A Mathematical Programming Approach for Calendar Generation

Lavinia Amorosi[a,*], Paolo Dell'Olmo[a], Giovanni Luca Giacco[b]

[a] *Dipartimento di Scienze Statistiche, University of Rome Sapienza, Italy*
[b] *Direzione Pianificazione Industriale, Trenitalia, Rome, Italy*

Abstract

This paper describes a new method for automatically generating a text to customers in a concise and clear way with a calendar represented by a boolean vector as its input. Our attention is focused on problems arising from calendars in the transportation field, in particular railway services. The aim is to verify if it is possible to optimize the quality of outcomes in terms of intelligibility with a mathematical programming approach. Two mathematical formulations for this challenging task have been proposed. The first one consists in using a specific set covering model with embedded generation of sets out of the ground set (integrated model). The second one combines a set covering model with a parallel vector generation algorithm. A Mixed Integer Programming solver is adopted to solve the two models, while a parallel algorithm for the vector generation problem is designed in the second approach. Both models have been tested on several real railway timetables, always obtaining optimal solutions which compare favorably with those produced using current practices. Moreover, an enhanced formulation for the integrated model with better performances is presented. Computational times show that both approaches are applicable in practical contexts.

Keywords: Mathematical Models, Optimization, Transportation Services.

*Corresponding author
Email addresses: lavinia.amorosi@uniroma1.it (Lavinia Amorosi), paolo.dellolmo@uniroma1.it (Paolo Dell'Olmo), g.giacco@trenitalia.it (Giovanni Luca Giacco)

1. Introduction

In many enterprise operational environments ICT systems adopt vectorial representations of calendars (i.e. boolean vectors where entries mean availability (1) or non availability (0) of a service in the day corresponding to a specific index of the vector). On the other hand, people (in particular customers) need a textual description which is clear, simple and easy to understand and remember, without redundancies. For this reason, railway undertakings and infrastructure managers are making a big effort to improve the quality of their communication with commuters, employees and travelers, especially in the frequent cases of path modifications implying scattered calendars and even worse descriptions. Textual calendars currently appear in various outputs such as websites, mobile devices, timetable boards, train transport diagrams and books. Therefore, producing a textual representation starting from a boolean vector is a real business priority. Indeed, the current state of the art on automation of organizational processes has almost completely canceled the existence of "readable" calendars and their manual management by the operators. Yet, there is the evident necessity of representing, in particular to the customers, a good quality description of the calendars related to the service offer as this is considered a substantial factor in the perception of the service quality. Moreover, it is also desirable to present a timetable in a compact way, showing how it usually operates or where and how it has changed. This complete format of communication is very often preferred to an individual day representation, used only in the case of particular journeys (leisure or business travel).

In general, a calendar can be textually represented in a variety of ways and different formats, but we wish to achieve the goal of generating the "best" representation by employing operations research methods. Despite its practical relevance, as this specific problem is relatively new, it has been dealt with in just one paper (see [1]) where an heuristic algorithm for producing readable text has been proposed. To the best of our knowledge, the problem has never been dealt with using a mathematical programming approach. For these reasons,

we propose two mathematical models in order to get the "optimal" quality of outcomes in terms of intelligibility. The first one adopts a customized formulation of a set covering model (see [2] for general set covering formulation) with embedded and constrained generation of sub-sets (integrated model), while the second one consists in combining a set covering formulation with a parallel vector generation algorithm. The computational time of both models required to reach the optimal solutions is, to our knowledge, compatible with the practical requirements. A proof of concept for this approach has been presented in [3], while in this paper, we describe the problem in details, give a formal assessment of the hypothesis underling the mathematical models, explain, by means of a complete example, how the sentences are created from the model solutions, produce an extensive computational analysis on a large set of difficult instances and perform also sensitivity analysis on some resolution parameters. Moreover, we introduce also a third model improving significantly the performance of the integrated one and compare the three models on all the problem instances.

The rest of the paper is organized as follows. Section 2 describes the problem and introduces the proposed methodology. Section 3 contains notation and the first mathematical formulation, while in Section 4 the second model and the external parallel vector generation algorithm are illustrated. A detailed example of how a readable text can be generated from the set covering solution is described in Section 5. Solution methods and the first computational results are reported in Section 6. Finally, in Section 7 we give an enhancement of the first model which improves significantly the performances of this approach and makes it competitive in several instances with the second model characterized by the external sub-vectors generation.

2. Problem Description and Methodology

Given a train calendar containing operating and non-operating days (see the example in Figure 1) we want to generate a text capable of describing, in a concise manner, when the train service will be provided.

