations, to maximize the quality of service offered to both waiting and traveling passengers, i.e., balancing the need for short waiting time on platforms and short travel time. On the other hand, information perceived as competition-critical may remain private, such as the value attributed by an operator to a service or the frequency at which this value shall be re-assessed. While remaining private, this information may be used to bias traffic management decisions: a freight train that needs to meet a multimodal connection on a very tight schedule may push toward a consensus solution in which it reaches its destination on time without disclosing the reason why this is particularly important. This may be extremely useful in the current European context, in which a major modal shift toward railway is aimed for freight transport (ERA, 2023). By effectively dealing with these three challenges, traffic self-organization may allow moving toward a flexible and very reactive railway transport system, participating in making the large European network efficient, interoperable and open to competition.

In this paper, we present the original design proposed for the self-organization process developed in SORTEDMOBILITY. It is based on the decomposition of this process in five separate modules, four executed individually by each train and a final, unifying one operated centrally in a traffic control center. These modules are devoted to (i) the identification of trains that need to reach a consensus at a given time, (ii) the individual generation of possible traffic management solutions, (iii) the sharing of compatible solutions among interacting trains, (iv) the selection of solutions by each train through the consensus process and (v) the centralized merge of the agreed individual solutions into a single traffic management plan to be deployed. Such a process formalization is an original contribution that clearly delineates how traffic self-organization may be integrated in the railway system. Moreover, it will allow future studies to focus on developing approaches for single modules relying on our process for its positioning within the overall traffic management problem. As a preliminary proof of concept on the applicability of the designed process, we propose a simple implementation of each module and we apply the process to a case study representing traffic in a French control area.

The rest of the paper is organized as follows. Section 2 describes the state of the art in railway traffic management, also focusing on decentralized techniques and consensus approaches. Section 3 presents the traffic management process we consider for introducing self-organization. Section 4 describes the modules of the devised approach, and Section 5 details their implementation. Section 6 discusses the proof of concept, and Section 7 concludes the paper.

2. State of the art

The literature on optimization algorithms for the rtRTMP is very rich. Most of it focuses on the proposal of Operations Research algorithms for managing railway traffic in case of small perturbations. These are disturbances that occur very often in operations: one or more trains suffer some delay due to an unpredicted event and this delay propagates to other trains in a snow-ball effect. This is typically due to train interdependencies (e.g., rolling stock re-utilization constraints) or to the occupation of unplanned paths by the delayed trains, where a path is defined in space and time. In the latter case, trains are said to be involved in *conflicts*. The objective of the algorithms is typically the minimization of delay propagation, although other criteria are sometimes considered, for example, explicitly related to passenger demand. Nearly all the existing algorithms for the rtRTMP tackle the train scheduling problem. Some works also consider train routing as a decision to be made, either sequentially with respect to the scheduling one or concurrently. Techniques that have been applied include Integer Linear Programming (ILP) (Caimi et al., 2012; Meng and Zhou, 2014; Toletti et al., 2020), Mixed Integer Linear Programming (MILP) (Luan et al., 2020; Fischetti and Monaci, 2017; Pellegrini et al., 2015; Törnquist and Persson, 2007; Lu et al., 2022; Reynolds and Maher, 2022; Leutwiler et al., 2023), alternative and disjunctive graphs (Corman et al., 2010; Lamorgese and Mannino, 2015; Mascis and Pacciarelli, 2002; Samà et al., 2017), time–space graphs (Bettinelli et al., 2017), and constraint programming (Rodriguez, 2007). We refer the interested reader to the review papers by Corman and Meng (2015), Cacchiani et al. (2014), Fang et al. (2015) and Lamorgese et al. (2018) for a wide analysis of this literature.

Among the proposed algorithms, one that is often considered to be part of the state of the art is named RECIFE-MILP (Pellegrini et al., 2015). It is a MILP-based algorithm that makes decisions on the best schedule and routes to be taken by a set of trains traversing a control area in a given time horizon. Scheduling and routing decisions are made simultaneously, considering a microscopic infrastructure representation. It is a two-step algorithm. In the first step, the pure scheduling problem is solved, in which trains use a previously defined route and schedules on common tracks must be decided. In the second step, the routing and scheduling problem is solved, in which the choice of one out of a set of alternative routes must be performed for each train. The solution of the first step is used to initialize the search in the second. The basic functioning of RECIFE-MILP consists in solving the MILP formulation detailed in Pellegrini et al. (2015) with a commercial solver. This is done in both steps. The objective function typically optimized is the minimization of total delay, but various others have been tested. RECIFE-MILP has been successfully used for optimizing traffic management in various French and European infrastructures (Quaglietta et al., 2016; Pellegrini et al., 2016).

Besides the classic research direction represented by this literature in which algorithms make centralized decisions on train routing and scheduling, a more recent trend explores the possibility to decentralize decision making (Marcelli and Pellegrini, 2020). Along this trend, Van Thielen et al. (2019) propose an approach in which each predicted conflict is tackled individually by rescheduling or rerouting the involved trains. Here, decisions are decentralized, but the focus is more on the dynamic decomposition of rtRTMP instances than on the identification of trains as decision makers. Nonetheless, the proposed dynamic instance decomposition and the considered deployment process have some remarkable similarities with our proposal, as discussed below. Differently, Shang et al. (2018) assume that trains are in charge of making decisions: each train optimizes its movement on a loop-shaped railway line based on the observation of the behavior of the train ahead. Considering an infrastructure topology including the junction of two lines, Yong et al. (2017) propose the use of swarm intelligence for traffic management: trains are grouped in platoons and face possible common conflicts as a swarm. Only scheduling options are considered; decisions are made by trains based on the application of rather simple rules after checking for the infrastructure occupation status. The most innovative contribution of this paper is the transfer of decision making to trains rather than to a centralized system.

Some recent papers following this trend focus on the exploitation of Artificial Intelligence (AI) techniques, and in particular of machine learning (ML) (Jusup et al., 2021). Among them, Khadilkar (2019) proposes a reinforcement learning (RL) approach: the system considers the train perspective, learns what choices bring to good performance, and tries to replicate them whenever suitable. The representation of the railway system is quite simplified, as the interest is in the novel application of RL rather than in direct deployment. Several applications of this type have been proposed in recent years, stimulated by the proposal of the Flatland challenges by a group of European railway infrastructure managers (Mohanty et al., 2020). Here, a simplified open source railway simulator is made available for the research community to propose, test and compare different RL approaches (The Community Flatland, 2022). Periodic challenges are launched, and novel approaches are designed to deal with larger and larger instances (Mohapatra et al., 2022). Indeed, RL allows dealing with extremely large instances in very short time, after a long training has been carried out. Shortly speaking, a policy must be devised based on the observation of a set of parameters describing the traffic state and the infrastructure, the performance of a corresponding action and the observation of a reward function. The actions that bring the highest rewards are learned to be good and will be performed when similar conditions occur. Although this approach has undoubtedly intuitive merits, it is still quite abstract. Much effort is needed before a RL approach can be implemented in practice. In fact, all approaches proposed so far provide no guarantee of finding overall feasible traffic management strategies. They will likely succeed in most of the cases, but in the worst case scenario they may end up producing deadlocks as they make decisions based on standard patterns and corresponding rewards, rather than on the detailed observation of specific situations. Moreover, the acceptability of approaches based on ML will be hard to achieve. Indeed, a characteristic of ML is that it operates as a black box, learning relations that are never made explicit for the user to understand. Having different actors involved in a competitive market accepting such a black box to manage traffic without being allowed to discuss the underlying principles will require much effort.

Self-organization is a promising option to exploit the AI ability to deal with large instances, while making decisions on specific situations following principles defined or at least accepted by individual actors. To the best of our knowledge, no self-organization approaches have been proposed for railway traffic management. The few attempts toward decision decentralization appearing in the literature do not consider intelligent trains with individual decision making possibilities (Marcelli and Pellegrini, 2020). However, this type of approach have been extensively studied in various other systems (e.g., multi-robot systems (Dorigo et al., 2021)) and they appear promising for dealing with traffic perturbations. A family of methods that appear particularly appealing to this aim is based on consensus protocols. A consensus protocol is nothing more than the set of rules that lead a multi-agent system to reach an agreement on some value. For instance, voting represents a mean to reach agreements in groups, and multiple protocols exist to implement it, such as a simple or a qualified majority voting system. However, voting requires a central authority that collects and counts all the votes to determine which alternative to select. We are interested here in decentralized consensus protocols, whereby agreement is achieved without any central authority. Note that the election of a central authority in a multi-agent system is also a consensus problem by itself, hence a truly decentralized approach is needed even to implement a voting system.

