Computers and Operations Research 102 (2019) 1–9

Contents lists available at ScienceDirect

Computers and Operations Research

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

Mathematical Models for On-Line Train Calendars 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

ARTICLE INFO

Article history:
Received 23 April 2018
Revised 18 September 2018
Accepted 18 September 2018
Available online 20 September 2018

Keywords:
Calendars
Mathematical Models
Transportation Services

ABSTRACT

In this paper we present a new model for train calendars textual generation, that is a method for automatically generating a text to customers in a concise and clear way with a service calendar represented by a boolean vector as its input. This problem arises in the transportation field, in particular in railway services. A new mathematical model which guarantees the optimality of solutions and good computational performances is described and tested on several real railway timetables, always obtaining optimal solutions. Moreover, it is extensively compared with existing models showing a significant reduction of computational times that makes it applicable in practical contexts.

1. Introduction

Railways services availability are seen by a traveler as the final result of a number of different and complex organization processes which have been widely studied in the literature, like timetable generation, train routing and scheduling and many others (see for example Arenas et al. (2018); Odijk et al. (2006); Samá et al. (2017) and Liebchen (2008)). In this paper we look a little step beyond all these processes focusing our attention on the problem of communicating a predetermined calendar to the passengers. Nowadays customers, always connected with smartphones and used to get real time information from web services, require to get information on availability of a transportation service in fast and "intelligent" format. In the case of railways, for instance, one would like to know the timetable of a given service in a very short format understandable and easy to remember. This format, say from the customer perspective, is not immediately available in the operational environment of railways. 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. Indeed, the ICT systems adopted in this context use a vectorial representations of calendars. Calendars for services are represented as boolean vectors whose entries mean availability (1) or non-availability (0) of the service in the day corresponding to the specific index of the vector. The actual practice suffers of a lack of intelligibility of calendars also increased by the high frequency of changes in train routes caused by the temporary restrictions of railways infrastructures. Indeed, the current systems are not able to produce a textual description for such a dynamic scenario. Moreover, the interface with customers is now realized by web services and apps working only on a single date, on which the user queries the database, while it is strongly required, especially by commuters, to buy traveling solutions not on a single date but over a given period.

As communication plays a great role in the perception of the quality of the offered service, railway companies are particular committed to improve the quality of their communication with all customers. As said, customers expect a textual description clear, simple, fluid, readable and without redundancies, hence there is the necessity to develop automatic tools to translate the boolean vector representation of a service into sentences having these characteristics of readability.

This problem belongs to Natural Language Generation (NLG) area whose approaches are known as the systematic way for producing human understandable natural language text based on non-textual data (see Gatt and Krahmer (2017) and Perera and Nand (2017) for recent surveys on the subject). According to Gatt and Krahmer (2017) and Perera and Nand (2017) NLG methods are generally organized in different phases; *Content Determination*, responsible for selecting information needed to be communicated through generated text; *Document Structuring*, that manages the structure of the information selected from content determination; *Lexicalization*, that operates on what words need to be included in the text; *Referring Expression Generation*, the process of determining how those words must be referred within generated text; *Aggregation* for structuring and ordering the sentence structures to build meaningful sentences; and, finally, *Linguistic and Structure Realization* accountable for producing final surface text and presenting it based on the requirements.

* Corresponding author.
E-mail address: paolo.dellolmo@uniroma1.it (P. Dell'Olmo).

https://doi.org/10.1016/j.cor.2018.09.009
0305-0548/

With respect to the mentioned phases our work faces a specific problem arising in the railways and makes reference mainly to the Aggregation and Linguistic Structure Realization phases (see Dalianis (1999) and Barzilay and Lapata (2006)), while, Content Determination and Document Structuring, as it will be seen in the paper, are basically embedded in our problem definition. Lexicalization and Referring Expression Generation are taken into account by means of the definition of clusters (see Section 2 and the Table 4 in the Appendix). In particular, we propose an exact optimization approach based on set covering formulations (see Nemhauser and Wolsey (1999) for general set covering formulation), which gets the maximum advantage from the specific problem structure, differently from works based on evolutionary algorithms like those in Hua and Mellish (2000) and Ervás and Gervás (2005) or set partioning Barzilay and Lapata (2006).