Day/Month	November					December			
Monday	31	7	14	21	28	5	12	19	26
Tuesday	1	8	15	22	29	6	13	20	27
Wednesday	2	9	16	23	30	7	14	21	28
Thursday	3	10	17	24	1	8	15	22	29
Friday	4	11	18	25	2	9	16	23	30
Saturday	5	12	19	26	3	10	17	24	31
Sunday	6	13	20	27	4	11	18	25	1

Figure 1: Example of train calendar

The time window in this example starts in November and ends in December. The days when the service is provided are colored in green, those in which it is not in red. As said before within the ICT systems this calendar is represented by a binary vector in which the zeros correspond to the days when the service is not provided (in red) and the ones to the days when the service is provided (in green), as shown in Figure 2.

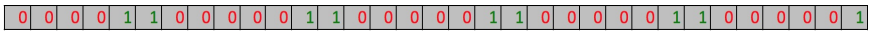

Figure 2: The binary vector corresponding to the example

The binary vector associated with the time window under consideration is called *periodicity*. From the periodicity we want to obtain the clearest and most concise text description of the corresponding calendar. A possible textual description for this example is the one below:

"The service is active on weekends from 5 November to 3 December".

In general, one can adopt different ways of representing the given calendar. In this case and throughout the paper, the positive way is used (based on operating days), but one could also use a negative way (based on non-operating days) or a mixed one (combination of positive and negative ways).

The solution idea, underlying the presented method, is to decompose the periodicity on the basis of 46 standard binary vectors, named *clusters*, each one of them referring to a particular sub-set of the calendar (see the 46 clusters represented in Table 5 in the Appendix). After chosing a family of sub-vectors resulting from the standard binary vectors, one has to find the best combination

out of this family to describe the given train calendar.

In more detail, this family can be obtained by decomposing the 46 clusters in subsequences having a minimum length and a given percentage of ones. Each of these sub-vectors can be associated to:

- a "quality", describing the type of extracted vector (Mondays, Tuesdays, holidays, etc.);
- a start date (date k if the first vector element is in position k);
- a final date (date h if the last vector element is in position h).

A detailed example describing how the text is obtained from a sub-vector solution is given in Section 5.

3. The First Approach

Following the reasoning of the previous section, we can formalize the problem of finding the minimum set of sub-vectors covering (i.e. representing) the whole periodicity, taking into account the necessity of using sub-vectors with "nice" characteristics (like the percentage of ones over a given threshold and with length larger than a given minimum). Since it is possible that a standard cluster (or its sub-vector) is used in a solution several times (e.g. the service is provided on Sundays from 1/1 to 1/31, from 3/15 to 7/25 and from 11/1 to 12/10, where the vector Sunday is used three times), we have to consider as an input data a predetermined number of copies of each standard cluster. This set of copies is related to the maximum number of times that it could be required to enter into a solution. The performed experiments show that the copies used in a solution are less than 10. For these reasons it was decided to copy each cluster 10 times.

3.1. Notation and Mathematical Formulation

We consider the following input data:

D = vector of dates associated with the periodicity;

C = set of clusters;

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

$m_{c,k}$ = size of the copy $C_{c,k}$;

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

l = minimum length that the sub-vectors must have to be feasible;

Tot = cardinality of the periodicity;

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

M = big integer.

We define the following integer variables:

$Y^I_{c,k}$ integer indicating the starting position of the sub-vector extracted from the k-th copy $C_{c,k}$ of the cluster c;

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

To ensure that every date $d \in D$ is covered, we need to know whether d is contained a given cluster, included in the indices proposed by the previous variables $Y^I_{c,k}$ and $Y^F_{c,k}$. We then define three types of binary variables:

$$KP_{d,c,k} = \begin{cases} 1 & \text{if } Y^I_{c,k} \leq d^*_{d,c,k} \\ 0 & \text{otherwise} \end{cases}$$

This variable represents whether the position index of the date d matches, or it is successive to the start index of the k-th copy $C_{c,k}$ of the cluster c.

$$KN_{d,c,k} = \begin{cases} 1 & \text{if } Y^F_{c,k} \geq d^*_{d,c,k} \\ 0 & \text{otherwise} \end{cases}$$

The previous variable represents whether the position index of the date d matches, or precedes the final index of the k-th copy $C_{c,k}$ of the cluster c.

$$K_{d,c,k} = \begin{cases} 1 & \text{if the date } d \text{ is covered by the } k\text{-th copy } C_{c,k} \text{ of the cluster } c \\ 0 & \text{otherwise} \end{cases}$$

