A branch-and-price algorithm for the aperiodic multi-period service scheduling problem

Citation for published version:
Fernandez, E, Kalcsics, J & Nunez-del-Toro, C 2017, 'A branch-and-price algorithm for the aperiodic multi-period service scheduling problem' European Journal of Operational Research, vol 263, no. 3, pp. 805-814. DOI: 10.1016/j.ejor.2017.06.008

Digital Object Identifier (DOI):
10.1016/j.ejor.2017.06.008

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
European Journal of Operational Research

A branch-and-price algorithm for the aperiodic multi-period service scheduling problem

Elena Fernández[a], Jörg Kalcsics[b], Cristina Núñez-del-Toro[a,*]

[a] *Statistics and Operations Research Department, Universitat Politècnica de Catalunya, Barcelona, Spain*
[b] *School of Mathematics, University of Edinburgh, Edinburgh, United Kingdom*

Abstract

This paper considers the multi-period service scheduling problem with an aperiodic service policy. In this problem, a set of customers who periodically require service over a finite time horizon is given. To satisfy the service demands, a set of operators is given, each with a fixed capacity in terms of the number of customers that can be served per period. With an aperiodic policy, customers may be served before the period were the service would be due. Two criteria are jointly considered in this problem: the total number of operators, and the total number of ahead-of-time periods. The task is to determine the service periods for each customer in such a way that the service requests of the customers are fulfilled and both criteria are minimized. A new integer programming formulation is proposed, which outperforms an existing formulation. Since the computational effort required to obtain solutions considerably increases with the size of the instances, we also present a reformulation suitable for column generation, which is then integrated within a branch-and-price algorithm. Computational experiments highlight the efficiency of this algorithm for the larger instances.

Keywords: combinatorial optimization, multi-period problems, service scheduling, column generation, branch-and-price

1. Introduction

In this paper we propose new and more efficient formulations and solution methods for the Aperiodic Multi-Period Service Scheduling Problem (A-MSSP), which was recently introduced in [13]. In the MSSP there is a set of customers who periodically require service over a finite time horizon.

*Corresponding author
Email address: `cristina.nunez@upc.edu` (Cristina Núñez-del-Toro)

To satisfy the service demands, a set of operators is given, each with a fixed capacity in terms of the number of customers an operator can serve per period. The task is to determine for each customer the periods in which he will be visited by an operator, called *service periods*, such that the service requests of the customers are fulfilled and the total number of operators used over the time horizon is minimal. The set of all service periods of all customers produce a *calendar* for MSSP. In the aperiodic version of the problem, the time between two consecutive services to a customer is not fixed in advance and can be of different length. The requirement in this case is that the maximum time between two consecutive service visits to the same customer never exceeds a given length, which is referred to as his *service interval.* Thus, it is permitted to visit a customer before the end of his service interval. If a service takes place before the end of the customer service interval, this is referred to as an *early service* or *early visit.* In case of an early service, the number of periods between the period where the early service occurs and the end of the customer service interval is referred to as the *earliness* of the visit. The earliness of a calendar for the A-MSSP is the total earliness of all visits scheduled in the calendar. We assume that all time periods have the same length and that all customers have been serviced just before the start of the planning horizon.

In the A-MSSP two criteria are jointly considered for minimization: the total number of operators, and the *earliness* of the calendar.

The A-MSSP appears as a core component in many practical applications from very diverse fields. An application in logistics refers to the collection or delivery of commodities, raw materials, or waste in which customers either produce or consume these items at a given rate per period and they can only store a certain amount at their location. The task is to determine in which periods to service each customer such that the storage limitations at the customers are fulfilled and as few tours as possible are needed. This problem occurs, for example, in the context of collection and recycling of waste electrical and electronic equipment, [13]. Another application appearsin the scheduling of preventive inspections of technical equipment, like production machines or airplanes. For this problem, the goal is to determine a maintenance schedule while minimizing costs, [1, 2, 8]. Similar problems arise in machine replacement, where regular schedules minimizing the variability between consecutive service periods have to be determined. In [10] the same objective is considered in the context of waste collection for rooms in a health care facility. Closely related is the windows scheduling problem in which transmissions of information pages have to be scheduled on broadcasting channels [3, 4]. Also related are replenishment problems in vendor managed inventory systems with direct deliveries [6]. In these problems, each retailer faces a constant demand, the inventory is replenished from a central distribution center and a vehicle can just visit one retailer per period. In [21] ahead of time service visits are allowed. Other related problems arise in task scheduling and periodic assignment problems [9, 11].