This specific problem has been faced in Greiner (2013) where an heuristic algorithm has been proposed and in Amorosi et al. (2015) where an exact approach is presented. In particular, in Amorosi et al. (2015) two mathematical models were described and preliminary computational results proved that both models deliver optimal solutions but the computational times vary a lot depending on the problem instance.

Although motivated by the railways sector, the technique presented in this paper can be adopted by any application for on-line textual description service availability also in other transportation means like, airplanes, ferry and long trip buses. Furthermore, it can be utilized to describe calendars of any event having a certain periodicity like sports events (e.g. soccer games of a team), or opening days of any entertainment and commercial activity.

The use of a mathematical programming approach is motivated by complex cases of service availability where a non-optimized sentence can be very long and not easy to understand and thus useless for the customer. In other contexts where NLG is applied to obtain text from data, like for instance in pollen forecast Turner et al. (2006) or in the weather forecast Goldberg et al. (1994); Sripada et al. (2014), the format of the output follows a standard template and its length is not so variable with the problem instance. Clearly, if a service is available always or on Sundays all year long, there is no need at all of optimization. However, this is not the case in practice and there is a large variety of possible distribution of service availability during the year depending both on customer demand, holidays, number of trains, limited resources and operational constraints. In this context, railways offer a large diversification of problem instances having several type of services and different classes of customer. The novel contributions of this paper can be summarized as follows:

- it solves a problem of practical importance in the transportation area and likely in other sectors where customers require a textual description of services availability or events over a certain time period;
- it gives a novel and exact method to textual calendar generation overcoming the difficulties of currently used heuristic algorithms;
- it proposes a mathematical programming approach which, for the third model, despite the NP-hardeness of solving exactly a set covering problem, has good computational performances also allowing an on-line practical use of the model and the implementation of a prototype software;
- it opens to further applications of Integer Linear Programming (ILP) to other Natural Language Generation problems where the length of the text has to be minimized;
- with respect to our previous conference contribution Amorosi et al. (2015), where models performances were too strongly dependent on the characteristics problem instances, it presents a new model which outperforms significantly the previous ones, presented in Amorosi et al. (2015), showing very limited execution time fluctuations over the whole set of 261 instances used to test it.

The rest of the paper is organized as follows. Section 2 describes formally the problem and introduces the proposed methodology. Sections 3 and 4 illustrate two mathematical formulations recalling and commenting the basic set covering model with external sub-vector generation and a slightly modified version of the integrated model proposed in Amorosi et al. (2015). A new integrated model is presented in Section 5. In Section 6 we perform a thorough computational comparison by testing the three models on 261 different instances. The new integrated model improves computational performances very significantly showing a limited computational time independent on the difficulty of the problem instance (maximum time values are reduced in percentage more than 85% with respect to the set covering model and more than 90% with respect to the integrated one) and it appears utilizable in practical cases.

2. Problem Description and Proposed Methodology

Starting from a train calendar (see the example in Fig 1 and Fig 2) we want to generate a concise and readable text describing when the train service is provided.

The time window in this example starts in November and ends in December. In green are represented the days when the service is provided and in red the days in which is not. 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 and the ones to the days when the service is provided, as shown in Fig 2.

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 5th November to 3rd December"; or, equivalently:

"The service is active on weekends from 11/5 to 12/3".

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 our method, is to decompose the periodicity on the basis of 46 typologies of binary vectors which refer to particular subsets of the calendar, named *Clusters*, that is, the sub-vector corresponding to all Mondays, all Tuesdays, ...all holidays, working days, and so on, currently adopted by the main Trenitalia's ICT systems (see Table 4 in the Appendix for the complete set of 46 typologies of Clusters). Once we constructed all the families of vectors derived from the periodicity as said, we have to search for the best covering of the periodicity on the basis of all sub-vectors that can be extracted from these Clusters with the following characteristics: at least a given length (number of days) and at least fixed percentage of ones (active days). Each of these sub-vectors can be associated to:

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

Our goal is to express in a text all the days in which the service is active, thus, as service availability is represented by the ones

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

Fig. 1. Example of train calendar

0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0	0	1

Fig. 2. The binary vector corresponding to the example

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

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				

Fig. 3. Illustrative example

(active days) in the periodicity, we must cover all the ones in the periodicity using sub-vectors corresponding to subsentences. These sub-vectors must have at least a given percentage of ones and a given length such to avoid too many sub-vectors in the solution which would lead to too many subsentences in the text. If a selected sub-vector has some zeros, these are reported in the corresponding sentence as exceptions. A detailed example describing how the text is obtained from a sub-vector solution is given next.