By means of this variable we indicate if the position index of the date d is

contained in the sub-vector extracted from the k-th copy $C_{c,k}$ of the cluster c and showing a start position $Y^I_{c,k}$ and final position $Y^F_{c,k}$. Finally, we introduce the binary variable allowing us to specify which copies and which clusters are chosen as solutions:

$$x_{c,k} = \begin{cases} 1 & \text{if the } k\text{-th copy } C_{c,k} \text{ of the cluster } c \text{ is chosen in the solution} \\ 0 & \text{otherwise} \end{cases}$$

Through these decisional variables we can formulate the following integer linear programming model:

$$\min \sum_{c \in C,k} x_{c,k} \tag{1}$$

subject to

$$\sum_{d \in D} K_{d,c,k} \geq \alpha * (x_{c,k} + Y^F_{c,k} - Y^I_{c,k}) \quad \forall c \in C \quad \forall k \tag{2}$$

$$Y^F_{c,k} \leq m_{c,k} * x_{c,k} \quad \forall c \in C \quad \forall k \tag{3}$$

$$Y^I_{c,k} \leq Y^F_{c,k} - (l-1) * x_{c,k} \quad \forall c \in C \quad \forall k \tag{4}$$

$$Y^I_{c,k} \geq x_{c,k} \quad \forall c \in C \quad \forall k \tag{5}$$

$$KP_{d,c,k} \geq \frac{(d^*_{d,c,k} - Y^I_{c,k}) + 1}{(Tot + 1)} \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{6}$$

$$KP_{d,c,k} \leq 1 + \frac{(d^*_{d,c,k} - Y^I_{c,k})}{(Tot + 1)} \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{7}$$

$$KN_{d,c,k} \geq \frac{(Y^F_{c,k} - d^*_{d,c,k}) + 1}{(Tot + 1)} \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{8}$$

$$KN_{d,c,k} \leq 1 + \frac{(Y^F_{c,k} - d^*_{d,c,k})}{(Tot + 1)} \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{9}$$

$$K_{d,c,k} \geq 1 - (1 - KP_{d,c,k}) * M - (1 - KN_{d,c,k}) * M$$
$$\forall d \in D \quad \forall c \in C \quad \forall k \tag{10}$$

$$K_{d,c,k} \leq KP_{d,c,k} \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{11}$$

$$K_{d,c,k} \leq KN_{d,c,k} \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{12}$$

$$\sum_{c \in C,k} K_{d,c,k} \geq 1 \quad \forall d \in D \tag{13}$$

The objective function minimizes the number of clusters required to describe the total periodicity. Constraint (2) ensures that the sub-vectors extracted have at least the predetermined percentage of ones represented by α. Constraints (3), (4) and (5) ensure that each sub-vector extracted from a cluster has the start and end indices included in the cluster's size and that its length is at least equal to the fixed minimum length l. Constraints (6) and (7) guarantee that the variable $KP_{d,c,k}$ assumes the value 1 if the position index of the date d matches, or it is successive to the start index of the sub-vector extracted from the k-th copy $C_{c,k}$ of the cluster c. Constraints (8) and (9) ensure that the variable $KN_{d,c,k}$ takes the value 1 if the position index of the date d matches, or precedes the final index of the sub-vector extracted from the k-th copy $C_{c,k}$ of the cluster c. Finally, constraints (10), (11) and (12) guarantee that the variable $K_{d,c,k}$ takes the value 1 if the corresponding $KP_{d,c,k}$ and $KN_{d,c,k}$ are both equal to 1. Consequently, if the variable assumes value 1, the index position of the date d belongs to the sub-vector extracted from the k-th copy $C_{c,k}$ of the cluster c, which has as start index $Y^I_{c,k}$ and as final index $Y^F_{c,k}$. Constraint (13) ensures that any given operating date is covered by at least one sub-vector of a cluster containing that date.

4. The Second Approach

As previously mentioned, the second approach proposed in this paper consists in using a set covering model with a parallel preprocessing algorithm for generating all feasible sub-vectors.

4.1. Notation and Mathematical Formulation

We consider the following sets:

D = set of dates associated with the periodicity;

O = set of operating days (in which the service is provided);

S = set of the feasible sub-vectors generated with the parallel algorithm from the periodicity.

Let us define a matrix whose entry $t_{s,d} \quad \forall s \in S \quad \forall d \in D$, is 1 if the date d is included in the set s and 0 otherwise. Now, we can introduce the following binary variable:

$$x_s = \begin{cases} 1 & \text{if the set } s \text{ is chosen in the solution} \\ 0 & \text{otherwise} \end{cases}$$

The problem formulation is given by the following set covering model:

$$\min \sum_{s \in S} x_s \tag{14}$$

subject to

$$\sum_{s \in S} t_{s,d} * x_s \geq 1 \quad \forall d \in O \tag{15}$$

$$x_s \in \{0, 1\} \quad \forall s \in S \tag{16}$$

By solving the above model we determine the minimum number of sub-vectors to describe the periodicity. Constraint (15) ensures that each operating day is included in at least one of the vectors that belong to the solution. In this approach a parallel vector generation algorithm creates all possible feasible sets.