As the original formulation proposed in [13] cannot solve instances with more than 30 customers optimally, in this paper we propose an alternative formulation in which customers with the same service interval are grouped into classes. The new formulation is then the basis for a reformulation suitable for column generation. In its turn, the column generation reformulation has been embedded in a branch-and-price solution algorithm, which is the main contribution of this work.

Branch-and-price is a solution method in combinatorial optimization that is widely used for solving Integer Linear Programming (ILP) problems with a large number of variables ([5]). This method has been applied to many different combinatorial optimization problems, e.g., cutting stock, graph coloring, routing and scheduling. For examples of branch-and price solution algorithms applied to scheduling problems the interested reader is referred to [17, 14, 16].

In this paper we present the master problem of the column generation reformulation and its associated pricing problem, for which we propose a polynomial-time exact solution algorithm. A comparison of the Linear Programming (LP) bounds of the proposed formulations shows that the latter outperforms the former. The details of the branch-and-price exact algorithm for the A-MSSP are also presented. To speed-up the column generation we apply a stabilization procedure. We analyze three different branching strategies for the exploration of the enumeration tree as well a procedure to handle infeasibilities in the master problem.

The rest of the paper is organized as follows. In Section 2 we recall the formal definition of the A-MSSP and we present an alternative formulation. In Section 3 we introduce the reformulation that is suitable for column generation, compare its LP bound to the alternative formulation, and discuss the pricing problem necessary for the generation of columns. Section 4 describes the branch-

and-price algorithm to solve the A-MSSP, including the algorithmic details of the implementation as well as the procedure to obtain initial feasible solutions. The results of the computational experiments as well as the comparison with the results of the formulation of Section 2 are shown in Section 5. The paper ends with some conclusions.

2. The Aperiodic Multi-Period Service Scheduling Problem

We use the following notation: T is the index set of (discretized) time periods; I denotes the index set of customers; t_i is the service interval (in number of periods) for customer $i \in I$; Q is the maximal number of customers an operator can serve per period; K is the index set of operators, with $|K| = \left\lceil \frac{|I|}{Q} \right\rceil$.

In the A-MSSP the following decisions must be made: a) determine the service calendar for the customers, i.e., the set of periods in which each customer will be served, given that the time between two consecutive visits does not exceed the service interval, and b) assign each service period of a customer to an operator taking into account the operator capacities. These decisions must be made such that a convex combination of the number of operators needed and the earliness of the calendar is minimized.

The ILP formulation for the A-MSSP that we present below is an alternative to the one proposed in [13]. Its rationale is based on the observation that the service intervals of several customers may coincide. According to the different values of the service intervals, t_i, we classify customers $i \in I$ into *interval classes* (or simply, *classes*). That is, customers with an identical service interval belong to the same class. We define J as the set of indices for the classes, where $|J| \leq |I|$. For each class $j \in J$, u_j denotes the service interval for class j, i.e., the common service interval for all customers of class j, and w_j, the size of class j, i.e., the number of customers with a service interval equal to u_j. Additionally, we define the following parameters: $H_j^t = \{1, \ldots, m_j^t\}$, with $m_j^t = \min\{u_j, |T| - t\}$ denoting the number of potential periods for serving customers of class $j \in J$ after a visit in period $t \in \{0, ..., |T| - 1\}$; $p_j^t = \min\{u_j, t\}$, the number of potential periods forserving customers of class $j \in J$ before a visit in period $t \in T$.

Since the formulation below takes advantage of the above classes of customers, we refer to it as the *class-based* formulation. We define the following sets of decision variables:

For $j \in J$, $t \in T \cup \{0\}$,

$x_j^t =$ number of customers of class j served in period t.

For $j \in J$, $k \in K$, $t \in T$,

$y_{jk}^t =$ number of customers of class j served by operator k in period t.