Decentralized consensus is largely studied in different contexts. The Paxos consensus algorithm (Pease et al., 1980; Lamport, 2006) is a well-known example from distributed computing, and has been followed by several similar approaches since its introduction. It is based on iterated message passing among agents within a network, and can deal with (temporary) failures and

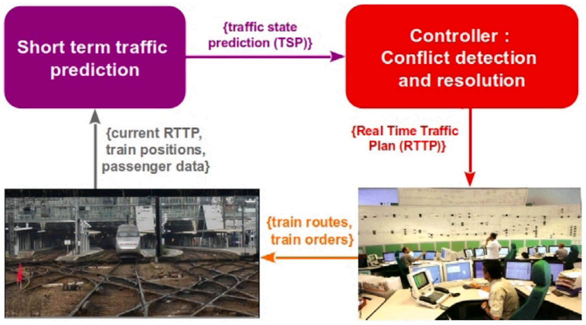

Short term traffic prediction
{traffic state prediction (TSP)}
Controller : Conflict detection and resolution
{current RTTP, train positions, passenger data}
{Real Time Traffic Plan (RTTP)}
{train routes, train orders}

Fig. 1. Graphical representation of the closed-loop framework.

communication losses. It guarantees the achievement of consensus, and provides recovery mechanisms to deal with different kinds of failures. Simpler consensus protocols are often inspired by the study of opinion dynamics in social systems (Castellano et al., 2009). Among these, the Voter Model strikes for its simplicity (Holley and Liggett, 1975): agents copy the opinion of other agents in a population. After a sufficiently large number of iterations, the population reaches consensus on the same opinion. Different dynamics are imposed by the interaction topology, such as lattices, random graphs and other heterogeneous networks (Baronchelli et al., 2011; Moretti et al., 2012). More complex opinion selection strategies can also be implemented, for instance looking at more than one neighbor at the time (Castellano et al., 2009). Variants also account for a quality value associated to alternatives (Valentini et al., 2014), which bring voter-like models closer to other collective decision algorithms studied in biological systems and robot swarms (Reina et al., 2015; Valentini et al., 2017). A different category of consensus algorithms derive from studies on the evolution of language and cognitive categories (Loreto et al., 2011; Baronchelli, 2017). In this context, consensus is necessary to determine a shared terminology (Steels, 1995) or a coherent categorization system (e.g., for colors (Baronchelli et al., 2010)). Stemming from studies in different disciplines, these decentralized consensus algorithms have been used in multi-agent systems and robot swarms (Cambier et al., 2020).

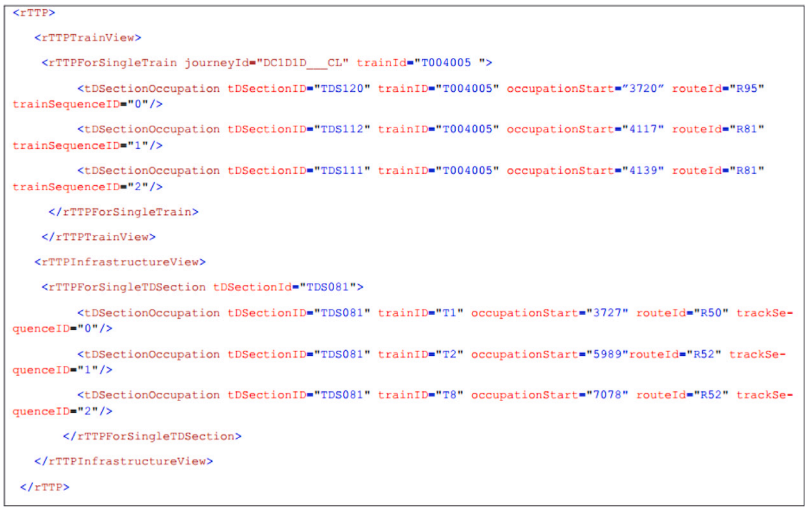
3. Railway traffic management process

The traffic management process we consider in this paper and in the SORTEDMOBILITY project is based on the closed-loop framework introduced in the project "Optimal Networks for Train Integration Management across Europe" (ONTIME, 2011) and showcased in Quaglietta et al. (2016). A similar framework is also considered in Van Thielen et al. (2019). Such a closed-loop framework represents one of the envisaged options for the industrial deployment of support algorithms for traffic management. In this framework, a conflict detection and resolution module (*controller*) periodically interacts with the field – or with a simulator replacing reality for testing purposes – receiving updated traffic conditions and providing a new traffic plan. Through repeated interactions, the traffic management decisions can be always based on the most recent traffic conditions. Fig. 1 depicts the closed-loop framework we consider. Here, the field supplies information on the currently implemented traffic management decisions, on the train positions and on the observed passenger data. Traffic management decisions consist in train routes and orders, and they are shared in the form of a Real Time Traffic Plan (RTTP). The RTTP includes information on train entrance time on each track detection section it is planned to traverse. From these, the train passing order can be derived. An example of RTTP is reported in Fig. 2. The so-called "train view" shows the sequence of track detection sections which compose the journey chosen for each train. The "infrastructure view", instead, shows the same information from the perspective of each track detection section, making train passing orders explicit. Note that block-sections are also mentioned in the RTTP with the name "route", following the British convention.

The information shared by the field allows a short term traffic prediction module to deduce where trains will be sometime in the near future, e.g., in ten minutes. This module may be based on a microscopic traffic simulator or any other prediction technique. The result of the deduction is included in the Traffic State Prediction (TSP). An example is reported in Fig. 3. Here, two trains (T1 and T2) are in the control area of interest. For each of them, the TSP indicates the track detection section in which the train is at the considered time and at which time the train accessed it. For trains entering the considered infrastructure in the near future (T3 in the example), the expected entrance time is reported.

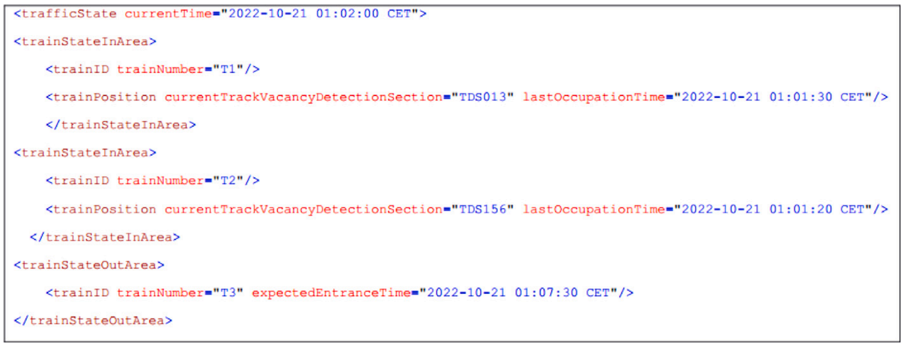
The TSP is used by the controller to detect and solve possible conflicts, and to produce a new RTTP. By considering the TSP, the relevant traffic situation is the one predicted for the near future, e.g., ten minutes from the current time. This time interval is sometimes referred to as *control delay* (Van Thielen et al., 2019), and is necessary to allow the controller to make sensible decisions, which may take a few minutes depending on the complexity of the situation to deal with. Moreover, in the same time period, the traffic control center has to implement the received RTTP, and possibly a dispatcher may have to validate it. Whether this validation is necessary depends on the chosen level of system automation. Such closed-loop can be repeated indefinitely and hence deployed for any time duration.