More precisely, from each standard cluster, all the sub-vectors which satisfy the following two properties are extracted:

- length at least equal to l;
- percentage of ones greater or equal to α.

The extraction takes place through a parallel algorithm which is composed of two simple steps: the first one leads to the building of a vector in which each entry represents the cumulative number of ones found in the original vector from the first to the current index; the second one considers only sub-vectors of the cumulative vector with length at least equal to l. On the vectors obtained in this way the condition of the percentage α must be verified. The final output is a list of pairs of integers representing the first and last index required to identify each feasible sub-vector. The sub-vectors are then used as variables of the set covering model. By requiring that the sub-vectors length is at least equal to a given length l, we guarantee that the periodicity is described through sub-vectors without being excessively fragmented. By requiring also that a sub-vector is feasible only if it contains a certain percentage of ones equal to α, we ensure that the periodicity is reconstructed through sets that can be expressed in a simple and clear way, with few exceptions.

5. Illustrative example

In this section it is shown how the proposed approaches work on a complete practical example. For this purpose we present the calendar in Figure 3.

Day/Month	December				January				February				March				April				May					June			
Monday	7	14	21	28	4	11	18	25	1	8	15	22	29	7	14	21	28	4	11	18	25	2	9	16	23	30	6	13	20	27
Tuesday	8	15	22	29	5	12	19	26	2	9	16	23	1	8	15	22	29	5	12	19	26	3	10	17	24	31	7	14	21	28
Wednesday	9	16	23	30	6	13	20	27	3	10	17	24	2	9	16	23	30	6	13	20	27	4	11	18	25	1	8	15	22	29
Thursday	10	17	24	31	7	14	21	28	4	11	18	25	3	10	17	24	31	7	14	21	28	5	12	19	26	2	9	16	23	30
Friday	11	18	25	1	8	15	22	29	5	12	19	26	4	11	18	25	1	8	15	22	29	6	13	20	27	3	10	17	24	1
Saturday	12	19	26	2	9	16	23	30	6	13	20	27	5	12	19	26	2	9	16	23	30	7	14	21	28	4	11	18	25	2
Sunday	13	20	27	3	10	17	24	31	7	14	21	28	6	13	20	27	3	10	17	24	1	8	15	22	29	5	12	19	26	3

In accordance with the previous notation, the service is provided for the days colored in green and is not provided for those colored in red. The periodicity is

Day/Month	July					August					September					October					November					December				
Monday	27	4	11	18	25	1	8	15	22	29	5	12	19	26		3	10	17	24	31	7	14	21	28		5	12	19	26	
Tuesday	28	5	12	19	26	2	9	16	23	30	6	13	20	27		4	11	18	25	1	8	15	22	29		6	13	20	27	
Wednesday	29	6	13	20	27	3	10	17	24	31	7	14	21	28		5	12	19	26	2	9	16	23	30		7	14	21	28	
Thursday	30	7	14	21	28	4	11	18	25	1	8	15	22	29		6	13	20	27	3	10	17	24	1		8	15	22	29	
Friday	1	8	15	22	29	5	12	19	26	2	9	16	23	30		7	14	21	28	4	11	18	25	2		9	16	23	30	
Saturday	2	9	16	23	30	6	13	20	27	3	10	17	24	1		8	15	22	29	5	12	19	26	3		10	17	24	31	
Sunday	3	10	17	24	31	7	14	21	28	4	11	18	25	2		9	16	23	30	6	13	20	27	4		11	18	25	1	

Figure 3: Illustrative example

from 13 December 2015 to 10 December 2016 and its dimension is of 364 entries. As we can observe, the service is provided in the holidays with one exception (the date 3/28/2016). Through the current system used for calendar generation, because of this exception, the corresponding string in natural language would be as follows:

```
"The service is provided on Sundays
and on the dates 12/25/2015, 12/26/2015,
1/1/2016, 1/6/2016, 4/25/2016, 6/2/2016,
8/15/2016, 11/1/2016 and 12/8/2016"
```

It is evident that this string is not optimal and could be expressed in a more clear and concise way. For this purpose, let us apply one of the two previous models. The solution obtained is as follows:

```
Optimal objective value = 1
Optimal variable values:
x[ <44 1 362>]=1
107
```

