Hindawi
Journal of Advanced Transportation
Volume 2021, Article ID 4664010, 18 pages
https://doi.org/10.1155/2021/4664010

Research Article

Linear Programming Model and Online Algorithm for Customer-Centric Train Calendar Generation

Tommaso Bosi and Andrea D'Ariano

Dipartimento Di Ingegneria, Sezione Di Computer Science and Automation, Università Degli Studi Roma Tre, Via Della Vasca Navale 79, Rome 00146, Italy

Correspondence should be addressed to Tommaso Bosi; tommaso.bosi@uniroma3.it

Received 26 April 2021; Accepted 31 May 2021; Published 21 June 2021

Academic Editor: Erfan Hassannayebi

An important objective for train operating companies is to let users, especially commuters, directly query the ICT system about trains' availability calendar, based on an online approach, and give them clear and brief information, expressed through "intelligent" phrases instead of bit maps. This paper provides a linear programming model of this problem and a fast and flexible heuristic algorithm to create descriptive sentences from train calendars. The algorithmic method, based on the "Divide and Conquer" approach, takes the calendar period queried in its whole and divides it into subsets, which are successively processed one by one. The dominant limitation of previous methods is their strong dependence on the size and complexity of instances. On the contrary, our computational findings show that the proposed online algorithm has a very limited and constant computation time, even when increasing the problem complexity, keeping its processing time between 0 and 16 ms, while producing good quality solutions that differ by an average surplus of 0.13 subsentences compared to benchmark state-of-art solutions.

1. Introduction

The European rail sector faces a number of important challenges that constitute together serious barriers for the enhancement of its attractiveness and competitiveness on the global market. This can be done through a comprehensive and coordinated approach to research and innovation and focusing not only on the needs of the rail system providers but also on the needs of the users.

Therefore, as reported in the Shift2Rail master plan [1], one of these is a quality of service challenge: rail still does not come across as a user-friendly transport mode, with 19% of Europeans simply avoiding taking trains because of accessibility issues. In today's hyper-connected society, railway customer service needs a radical rethinking to be adapted to the constant and rapid evolution of quality expectations of travellers.

Into the Shift2Rail framework, we can identify five main asset-specific Innovation Programmes (IPs), covering all the different structural and functional subsystems of the rail system, as illustrated in Figure 1. These five IPs are not independent of one another and, into each of them, customer satisfaction represents one of the major keys. By means of this interdependence, evolutions in the technology in one part of the system can lead to changes in performance in another part. Starting from this new viewpoint, optimization covers each phase of the railway organization process moving toward perceived service quality by rail customers.

Many examples of this new research approach can be recognized in the literature: to improve travel safety, Yin et al. [2] propose a mathematical formulation to minimize the crowdedness of stations during peak hours to synchronously generate the optimal coordinated train timetables; "To Wait or Not to Wait?" is the question submitted by Schanchtebeck and Schöbel [3] for the delay management problem, in order to satisfy two different customer categories; analyzing and seeking an equilibrium point between the optimization of reordering choices of train dispatchers and the route choice of passengers in the

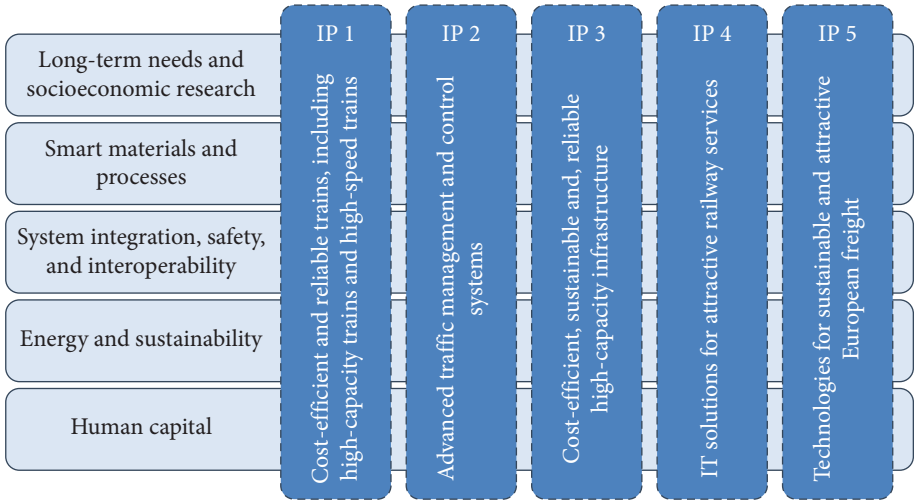

Figure 1: The five IPs of the Shift2Rail framework (source: Shift2Rail master plan).

available services of the railway transport network is rather the aim of Corman [4]; Hong et al. [5] define a method to solve train rescheduling and passenger reassignment simultaneously to facilitate real-time re-ticketing; some more research papers that aim to improve quality service experienced by users were selected by the special issue by D'Ariano et al. [6]; like Li et al. [7] where smart card data transactions are converted to a model of route choice, throughout the network, to improve the amount of insight into the travellers' behavior; D'Acierno et al. [8] aim to provide operations' parameters for metro systems so as to support the planning and implementation of energy-saving strategies while maintaining a targeted service level; a scalable method for dynamic profiling is introduced by Toader et al. [9] that aims to discover knowledge, like travel habits, from data in motion, and provide faster sharing mobility services in dynamic contexts; Botte et al. [10] focus on a Neighborhood Search Algorithm for optimal intervention strategies in the case of a metro system failure with the purpose to keep a certain service quality level; European journals themselves began to actively promote an ongoing special issue in order to "provide the highest quality with minimal travel cost and time to satisfy the ultimate needs of customers" (as mentioned in the call for papers of Journal of Advanced Transportation: *Advances in Modelling and Data-Driven Optimization of Urban Transport and Logistics*).

The problem studied in this paper belongs to IP2 and IP4 programmes of Shift2Rail. These two IPs aim to increase punctuality and the use of accurate and real-time data for improved passenger information, to minimize travellers' inconvenience. Based on the user's preferences, personal and secure mobile *Travel Companion* will store and share their personal preferences in a wallet, providing personalized journey and messaging help.

One of the most important and basic information for users about railways services' availability is the train calendar, which is seen by travellers as the final result of several different and complex organization processes, from train routing to scheduling, timetable generation, and so on [11–14]. Along with basic information, such as arrival and departure times, the first data that users need to know is on which days the service is offered. Due to the present-day pace of life, it could be too much effort for a person to check a train calendar if it is expressed in the numerical form. This could bring in mistakes and forgetfulness, leading to travellers' inconvenience. Therefore, we desire to generate a more readable way to communicate trains' availability in order to ease its comprehension and memorization by users. Actually, railway companies, such as Trenitalia S.p.A., already use web services to provide trains availability information related to predetermined time-lapse through bit maps accompanied by descriptive sentences. Nevertheless, this time-lapse is defined by the company, so the customer cannot directly query the ICT system about a specific period of interest. Our purpose is to develop an online train calendar generation tool that permits, especially commuters, to ask when service is offered into an arbitrary period. Even if the problem is a niche one, the little previous literature has approached it mostly through linear programming, developing mathematical models that give optimal solutions based on assumptions made, but with temporal complexity strongly depending on the sizes and characteristics of the instances. To avoid this negative scalability feature, we propose to solve the problem with a new fast algorithmic method, developed through *C* programming language, which has shown constant complexity and produces good quality solutions. To test the effectiveness of this method, we have compared its performance on 264 instances, with a mathematical model strongly inspired by the third and most efficient one proposed by Amorosi et al. [14], noting how the algorithm returns similar solutions while reducing the average processing time from 381 ms to 2.32 ms.

In addition to this improvement in temporal performances, the flexibility provided by this new approach allows it to print out more detailed and suitable sentences, based on the specific transport field considered, by modulating the

core body of the algorithm code but without altering the computational complexity. A natural extension in the application of this approach can be the literal description of urban rail and bus systems' calendars, where a periodicity on the service availability also exists. In this case, given the smaller scale of instances, one could enrich the sentences' descriptions by considering, besides days' timetables, hours' ones as well, in order to share more compact availability information with passengers and to better satisfy the customer needs, which is the ultimate goal of train operating companies.

The major contribution of this paper lies in a new solving method applicable to quickly generate descriptive sentences from event calendars expressed through bit maps, in a quicker and more adaptable way than the earlier ones, in order to be applied as an online tool. Therefore, the technique presented can be extended to any application for online textual description service availability, also in other transport fields such as airplanes, ferry, and long trip buses. But this technique can be also adopted to describe the frequency of any event having a certain periodicity like sports events, opening days of commercial activities, and so forth.

To summarize, this paper considers the problem of generating descriptive sentences from train calendars, especially for commuters. The state of the art proposed a linear programming model which inspired us to create an improved model that can be easily transformed into an online tool. The main challenge of this modeling approach is the strong dependence of processing times on the complexity of the instances. To fill this gap, we propose a fast heuristic algorithm based on the "Divide and Conquer" approach, which is implemented in C language. This has been tested on 264 instances. The results report processing time much shorter than the ones required by CPLEX to solve the model, thus proving a better propensity to be used as a practical online tool.