The controller can be based on either an algorithm as the ones described in Section 2, or a self-organization process as the one detailed in Section 4. This module in particular can be decomposed in various sub-modules. For example, conflict detection may



```
<rTTP>
   <rTTPTrainView>
    <rTTPForSingleTrain journeyId="DC1D1D___CL" trainId="T004005 ">
         <tDSectionOccupation tDSectionID="TDS120" trainID="T004005" occupationStart="3720" routeId="R95"
trainSequenceID="0"/>
         <tDSectionOccupation tDSectionID="TDS112" trainID="T004005" occupationStart="4117" routeId="R81"
trainSequenceID="1"/>
         <tDSectionOccupation tDSectionID="TDS111" trainID="T004005" occupationStart="4139" routeId="R81"
trainSequenceID="2"/>
     </rTTPForSingleTrain>
    </rTTPTrainView>
   <rTTPInfrastructureView>
    <rTTPForSingleTDSection tDSectionId="TDS081">
         <tDSectionOccupation tDSectionID="TDS081" trainID="T1" occupationStart="3727" routeId="R50" trackSe-
quenceID="0"/>
         <tDSectionOccupation tDSectionID="TDS081" trainID="T2" occupationStart="5989"routeId="R52" trackSe-
quenceID="1"/>
         <tDSectionOccupation tDSectionID="TDS081" trainID="T8" occupationStart="7078" routeId="R52" trackSe-
quenceID="2"/>
       </rTTPForSingleTDSection>
   </rTTPInfrastructureView>
  </rTTP>
```

Fig. 2. Example of real time traffic plan.



```
<trafficState currentTime="2022-10-21 01:02:00 CET">
<trainStateInArea>
    <trainID trainNumber="T1"/>
    <trainPosition currentTrackVacancyDetectionSection="TDS013" lastOccupationTime="2022-10-21 01:01:30 CET"/>
    </trainStateInArea>
<trainStateInArea>
    <trainID trainNumber="T2"/>
    <trainPosition currentTrackVacancyDetectionSection="TDS156" lastOccupationTime="2022-10-21 01:01:20 CET"/>
  </trainStateInArea>
<trainStateOutArea>
    <trainID trainNumber="T3" expectedEntranceTime="2022-10-21 01:07:30 CET"/>
</trainStateOutArea>
```

Fig. 3. Example of traffic state prediction.

be separated from conflict resolution, or a decentralized decision making system can be defined taking place among the intelligent trains. Whatever the possible decomposition, it is in charge of defining the RTTP to be implemented in coherence with the latest TSP.

In SORTEDMOBILITY, the EGTrain simulator shares information on the observed train and passenger travels. It does so either with a fixed time periodicity, or when a triggering event occurs. To ease the implementation, when this sharing takes place, we pause the simulation. This allows the consideration of the current traffic state as being the predicted one. EGTrain hence directly produces a TSP. This is the usual practice for testing the closed-loop framework in a laboratory environment, although this may favor the optimization performance with respect to actual deployment, as it eliminates any prediction error and impact of possible unexpected events. Based on the shared TSP, the controller makes decisions on train routes and orders to apply. It shares it with a traffic control center, which ensures its consistency and sends it to EGTrain for implementation. Let us remark that, following the literature, all the infrastructure data format described are defined for a fixed-block signaling system. In a moving-block context, track detection sections, block sections and signals are not defined. Here, we adapt the definition considering an infrastructure representation based on track discretization.

4. Self-organized decision making process

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 this paper, but we think the proposed design is compatible with most envisageable options.

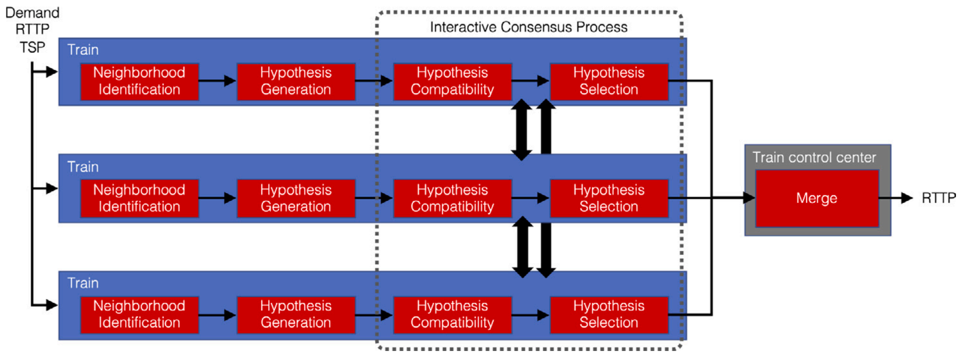


Fig. 4. Conceptual representation of the self-organized traffic management process.

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. Fig. 4 depicts this process, an example of whose implementation will be provided in Section 5.

The process starts with the reception of the current RTTP and TSP, along with the prediction of the passenger demand. These 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 rtRTMP (e.g., including or excluding rerouting possibilities) and the process design (e.g., frequency of traffic monitoring). This neighborhood identification has some similarities with respect to the *dynamic impact zone* determination of Van Thielen et al. (2019), as it states the subsets of somehow linked trains and it depends on the current train positions and schedules. However, a difference exists between the two concepts. On the one hand, the dynamic impact zone intervenes after the solution of a conflict, to determine trains that will be involved in future conflicts given the solution of the current one. On the other hand, our neighborhood identification is independent of conflicts: when a decision is triggered, each train checks the traffic state to determine which train it may interact with, be there expected conflicts or not. Indeed, we consider that it is possible that no conflict is expected if the traffic evolves as currently planned, but a change of plan may still be desirable for a train. For example, the currently planned train route may have appeared to be the best in the past, given the available expectations, but a different one may be better now. This may happen if a longer route is planned at some point to avoid expected conflicts, but then it turns out that no conflict will occur if the train takes a shorter one. In this case, defining the neighborhood identification independently on expected conflicts allows the train to possibly change its route.

Once its neighborhood identified, each train performs a *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 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.

Fig. 5 depicts an example of application of the designed process. Here, four trains are traveling in a simple control area. Trains A and B identify the same neighborhood, including the two of them as well as train C. Train C identifies as possibly interacting train also D, as it may either continue traveling on its current track or move to the lower one. Considering the possible routing of C, train D includes C in its neighborhood. After generating the hypotheses and sharing them to check for compatibility, the consensus process takes place. Remark that, as neighborhoods are train-specific, A interacts with B and C, B interacts with A and C, C with all trains and D only with C. In the consensus process, each train iteratively selects the preferred hypothesis depending on its value and on the hypotheses selected by each neighbor. The number of iterations performed by each train depends on the specific consensus approach implemented, on the neighborhoods, and on the available hypotheses. As the process is decentralized and asynchronous, different trains may perform a different number of iterations. After consensus, the selected hypotheses for each train/neighborhood are communicated to the traffic control center and merged.