For $k \in K$, $t \in T$,

$$z_k^t = \begin{cases} 1 & \text{if operator } k \text{ is used in period } t \\ 0 & \text{otherwise} \end{cases}$$

For $j \in J$, $t \in \{0, \ldots, |T| - 1\}$, $h \in H_j^t$,

$f_j^{th} =$ number of customers of class j consecutively served in periods t and $t + h$.

The class-based ILP formulation for the A-MSSP is the following:

$$(AS^c) \quad \min \ \beta \sum_{t\in T}\sum_{k\in K} z_k^t + (1-\beta)\sum_{j\in J}\sum_{t=1}^{|T|-1}\sum_{h\in H_j^t}(u_j - h) f_j^{th} \tag{1}$$

$$s.t. \quad \sum_{h=1}^{u_j} f_j^{0h} = w_j \qquad j \in J \tag{2}$$

$$x_j^t = \sum_{h=1}^{u_j} f_j^{th} \qquad j \in J, t \in \{0,...,|T|-u_j\} \tag{3}$$

$$\sum_{h=1}^{p_j^t} f_j^{t-h,h} = x_j^t \qquad j \in J, t \in T \tag{4}$$

$$z_k^t \leq \sum_{j\in J} y_{jk}^t \qquad k \in K, t \in T \tag{5}$$

$$x_j^t = \sum_{k\in K} y_{jk}^t \qquad j \in J, t \in T \tag{6}$$

$$\sum_{j\in J} y_{jk}^t \leq Q z_k^t \qquad k \in K, t \in T \tag{7}$$

$$Q z_k^t \leq \sum_{j\in J} y_{j,k-1}^t \qquad k \in K \setminus \{1\}, t \in T \tag{8}$$

$$z_k^t \in \{0,1\} \qquad k \in K, t \in T \tag{9}$$

$$x_j^t,\, y_{jk}^t \in \ \mathbb{Z}_0^+ \qquad j \in J, k \in K, t \in T \tag{10}$$

$$f_j^{th} \in \ \mathbb{Z}_0^+ \qquad j \in J, t \in \{0,...,|T|-1\}, h \in H_j^t. \tag{11}$$

Objective (1) minimizes a weighted sum of the total number of operators used in the time horizon and the total earliness. By assigning different values to $\beta \in [0,1]$, both criteria can be considered within different scenarios. Constraints (2) guarantee that the first service period for customers in class j occurs within their service interval. Constraints (3) are logical constraints, which relate the x and f variables. Constraints (4) force that customers in class j being served in period t have a previous service in no more than p_j^t periods before. Constraints (5) ensure that no idle operator is considered as active. The rationale behind these constraints is to strengthen the formulation. Constraints (6) guarantee that customers served in period t are assigned to some operator in that period. Constraints (7) are capacity constraints that ensure that the number of customers assigned to each operator in a given period must not exceed her capacity. Constraints (8) are symmetry breaking constraints imposing that in each period operator k is not used unless operators $1,\dots,k-1$

are full, i.e., each of them has Q assigned customers. Finally, Constraints (9) enforce variables z_k^t to be binary, while Constraints (10) and (11) enforce integrality on the variables x_j^t, y_{jk}^t and f_j^{th}. Formulation AS^c contains a smaller number of variables and constraints than formulation AS in [13], which does not make use of the classes, but uses binary variables only.

3. Column generation formulation

In this section we propose a different formulation for the A-MSSP that is suitable for column generation. Columns of this formulation correspond to *patterns* of services. A pattern is represented by a vector $c = (a_1^c, \dots, a_j^c, \dots, a_{|J|}^c)$ whose *j-th* component, $a_j^c \leq w_j$, indicates the number of customers of class j that are served in a given period. C is defined as the set of all patterns c for any period t. The cost of a pattern $c \in C$ is given by $n_c = \left\lceil \frac{1}{Q} \sum\limits_{j=1}^{|J|} a_j^c \right\rceil$. For each period $t \in T$, the number of all possible patterns is $|C| = \prod\limits_{j=1}^{|J|} (w_j + 1)$. For the column generation formulation we define the following sets of decision variables:

For $t \in T \cup \{0\}$, $c \in C^t$,

$$x_c^t = \begin{cases} 1 & \text{if the pattern } c \text{ is served at period } t \\ 0 & \text{otherwise} \end{cases}$$