2.1. Illustrative example

In this section it is shown how the proposed methodology works on a complete practical example. For this purpose we present the calendar in Fig 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 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 clear that this string is not concise, difficult to understand and to remember, and it could be expressed in a more compact way. Following the proposed approach we build for this example all the clusters from the periodicity. In Figs 4 and 5 two of them are represented, that is cluster 29 corresponding to Monday-Friday (all sets of 5 days) and cluster 44 identifying holidays including Sundays. Adopting a set covering approach, sub-vectors of all clusters are selected to cover the whole periodicity. The minimum number of sub-vectors covering the periodicity will permit to build the most concise sentence expressing the days in which the service is available. Note that if this problem is solved optimally no shorter sequence can exist. In this example the whole periodicity is covered by only one sub-vector with one exception.

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

This sub-vector is extracted from cluster 44 (see Fig 5) which has the start index (1) corresponding to the first date of the periodicity and the final index corresponding to the 362nd date of the periodicity. The exception (with index 107) is represented by the date 3/28/2016 (this is the Monday after Easter that was holiday in Italy in 2016). Therefore, from this solution it is possible to con-

2	3	4	5	6	9	10	11	12	13	16	17	18	19	20	23	24	25	26	27	30	31	32	33	34	37	38	39	40	41	44	45	46	47	48
0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

51	52	53	54	55	58	59	60	61	62	65	66	67	68	69	72	73	74	75	76	79	80	81	82	83	86	87	88	89	90	93	94	95	96	97
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

100	101	102	103	104	107	108	109	110	111	114	115	116	117	118	121	122	123	124	125	128	129	130	131	132	135	136	137	138	139	142	143	144	145	146
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

149	150	151	152	153	156	157	158	159	160	163	164	165	166	167	170	171	172	173	174	177	178	179	180	181	184	185	186	187	188	191	192	193	194	195
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

198	199	200	201	202	205	206	207	208	209	212	213	214	215	216	219	220	221	222	223	226	227	228	229	230	233	234	235	236	237	240	241	242	243	244
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

247	248	249	250	251	254	255	256	257	258	261	262	263	264	265	268	269	270	271	272	275	276	277	278	279	282	283	284	285	286	289	290	291	292	293
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

296	297	298	299	300	303	304	305	306	307	310	311	312	313	314	317	318	319	320	321	324	325	326	327	328	331	332	333	334	335	338	339	340	341	342
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

345	346	347	348	349	352	353	354	355	356	359	360	361	362	363
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Fig. 4. Cluster 29 for the illustrative example

1	8	13	14	15	20	22	25	29	36	43	50	57	64	71	78	85	92	99	106	107	113	120	127	134	135	141	148	155	162	169	173
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1

176	183	190	197	204	211	218	225	232	239	246	253	260	261	267	274	281	288	295	302	309	316	323	330	337	339	344	351	358	365	372	376
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fig. 5. Cluster 44 for the illustrative example

struct 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"

To construct this sentence from the above solution a simple program is necessary which builds the sentence starting with "The service is provided on" and associates the number of the cluster (in this example 44) to the cluster typology (in this example "holidays") followed by "from" the starting day (in this example the day corresponding to index 1), and "to" the final day (in this example the day corresponding to index 362).

As we can observe, the string built from the solution provided by this optimal covering approach is more concise and intelligible than the one produced by the current procedure but to be usable in practice, the solutions has to be obtained in limited computational time. As it is well known, the set covering problem is NP-complete (see Nemhauser and Wolsey (1999)) and its resolution on instances of big size can be incompatible with practical applications and this was the case of the results presented in Amorosi et al. (2015). Hence here our main concern is if this approach can be solved in computational times which are compatible with real operating requirements.