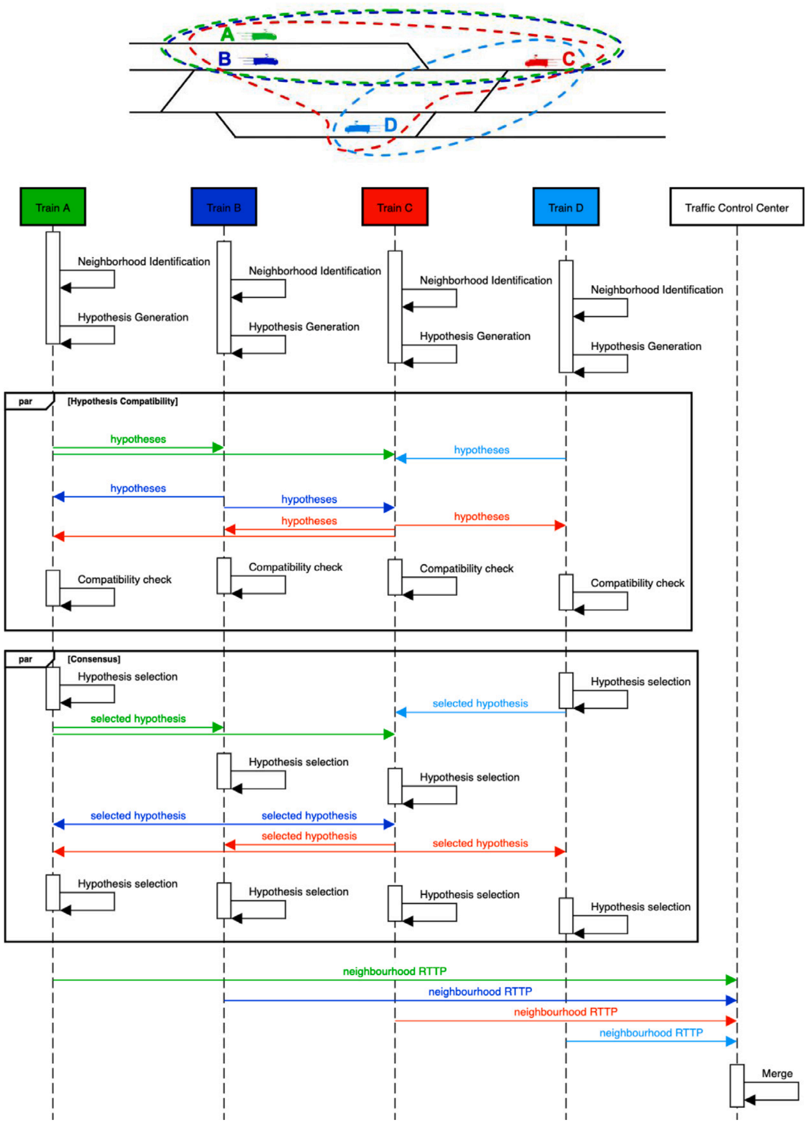


Fig. 5. Top: Set of trains traveling in a simple control area. For each train, the neighborhood is indicated using the corresponding color. Bottom: sequence diagram of the self-organization process indicating, for each train, interactions and control processes.

5. Proof-of-concept implementation

In this section, we describe the current implementation of the software pipeline, which emulates the distributed process and centralized merge discussed above, and is graphically rendered in Fig. 6. Here, a single *neighborhood identification* module identifies the neighborhood of each train, based on the current RTTP and TSP. It generates *neighborhood TSPs*, indicating the composition of the neighborhood for each *focal* train. This information is also used to define the *interaction graph*, which is used for the *hypothesis compatibility* and *consensus* modules. The *hypothesis generation* module is run separately for each train and produces multiple *neighborhood RTTPs*, each one representing an hypothesis for a specific train neighborhood. It uses the specific neighborhood TSP and the original RTTP, and it interacts with the *demand prediction* module to evaluate the impact on passenger choices (not part of the current implementation). With the interaction graph and all neighborhood RTTPs, the *hypothesis compatibility* module generates an *hypothesis graph* that enables to evaluate the algorithm for *consensus*. A *merge* module collects all selected neighborhood RTTPs and produces the *updated RTTP* to be deployed. The current implementation of each module is described in the following.

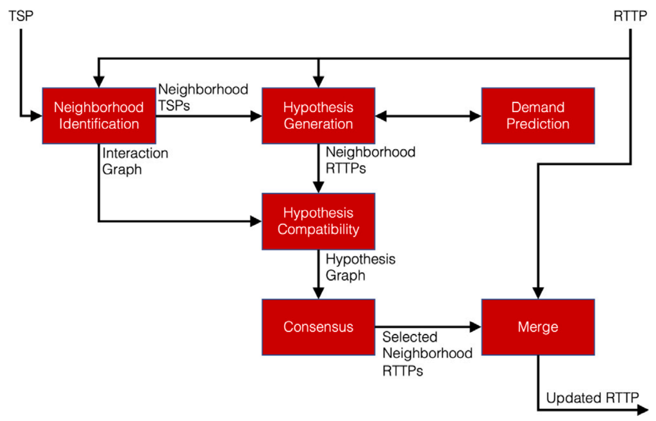


Fig. 6. Software pipeline for the implementation of the self-organization process. The modules that are supposed to be executed in a decentralized way implement a multi-agent system. A demand prediction module is also considered that interacts with the hypothesis generation to determine how hypotheses impact the predicted demand.

5.1. Neighborhood selection

Neighborhood selection starts with the update of the RTTP considering the TSP information: all train paths are checked to verify the consistency of their path as described in the RTTP and the last track detection section occupation starts as reported in the TSP. If necessary, the train paths are shifted to make the times consistent. Then, the following procedure is operated for each train iteratively. Once identified the focal train t, its updated path is observed for the next $\bar{s}$ seconds. All track detection sections that t will use in the $\bar{s}$ seconds following the time indicated in the TSP are considered one by one: for each of them, if another train t' uses it within the same interval of $\bar{s}$ seconds following its updated path, t' is included in t's neighborhood. For these trains t', a new attribute `InNeighborhood=TRUE` is set in the TSP. For all other trains, this attribute is set to `FALSE`. The original TSP augmented with these attributes as well as with the information of the focal train constitutes the Neighborhood TSP of t. Remark that this neighborhood selection is quite naive: it does not consider train re-ordering and re-rerouting when selecting trains with which an interaction may occur. Moreover, the resulting neighborhoods are dependent on the value set for parameter $\bar{s}$: in general, the smaller this value, the smaller the neighborhood. This is a preliminary, baseline implementation on top of which we will improve in the future owing to dedicated research activities.

The identification of the neighborhoods for each train leads also to the generation of the interaction graph $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{E}_T)$, where $\mathcal{V}_T$ is the set of nodes, each representing a train (i.e., $\mathcal{V}_T = T$, with T the set of trains in the TSP), while $\mathcal{E}_T$ is the set of edges (t_1, t_2), indicating that two trains are in the same neighborhood. The set of trains within the neighborhood of train t are represented as $\mathcal{N}_t$ ($t \in \mathcal{N}_t$). Note that $\mathcal{G}_T$ may not be a connected graph, and multiple connected components may be present. This means that there are groups of trains that may not interfere in the (near) future, and consequently do not need to interact for the self-organized traffic management activities.

5.2. Hypothesis generation

The hypothesis generation module is a new variant of RECIFE-MILP[1] presented in Section 2 (Pellegrini et al., 2015). In this variant, the set of trains composing an instance are all the ones included in the input neighborhood TSP considered. The routing and ordering of the trains for which the `InNeighborhood` attribute is false are treated as constraints, while the ones for which it is true are optimized. This choice is motivated by two observations. First, trains not in the neighborhood have no expected interaction with the focal train, hence fixing their routes and orders should not limit the traffic management strategies of the trains in the neighborhood. Second, considering these trains allows the focal train to assess its hypotheses also considering system-level observations. For example, if passenger travel time is to be taken into account, it may be important to consider also trains to which passengers will transfer in the future, although no interaction is expected in the short term.

The objective function currently optimized is the minimization of the weighted sum of three components for each train: delays at their arrival at planned stops within the control area; entrance delay in the control area; exit delay from the control area. In particular, we suppose that each train weights its own delay twice as much as the delay of other trains. This is because we assume that each train, as an independent agent, is more interested in minimizing its own delay rather than the one of all trains in the neighborhood. This is particularly realistic when trains are operated by different railway undertakings, which is often the case, at least when mixed freight and passenger traffic share a line. The choice of doubling the importance of a train's own delay is a

[1] In the current implementation, the commercial solver we exploit is CPLEX version 12.8.

simplification of reality, where each train may have various reasons to account differently for different delays. For example, trains of the same railway undertaking may weigh their delays equally, while considering them much more important than delays of trains of a competing undertaking. In this preliminary implementation of the hypothesis generation module, we do not study the sensitivity of the results to the weight of each train delay: this will make the object of future studies.

Finally, a further new extension of RECIFE-MILP makes it produce multiple solutions, in the form of Neighborhood RTTPs. In particular, when it proves optimality or when it reaches the available time limit, it returns the best solutions it found which are at most $p\%$ worse than the final lower bound, which corresponds the optimal solution if available. The *acceptability threshold* p is a parameter of the algorithm. A further parameter is $\bar{h}$, which indicates the maximum cardinality of the set of solutions returned. We filter solutions in which timing decisions change but routings and orderings are the same: only the best of them is returned. We refer to the set of hypotheses generated for train t as H_t, with $|H_t| \leq \bar{h}$.