We next introduce a brief overview of the paper structure: Section 2 reviews the most related literature, specifically the third model proposed by Amorosi et al. [15], as this is the only available research paper (to the best of our knowledge) that considers the specific train calendar generation problem in the context of the Italian railways; Section 3 describes how the ICT system works and at what point of the process of sharing our algorithm is inserted, and provides the Integer Linear Programming (ILP) model introduced to model the studied problem, which will be compared in Section 4 with the developed online algorithmic method (i.e., the heuristic algorithm) and outlines the proposed method in all its key phases through a high-level flowchart; Section 4 shows numerical simulations performed, arguing some observations about the performance values achieved by the heuristic algorithm and the ILP model and defining strengths and weaknesses of each approach; Section 5 concludes the paper by summarizing the main findings, by suggesting directions to improve the heuristic algorithm, and by summing up other possible applications of the proposed tool.

2. Literature Review

This section is mainly focused on the third model presented by Amorosi et al. [15]. The reason is related to three different considerations: first, the problem is a niche one, so there is very little material about possible solving methods [16]); second, this paper is the only one that considers the Italian railway context, with its particular passenger timetables; third, the third model is the fastest of the three models presented in their paper. Furthermore, the simulations studied in Amorosi et al. [15] are created with the active participation of Trenitalia, so they can be considered as practical cases, in which real-world characteristics have been considered for the 264 instances used in our numerical experiments.

The idea underlying the method proposed by Amorosi et al. [15] consists of several phases. First of all, the method takes as input a *periodicity*, which is a binary vector that represents the availability of the train in the time-window queried, and associates with each of its entry a progressive index (we will discuss these input data more in detail in section 3). Then, from the periodicity, only some days, in which the service is available, are extracted. The periodicity is decomposed based on 46 typologies of binary vectors, called *clusters*, which refer to a particular availability frequency, such as "*all Mondays*," "*all Tuesdays*," . . ., "*Holidays*" and so on, currently adopted by the main Trenitalia's ICT systems. An example of this relevant passage for the train calendar can be seen in Figure 2, considering the only "*Weekends*" and "*Wednesdays-Thursdays*" clusters.

Once all the clusters are created for the periodicity in the input, some copies are replicated for each cluster. These copies are eventually used in the case of a particular cluster that is used to represent more than one subperiod of the periodicity. The simulations show that a number of 5 copies should be enough for any kind of instance. The input data are given to a mathematical model implemented in the solver *IBM ILOG CPLEX*, which returns the minimum number of sub-vectors extracted from the clusters-copies chosen.

To choose from which cluster-copy the sub-vectors must be extracted, the following information is defined:

(i) A quality threshold α, forcing the minimum percentage of ones that a sub-vector must contain to be feasible

(ii) The minimum length l of the sub-vector extracted

(iii) The start date $Y^I_{c,k}$ of the sub-vector extracted from the cluster c copy k, which is an integer variable

(iv) The final date $Y^F_{c,k}$ of the sub-vector extracted from the cluster c copy k, which is an integer variable too

The adopted solver generates from each cluster c copy k all feasible subvectors based on the first two parameters α and l. Then, defining the start date $Y^I_{c,k}$ and the final date $Y^F_{c,k}$, the solver decides from which cluster c copy k to extract the most suitable sub-vectors for each sub-period of the periodicity. If the sub-vector chosen is populated by some zeros, those are described in the corresponding sentence as exceptions.

3	4	10	11	17	18	24	25	31	32	38	39
1	1	1	1	0	1	0	0	0	0	0	0

(a)

1	7	8	14	15	21	22	28	29	35	36	42	43	49	50
0	0	0	0	0	1	1	1	1	1	1	1	1	1	1

(b)

Figure 2: Examples of "Weekends" (a) and "Wednesdays-Thursdays" (b) clusters extracted from the practical case.

In the absence of the original implementation, to build up a comparison as accurate as possible and directly check the strengths and weaknesses of this approach, we have personally developed a mathematical model strongly inspired by the Amorosi et al.'s [15] model (a detailed description of this model will be given in section 4). This model has three fundamental differences with the work done in Amorosi et al. [15]:

(1) Our model is implemented through Python, importing and using the *docplex* module, and it can thus be thought of as a stand-alone tool, implemented into a single environment. On the contrary, Amorosi et al.'s [15] model was being fed through data generated in *Microsoft .NET*, solved in *IBM ILOG CPLEX*, and the numerical solutions were then translated into train calendars by an external script. Using the *docplex* module, we have linked our optimization model with the preprocessing and postprocessing phases, effectively making it an online tool, which receives input data about periodicity queried by the users and returns the corresponding descriptive sentence (Figure 3).

(2) Before giving any input data to Amorosi et al.'s [15] model, the previous approach applied preprocessing functions that allow their model to know when a specific day was associated with a cluster and when it was not associated. On the contrary, in the new model presented in this paper, this information is expressed by a binary data $c^*_{d,c,k}$. When $c^*_{d,c,k}$ for the day d, cluster c, copy k is equal to 1, this means that day d is covered by that cluster-copy, 0 otherwise. This additional input data allow our model to avoid using a preprocessing phase and directly transfer the information on whether or not a day belongs to a cluster.

(3) In our model, the parameter l considered by Amorosi et al. [15] to extract sub-vectors is removed, along with constraints (1). The removal of this parameter allows to lighten the model of $|C| * |K|$ constraints, where C is the set of clusters and K the set of the clusters-copies used by the model.

$$Y^F_{c,k} - Y^I_{c,k} \geq (l-1) * x_{c,k} \quad \forall c \in C \ \forall k \in K. \tag{1}$$

In constraints (1), to consider single scattered days also, we need to set the parameter l equal to 1, resulting in the argument to the right to be set equal to zero. With this setting, the only case in which any of constraints (1) can be violated is when the difference between $Y^F_{c,k}$ and $Y^I_{c,k}$ is negative, that is, when $Y^I_{c,k}$ is associated with a date preceding the one associated with $Y^F_{c,k}$. However, this case is not allowed by constraints (4) and (5) in our model. As a result, there is no reason to enforce constraints (1), and we thus remove them from our model.

The simulations in Amorosi et al. [15] showed four main limitations of their approach:

(i) The processing times are strongly dependent on the size and complexity of instances in the input, and this is not so good for an online tool. In Figure 4, we can see a scatter plot of the processing time for each of the 264 instances, expressed in milliseconds. The chart shows significant variability in the distribution of the values, with peaks up to 4.5 sec. These peaks correspond to instances populated by single scattered days. This fact is one of the weaknesses of their approach along with the low-quality solutions associated with some instances.

(ii) The use of a percentage threshold α (based on the size of the sub-vector considered) could be a double-edged sword as, if the periodicity is quite long, the solver could not extract the most properly cluster, and add many exceptions to the sentence printed in output. During the simulations, we tried out α values equal to 80% and 90%, noting that: in the first case, the solver does not tend to extract the most suitable descriptive clusters and/or add several exceptions, when the periodicity size is particularly extended. This tendency is illustrated by Figure 5; in the second case, decreasing the number of exceptions permitted, their model has difficulties in the processing instances, which are characterized by single scattered days, both in terms of quality and computation time. For example, if the briefest description for the periodicity is *"The service is provided from Monday to Saturday from x/y to z/w"* but the time window considered is wide enough, their model solution could take a sub-vector of the cluster-copy *"Monday-Friday"*, while the corresponding solver prints out *"The service is provided from Monday to Friday from x/y to z/w except for Saturday t/y, Saturday h/y…"*. However, for the comparison between the different methods, we fixed the threshold to 90%, in order to prevent the solver from using too many exceptions, lowering the quality-related performance.