Therefore, a formulation for the A-MSSP is as follows:

$$(MP^c) \ \min \beta \sum_{t \in T} \sum_{c \in C} n_c x_c^t + (1-\beta) \sum_{j \in J} \sum_{t=1}^{|T|-1} \sum_{h \in H_j^t} (u_j - h) f_j^{th} \tag{12}$$

$$s.t. \quad \sum_{h=1}^{u_j} f_j^{0h} = w_j \qquad j \in J \tag{13}$$

$$\sum_{c \in C} x_c^t \leq 1 \qquad t \in T \tag{14}$$

$$\sum_{c \in C} a_j^c x_c^t = \sum_{h=1}^{u_j} f_j^{th} \qquad j \in J, t \in \{0, ..., |T| - u_j\} \tag{15}$$

$$\sum_{h=1}^{p_j^t} f_j^{t-h,h} = \sum_{c \in C} a_j^c x_c^t \quad j \in J, t \in T \tag{16}$$

$$x_c^t \in \{0, 1\} \qquad t \in T, c \in C \tag{17}$$

$$f_j^{th} \in \mathbb{Z}_0^+ \qquad j \in J, t \in \{0, ..., |T| - 1\}, h \in H_j^t. \tag{18}$$

The objective function (12) again minimizes a weighted sum of the total number of operators used in the time horizon and the total earliness. Constraints (13) are identical to (2), and Constraints (14) are logical constraints for the columns. Without loss of generality, we can suppose that, for every period, a single pattern is selected. Therefore, due to the definition of the x variables, the latter constraints also define an optimality cut for MP^c. As $x_j^t = \sum\limits_{c \in C} a_j^c x_c^t$ for all $j \in J, t \in T$, Constraints (15)- (16) are the same as (3)-(4). Finally, Constraints (17) enforce variables x_c^t to be binary, while Constraints (18) enforce the integrality on the variables f_j^{th}.

3.1. Comparison of formulations

Next, we compare the LP bounds of formulations MP^c and AS^c. As we will see MP^c may produce better bounds.

Proposition 1. *Formulation MP^c is at least as tight as formulation AS^c, in the sense that the LP bound of MP^c is greater than or equal to the one of AS^c.*

Proof. To see that, we will prove in the following that for every feasible solution of the LP relaxation of MP^c, $(\overline{x}, \overline{f})$, there exists a feasible solution of the LP relaxation of AS^c, $(\widetilde{x}, \widetilde{y}, \widetilde{z}, \widetilde{f})$ whose objective function value is smaller than or equal to the one of $(\overline{x}, \overline{f})$.

Let $(\overline{x}, \overline{f})$ be a feasible solution of the LP relaxation of MP^c. Let $\overline{C}^t = \{c \mid \overline{x}_c^t > 0\}$ and $J^t = \{j \in J \mid a_j^c > 0, c \in \overline{C}^t\}$, with $t \in T$. Then, we define the following vector $\widetilde{x}$:

$$\widetilde{x}_j^t = \begin{cases} \sum\limits_{c \in \overline{C}^t} \overline{x}_c^t a_j^c & \text{for } j \in J^t,\ t \in T \\ 0 & \text{otherwise} \end{cases}$$

By construction, and because $(\overline{x}, \overline{f})$ satisfies (13)-(16), it is clear that $(\widetilde{x}, \widetilde{f})$ satisfies constraints (2) -(4).

To define the vector $\widetilde{z}^t$ associated with a given $t \in T$, we set $K^t = \frac{1}{Q} \sum\limits_{j \in J^t} \widetilde{x}_j^t$. Then:

$$\widetilde{z}_k^t = \begin{cases} 1 & \text{for } k = 1 \dots \lfloor K^t \rfloor \\ K^t - \lfloor K^t \rfloor & \text{for } k = \lfloor K^t \rfloor + 1 \\ 0 & \text{otherwise} \end{cases}$$

Observe that, by construction, for each $t \in T$, we have $\sum\limits_{k \in K} \widetilde{z}_k^t \leq \sum\limits_{c \in \overline{C}^t} n_c \overline{x}_c^t$. Thus, the objective value (1) for $(\widetilde{x}, \widetilde{f})$ is smaller than or equal to the objective value (12) for $(\overline{x}, \overline{f})$.