3. The Set Covering Model

As previously mentioned, the problem can be seen as a set covering model where the set to be covered is the periodicity and the subsets are sub-vectors with specific characteristics generated as described in the sequel.

3.1. 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 a 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:

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

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

$$\min \sum_{s \in S} x_s \tag{1}$$

subject to

$$\sum_{s \in S} t_{s,d} * x_s \geq 1 \quad \forall d \in O \tag{2}$$

$$x_s \in \{0, 1\} \quad \forall s \in S \tag{3}$$

By solving the above model we determine the minimum number of sub-vectors to describe the periodicity. Constraint (2) ensures that each operating day is included in at least one of the sub-vectors that belong to the solution. For this model a preprocessing parallel vector generation algorithm (technical details in Section 6) creates all possible feasible subsets. 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.

We generate the sub-vectors in the following way: first we build a vector in which each entry represents the cumulative num-

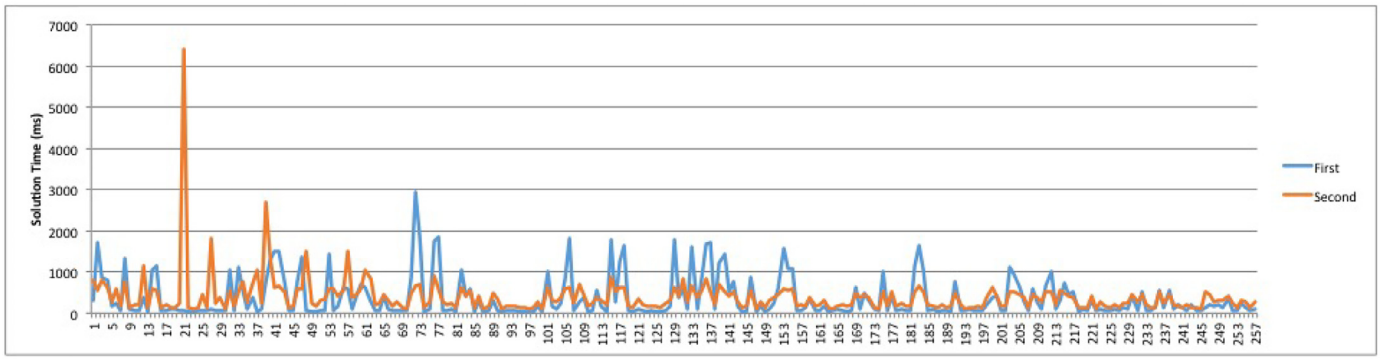

Fig. 6. Solution times comparison between the Set Covering Model and the Integrated Model

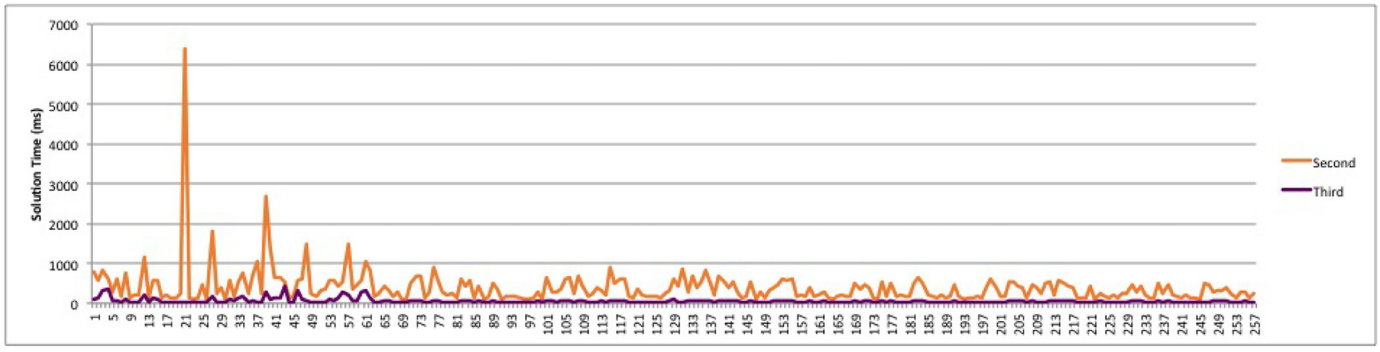

Fig. 7. Solution times comparison between Integrated Model and the New Model

ber of ones found in the original vector from the first to the current index; then we consider only sub-vectors of the cumulative vector with length at least equal to l. From the sub-vectors so generated we select only those with a percentage of ones at least equal to α. These sub-vectors are then used as variables of the set covering model.