(iii) The parameter l, being a fixed value, must be equal to 1, in order to cover the periodicity populated by single days as well. Due to this necessity, their model has several difficulties both in the timely processing and in

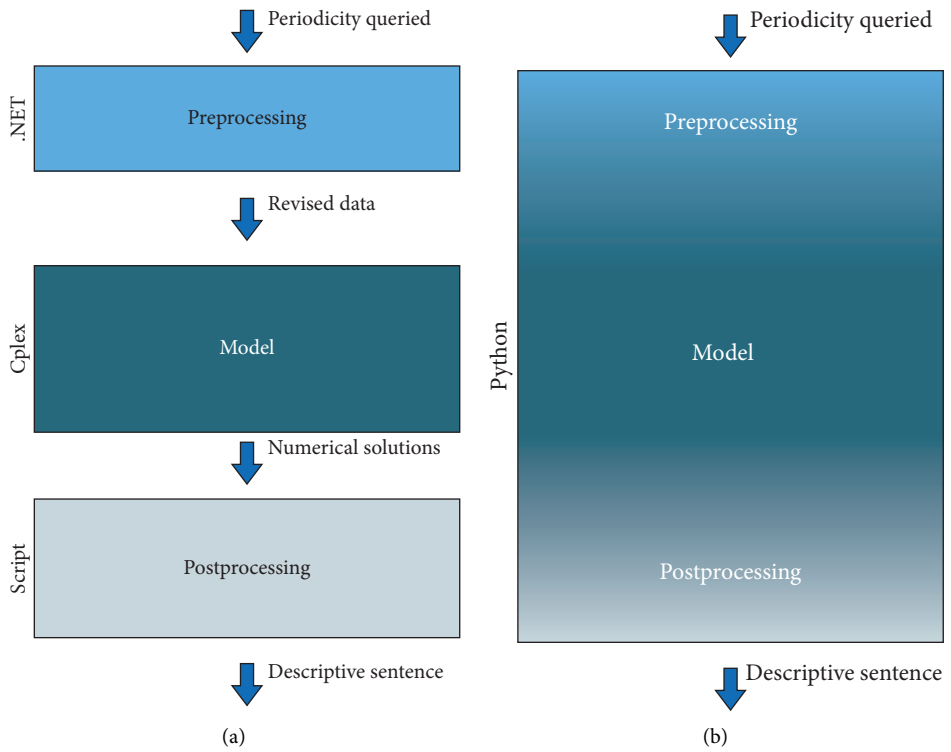

Figure 3: Structural difference between the previous model (a) and the newly developed model (b).

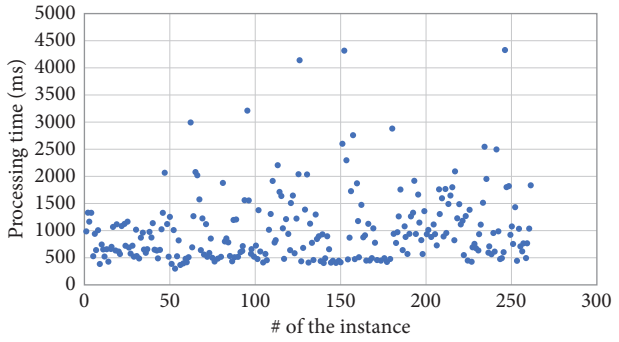

Figure 4: Scatter plot on the running times for the 264 instances of Amorosi et al. [15].

the association of sentences with instances with single scattered days. When their model has to process these types of instances, we observe both a considerable increase in processing times and the following solver's decision has to extract whatever cluster-copy that covers those days, simply by modulating the start and final dates of the sub-vector. For example, if in a week the service is available only on Wednesday, their model could take a copy of the cluster *"Working days,"* while the corresponding solver prints out *"The service is provided on working days from 13th May to 13th May."* Even though the last sentence is correct, the sentence is clearly not so effective and might confuse the users.

(iv) Their model does not consider the possibility of inserting new clusters without involving a significant increase in processing times or storing and adding extra days to the subsentences, instead of using one or more additional clusters. During a subperiod of the periodicity, for example, if the service is provided on Monday, Tuesday and Saturday, the sentence in output will be *"The service is provided on Monday and Tuesday from x/y to z/w; on Saturday from x/y to z/w,"* that is, the solver will consider two different clusters-copies. On the contrary, we could insert new clusters such as *"Monday-Tuesday and Saturday"* to use one cluster only and make the sentence more readable and effective. Why was this not performed in the approach proposed in the literature? Probably because considering all possible combinations among the week days would significantly increase the computational complexity of their approach. Otherwise, if the extra day concerning the cluster associated with that week is exactly one, we could print "*The service is provided on Monday and Tuesday from x/y to z/y including Saturday t/y*" in output.

These limitations and new features can be, respectively, eliminated and realized through the speed and flexibility of our algorithmic method, which is based on *C* programming language, as described in Section 3.

3. Materials and Methods

As we anticipated in the previous section, starting from a train calendar expressed through a bit map (Figure 6), we want to query a defined time window and generate the

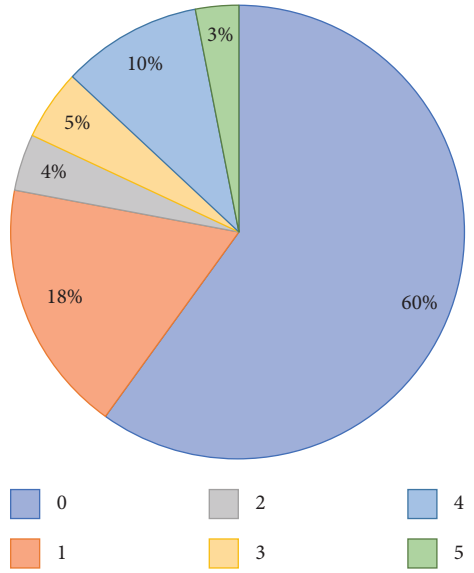

Figure 5: Percentages of the number of exceptions printed out by the model when the solutions differ.

clearest and briefest sentence describing the train availability. Within the ICT system, the calendars are represented by binary vectors, in which the zeros correspond to the days on which the service is not provided and the ones on which it is provided.

The binary vector based on the time window of interest and extracted from this calendar form is called *periodicity*. Figure 7 is an example of periodicity referred to the time window introduced in Figure 6. Due to the intersection between mathematical and natural language domains, there are many different ways to express the same periodicity.

In this case, for example, to represent the periodicity through a sentence, we can use the positive way,

"The service is provided on working days from 1st May to 30th June."

Or the negative one,

"The service is not provided on weekends from 1st May to 30th June."

However, throughout this paper, the positive way is used. For the positive way itself, different representations of the same information could be implemented for the same instance. We could say that *"The service is provided from Monday to Thursday from 1st May to 30th June except for 1st May, 8th May,..."* and so on, or, as we reported above, *"The service is provided on working days from 1st May to 30th June."*

The most readable among them is for sure the second one. For this reason, we want to develop a fast tool that can automatically recognize which representation is better and prints it out. According to our study as well as previous studies, the cleverest descriptive phrase is the briefest one, which is more readable and storable.

Obviously, if a service is available always or on Monday all year long, there is no need at all for optimization. However, the use of a heuristic algorithm or mathematical programming approach is motivated by complex cases of service availability, where a nonoptimized sentence can be very long and not easy to understand. This can thus be even useless in some cases for the customers. The practice case considers the distribution of service availability during the year depending on many variables, such as customer demand, holidays, the number of trains, limited resources, and operational constraints [5, 17].

Following the practical case (as shown in Figure 8), we can see a more likely train calendar queried, along with its periodicity, populated by different subperiods of train availability and by exceptions and days in surplus.

The most concise descriptive phrase for the periodicity below could be:

"*The service is provided on the weekends from 3rd July to 18th July except for Saturday 17th July; on Wednesdays and Thursdays from 21st July to 5th August including Friday 6th August; on working days from 9th August to 20th August.*"

We generate this type of sentence by inserting the train calendar's periodicity as input, which is a binary vector. In Section 2, we mentioned the ILP model implemented to solve the studied problem, in order to compare the literature approach with the algorithmic approach presented in this paper. In the following, we provide its description.

3.1. Notations and Mathematical Formulation. We consider the following input data:

O = set of operating days associated with the periodicity (in which the service is provided).

C = set of clusters.

$C_{c,k}$ = k-th copy of cluster $c \in C$.

$d^*_{d,c,k}$ = (position of the date d in $C_{c,k}$) +1.

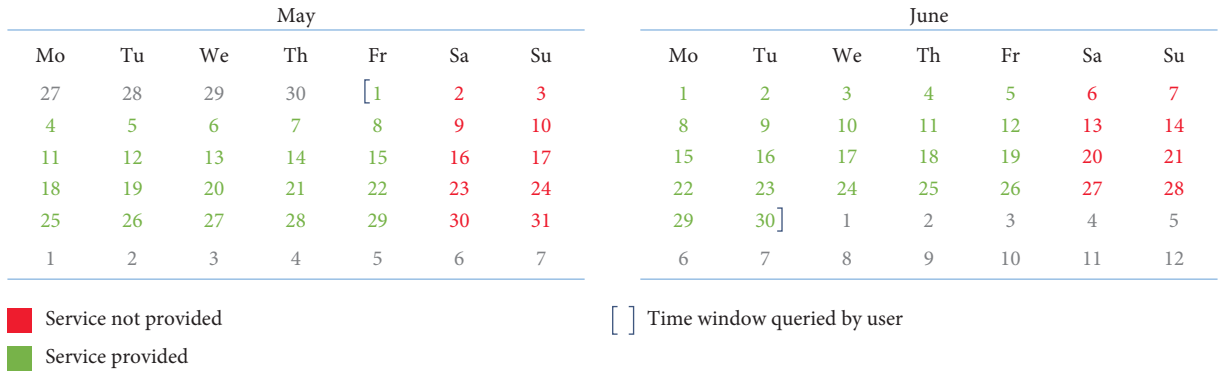

Figure 6: Example of a train calendar.