To define an appropriate vector $\widetilde{y}$, we first set $\widetilde{y}_{jk}^t = 0$, for all $t \in T$, $j \notin J^t$, $k \in K$. Then, we consider the elements of J^t by increasing order of their indices. Let j_r be the *r-th* element of J^t. We define $k(r) = \arg\min \left\{ k \mid \sum\limits_{s=1}^{r} \widetilde{x}_{j_s}^t \leq kQ \right\}$. Then, we define the following vector:

For $\quad r = 1$:

$$\widetilde{y}_{j_r,k}^t = \begin{cases} Q & \text{for } k < k(r) \\ \widetilde{x}_{j_r}^t - \sum\limits_{k' < k} \widetilde{y}_{j_r,k'}^t & \text{for } k = k(r) \\ 0 & \text{otherwise.} \end{cases}$$

For $\quad r > 1$:

$$\widetilde{y}_{j_r,k}^t = \begin{cases} Q - \sum\limits_{s<r} \widetilde{y}_{j_s,k}^t & \text{for } k(r-1) = k < k(r) \\ Q & \text{for } k(r-1) < k < k(r) \\ \widetilde{x}_{j_r}^t - \sum\limits_{k' < k} \widetilde{y}_{j_r,k'}^t & \text{for } k(r-1) < k = k(r) \\ \widetilde{x}_{j_r}^t & \text{for } k(r-1) = k = k(r) \\ 0 & \text{otherwise.} \end{cases}$$

The reader can check that above the vector $\widetilde{y}$ is compatible with $(\widetilde{x}, \widetilde{f})$, so they jointly satisfy constraints (5)-(8). □

3.2. The Pricing Problem

Formulation MP^c has an exponential number of x variables and $O(n^3)$ variables f. The restriction of MP^c to only a subset of x variables is called the Restricted Master Problem RMP^c. We use the following notation for the dual variables associated with the linear programming (LP) relaxation of RMP^c ($LRMP^c$):

γ^t: $t \in T$, for Constraints (14).

π_j^t: $j \in J, t \in \{0, ..., |T| - u_j\}$, for Constraints (15).

σ_j^t: $j \in J, t \in T$, for Constraints (16).

We formulate the pricing problem for finding patterns with negative reduced costs for every period $t \in T$. The formulation uses the following sets of decision variables:

v = number of operators needed.

For $j \in J$,

y_j = number of customers of class j served.

Then, the pricing problem associated with period t is:

$$\left(P_c^t\right) \;\; z_c^t = \gamma^t + \min \; \beta v - \sum_{j \in J} \alpha_j y_j \tag{19}$$

$$s.t. \;\; \sum_{j \in J} y_j \leq Qv \tag{20}$$

$$y_j \leq w_j \qquad j \in J \tag{21}$$

$$v, y_j \in \mathbb{Z}_0^+ \qquad j \in J \tag{22}$$

with $\alpha_j = \pi_j^t + \sigma_j^t$.

Objective (19) minimizes the reduced cost of any possible service pattern for period t relative to the dual multipliers vector $(\gamma^t, \pi_j^t, \sigma_j^t)$. Constraints (21) avoid multiple services to customers. Constraints (20) are capacity constraints that ensure that the number of customers assigned to operators does not exceed their capacity. Finally, Constraints (22) enforce integrality on the variables.

If $z_c^t < 0$, the pattern associated to the optimal solution defines a new variable, which may improve the value of the current $LRMP^c$. The new variable x_c^t is then defined by $c = (a_j^c)_{j \in J}$, where $a_j^c = y_j$, for all $j \in J$, with cost $n_c^t = v$.