5.3. Hypothesis compatibility

The hypothesis compatibility module considers all pairs of trains that simultaneously belong to at least one neighborhood, i.e., they are connected in the interaction graph $\mathcal{G}_T$. All hypotheses of these pairs of trains are checked for compatibility. In a nutshell, two hypotheses are compatible if there is no train in one hypothesis that has overlapping schedule with any other train of the second hypothesis. Given two trains t_1 and t_2, where $(t_1, t_2) \in \mathcal{E}_T$, let h_1 be a hypothesis of t_1 and h_2 a hypothesis of t_2. The hypothesis compatibility module starts considering h_1 as the basis, and it iteratively replaces the path of one train at a time with the corresponding path in h_2. If this path overlaps with at least another one in h_1, then h_1 and h_2 are declared incompatible. Otherwise, h_1 is re-established to its original version and the next train path is replaced. When all trains are tested, if no incompatibility is detected, then h_2 becomes the basis and the replacements are checked with paths of h_1. If all replacements are tested and no incompatibility is detected, h_1 and h_2 are compatible. Note that this compatibility check is quite strict, and possibly some tolerance may be considered. The study of other hypothesis compatibility implementations will be considered in the future.

Following the compatibility check among all paired hypotheses, a *hypothesis graph* $\mathcal{G}_H = (\mathcal{V}_H, \mathcal{E}_H)$ is created, in which $\mathcal{V}_H = \bigcup_{t \in T} H_t$ represents the set of all hypotheses from all trains, while $\mathcal{E}_H$ represents the set of edges, each corresponding to the detected compatibility between two hypotheses: (h_1, h_2). The graph is multi-partite: vertexes corresponding to hypotheses formulated by a train can only be connected to vertexes representing hypotheses of other trains. In other words, the vertices of the hypothesis graph can be easily partitioned into independent sets by grouping together the hypotheses of the same train. Furthermore, vertexes associated with a specific train t can only be connected to vertexes associated to a train belonging to t's neighborhood. Hence, each hypothesis $h \in H_t$ may have a set of compatible hypotheses $C_{h,t'}$ from train $t' \in \mathcal{N}_t, t' \neq t$ defined as follows:

$$C_{h,t'} = \{h' \in H_{t'} : (h, h') \in \mathcal{E}_H\} \tag{1}$$

Also, a cost c_h is associated to each vertex $h \in \mathcal{V}_H$, i.e., the cost for implementation of the hypothesis h as computed by the corresponding train. In particular, in the current implementation, if h is generated by train t, this cost corresponds to the objective function described in Section 5.2: it includes all train delays at entrance, arrival at schedule stops and exit from the control area, with the delay of t weighted by a factor two.

5.4. Hypothesis selection through consensus

The consensus module aims at finding a configuration in which each train selects a hypothesis – a neighborhood RTTP – that is compatible with the hypothesis of any other train in its neighborhood. Moreover, the configuration should minimize the cost of the selected hypotheses. Considering the hypothesis graph $\mathcal{G}_H$, a solution to the problem corresponds to a sub-graph where the set of vertexes contains one and only one hypothesis for each train, and the edges are chosen in a way to ensure that the hypothesis of any train is compatible with the hypothesis selected by all trains in its neighborhood. The cost of a solution is the sum of the cost of all vertexes it includes.

To determine whether the problem admits a solution, and how good this solution is, we take a centralized perspective. We assume that the objective is finding the solution with minimum cost, if at least a solution exists. We model it as an integer linear program (ILP). First, we define the binary decision variables x_h as

$$x_h = \begin{cases} 1 & \text{if vertex } h \text{ is selected in the solution} \\ 0 & \text{otherwise} \end{cases} \qquad \forall h \in \mathcal{V}_H \tag{2}$$

The centralized optimization problem consists in finding the appropriate selection of hypothesis to minimize the following objective function:

$$\min \sum_{h \in \mathcal{V}_H} x_h \, c_h \tag{3}$$

provided that the following constraints are satisfied:

$$\sum_{h \in H_t} x_h = 1, \qquad \forall t \in T \tag{4}$$

$$\sum_{h' \in C_{h,t'}} x_{h'} \geq x_h \qquad \forall h \in \mathcal{V}_H, t, t' \in T : h \in H_t, t' \in \mathcal{N}_t, t' \neq t \tag{5}$$

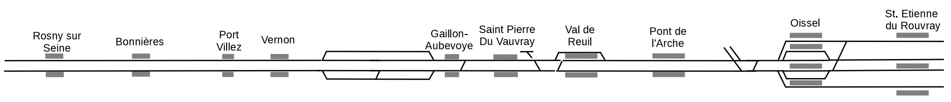


Fig. 7. Schematic representation of the control area considered.

Constraints (4) ensure that exactly one hypothesis per train is selected. Constraints (5) state that, if vertex h is selected and is associated to train t, then for each neighboring train in $\mathcal{N}_t$ a hypothesis is selected that is compatible with h. This formulation corresponds to the *consensus problem*.

In the decentralized approach implemented in the consensus module, each train t must select one and only one hypothesis $\hat{h}_t$ ($x_{\hat{h}_t} = 1$) aiming to overall compatibility and individual cost minimization. We implement a stochastic, iterative consensus process inspired by the voter model, introduced in Section 2. At initialization, each train t selects its best hypothesis: $\hat{h}_t = \arg\min_{h \in H_t} c_h$. In case of ties, a random hypothesis is chosen among the equal-best alternatives. Then, at each iteration, a random train t is chosen to update its selected hypothesis. This enables to simulate asynchronous decisions, assuming that the time required to select a hypothesis is negligible. Then, a random train $t' \in \mathcal{N}_t$ is selected, and compatibility is checked. Specifically, train t executes the following rules:

1. if $(\hat{h}_t, \hat{h}_{t'}) \in \mathcal{E}_H$, the two trains already selected hypotheses that are compatible, and t does not change its choice.
2. if $(\hat{h}_t, \hat{h}_{t'}) \notin \mathcal{E}_H$ and $|C_{\hat{h}_{t'},t}| > 0$, then there exist hypotheses from train t that are compatible with the hypothesis selected by t'. Hence, t tries to reach compatibility with t' by selecting $h \in C_{\hat{h}_{t'},t}$, proportionally random to the benefit $1/c_h$.
3. if $(\hat{h}_t, \hat{h}_{t'}) \notin \mathcal{E}_H$, $|C_{\hat{h}_{t'},t}| = 0$ and $\sum_{h' \in H_{t'}} |C_{h',t}| > 0$, then there exist hypotheses from train t' that are compatible with at least one hypothesis from train t. Hence, t tries to prepare for future compatibility with t' by selecting $h \in \bigcup_{h' \in H_{t'}} C_{h',t}$, proportionally random to the benefit $1/c_h$.

If none of the above rules can be executed, then there is no possibility of hypothesis compatibility between t and t'. Hence, the consensus problem does not admit a solution. In such case, train t excludes t' from the list of neighbors, and tries to reach consensus with the remaining trains in its neighborhood. The iterative process is carried out until each train verifies compatibility with every other train in its neighborhood, or until a maximum number of iterations are performed (equivalent to a maximum decision time). Note that, when any consensus configuration is achieved, the process terminates and no further change is possible. Indeed, at consensus any train t would always verify the conditions 1 above, which leads to no change. When this happens, we consider that the consensus process reached convergence for a solution, although it may be a suboptimal one. Indeed, the stochastic nature of the process guarantees convergence (albeit in a very long time), but does not guarantee optimality. Future work will determine the accuracy of this process, which constitutes the baseline for more complex consensus protocols to be adopted by the self-organizing trains.

5.5. *Merge*

Eventually, the merge module receives all selected hypotheses in the form of neighborhood RTTPs. If consensus was reached, it computes a new RTTP from the neighborhood RTTPs. Otherwise, it uses the currently deployed RTTP. Remark that, if consensus is reached, necessarily all selected hypotheses are compatible within a neighborhood. In the current implementation, we use the currently implemented RTTP as a basis. For each neighborhood RTTP, we take the path of the focal train and we replace the one present in the current RTTP with the former. Remark that in the current version we do not include any consistency check in this module, completely relying on hypothesis compatibility. Consistency checks and repair procedures will be proposed in further research.

6. Preliminary results

In this section, we propose a proof of concept showing that the process proposed in Section 4, and in particular its implementation of Section 5, actually allows for traffic self-organization. To do so, we consider traffic in a railway control area covering a 80 km portion of the line connecting Paris and Le Havre, in France. In particular, we consider the line portion between Rosny sur Seine and St. Etienne de Rouveray. Its layout is shown in Fig. 7. This line includes ten stations and a four track portion. While most stations only have two platforms, Oissel and St. Etienne de Rouveray are larger, with six and four platforms respectively. Conventional and high speed passenger trains circulate on this line, in addition to freight ones. At peak time, approximately 20 trains traverse the control area. We consider a perturbed scenario in which 20% of trains, randomly selected, suffer an entrance delay between 5 and 15 min when reaching the control area.