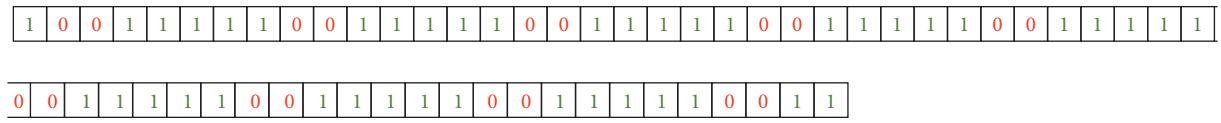

Figure 7: Example of periodicity extracted from the train calendar.

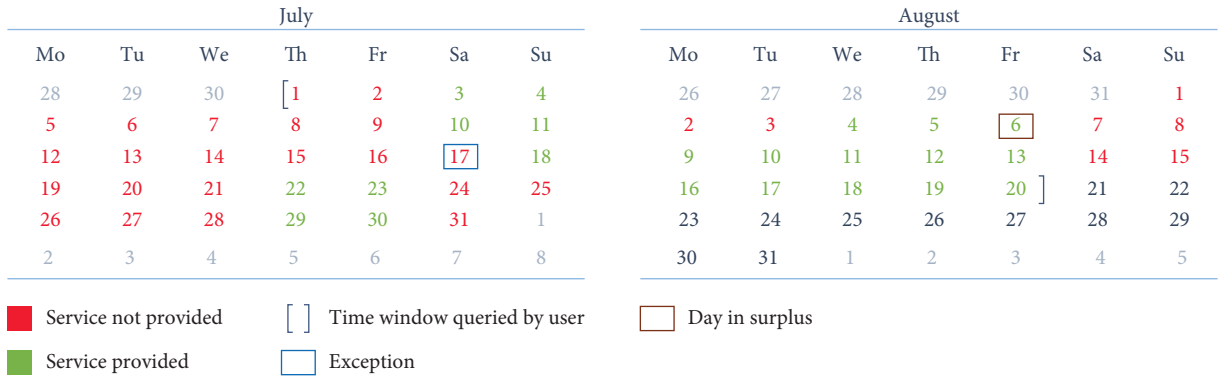

Figure 8: A practical case.

$c^*_{d,c,k} = \{1,$ *if the date* d is covered by the copy $C_{c,k}$ of cluster c, 0, otherwise.

α = minimum percentage of ones that the sub-vectors must have to be feasible.

Tot = cardinality of the periodicity.

M = big integer.

We define the following decision variables.

(i) Integer variables:

$Y^I_{c,k}$ = integer representing the start position of the sub-vector extracted from the copy $C_{c,k}$ of cluster c

$Y^F_{c,k}$ = integer representing the final position of the sub-vector extracted from the copy $C_{c,k}$ of cluster c

(ii) Binary variables:

$K_{d,c,k} = \{1,$ *if the date* d is covered in the *solution* by the copy $C_{c,k}$ of cluster c, 0, otherwise.

$x_{c,k} = \{1,$ if the the copy $C_{c,k}$ *of cluster* c is chosen in the solution, 0, otherwise.

Through these input data and decision variables, we present the following ILP model:

$$\min \sum_{c \in C,k} x_{c,k}. \quad (2)$$

subject to

$$\sum_{d\in O} K_{d,c,k} * c^*_{d,c,k} \geq \alpha * \left(x_{c,k} + Y^F_{c,k} - Y^I_{c,k} \right) \quad \forall c \in C \quad \forall k \tag{3}$$

$$Y^F_{c,k} \geq d^*_{d,c,k} * K_{d,c,k} \quad \forall d \in O \ \forall c \in C \quad \forall k \tag{4}$$

$$Y^I_{c,k} \leq (1 - M) * d^*_{d,c,k} * K_{d,c,k} + d^*_{d,c,k} * M \quad \forall d \in O \quad c \in C \quad \forall k \tag{5}$$

$$\sum_{c\in C,k} K_{d,c,k} * c^*_{d,c,k} \geq 1 \quad \forall d \in O \tag{6}$$

$$x_{c,k} \geq \frac{\sum_{d\in O} K_{d,c,k}}{Tot} \quad \forall c \in C \quad \forall k \tag{7}$$

As we can see in the mathematical models (2)–(7), the objective function minimizes the number of sub-vectors extracted, represented by $x_{c,k}$, the binary variable associated with the cluster c and copy k (2). The following constraints explain, respectively, that the percentage of ones into a feasible sub-vector must be greater than or equal to the threshold α (3); constraints (4) and (5) impose that the two integer variables, $Y^I_{c,k}$ and $Y^F_{c,k}$, of a feasible sub-vector take progressive indexes, respectively, before the first date and after the last date covered by the sub-vector that they are associated with; every date in which the service is available must be covered at least one time (6); if a cluster c and copy k are chosen in the solution to cover at least one date of the periodicity, the variable $x_{c,k}$ associated with this must be activated (7). Based on preliminar numerical simulations, we have set the threshold α equal to 0.9. The reason is that if we set this threshold to a lower value, the solver would generate a less fitting sub-vector along with many exceptions; on the contrary, if we set it to a higher value, the solver would take too much time in processing the tested instances.

We will next refer to this mathematical model as "the model," in order to get the reading smoother.

At this point, we can describe the proposed heuristic algorithm. First of all, as an online tool, it interacts with users by accepting the input data queried: start and final dates of the time window of interest. To avoid the use of external one-time filled calendars, as happened in previous literature, the second step plans to generate an internal calendar through mathematical formulas. Then, the periodicity generated by the ICT systems is taken and divided into the weeks that make it up. We can see how the algorithm approach is quite different from the previous one, indeed; while the model considers the periodicity as a whole, the algorithm follows the "Divide and Conquer" approach [18]: taking the problem in its entirety and dividing it into less complex subproblems. The idea here comes from the *Work Breakdown structure* [19] activity, used to divide large projects into project segments, called *leaves*, to be assigned to each operating unit.

Our algorithmic method is implemented in *C* language. The decision to utilize a programming language, like *C*, is related to two different reasons: this is a specific request of the train operating company; it is a compiled one, therefore it is faster than a programming language interpreted [20]. Our first aim was to mitigate the variability and the duration of processing time employed by past literature, in order to develop a more practical tool. Furthermore, we wanted the possibility to insert more clusters based on the particular railway context to improve the briefness of sentences printed and the chance to create useful functions, such as the one related to extra days. But how does the heuristic algorithm work? The code is divided into two macro-phases: a preparatory one that manipulates the periodicity, and the next one that processes the periodicity itself. The main phases of the proposed heuristic are illustrated by the flowchart in Figure 9.

After we have carefully manipulated the input data, we need a tool that transforms the subperiods of initial periodicity into sentences. This one is a classical *C* data structure called the *Box*. The Box is like a single machine that receives the WIPs, that is the weeks, through a conveyor belt and does something and pulls out the final products, which are the sentences. The conveyor belt is represented by a while loop, which is the core of the second phase. The exit condition for the while loop is connected with the number of weeks of the input periodicity: the while loop goes on until the last week has not been processed.

What does the machine do exactly to generate a solution to the studied problem? Into the Box, the weeks are stored three at a time. A descriptive cluster of the days, on which the service is offered, is assigned with the week in the first position. Later, the weeks stored in the second and third positions are compared with the one in the first position, depending on the case, to check if they are different, similar, or equal.

This comparison is done based on the days of the week in which the service is available:

(i) If two weeks have the same days on, this means that the service is offered on the same days and so the compared weeks can be considered as equal. The second week is thus "incorporated" into the first one. This means that the cluster associated with the first week is also associated with the second one, and when the sentence, which describes the service availability of these two weeks (based on the cluster associated) is printed out, a specific time window will be considered, which starts from the first day of availability of the first week to the last day of availability of the second week.

(ii) If two weeks differ from each other by only one day of unavailability or availability, this means that this day could be, respectively, an exception or an extra day. If this is the case, these weeks can be considered as *similar*. To check if it is true, we compare also the first week, with the third one, and if they are the same, the similarity is confirmed. The second week is thus incorporated into the first one, and the day apart is labeled as an extra one or an exception, based on the case.