4. The Integrated Model

To avoid the use of a parallel preprocessing algorithm external to the model, we can formalize the problem of finding the minimum set of sub-vectors covering 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). Note that as it is possible that a standard cluster (or its sub-vector) is used in a solution more than once (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 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 this reason it was decided to copy each cluster 10 times. In the previous model the preprocessing parallel generation algorithm was in charge of creating copies, while in this model explicit constraints of the number of the copies of the clusters must be formalized.

4.1. 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 $C_{c,k}$ of cluster c;
$d^*_{d,c,k} =$ (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;
$\alpha =$ 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 denoting 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 denoting 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, 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:

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

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.

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

This 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.

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

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 having start position $Y^I_{c,k}$ and final position $Y^F_{c,k}$. Eventually, we introduce the binary variable allow-

ing 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:

$$\min \sum_{c \in C,k} x_{c,k} \tag{4}$$

subject to

$$\sum_{d \in D} K_{d,c,k} \geq \alpha * \left(x_{c,k} + Y^F_{c,k} - Y^I_{c,k}\right) \quad \forall c \in C \quad \forall k \tag{5}$$

$$Y^F_{c,k} \leq m_{c,k} * x_{c,k} \quad \forall c \in C \quad \forall k \tag{6}$$

$$Y^I_{c,k} \leq Y^F_{c,k} - (l-1) * x_{c,k} \quad \forall c \in C \quad \forall k \tag{7}$$

$$Y^I_{c,k} \geq x_{c,k} \quad \forall c \in C \quad \forall k \tag{8}$$

$$KP_{d,c,k} \geq \frac{\left(d^*_{d,c,k} - Y^I_{c,k}\right) + 1}{(Tot+1)} \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{9}$$

$$KP_{d,c,k} \leq 1 + \frac{\left(d^*_{d,c,k} - Y^I_{c,k}\right)}{(Tot+1)} \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{10}$$

$$KN_{d,c,k} \geq \frac{\left(Y^F_{c,k} - d^*_{d,c,k}\right) + 1}{(Tot+1)} \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{11}$$

$$KN_{d,c,k} \leq 1 + \frac{\left(Y^F_{c,k} - d^*_{d,c,k}\right)}{(Tot+1)} \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{12}$$

$$K_{d,c,k} \geq 1 - \left(1 - KP_{d,c,k}\right) * M - \left(1 - KN_{d,c,k}\right) * M \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{13}$$

$$K_{d,c,k} \leq KP_{d,c,k} \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{14}$$

$$K_{d,c,k} \leq KN_{d,c,k} \quad \forall d \in D \quad \forall c \in C \quad \forall k \tag{15}$$

$$\sum_{c \in C,k} K_{d,c,k} \geq 1 \quad \forall d \in D \tag{16}$$

The objective function minimizes the number of clusters required to describe the total periodicity. Constraint (5) ensures that the sub-vectors extracted have at least the predetermined percentage of ones represented by α. Constraints (6), (7) and (8) 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.

Now there is the problem of determining if a specific date d is included or not in a given sub-vector with starting index $Y^I_{c,k}$ and the final index $Y^F_{c,k}$. For this purpose constraints (9) and (10) 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 starting index of the sub-vector extracted from the k-th copy $C_{c,k}$ of the cluster c. Analogously, constraints (11) and (12) 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. Eventually, the definition of the above variables $KP_{d,c,k}$ and $KN_{d,c,k}$ permits to set the variable $K_{d,c,k}$ by means of the inequalities (13), (14) and (15). Consequently, if the variable $K_{d,c,k}$ 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 (16) ensures that any given operating date d is covered by at least one sub-vector of a cluster containing that date.