It can be seen that the whole periodicity is covered by only one sub-vector with one exception. That is, a sub-vector extracted from cluster 44 (holidays, including Sundays) which has a start index corresponding to the first date of the

periodicity and a final index corresponding to the 362nd date of the periodicity. The exception is represented by the integer number 107 which, in this example, corresponds to the date 3/28/2016. Therefore, from this solution it is possible to form the sentence that describes in natural language the whole periodicity:

```
"The service is provided on holidays
from 12/13/2015 to 12/8/2016
with the exception of 3/28/2016"
```

As we can observe, the string built from the solution provided by the model is more concise and intelligible than the one produced by the current procedure.

6. Solution Methods and Computational Results

We solved the models by means of a Mixed Integer Programming solver, i.e., CPLEX. We recall that there is a major difference in the two formulations proposed, as in the second one we have to generate sub-vectors before launching the solver. Therefore, to solve the model presented in the second solution approach, a parallel vector generation algorithm has been designed to create all possible feasible sets. As the possible number of vectors is very large, we adopted a parallel solution method that turns out to be quite effective for large combinatorial optimization problems. The reader interested in parallel computation can refer, for instance, to the book of Bisseling [4]. In general, parallel computing implies the simultaneous use of more processes to solve a single computational problem. A problem is divided into discrete parts that can be solved concurrently through multiple processes. This method allows the process to overcome the limitations of memory and speeds up the computational time. We can distinguish two main different logics for developing parallel codes: the message-passing approach and the multithreading approach. In the *message-passing* approach, the processes cooperate by means of exchanging messages. In this case, each process has access to its own memory for reading and writing data. If a process needs to

access data present in the memory of another process or if more processes need to synchronize the execution of a set of instructions, it is necessary to exchange messages between the processes involved. An API (Application Programming Interface) that allows us to implement this parallel computation method is MPI (Message Passing Interface). This approach has some disadvantages. Each MPI process can only access its own local memory and the data to be shared must be exchanged with explicit inter-process communications (messages). The communications have a time cost and therefore for problems with a large amount of data, is more efficient to write parallel applications with shared memory. In this second approach, called *multithreading*, all processors may access the whole main memory. The process contains several concurrent execution flows (threads) and the instructions executed by a thread can access the process global memory (data) and the thread local stack (see the figure below).

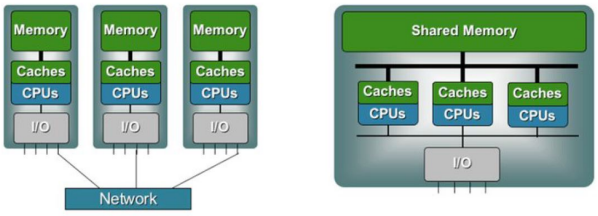

Figure 4: Shared memory architectures

In our case we chose the .NET environment preferring a multithreading approach compared to the MPI one. In such a way it is not necessary to transfer data as sharing memory among processes is more suitable. We set a number of processes equal to the number of standard vectors (46) and managed the control by means of a master procedure and the use of barriers to avoid asynchronous behavior.

The two approaches were tested on 261 instances related to different time windows. The length of the time windows (expressed in number of days of interest) is in the range [90, 365]. The minimum percentage of ones was set equal

to 80% and the minimum length of each sub-vector was fixed equal to 2. Many of these instances were chosen based on two criteria: the high segmentation of the corresponding periodicity and the difficulty of finding a concise way of translating the instances into natural language. All tests were performed on a Windows computer with 4 Intel i7 2.3 GHz and 8 GB of RAM. As far as the problem solver is concerned, CPLEX 12.4 has been chosen with a multithread strategy solution. The second approach, with external vectors generation, performed significantly better than the first one where the vectors generation is embedded in the formulation. For the second approach the average computational time over the 261 tests was 0.6 seconds with values ranging from 0.058 to 4.87 seconds, while for the first one we obtained over the same set of instances with 0.89 seconds as an average value of the computational time with values ranging from 0.075 to 10.6 seconds. The solution times are summarized in the Table 1 and in Figure 5. These include the preprocessing time, respectively cluster processing and model writing for the first approach, and all possible sub-vectors generation, through the parallel algorithm, as well as model writing for the second approach.