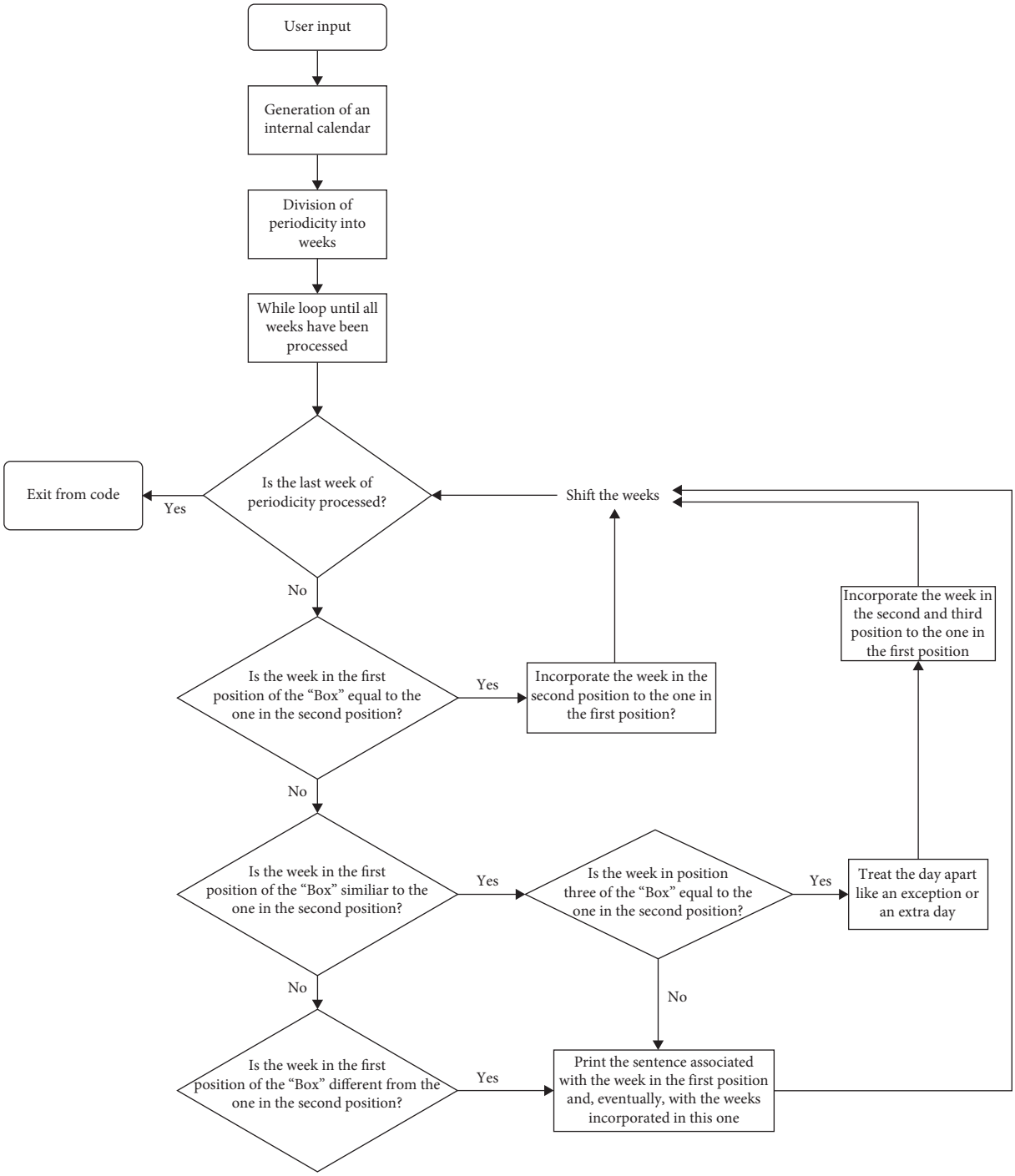

Figure 9: Flowchart of the proposed heuristic algorithm.

(iii) If two weeks differ from each other by two or more days of unavailability or availability, we consider these as different. In this case, we do not incorporate the second week with the first one, and we represent the two weeks, and the ones incorporated with them, with different clusters. Therefore, the algorithm will print out different sentences from these two subperiods.

An example for each case is shown in Figure 10. Once we know if the week in the second position of the Box is different (c), similar (b), or equal (a) to the one in the first position, a function scrolls the weeks in order to sequentially process all the weeks of the periodicity.

Once all the weeks have been processed and the sentences associated with the subperiods have been printed out, the code exits from the while loop. However, for the practical use of this

tool, we will consider the implementation of an infinite loop, such as a *while True*, containing the whole code along with a break condition. Another important feature (that solved an important limitation of the previous method) is that, thanks to the flexibility of the *C* programming language, we inserted other clusters in addition to the starting 46, such as the cluster *"Monday-Tuesday and Saturday"* and all possible combinations between different days of the week, without altering the computational complexity, as for the code they are only strings to search. This added feature allows us to reduce the number of clusters used to print out a solution. As we will discuss in the next section, this approach ensures that a good constructive solution is generated with constant complexity and the clusters are adapted to the specific rail context in which the train operating company is part.

To better explain how the workflow of our algorithm operates, a step-by-step description of the activities performed on a trial instance is presented. We assume that the period queried by a user is composed of only four weeks, from 1st March to 28th March (Figure 11).

First of all, the algorithm takes the input data, start and final dates of the time window, along with the periodicity associated with the specific train requested, and creates the internal calendar, which is a matrix. Then, the periodicity is divided into its weeks based on the weekdays assigned to each progressive index through the following mathematical formula. The variables in this formula are described in Table 1.

$$d_w = \mathrm{mod}\left(\frac{y + (y - 1/4) - (y - 1/100) + (y - 1/400) + d_c}{7}\right). \tag{8}$$

For example, to figure out which day of the week corresponds to 10th January 2020, we will consider $y = 2020$ and $d_c = 10$, obtaining a d_w equal to 6, that is, Friday.

The data structure "Box" takes the first three weeks, associates the cluster "Monday-Wednesday" with the week in the first position, and starts the comparison between the latter and the one in the second position (Figure 12).

Since they are not equal but differ by one day only, Wednesday 10th March, the algorithm checks if they could be similar or different by comparing the week in the first position with that in the third position. The latter is equal to the week in the first position, so the similarity is confirmed. Consequently, the weeks in the second and third positions are incorporated with the one in the first position. Wednesday 10th March is stored as an exception and the time-lapse of cluster "Monday-Wednesday" is updated, starting from Monday 1st March to the last day of the week in the third position, which is Wednesday 17th March. The weeks in the second and third positions are scrolled, and the fourth week enters the "Box." The latter is compared with the week in the first position and, as they have only two days on in common, they are classified as "different." The cluster associated with the week in the first position is then printed out as a subsentence of the periodicity, the fourth week is inserted in the first position of the "Box," and a new cluster is attached with it, that is, the cluster "on Weekends" on Saturday 27th and Sunday 28th March.

Since all the weeks are processed, the counter meets the exit condition and the while loop ends. The sentence connected to the periodicity is thus displayed, as we can see in Figure 13.

Once the algorithm has been described, some differences with the literature approach can be identified:

(i) The modeling approach looks at the instance in its entirety and creates the clusters-copies over the overall time window queried. This leads to an increase in the computational effort to compute the best possible solution. On the contrary, the heuristic algorithm looks at the instance week by week, generating and associating, when required, a cluster of seven days with the week in the first position of the box (Figure 14). This cluster is compared with the weeks ahead, and no new cluster is generated until there exists a significant difference between two weeks, in terms of days of service availability. This trade-off in the view and segmentation of the periodicity allows to sharply reduce the processing times.

(ii) Through the modeling approach, introducing a new cluster c_1 to improve the solution quality would lead to the computation of that cluster over the whole instance along with the replication of all its copies. Differently, the proposed heuristic algorithm implements clusters through strings and, therefore, the most considerable strain lies in the association of each week with a cluster, performed through the sorting of arrays of a length of seven. This allows to easily insert new clusters with a considerably reduced computational effort compared to the model, in order to improve the quality of descriptive sentences.

(iii) Another relevant difference between the model and the algorithm is the introduction of a function that considers extra days between weeks. Let us assume that, in the period of interest, the service is offered with the same frequency, except for some scattered weeks, in which there is an extra day, as in the case of festivities. In this case, the modeling approach will associate at least two clusters with the periodicity, and therefore, two subsentences. Through the extra days' function implemented in the algorithm, these days will be stored as extra days and added to the single cluster choices to describe the periodicity. This sample function is an example of the flexibility of this type of tool and its potential to be adapted to different public transport contexts, to increase the service quality perceived by the users.

4. Results and Discussion

In order to compare how the two methods are presented in this work, we tested them on 264 instances imported from a .csv file. While the model was developed and executed on the IDE Pycharm 2020.3.3, for the heuristic algorithm we used Code:Blocks 20.03. All tests were performed on a Windows

	Week 1		Week 2
Monday	1	=	1
Tuesday	1	=	1
Wednesday	1	=	1
Thursday	0	=	0
Friday	1	=	1
Saturday	0	=	0
Sunday	0	=	0

(a)

	Week 1		Week 2
Monday	1	=	1
Tuesday	1	=	1
Wednesday	1	=	1
Thursday	0	=	0
Friday	0	!–	1
Saturday	0	=	0
Sunday	0	–	0

(b)

	Week 1		Week 2
Monday	1	=	1
Tuesday	1	=	1
Wednesday	1	=	1
Thursday	0	!=	1
Friday	0	!=	1
Saturday	0	!=	1
Sunday	0	=	0

(c)

Figure 10: Examples of the three possible comparison results.

operative system Intel i7 processor with 2.6 GHz and 16 GB of RAM. The whole 264 instances are created based on the ones presented in the study of Amorosi et al. [15] and on the potential weakness of each method, that are, respectively, the less effective processing of the descriptive cluster "Every day," when it is used as a solution by the heuristic algorithm; the poor performances, in terms of efficiency and effectiveness, generated by the model approach in processing periodicities populated by single scattered days.