We simulate traffic starting at 6am with the OpenTrack microscopic simulator (Nash and Huerlimann, 2004). We pause the simulation and produce the corresponding TSP at 10am. At this time, 19 trains are present in the control area or expected to enter it within the next hour. All these trains are included in the TSP.

We set the duration of the interval considered for neighborhood selection ($\bar{s}$) to 50 min. This results in neighborhoods including up to 10 trains. Five trains are isolated, the remaining 14 have in average 5.6 trains in their neighborhood. The acceptability threshold for hypothesis generation (p) is 40%.

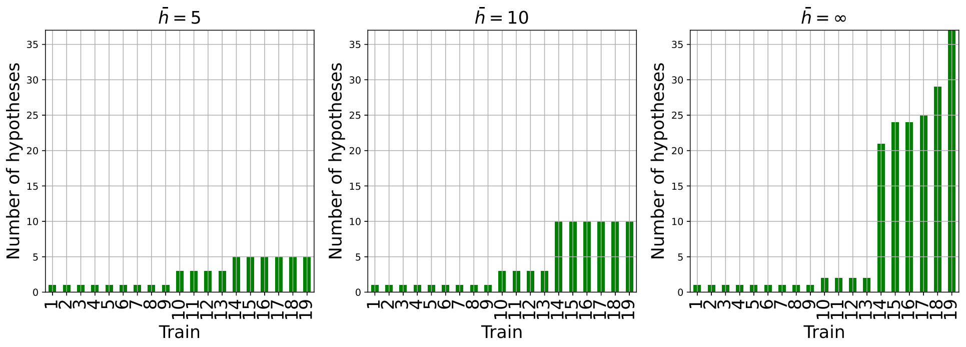


Fig. 8. Number of hypotheses per train.

We assess the performance of the self-organization process considering a maximum number of hypotheses per train ($\bar{h}$) equal to 5, 10 and ∞. In the third case, all sufficiently different solutions found by the hypothesis generation within the acceptability threshold are returned. When $\bar{h} = \infty$, trains have on average nine hypotheses: nine trains have a single one, and the train with the larger hypothesis set has 37 of them. When $\bar{h} = 5$ and $\bar{h} = 10$ trains have on average three and four hypotheses. Fig. 8 reports the number of hypotheses considered for each train with $\bar{h} = 5, 10, \infty$. For the three values of this parameter, the resulting hypothesis graphs have 47, 77 and 177 vertexes, and 427, 1557 and 8634 edges, respectively. No vertex is isolated, i.e., all hypotheses have at least one compatible hypothesis from a neighbor train. When $\bar{h} = 5$, for all trains, all hypotheses are compatible with all hypotheses of all neighbor trains. When $\bar{h} = 10$, in average, each hypothesis is compatible with 98% of the hypotheses of neighbor trains. This percentage decreases to 85% with $\bar{h} = \infty$. Recall that, as described in Section 5.2, hypothesis generation relies on the RECIFE-MILP exploration of the search space. This is performed through the CPLEX MILP solver, version 12.8. As this solver includes some stochastic features to speed up its search, it happens sometimes that the set of solutions visited differs from run to run. Hence, the number of hypotheses per train may sometimes decrease although $\bar{h}$ increases. For example, here, trains 10 and 13 have fewer available hypotheses when $\bar{h}$ is set to ∞ than in the other two cases.

We run the consensus process 100 times for each $\bar{h}$ and we summarize the results in Table 1. It reports the minimum, average and maximum value of solutions returned by the consensus algorithm, i.e., the sum over all trains of the value it associates to the selected hypothesis. The last column reports the centralized optimal solution value, computed by solving the ILP described in Section 5.4. Consensus is reached in all runs.

The results show that, when $\bar{h}$ increases, consensus solution typically worsens. Indeed, the greedy nature of the currently implemented approach does not manage to make the best of the larger search space provided by higher $\bar{h}$. Nonetheless, our approach always succeeds in returning solutions that are quite close to the optimal ones: the percentage difference is always below 3%, and in average it is below 1.48% regardless of $\bar{h}$. In most cases, the difference is due to the slight increase of the delay of a few trains in consensus solutions. In few runs, however, the delay of some trains is remarkably different. Fig. 9 shows the Gantt diagram of the worst run for consensus (top) against the one of the optimal solution (bottom). It is focused on the path of the green train, that is the train that suffers a remarkably higher delay after consensus. It travels from St. Etienne du Rouvray to Rosny sur Seine. In the Gantt, a line is associated to each TDS traversed by the green train, and rectangles show the occupation (dark color) and utilization (light color) by trains. If a yellow rectangle is displayed, it means the train spends in the corresponding TDS a longer time than its scheduled running time, be it for dwelling at a stop or due to traffic. The green train suffers for a slight entrance delay in the control area, due to the perturbation. This creates a conflict with the following gray train in the first part of the path. In both solutions, the green train passes first, only slightly delaying the gray one. Then, no further conflicts exist, and the two solutions take two different routes. In the consensus solution, the train takes the default route, which does not allow it to catch up the delay despite using the buffer time at the scheduled stop at Port Villez (in correspondence of the yellow rectangle after about two thirds of the train path). In the optimal solution, instead, the train takes a different, slightly shorter route (using the upper track between Gaillon-Aubevoye and Vernon). This allows it to arrive at Port Villet only slightly late, and to completely catch up its delay before exiting the control area thanks to the buffer time. In this unfortunate run, hence, the consensus is reached with the green train using a sub-optimal solution which is, however, compatible with the ones of all the other trains. In most other runs, the green train actually chooses an hypothesis using the shorter route.

As a further comparison, we assess the total train delay corresponding to the merged RTTP. Indeed, the selected hypotheses are merged into consistent RTTPs in this instance. We compare the total train delay obtained through consensus and through the centralized solution of the consensus problem with the centralized optimal solution found by RECIFE-MILP. In addition, we measure the total delay resulting from the application of the *timetable order*. Here, no traffic management decision is made to cope with the perturbation: each train uses the route assigned in the timetable, and if a train was originally planned to pass before another one in a given location, it will do so independently on delays. Table 2 reports these results. The total delay returned by RECIFE-MILP is about 18% smaller than the one obtained through the consensus problem, either centralized or decentralized. However, it is of the same order of magnitude, especially if considering that about 2400 seconds of delay are due to the considered perturbation. Indeed,