Approach	Min	Max	Average
First	75.0049 ms	10625.6706 ms	897.5859211 ms
Second	58.0041 ms	4875.1411 ms	611.3748065 ms

Table 1: Solution times of the two approaches

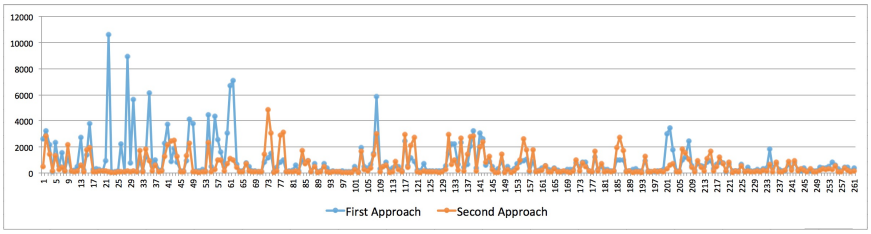

Figure 5: Solution times comparison between the two approaches

7. Enhanced First Model and Final Computational Results

In this section we propose an enhanced version of the first integrated model, which allows us to improve the efficiency of the first formulation. This enhancement is actually comparable, in terms of computational efficiency, with the second model. The main improvement is obtained by means of a reduction in the number of the binary variables.

7.1. Notation and Mathematical Formulation

We consider the following input data:

D = set of dates associated with the periodicity;

C = set of clusters;

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

$m_{c,k}$ = size of the copy $C_{c,k}$;

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

l = minimum length that the sub-vectors must have to be feasible;

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

Tot = cardinality of the periodicity;

M = big integer.

We define the following integer variables:

$Y^I_{c,k}$ integer indicating the starting position of the sub-vector extracted from the k-th copy $C_{c,k}$ of the cluster c;

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

To ensure that every date $d \in D$ is covered, we need to know whether d is contained in a given cluster, defined by the indices proposed by the previous variables $Y^I_{c,k}$ and $Y^F_{c,k}$. We then define the following binary variable:

$$K_{d,c,k} = \begin{cases} 1 & \text{if the date } d \text{ is covered by the } k\text{-th copy } C_{c,k} \text{ of the cluster } c \\ 0 & \text{otherwise} \end{cases}$$

Finally, we introduce the binary variable allowing us to specify which clusters are chosen in the solution:

$$x_{c,k} = \begin{cases} 1 & \text{if the } k\text{-th copy } C_{c,k} \text{ of the cluster } c \text{ is chosen in the solution} \\ 0 & \text{otherwise} \end{cases}$$

Through these decisional variables we can formulate the following integer linear programming model:

$$\min \sum_{c \in C,k} x_{c,k} \tag{17}$$

subject to

$$\sum_{d \in D} K_{d,c,k} \geq \alpha * (x_{c,k} + Y^F_{c,k} - Y^I_{c,k}) \quad \forall c \in C \quad \forall k \tag{18}$$

$$Y^F_{c,k} \geq d^*_{d,c,k} * K_{d,c,k} \quad \forall c \in C \quad \forall k \quad \forall d \in D \tag{19}$$

$$Y^I_{c,k} \leq (1 - M) * d^*_{d,c,k} * K_{d,c,k} + d^*_{d,c,k} * M \quad \forall c \in C \quad \forall k \quad \forall d \in D \tag{20}$$

$$Y^F_{c,k} - Y^I_{c,k} \geq (l - 1) * x_{c,k} \quad \forall c \in C \quad \forall k \tag{21}$$

$$\sum_{c \in C,k} K_{d,c,k} \geq 1 \quad \forall d \in D \tag{22}$$

$$x_{c,k} \geq \frac{\sum_{d \in D} K_{d,c,k}}{Tot} \quad \forall c \in C \quad \forall k \tag{23}$$

As we can see, in this new formulation only the binary variable $K_{d,c,k}$ is associated with each date $d \in D$. That is, from the original version of the model, the variables $KP_{d,c,k}$ and $KN_{d,c,k}$ have been eliminated. As a consequence, also part of the constraints were changed. Constraint (18) ensures that the sub-vectors extracted have at least the predetermined percentage of ones represented

by α. Constraints (19) and (20) guarantee that if $K_{d,c,k}$ is equal to 1 (d is covered by a sub-vector extracted from the k-th copy $C_{c,k}$ of the cluster c), then the sub-vector extracted from $C_{c,k}$ has to contain the date d. Constraints (21) and (22), as in the original formulation, impose that the minimum length of each sub-vector is at least equal to l and that each date $d \in D$ is covered. Finally, constraint (23) ensures that if the variable $K_{d,c,k}$ is equal to 1 for at least one date $d \in D$, then the variable $x_{c,k}$ has to be equal to 1, that is the k-th copy $C_{c,k}$ of the cluster c has to be in the solution.