We test the ILP model and the heuristic algorithm for various possible situations in order to quantitatively evaluate their potential. We also investigate the exponential time and the high number of clusters used by the model of Section 3 to process periodicity populated by single days. This creates "rare events" that significantly alter the performance indexes employed. Therefore, the periodicities described in whole or in part by the cluster "Every day" represent 24%, while the ones populated by single days are 4% of all the tested cases.

The experiments consist of three different sets of periodicities, updated to 2020: the first one is composed of 62 instances ranging from 13th September to 24th November 2020; the second one contains 88 instances ranging from 3rd May to 2nd August 2020; the last one is made up of 113 instances from 2nd February to 20th May 2020. So, the sets cover, respectively, segments with a length equal to two, three, and almost four months.

We considered both temporal and quality indexes, precisely: average, maximum, and minimum processing times; average, maximum, and minimum numbers of descriptive clusters used. The values obtained for each index are indicated in Table 2.

Two other important indexes that we can consider are: the number of different clusters used by each method when two different solutions are printed out; the number of exceptions used by each method when the solutions differ by the number of clusters used. This last index is important due to the

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	1	1	0	0	0	0	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	1	1

Figure 11: Train calendar and periodicity of the trial instance.

Table 1: List of symbols used in mathematical formula (8).

Symbol	Description
d_w	Values ranging from 0 to 6, where 0 is associated with the day of week "Saturday" and 6 with "Friday". Therefore, for "Monday" this value will be equal to 2.
y	Current year. For the simulations, we consider this value equal to 2020.
x	The integer portion quotient x inside the brackets.
d_c	The number of the year-to-date days, starting from 1st January to which we associate the value 1.
mod	The modulo operator, which returns the remainder of the division inside the round brackets.

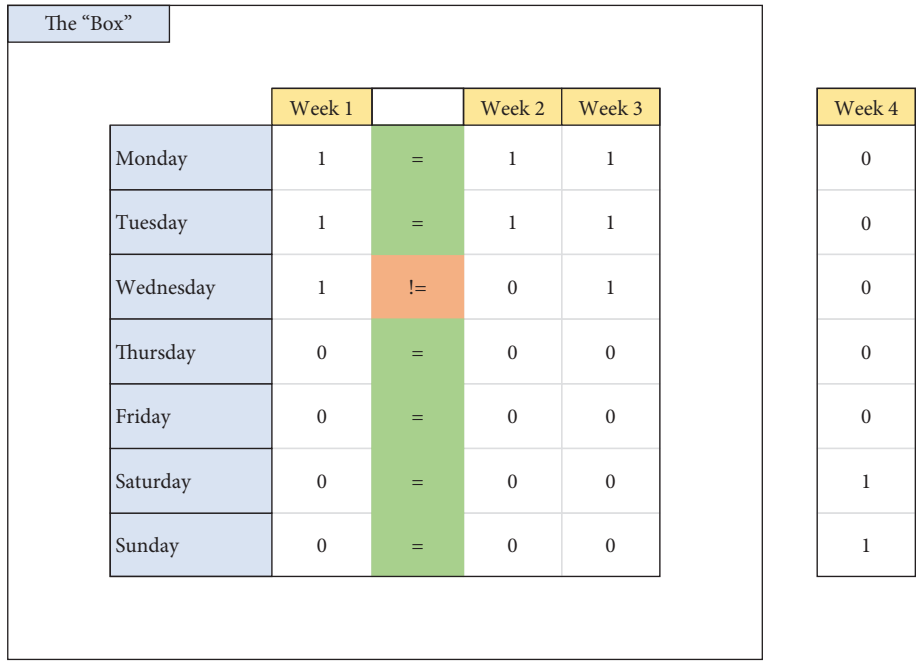

Figure 12: Illustration of the data structure "Box."

Week 1	Week 2	Week 3	Week 4
1	1	1	0
1	1	1	0
1	0	1	0
0	0	0	0
0	0	0	0
0	0	0	1
0	0	0	1

The service is provided: from Monday to Wednesday from 1st March to 17th March expet for 10th March; on the weekend of 27th and 28th March.

Figure 13: The sentence printed out for the considered trial instance.

Table 2: Values measured for each method based on the performance indexes considered.

Time	ILP model	Heuristic algorithm
Average	1000	2.32
Max	4334	16
Min	300	0
Number of clusters		
Average	1.19	1.32
Max	6	5

Note: time is expressed in ms.

limitation identified with the use of a percentage threshold by the model. In fact, what can happen is that the model may use fewer clusters, but along with many more exceptions.

As we can see in Figure 15, the percentage of periodicities to which the two methods associate different number of clusters is 30% of the whole tests. Eighty-two percent of the solutions of this 30% differ by one cluster, 9% differ by two clusters, and the other 9% differ by three or more clusters.

However, the latter 9% can be traced back to the difficulties of the model to process instances populated by single days. The following figure (Figure 5), rather, represents the percentages of the number of exceptions inserted in the sentences printed out by the model when the two methods used a different number of clusters to describe each periodicity.

The following plots (Figure 16) consider, respectively, the processing time of each periodicity tested, and the number of clusters used to print out the solution by each method.

The results in Figure 16 show how the proposed heuristic algorithm solves our initial research questions and the limitations identified from the past literature. First, we were looking for an online tool that could interact with rail users. Due to this specific feature, the tool has to be very fast. To achieve this goal, the past literature made use of external tools, such as.NET and scripts in addition to their model, thus increasing the resulting processing times. The model and the heuristic algorithm proposed in this paper are both stand-alone tools. Figure 17 illustrates the distribution of processing times into specific time frames and, as we can see, 40% of them exceeds the seconds up to 4.3. This can be a problem in practice, to maintain fast response times, while considering the time required for transfer of information to and from the server. Unlike the model, the heuristic algorithm maintains a constant complexity, as proved by Figure 16 and Table 2, which is independent from the length or complexity of the considered instance, with processing times ranging from 0 to 16 ms and an average processing time of 2.32 ms. A constant computational complexity means that developing an online tool based on our algorithm would not be subject to significant variability of the response times to the users' queries.

Second, the number of clusters used by the model is up to 6 per instance, while the one used by our heuristic algorithm is up to 5 per instance. Even though this is a slight difference, to properly assess the quality of the provided solutions, we should consider which clusters are used to print out the descriptive sentences as well. Indeed, looking at Figure 18 on the distribution of the number of clusters, in correspondence with the solutions that exceed the three clusters used by the model, there are instances populated by single scattered days, for which the solver tends to extract whatever cluster covers the corresponding single day, regardless if this is the most proper one, by

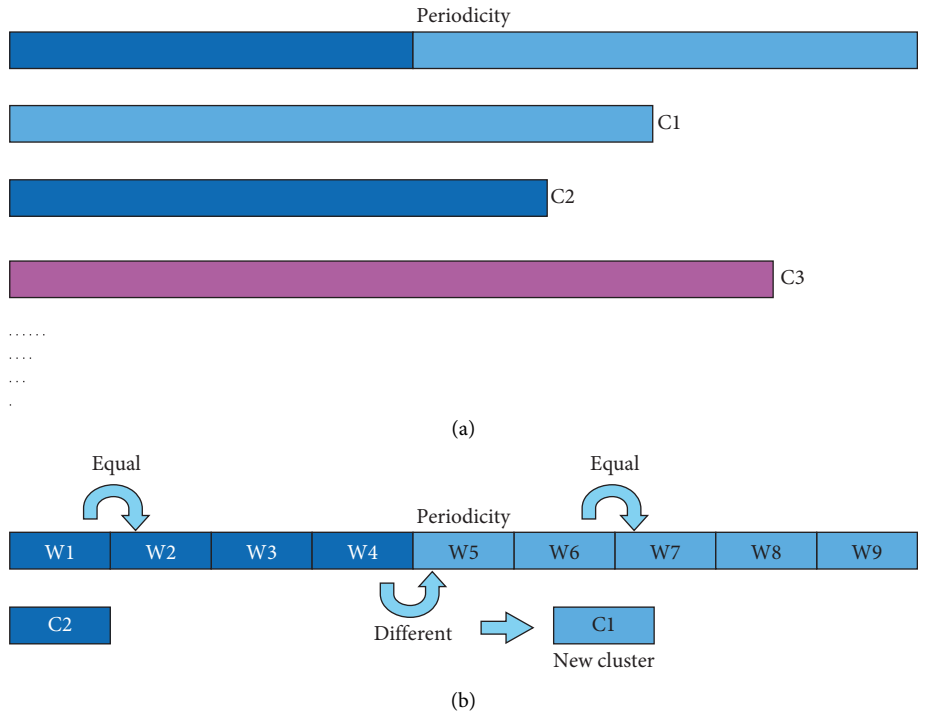

Figure 14: Methodological differences between the model (a) and the heuristic algorithm (b).