5. A New Integrated Model

In this section we propose a new model which significantly improves the computational performances.

5.1. 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;
$\alpha =$ 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 denoting 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 denoting 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

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

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

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

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

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

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

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, with the respect of the Integrated Model, the variables $KP_{d,c,k}$ and $KN_{d,c,k}$ have been eliminated. As a consequence, also part of the constraints are 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), impose that the minimum length of each sub-vector is at least equal to l and that each date $d \in D$ is covered. Eventually, 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.

6. Solution Methods and Computational Results

The three models have been tasted and compared on 261 different instances. We recall that in the set covering formulation sub-vectors have to be generated before launching the solver. Therefore, a vector generation algorithm has been designed to create all possible feasible sets. As the number of sub-vectors may be huge, we adopted a parallel solution process that turns out to be quite effective for large combinatorial optimization problems. General issues in parallel computation are dealt with in Bisseling (2004). Parallel computing requires the simultaneous use of more than one process to solve a single computational problem and in our case the generation of sub-vectors from clusters.

This approach permits to reduce the limitations of memory and the computational time. Two main different logics for developing parallel codes can be identified: the *message-passing* (MP) approach and the *multithreading* approach. In the MP approach, each process has access to its own memory for reading and writing data and the processes cooperate by means of exchanging messages. In the second approach all processes access in the same way to all the main memory.

For our purposes the .NET environment is chosen preferring a multithreading approach compared to the MP one. In doing so, as sharing memory among processes is more suitable, it is not necessary to transfer data. We assigned a number of processes equal to the number of clusters (46) and managed the control by means of a master procedure. The first two models were tested on a more extensive set of instances than those utilized in Amorosi et al. (2015), consisting in 261 instances related to different time windows with length ranging in [90, 365] days. The general objective of test problems selection is to include all the complex instances of any possible practical case. 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. As each solution obtained solving optimally the model is not a sentence, we designed a script that, taking in input the solution and the cluster table 4 in the Appendix, writes the corresponding sentence in natural language. We tested it on all the instances verifying the sentences syntax and intelligibility. More in details, this script reads the solution of the model in terms of activated clusters and starting and final indexes of the extracted sub-vectors and produces a sentence describing when the service is active. For instance, if the selected cluster in the solution is cluster number 5 (Friday) and starting and final indexes of the extracted sub-vector correspond respectively to the 2nd January and the 31st of May, the script produces the sentence "The service is provided all Fridays from 2nd January to 31st May". In case the extracted sub-vector contains zeros, these represent exceptions and the script identifies the corresponding dates and adds to the sentence the days in which the service is not active, for example if the zero is in correspondence of the 22nd of April the previous sentence will become "The service is provided all Fridays from 2nd January to 31st May except the 22nd of April".

Table 1
Solution times of the Set Covering Model and the Integrated Model

Model	Min	Max	Average
First	34,89 ms	2932,68 ms	372,31 ms
Second	101,67 ms	6391,95 ms	394,84 ms

Table 2
Solution times of the Integrated Model and the New Model

Model	Min	Max	Average
Second	101,67 ms	6391,95 ms	394,84 ms
Third	15,45 ms	411,80 ms	54,18 ms

Table 3
Solution times of the three models

Model	Min	Max	Average
First	34,89 ms	2932,68 ms	372,31 ms
Second	101,67 ms	6391,95 ms	394,84 ms
Third	15,45 ms	411,80 ms	54,18 ms

All tests were performed on a Windows computer with 8 Intel i7 processors 2.6 GHz and 16 GB of RAM. As far as the problem solver is concerned, Cplex 12.6.1 has been chosen with a multithread strategy solution. They confirm that, as in Amorosi et al. (2015), the set covering model, with external vectors generation, performs better than the integrated one where the vectors generation is embedded in the formulation. Indeed, although the average computational time over the 261 tests is around 0.3 seconds for both models, the range in which the solution times vary is [0.03, 3] seconds for the set covering model and [0.1, 6] seconds for the integrated one as summarized in Table 1 and Fig 6. These times include the preprocessing time, all possible sub-vectors generation through the parallel algorithm as well as model writing for the set covering model, cluster processing and model writing for the integrated model, respectively.