7.2. Final Computational Results

The enhanced model was tested on the same set of 261 instances used for the first computational results. In Table 2 and in Figure 6 we can see the solution times of this formulation, including preprocessing and the comparison with the previous version. As we can observe, the enhanced formulation of the first approach is faster on average than the original formulation. In particular, from the Figure 6 it is possible to see that, with the exception of the instance 22, the enhanched formulation tends to reduce the high time values of the original model. Subsequently, Table 3 and Figure 7 show the comparison between the enhanced formulation and the first and second approach.

Approach	Min	Max	Average
First	75.0049 ms	10625.6706 ms	897.5859211 ms
First enhanced	144.9924 ms	13473.503 ms	672.6719084 ms

Table 2: Solution times of the two models for the first approach

Approach	Min	Max	Average
First	75.0049 ms	10625.6706 ms	897.5859211 ms
First enhanced	144.9924 ms	13473.503 ms	672.6719084 ms
Second	58.0041 ms	4875.1411 ms	611.3748065 ms

Table 3: Solution times of the three models

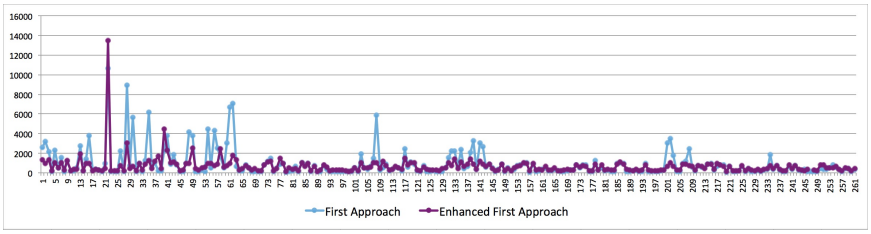

Figure 6: Solution times comparison between the two models for the first approach

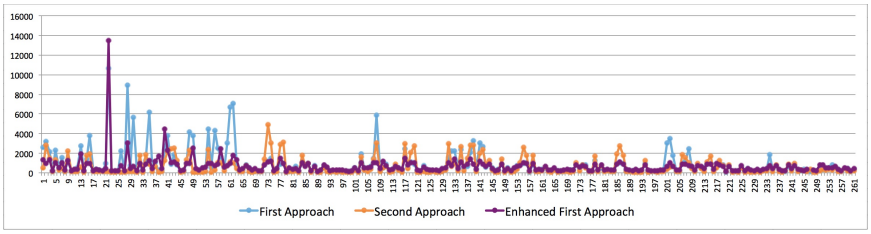

Figure 7: Solution times comparison between the three models

As previously mentioned, the solution time of the enhanced formulation is comparable on average with that of the second approach. From Figure 7 we can see that the enhanced formulation also shows the best behavior on the instances where the other two approaches show high time values, with the exception of the instance 22. To better compare the performances of the two approaches, we also analyzed their sensitivity to the parameter l (minimum length that the sub-vectors must have to be feasible). We considered five instances by varying the value of l in the set $\{1, ..., 5\}$ (the instance with minimum length equal to 2 is the instance number 22 of the original set on which the models were tested).

As we can see from Figure 8, the enhanced formulation is very sensitive to the values of the parameter l. The latter is more variable than the set covering of the second approach, to which it converges for values of $l \geq 3$. Moreover, on the same five instances we applied a different tuning in CPLEX, through which it is possible to further improve the performances of the enhanced formulation.