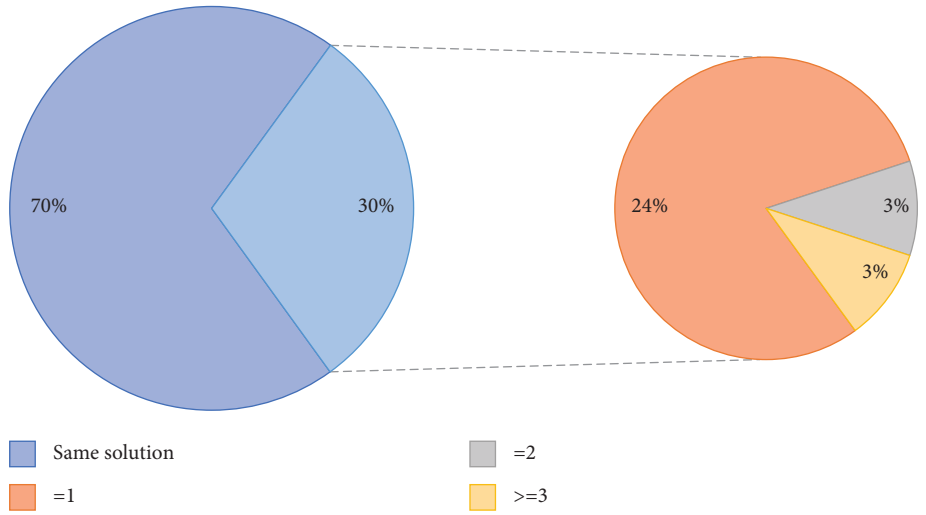

Figure 15: Percentage of different solutions used by each method, along with the number of different clusters used.

modulating the endpoints $Y^I_{c,k}$ and $Y^F_{c,k}$. This leads to a lowering of solution-quality and an increase in the likelihood of confusing the users. Differently, the algorithm works very well with single days, since this does not require the use of any threshold value, which could insert many exceptions, to minimize the studied objective function. Furthermore, our algorithm returns the most effective descriptive cluster that could be associated with these subperiods, keeping away from the modulation of the start and final dates of whatever cluster covers them. For example, if the single day on which the service is offered is Wednesday, the algorithm will associate the cluster "Wednesday" with that week instead of using the cluster "Working days" and thus choosing the $Y^I_{c,k}$ and $Y^F_{c,k}$ values which correspond to that particular Wednesday.

When looking at the quality performance measurements in Table 2 and Figure 18, the heuristic algorithm makes use of 0.13 extra clusters compared to the model, which computes the optimal solution. Furthermore, even though the distribution of the instances with two clusters is higher in the solutions provided by the heuristic algorithm compared to the ones computed by the model, the algorithm avoids exceeding three clusters to solve these benchmark instances. This means that the algorithm behaves more consistently, avoiding rare events that lead to

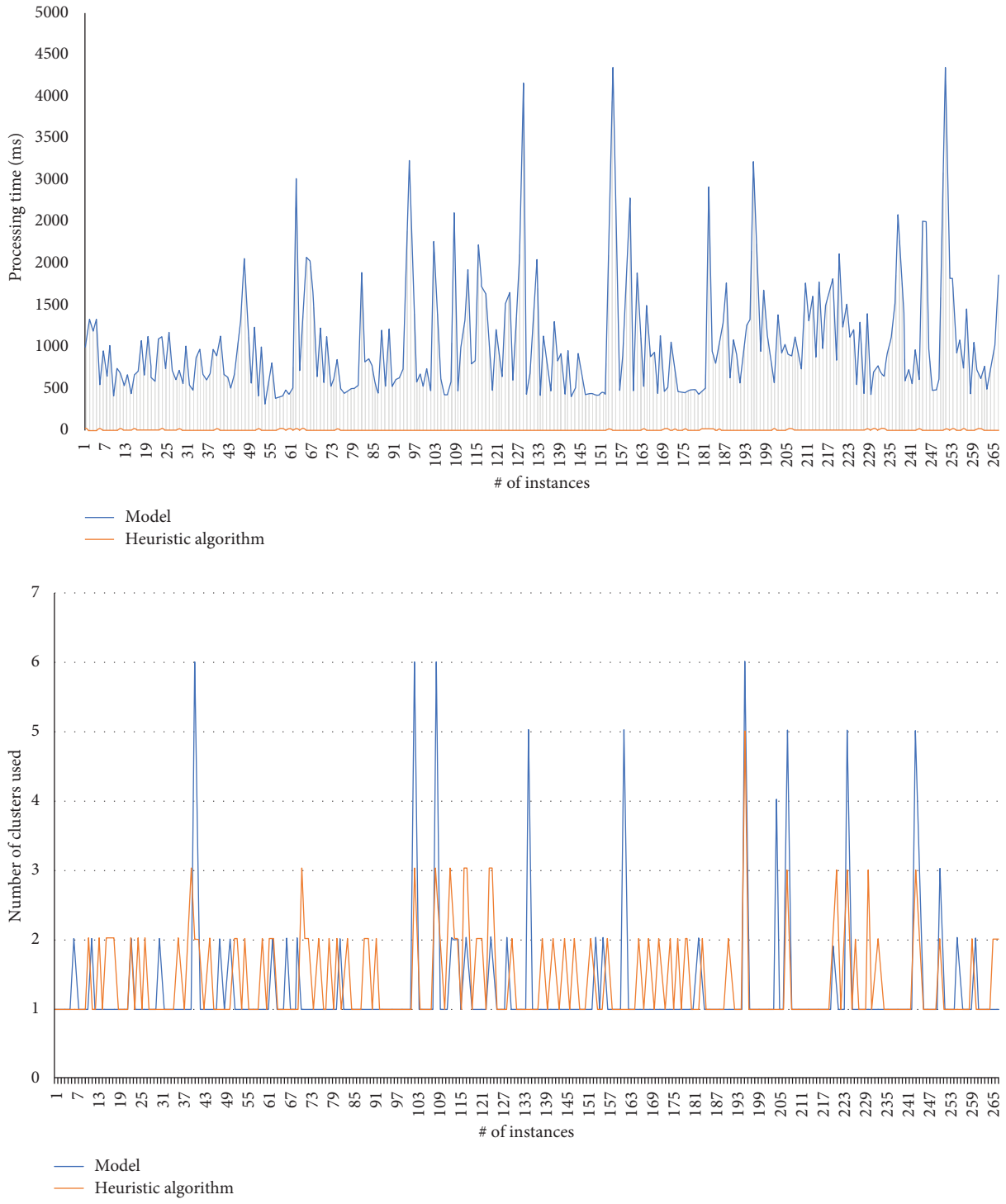

Figure 16: Running times and clusters used for the 264 instances.

important disservices, while sharing information with the users. Furthermore, the algorithm identifies more descriptive clusters and introduces new and more flexible functions (e.g., information on the extra days) without increasing the overall processing time. This additional flexibility reduces the number of clusters associated with each solution and improves the resulting solution quality. Differently, in the modeling approach, these additional functions might lead to new constraints and variables, thus potentially increasing the computational complexity.

However, there are two main limitations that we have met: the "*Every day*" cluster is worked more effectively by the model, while the algorithm tends to unpack the beginning and end times of each period. For example, if the solution generated by the model is "*The service is provided*

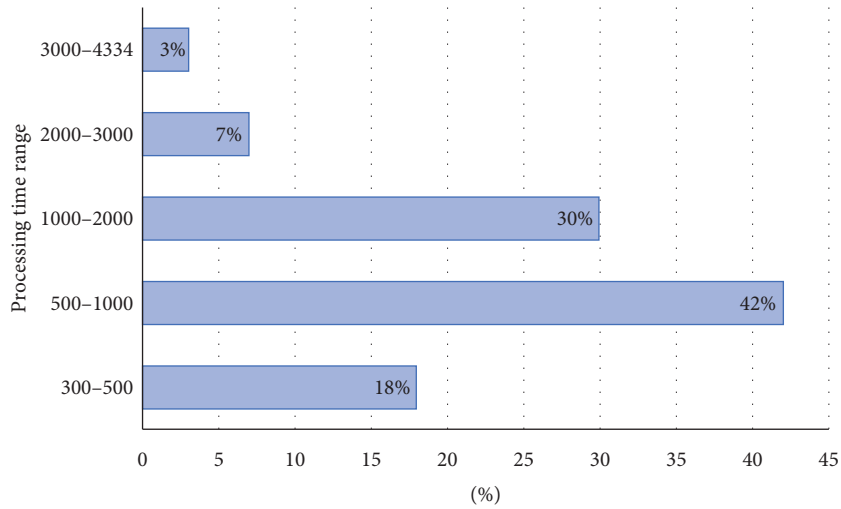

Figure 17: Distribution of processing times for the model.