However, the maximum solution times of the Set Covering model (having the best performance) reveals that it is not enough stable, with respect to variability of instances, for practical usability (indeed the waiting time for the customer can be greater than three seconds). For this reason we have introduced the new integrated formulation which overcomes this drawbacks by reducing the number of required variables and constraints with respect to the previous Integrated Model and whose computational performances are analyzed next.

In Table 2 and in Fig 7 we can see the solution times of this third formulation and the comparison with the Integrated Model, that is the comparison of the models which do not require the preprocessing parallel algorithm. Note that having less variables and less constraints makes this formulation much faster to be solved than the Integrated Model. In particular maximum time values are always much lower for the new model. Note that for instance 23 the time reduction is in percentage more than 90%. The average solution time value is of 54.18 ms as opposed to 394,84 ms having a percentage reduction of more than 86%.

The new model performs better also than the set covering model with preprocessing parallel algorithm as shown in Table 3 and Fig 8. It can be observed that the maximum time value of the new model is much lower than the set covering one and its percentage reduction is almost 86%. Average value of the new model

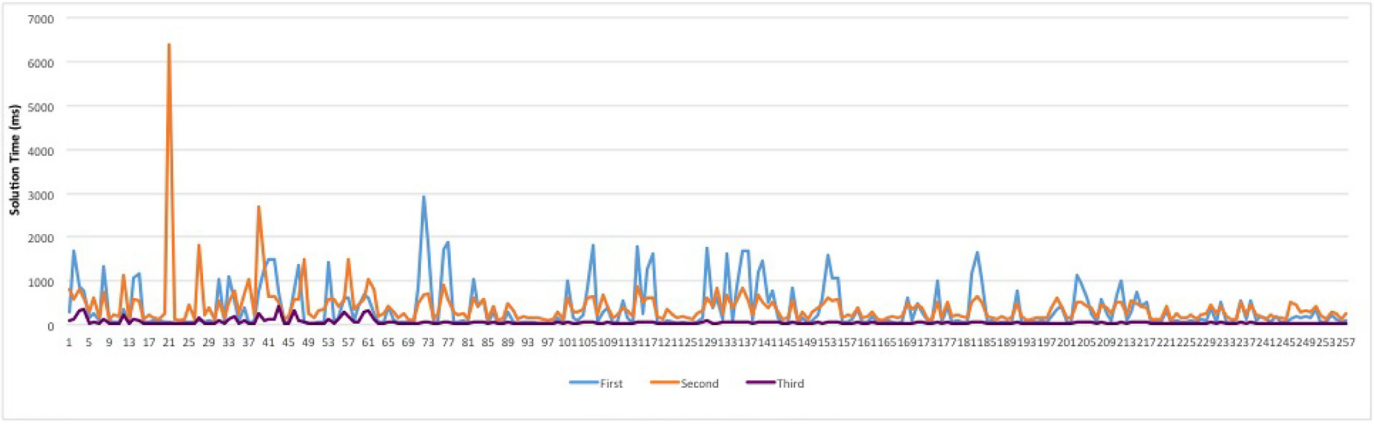

Fig. 8. Solution times comparison between the three models

is also very low and time reduction in percentage is more than 85%.

Thus, we can conclude that the new model compares very favorably with the other two in all instances and it appears very stable with the respect to different problem instances.