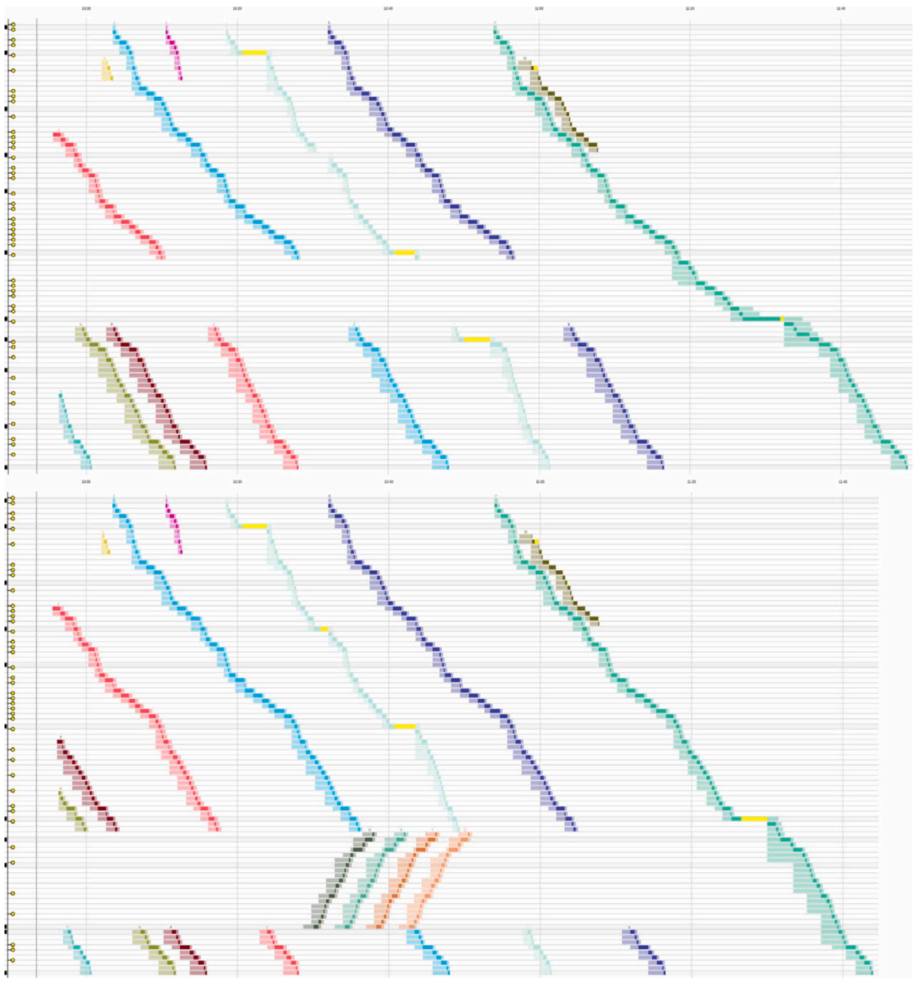


Fig. 9. Representation of the Gantt diagrams representing train paths in consensus (top) vs optimal (bottom) solutions in the worst run for consensus.

Table 1
Solution value returned by consensus algorithms and centralized optimization. Recall that (i) the solution value is the sum of the cost of the selected hypotheses, and (ii) the cost of a hypothesis is the total weighted delay of all trains of the instance, with all weights set to one but for the delay of the focal train, which is set to two. Hence, the solution cost counts for the delay of every train as many times as the total number of trains plus one. This explains the high values observed.

$\bar{h}$	Consensus			Centralized
	min	average	max	solution
5	123 793	124 049	124 212	123 750
10	123 983	124 294	124 762	123 750
∞	124 146	125 579	127 231	123 750

Table 2
Total train delay returned by consensus algorithm, centralized optimization of consensus problem and RECIFE-MILP.

$\bar{h}$	Consensus			Centralized	RECIFE-MILP	Timetable
	min	average	max	solution		order
5	6325	6376	6425	6325	5540	6650
10	6325	6391	6459	6325	5540	6650
∞	6325	6535	6877	6325	5540	6650

excluding entrance delay, all decision making methods find solutions in which the average delay per train is around three minutes. Remark that the optimal centralized solution of the consensus problem implies the same total delay as the best set of hypotheses returned by the decentralized consensus process ("min" column), while this does not hold in Table 1. The different relative behavior is due to the fact that, here, we apply the merge procedure while in Table 1 we simply summed hypothesis values. When merging, we extract from each chosen hypothesis only the path of the focal train. Then we merge all these focal train paths in a single RTTP, for which we compute the total delay. Indeed, in the best decentralized solution, all trains choose a hypothesis in which their path corresponds to the one which is optimum for the centralized consensus problem. Instead, some of the non-focal train paths in these hypotheses do not correspond to the optimal centralized ones.

Despite the worsening of the total delay due to decision decentralization, it is remarkable that the consensus solutions are always better than the timetable order: also from a pure system-level perspective, it is convenient to let train self-organize rather than applying the original plan. Indeed, the timetable order does not really challenge good optimization algorithms. However, it is a relevant benchmark to show that the current implementation of self-organization is capable of improvement over the baseline benchmark, despite suffering from the myopic decisions due to the local perspective of the trains and from the simplicity of the current module implementations, which keeps it quite far from centralized optimization.

7. Conclusions

In this paper, we proposed an original approach to railway traffic management based on self-organization. We consider intelligent trains that can (i) identify other trains with which they may possibly interact in the near future, (ii) formulate hypotheses of desirable traffic evolution, and (iii) reach a consensus with all possibly interacting trains. The proposal and assessment of such a self-organizing train traffic management is the ultimate goal of the European project SORTEDMOBILITY.

We formally designed self-organized railway traffic management as a modular process, which had never been done in the related literature before. We also proposed a preliminary proof of concept of the applicability of self-organization based on a simple implementation of each module and on a case study representing traffic in a French railway control area. We compared the performance of our stochastic, decentralized consensus algorithm with the centralized optimal solution of the consensus problem. Here, consensus achieves solutions that are extremely close to the centralized optimal ones. Moreover, we compared the total train delay corresponding to these solutions with the one that can be obtained by centrally optimizing traffic. The results show that, despite the current implementation of most modules being quite naive, trains can self-organize by exploiting the designed process. The total delay resulting from the consensus process is of the same order of magnitude as the centralized optimal one. Moreover, it is lower than the one achievable by implementing the planned train routes and precedences, which corresponds to the situation in which no active traffic management is deployed. In conclusion, the proposed process appears promising, although the experimental results indicate that a smart and well-designed implementation of each module will be necessary to make it competitive with respect to the centralized optimal traffic management in situations where the latter is not penalized by the size of the instance to be solved.

In future research, we will work to improve the implementation of all self-organization modules. In particular, the neighborhood selection will need to take into account possible interactions due to re-routing and re-ordering decisions. Hypothesis generation will need to consider passenger demand in the solution assessment, and hypothesis compatibility may benefit from the introduction of some tolerance. Various consensus algorithms will be tested, and a consistency check and repair mechanism will be included in the merge module. Moreover, the whole process will be integrated in a closed-loop optimization including a railway and a passenger simulator. Three case studies will make the object of comprehensive analysis, allowing the production of guidelines and recommendations for the deployment of self-organizing traffic management. A thorough experimental study will allow making sound conclusions on the potential of self-organizing railway traffic management. Indeed, self-organization may bring intuitive benefits in terms of good scalability, extreme flexibility and consideration of possible conflicting interests of different railway undertakings. However, this may come at the cost of myopic decision making due to the problem decomposition in train-centric sub-problems, or of difficulty of achieving efficient traffic management due to the stubbornness of some undertakings, for example. What kind of mechanisms will be able to tip the scale toward the benefits rather than toward the costs will most likely depend on the characteristics of the intended traffic.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work presented in this paper has been carried out in the context of the SORTEDMOBILITY project. This project is supported by the European Commission and funded under the Horizon 2020 ERA-NET Cofund scheme under grant agreement N 875022. Vito Trianni and Leo D'Amato acknowledge partial support by TAILOR, a project funded by EU Horizon 2020 research and innovation program under GA No 952215.

References

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

Baronchelli, A., 2017. The emergence of consensus: a primer. R. Soc. Open Sci. 5 (2), 172189.

Baronchelli, A., Castellano, C., Pastor-Satorras, R., 2011. Voter models on weighted networks. Phys. Rev. E 83 (6), 066117.

Baronchelli, A., Gong, T., Puglisi, A., Loreto, V., 2010. Modeling the emergence of universality in color naming patterns. Proc. Natl. Acad. Sci. USA 107 (6), 2403–2407.

Bettinelli, A., Santini, A., Vigo, D., 2017. A real-time conflict solution algorithm for the train rescheduling problem. Transp. Res. B 106, 237–265.

Cacchiani, V., Huisman, D., Kidd, M., Kroon, L., Toth, P., Veelenturf, L., Wagenaar, J., 2014. An overview of recovery models and algorithms for real-time railway rescheduling. Transp. Res. B 63, 15–37.

Caimi, G., Fuchsberger, M., Laumanns, M., Lüthi, M., 2012. A model predictive control approach for discrete-time rescheduling in complex central railway station areas. Comput. Oper. Res. 39 (11), 2578–2593.

Cambier, N., Miletitch, R., Frémont, V., Dorigo, M., Ferrante, E., Trianni, V., 2020. Language evolution in swarm robotics: A perspective. Front. Robot. AI 7, 12.

Castellano, C., Fortunato, S., Loreto, V., 2009. Statistical physics of social dynamics. Rev. Modern Phys. 81 (2), 591–646.

Corman, F., D'Ariano, A., Pacciarelli, D., Pranzo, M., 2010. A tabu search algorithm for rerouting trains during rail operations. Transp. Res. B 44 (1), 175–192.