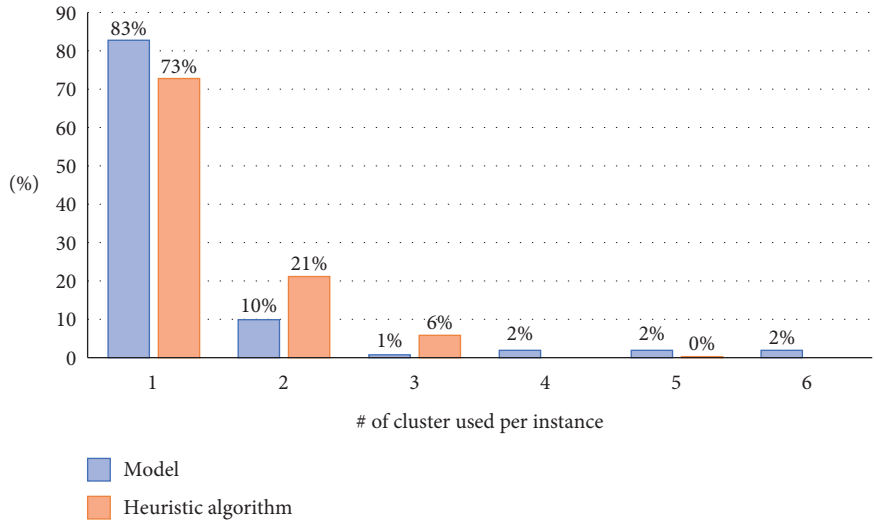

Figure 18: Percentage distribution of the number of clusters used by the model.

every day from x/y to z/w," the heuristic algorithm, which works week by week, could break it down into two different clusters; the second limitation is actually linked to the characteristics requested by the train operating company itself. In fact, while the model can work on a periodicity over the years, the algorithm can only consider one year at a time. This is because the train operating company requests a tool that would work from six months to six months.

5. Conclusions

In this paper, we developed a fast and flexible alternative method to the approach of state-of-the-art for train calendars' textual generation that mainly allows us to maintain a constant computation time, return good quality solutions, and introduce new functions to enhance the effectiveness of the sentences to be printed out. Theoretical and practical contributions lie in a new ILP model along with a fast heuristic algorithm for solving the online train calendar generation problem. The ILP model has three main differences with Amorosi et al.'s [15] model: our model considers a new data $c^*_{d,c,k}$ with the aim of avoiding preprocessing functions which were used by Amorosi et al. [15] to filter input data; the parameter l in their model, employed to define a bond to the length on the subsentences extracted, is not considered in our model along with the $|C| * |K|$ constraints involving this parameter; our model is embedded in Python, which allows the integration both of a preprocessing phase of the input data and a postprocessing phase of output data into a single environment, thus improving the online interaction with the users and the speed of generating train calendar descriptive sentences that were performed by an external script in Amorosi et al. [15].

Regarding the algorithmic contribution, the heuristic proposed in our paper exploits a "Divide and Conquer" logic to tame the whole periodicity through the generation of week-by-week solutions. From the computational

experiments, our algorithm presents, on average, a strong (equal to 99.8%) processing time reduction compared to the modeling approach, while remaining below 1.32 subsentences per periodicity. Due to its fast response time and its ability to compute good quality solutions, the proposed algorithm can be considered as the best choice to develop a valid online tool for train calendar generation.

Is it possible to go further in this direction? Of course, there are some open issues. The first thing that we could improve is the processing of the "Every day" cluster, giving the algorithm the possibility to predict that. What we mean is to allow the algorithm to intelligently understand if a subperiod that can be effectively described by the "Every day" cluster has begun. This should further reduce the average number of clusters used.

Another important way to improve the current results could be to reconsider the initial assumptions on what we consider a readable and intelligent sentence based on a more realistic perceived quality by the user, for example, through the perceived service quality model proposed by [21]. What if the minimum number of clusters is not a quality parameter for the sentences? Before we began developing the current algorithm, we intended to start a survey campaign among university commuter students, but this was not fully possible due to the restrictions imposed by the covid-19 situation. This could have allowed us to confirm our logical assumptions or reshape them based on new considerations from the perceived quality of service.

Moreover, the flexibility demonstrated by the proposed algorithm enables it to be adopted to develop online tools for event calendars' description in other public transport contexts with different problem specifications and descriptive sentences to be printed out. From the ground to air transport, the procedural method expressed by the main body of the algorithm could be populated by additional features, and be addressed not only to external users but also to the transfer of information for internal staff, as in the case of freight transport sector.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

The authors would like to thank Dr. Gianluca Giacco (Trenitalia senior manager) and Dr. Lavinia Amorosi (Sapienza University researcher) for their feedback on the problem and for generously sharing railway data.

References

[1] Strategic Master Plan, *Shift2Rail Joint Undertaking*, Strategic Master Plan, Brussels, Belgium, 2015.

[2] J. Yin, A. D'Ariano, Y. Wang, L. Yang, and T. Tang, "Timetable coordination in a rail transit network with time-dependent passenger demand," *European Journal of Operational Research*, 2021.

[3] M. Schachtebeck and A. Schöbel, "To wait or not to wait—and who goes first? delay management with priority decisions," *Transportation Science*, vol. 44, no. 3, pp. 291–428, 2010.

[4] F. Corman, "Interactions and equilibrium between rescheduling train traffic and routing passengers in microscopic delay management: a game theoretical study," *Transportation Science*, vol. 54, no. 3, pp. 565–853, 2020.

[5] X. Hong, L. Meng, A. D'Ariano et al., "Integrated optimization of capacitated train rescheduling and passenger reassignment under disruptions," *Transportation Research Part C: Emerging Technologies*, vol. 125, p. 103025, 2021.

[6] A. D'Ariano, F. Corman, T. Fujiyama, L. Meng, and P. Pellegrini, "Simulation and optimization for railway operations management," *Journal of Advanced Transportation*, vol. 2018, Article ID 4896748, 3 pages, 2018.

[7] W. Li, Q. Luo, Q. Cai, and X. Zhang, "Using smart card data trimmed by train schedule to analyze metro passenger route choice with synchronous clustering," *Journal of Advanced Transportation*, vol. 201813 pages, 2018.

[8] L. D'Acierno, M. Botte, M. Gallo, and B. Montella, "Defining reserve times for metro systems: an analytical approach," *Journal of Advanced Transportation*, vol. 201815 pages, 2018.

[9] B. Toader, A. Moawad, T. Hartmann, and F. Viti, "A data-driven scalable method for profiling and dynamic analysis of shared mobility solutions," *Journal of Advanced Transportation*, vol. 202115 pages, 2021.

[10] M. Botte, C. Di Salvo, A. Placido, B. Montella, and L. D'Acierno, "A neighbourhood search algorithm for determining optimal intervention strategies in the case of metro system failures," *International Journal of Transport Development and Integration*, vol. 1, no. 1, pp. 63–73, 2017.

[11] M. Samà, A. D'Ariano, F. Corman, and D. Pacciarelli, "A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations," *Computers and Operations ResearchComputers & Operations Research*, vol. 78, pp. 480–499, 2017.

[12] V. Cacchiani and P. Toth, "Nominal and robust train timetabling problems," *European Journal of Operational Research*, vol. 219, no. 3, pp. 727–737, 2012.

[13] Y. Wang, Y. Gao, X. Yu, I. A. Hansen, and J. Miao, "Optimization models for high-speed train unit routing problems," *Computers and Industrial EngineeringComputers & Industrial Engineering*, vol. 127, pp. 1273–1281, 2019.

[14] F. Corman, A. D'Ariano, A. D. Marra, D. Pacciarelli, and M. Samà, "Integrating train scheduling and delay management in real-time railway traffic control," *Transportation Research Part E: Logistics and Transportation Review*, vol. 105, pp. 213–239, 2017.

[15] L. Amorosi, P. Dell'Olmo, and G. L. Giacco, "Mathematical models for on-line train calendars generation," *Computers and Operations ResearchComputers & Operations Research*, vol. 102, pp. 1–9, 2019.

[16] H. Bachraty, E. Krsak, and M. Tavac, "Algorithm for generating text descriptions of bit calendars," *Communications*, vol. 11, no. 3, pp. 54–62, 2009.

[17] N. Kumar and A. Mishra, "A multi-objective and dictionary-based checking for efficient rescheduling trains," *Alexandria Engineering Journal*, vol. 60, no. 3, pp. 3233–3241, 2021.

[18] N. Dethlefs, A. Schoene, and H. Cuayáhuitl, "A divide-and-conquer approach to neural natural language generation from structured data," *Neurocomputing*, vol. 433, pp. 300–309, 2021.

[19] E. I. A. E. Lester, "Work breakdown structures," *Project Management, Planning and Control*, pp. 53–59, Butterworth-Heinemann, Oxford, UK, 2017.

[20] C. Wootton, "Compiled and interpreted languages," in *Developing Quality Metadata: Building Innovative Tools and Workflow Solutions*, pp. 299–308, CRC Press, Boca Raton, FL, USA, 2007.

[21] I. G. M. Y. Bakti, T. Rakhmawati, S. Sumaedi, and S. Damayanti, "Railway commuter line passengers' perceived service quality: hedonic and utilitarian framework," *Transportation Research Procedia*, vol. 48, pp. 207–217, 2020.