7. Conclusions

In this paper we present a new mathematical model for train calendars textual generation based on a specific set covering formulation of a boolean vector with embedded sub-vectors generation. Indeed, although the set covering formulation with external sub-vectors generation could seem rather straightforward, from the methodological point of view it is not obvious how to formulate sub-vectors generation within the same set covering model. As it happens for others ILP problems, different alternative formulations providing the same optimal solutions are possible, yet with different computational performances. It appears that this new model allows to formulate the whole problem by means of variables and constraints which make its computational times quite independent on the characteristics of the problem instances. This model is tested on 261 different instances and compared with the models presented in Amorosi et al. (2015) for which new extensive computational results are performed and included in this paper. The quality of textual representation obtained by all models always compares favorably with texts used in the current practice in all tests. The first formulation (set covering model) performs better than the second one (integrated model), however for both models practical usability has some limits due to computational times (in particular the maximum values). The new model improves computational performances very significantly (maximum time values are reduced in percentage more than 85% with respect to the set covering model and more than 90% with respect to the integrated one) and it appears utilizable in practical cases. Currently, in the operational environment we refer to, the generation of train calendars is based on a heuristic algorithm affected by several issues in complex cases. The first two models proposed were not implemented because of the long processing time. The new one presented in this paper is likely to be implemented in a prototype software as the computational times make the approach applicable in real practical context. The economic impact of this solution, beyond the simple reduction of some direct costs (like for instance the printing of timetables) is mainly related to the quality of customer-enterprise communication services. A description of timetables which is not clear enough is often perceived as a low quality of service in general or a limited care of the customer. Successive researches will be dedicated to apply this methodology to other text generation problems.

Appendix

Table 4
Typology of Clusters

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

References

Amorosi, L., Dell'Olmo, P., Giacco, G., 2015. A new approach for train calendar description generation. In: Models and Technologies for Intelligent Transportation Systems (MT-ITS), Budapest, 3-5 June 2015. IEEE, pp. 280–286.

Arenas, D., Pellegrini, P., Hanafi, S., Rodriguez, J., 2018. Timetable rearrangement to cope with railway maintenance activities. *Computers & Operations Research* Vol. 95, 123–138.

Barzilay, R., Lapata, M., 2006. Aggregation via set partitioning for natural language generation. In: Proc. HLT-NAACL-06, pp. 359–366.

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

Dalianis, H., 1999. Aggregation in natural language generation. *Computational Intelligence* Vol. 15, 384–414.

Ervás, R., Gervás, P., 2005. Case retrieval nets for heuristic lexicalization in natural language generation. In: Proceedings of the Twelfth Portuguese Conference on Artificial Intelligence (EPIA 2005), Covilha, Portugal, pp. 55–66.
Gatt, A., Krahmer, E., 2017. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research* Vol. 60, 1–118.
Goldberg, E., Driedger, N., Kittredge, R., 1994. Using natural-language processing to produce weather forecasts. IEEE Expert Vol. 9, 45–53.
Greiner, M., 2013. Algorithm for generating train calendar texts. *Promet-Traffic&Transportation* Vol. 25, 99–107.
Hua, C., Mellish, C., 2000. Capturing the interaction between aggregation and text planning in two generation systems. In: Proceedings of the First International Conference on Natural Language Generation (INLG00), Mitzpe Ramon, Israel, pp. 186–193.
Liebchen, C., 2008. The first optimized railway timetable in practice. *Transportation Science* Vol. 42, 420–435.
Nemhauser, G.L., Wolsey, L.A., 1999. Integer and Combinatorial Optimization. J. Wiley and Sons.
Odijk, M.A., Romeijn, H.E., van Maaren, H., 2006. Generation of classes of robust periodic railway timetables. *Computers & Operations Research* Vol. 33, 2283–2299.
Perera, R., Nand, P., 2017. Recent advances in natural language generation: a survey and classification of empirical literature. *Computing and Informatics* Vol. 36, 1–32.
Samá, M., D'Ariano, A., Corman, F., Pacciarelli, D., 2017. A variable neighbourhood search for fast train scheduling and routing during disturbed railway traffic situations. *Computers & Operations Research* Vol. 78, 480–499.
Sripada, S., Burnett, N., Turner, R., Mastin, J., Evans, D., 2014. Generating a case study: Nlg meeting weather industry demand for quality and quantity of textual weather forecasts. In: Proceedings of the 8th International Natural Language Generation Conference. Association for Computational Linguistics, USA 2014., pp. 1–5.
Turner, R., Sripada, S., Reiter, E., Davy, I., 2006. Generating spatio-temporal descriptions in pollen forecasts. In: EACL '06 Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics. Association for Computational Linguistics, USA 2006, pp. 163–166.