Corman, F., Meng, L., 2015. A review of online dynamic models and algorithms for railway traffic management. IEEE Trans. Intell. Transp. Syst. 16 (3), 1274–1284.

Dorigo, M., Theraulaz, G., Trianni, V., 2021. Swarm robotics: Past, present, and future. Proc. IEEE 109 (7), 1152–1165.

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).

Fang, W., Yang, S., Yao, X., 2015. A survey on problem models and solution approaches to rescheduling in railway networks. IEEE Trans. Intell. Transp. Syst. 16 (6), 2997–3016.

Fischetti, M., Monaci, M., 2017. Using a general-purpose mixed-integer linear programming solver for the practical solution of real-time train rescheduling. European J. Oper. Res. 263 (1), 258–264.

Hamann, H., Reina, A., 2022. Scalability in computing and robotics. IEEE Trans. Comput. 71 (6), 1453–1465. http://dx.doi.org/10.1109/tc.2021.3089044.

Holley, R.A., Liggett, T.M., 1975. Ergodic theorems for weakly interacting infinite systems and the voter model. Ann. Probab. 3 (4), 643–663.

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

Jusup, M., Trivella, A., Corman, F., 2021. A review of real-time railway and metro rescheduling models using learning algorithms. In: 21st Swiss Transport Research Conference (STRC 2021). p. 27.

Khadilkar, H., 2019. A scalable reinforcement learning algorithm for scheduling railway lines. IEEE Trans. Intell. Transp. Syst. 20 (2), 727–736.

Lamorgese, L., Mannino, C., 2015. An exact decomposition approach for the real-time Train Dispatching problem. Oper. Res. 63 (1), 48–64.

Lamorgese, L., Mannino, C., Pacciarelli, D., Törnquist Krasemann, J., 2018. Train dispatching. In: Borndörfer, R., Klug, T., Lamorgese, L., Mannino, C., Reuther, M., Schlechte, T. (Eds.), Handbook of Optimization in the Railway Industry. Springer International Publishing, Cham, pp. 265–283.

Lamport, L., 2006. Fast paxos. Distrib. Comput. 19 (2), 79–103.

Leutwiler, F., Bonet Filella, G., Corman, F., 2023. Accelerating logic-based benders decomposition for railway rescheduling by exploiting similarities in delays. Comput. Oper. Res. 150, 106075.

Loreto, V., Baronchelli, A., Mukherjee, A., Puglisi, A., Tria, F., 2011. Statistical physics of language dynamics. J. Stat. Mech. Theory Exp. 2011 (04), P04006.

Lu, G., Ning, J., Liu, X., Nie, Y.M., 2022. Train platforming and rescheduling with flexible interlocking mechanisms: An aggregate approach. Transp. Res. E 159, 102622.

Luan, X., De Schutter, B., Meng, L., Corman, F., 2020. Decomposition and distributed optimization of real-time traffic management for large-scale railway networks. Transp. Res. B 141, 72–97.

Marcelli, E., Pellegrini, P., 2020. Literature review toward decentralized railway traffic management. IEEE Intell. Trans. Syst. Mag. 13 (3), 234–252.

Mascis, A., Pacciarelli, D., 2002. Job-shop with blocking and no-wait constraints. European J. Oper. Res. (143), 498–517.

Meng, L., Zhou, X., 2014. Simultaneous train rerouting and rescheduling on an N-track network: A model reformulation with network-based cumulative flow variables. Transp. Res. B 67, 208–234.

Mohanty, S., Nygren, E., Laurent, F., Schneider, M., Scheller, C., Bhattacharya, N., Watson, J., Egli, A., Eichenberger, C., Baumberger, C., Vienken, G., Sturm, I., Sartoretti, G., Spigler, G., 2020. Flatland-RL: Multi-agent reinforcement learning on trains. arXiv:2012.05893.

Mohapatra, D., Ojha, A., Khadilkar, H., Ghosh, S., 2022. Gatekeeper: A deep reinforcement learning-cum-heuristic based algorithm for scheduling and routing trains in complex environments. In: 2022 International Joint Conference on Neural Networks (IJCNN). p. 7.

Moretti, P., Liu, S.Y., Baronchelli, A., Pastor-Satorras, R., 2012. Heterogenous mean-field analysis of a generalized voter-like model on networks. Eur. Phys. J. B 85 (3), 1–6.

Nash, A., Huerlimann, D., 2004. Railroad simulation using OpenTrack. In: Brebbia, C., Allan, J., Sciutto, G., Scone, S. (Eds.), Computers in Railways IX. WIT Press, Southampton, United Kingdom, pp. 45–54.

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

Pease, M., Shostak, R., Lamport, L., 1980. Reaching agreement in the presence of faults. J. ACM 27 (2), 228—234.

Pellegrini, P., Marlière, G., Pesenti, R., Rodriguez, J., 2015. RECIFE-MILP: an effective MILP-based heuristic for the real-time railway traffic management problem. IEEE Trans. Intell. Transp. Syst. 16 (5), 2609–2619.

Pellegrini, P., Marlière, G., Rodriguez, J., 2016. A detailed analysis of the actual impact of real-time railway traffic management optimization. J. Rail Transp. Plan. Manag. 6 (1), 13–31.

Quaglietta, E., 2013. A simulation-based approach for the optimal design of signalling block layout in railway networks. Simul. Model. Pract. Theory 46, http://dx.doi.org/10.1016/j.simpat.2013.11.006.

Quaglietta, E., Pellegrini, P., Goverde, R., Albrecht, T., Jaekel, B., Marlière, G., Rodriguez, J., Dollevoet, T., Ambrogio, B., Carcasole, D., Giaroli, M., Nicholson, G., 2016. The ON-TIME real-time railway traffic management framework: A proof-of-concept using a scalable standardised data communication architecture. Transp. Res. C 63, 23–50.

Reina, A., Valentini, G., Fernández-Oto, C., Dorigo, M., Trianni, V., 2015. A design pattern for decentralised decision making. PLoS One 10 (10), e0140950 – 18.

Reynolds, E., Maher, S.J., 2022. A data-driven, variable-speed model for the train timetable rescheduling problem. Comput. Oper. Res. 142, 105719.

Rodriguez, J., 2007. A constraint programming model for real-time train scheduling at junctions. Transp. Res. B 41, 231–245.

S2R, 2022. Shift2Rail. URL https://rail-research.europa.eu/about-shift2rail/. (Accessed October 20th, 2022).

Samà, M., D'Ariano, A., Corman, F., Pacciarelli, D., 2017. A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. Comput. Oper. Res. (78), 480–499.

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

Shang, F., Zhan, J., Chen, Y., 2018. Distributed model predictive control for train regulation in urban metro transportation. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, pp. 1592–1597.

SORTEDMOBILITY, 2021. Self-organized rail traffic for the evolution of decentralized MOBILITY. URL https://www.sortedmobility.eu/. (Accessed October 20th, 2022).

Steels, L., 1995. A self-organizing spatial vocabulary. Artif. Life 2 (3), 319–332.

The Community Flatland, 2022. Flatland. URL https://flatland.aicrowd.com/intro.html. (Accessed 11/29/2022).

Toletti, A., Laumanns, M., Weidmann, U., 2020. Coordinated railway traffic rescheduling with the resource conflict graph model. J. Rail Transp. Plan. Manag. 15, 100173.

Törnquist, J., Persson, J.A., 2007. N-tracked railway traffic re-scheduling during disturbances. Transp. Res. B 41 (3), 342–362.

Valentini, G., Ferrante, E., Dorigo, M., 2017. The best-of-n problem in robot swarms: Formalization, state of the art, and novel perspectives. Front. Robot. AI 4, 1–43.

Valentini, G., Hamann, H., Dorigo, M., 2014. Self-organized collective decision making: The weighted voter model. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems. pp. 45–52.

Van Thielen, S., Corman, F., Vansteenwegen, P., 2019. Towards a conflict prevention strategy applicable for real-time railway traffic management. J. Rail Transp. Plan. Manag. 11, 100139.

Yong, C., Ullrich, M., Jiajian, L., 2017. Decentralized, autonomous train dispatching using swarm intelligence in railway operations and control. In: 7th International Conference on Railway Operations Modelling and Analysis - RailLille2017. IAROR, pp. 521–540.