In particular, we can observe from Figure 9 that the latter formulation also

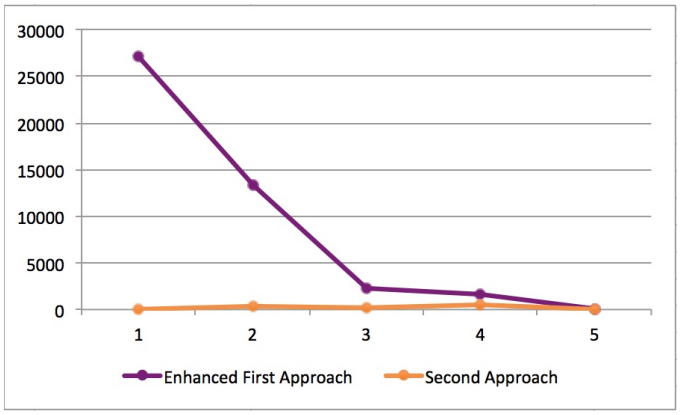

Figure 8: Solution times comparison with standard tuning

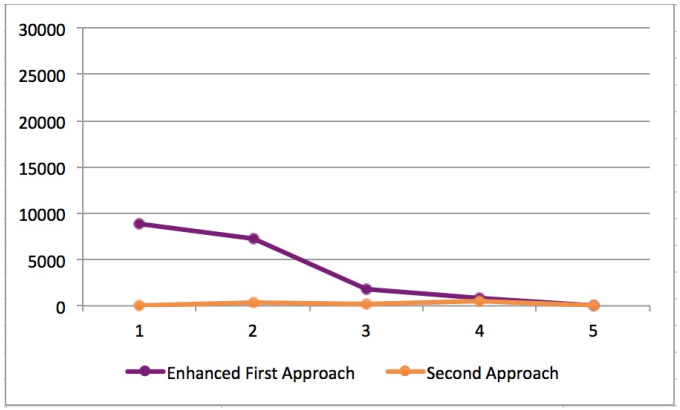

Figure 9: Solution times comparison with best tuning

performs significantly better with this tuning on the instance 22, for which the computational time is almost halved in comparison with the original tuning. For each instance in Table 4, the following are reported: value of the parameter l, the solution time of the enhanced first approach with the standard tuning, the solution time of the enhanced first approach with the best tuning, the solution time of the second approach (for which the best tuning does not have effect) and the value of the objective function.

From these results we can conclude that for instances in which the periodicity

l	1st En.Mod. Std Tuning	1st En.Mod. Best Tuning	2nd Mod.	Obj
1	27267.7106 ms	8949.0066 ms	249.0142 ms	3
2	13473.503 ms	7314.0028 ms	592.9644 ms	3
3	2304.106 ms	1874.0032 ms	437.0467 ms	3
4	1755.0683 ms	1012.007 ms	651.0411 ms	3
5	217.0073 ms	129.0039 ms	147.0078 ms	INF

Table 4: Sensitivity analysis

is fragmented, like many of the instances in the set used to test the models, the first approach, even in its enhanced version, performs worse than the standard set covering if the parameters are not set in a restrictive way. Otherwise, with more restrictive value of parameter l, the enhanced formulation is competitive with the standard set covering model and it performs even better on many instances on which the set covering shows high time values.

8. Conclusions

In this paper we presented a novel approach for train calendar textual generation. Two mathematical programming formulations have been introduced. The two approaches were tested on 261 different instances and the quality of solutions always compared favorably with those used in the real practice in all tests. The second approach (set covering model) performs better than the first one (integrated model), although in the enhanced version the integrated model has significantly improved performance and in a number of cases performs better than the set covering model. Successive researches will be dedicated to further improving the integrated model and applying this methodology to other text generation problems.

References

[1] M. Greiner, Algorithm for Generating Train Calendar Texts, *Promet-Traffic&Transportation* Vol. 25 (2013) 99–107.

[2] G. L. Nemhauser, L. A. Wolsey, Integer and Combinatorial Optimization, J. Wiley and Sons, 1999.

[3] L. Amorosi, P. Dell'Olmo, G. Giacco, A New Approach for Train Calendar Description Generation, in: Models and Technologies for Intelligent Transportation Systems (MT-ITS), Budapest, 3-5 June 2015, IEEE, 280–286, 2015.

[4] R. H. Bisseling, Parallel Scientific Computation: A Structured Approach using BSP and MPI (PSC), Oxford University Press, 2004.

Appendix

1. Monday (all Mondays in the periodicity)	24. Wednesday-Saturday
2. Tuesday	25. Thursday-Sunday
3. Wednesday	26. Friday-Monday
4. Thursday	27. Saturday-Tuesday
5. Friday	28. Sunday-Wednesday
6. Saturday	29. Monday-Friday (all sets of 5 days)
7. Sunday	30. Tuesday-Saturday
8. Monday-Tuesday (all sets of 2 days)	31. Wednesday-Sunday
9. Tuesday-Wednesday	32. Thursday-Monday
10. Wednesday-Thursday	33. Friday-Tuesday
11. Thursday-Friday	34. Saturday-Wednesday
12. Friday-Saturday	35. Sunday-Thursday
13. Saturday-Sunday (weekends)	36. Monday-Saturday (all sets of 6 days)
14. Sunday-Monday	37. Tuesday-Sunday
15. Monday-Wednesday (all set of 3 days)	38. Wednesday-Monday
16. Tuesday-Thursday	39. Thursday-Tuesday
17. Wednesday-Friday	40. Friday-Wednesday
18. Thursday-Saturday	41. Saturday-Thursday
19. Friday-Sunday	42. Sunday-Friday
20. Saturday-Monday	43. days before holidays, including Saturdays (not holidays)
21. Sunday-Tuesday	44. holidays, including Sundays
22. Monday-Thursday (all sets of 4 days)	45. working days
23. Tuesday-Friday	46. all days

Table 5: Clusters