Note that P_c^t has a very simple structure, as all y variables have the same coefficient in Constraints (20). Hence, those constraints basically become cardinality constraints, once the number of operators v to be used has been fixed. Taking the objective function into account, any optimal solution to P_c^t will serve as many customers as possible from classes with positive α coefficients, and it will be customers from classes with the largest α_j coefficients first. Thus, in the following we sort the indices of classes by non-increasing α values, i.e., $\alpha_{j_1} \geq \alpha_{j_2} \geq \ldots \geq \alpha_{j_s} > 0$, with ties arbitrarily broken. In particular, P_c^t exhibits the following properties:

1. There always exists an optimal solution to P_c^t where $y_j > 0$ only if $\alpha_j > 0$. If $y_j > 0$ with $\alpha_j = 0$ for some j, then we can set $y_j = 0$ without changing the objective function value.
2. We can also assume that there is an optimal solution with at most one class j for which some but not all customers are served, i.e., $0 < y_j < w_j$. Suppose otherwise that there exist two such classes j_l and $j_{l'}$ where $l < l'$.Let $\Delta = \min\{w_{j_l} - y_{j_l}, y_{j_{l'}}\}$. Then, the solution with $y_{j_l} = y_{j_l} + \Delta$ and $y_{j_{l'}} = y_{j_{l'}} - \Delta$ is also feasible, has an objective function value that is at least as good as the previous one and has one class less with $0 < y_j < w_j$.

Taking into account the above properties, an optimal solution P_c^t can be obtained by trying to increase the number of operators to be used one at a time. After each increase in the number of operators, we increase the number of served customers from classes with $y_j < w_j$, starting with class j_1 and then following the ordering, until either the operator capacity is again reached or there are no more classes with $\alpha_j > 0$. If this results in a solution with a smaller objective function value, we repeat the process. If not, we undo the last increase and stop. Next, we give a formal description of this process, see also Algorithm 1.

Let w'_j, for all $j \in J$, be the number of customers of class j not yet assigned to an operator. Initially, $w'_j = w_j$, for all $j \in J$. At each iteration we try to use a new operator and to assign up to Q unassigned customers to her. Customers of class j are tentatively assigned according to their ordering, starting with customers of class j_1. If j_r denotes the index of the last class assigned at iteration $k-1$, then the set of classes with customers that can be assigned at iteration k is given by $S^k = \{q : r \leq q \leq l\}$ where:

$$l = \begin{cases} s & \text{if } \sum\limits_{h=r}^{s} w'_{j_h} \leq Q \\ \min\left\{l' : \sum\limits_{h=r}^{l'} w'_{j_h} \geq Q\right\} & \text{otherwise} \end{cases} \tag{23}$$

Then, at iteration k we use a new operator and assign customers of classes in S^k to her, provided that $\alpha_{j_l} \min\left\{w'_{j_l}, Q - \sum\limits_{q \in S^k \setminus l} w'_{j_q}\right\} + \sum\limits_{q \in S^k \setminus l} w'_{j_q}\alpha_{j_q} > \beta$. Otherwise the process terminates and the output is given by the set of operators in use so far, together with their assigned customers. Since

Algorithm 1 ensures that a new operator is only used if, together with its tentative assignments, it would improve the objective function value, it produces an optimal solution. The complexity of Algorithm 1 is bounded by $O(|J|log|J|)$, the time complexity of the sorting step. All other steps can be done incrementally in $O(|J|)$ total time.

Algorithm 1: Solution algorithm for P_c^t

```
   Data: $\beta$, $\alpha_j$, $w_j$, $j \in J$, $K$
1  sort $\alpha_{j_1} \geq \alpha_{j_2} \geq \ldots \geq \alpha_{j_s} > 0$, breaking ties arbitrarily;
2  $w'_j \leftarrow w_j$, $j \in J$;
3  stop $\leftarrow$ false;
4  $v \leftarrow 0$; $k \leftarrow 1$; $r \leftarrow 1$;
5  while not stop do
6      if $\sum_{h=r}^{s} w'_{j_h} \leq Q$ then
7          $l \leftarrow s$;
8      else
9          $l \leftarrow \min\left\{l' : \sum_{h=r}^{l'} w'_{j_h} \geq Q\right\}$;
10     end
11     $S^k \leftarrow \{q : r \leq q \leq l\}$;
12     if $\alpha_{j_l} \min\left\{w'_{j_l}, Q - \sum_{q \in S^k \setminus l} w'_{j_q}\right\} + \sum_{q \in S^k \setminus l} w'_{j_q}\alpha_{j_q} > \beta$ and $k \leq |K|$ then
13         $v \leftarrow v + 1$;
14         $y_{j_q} \leftarrow w'_{j_q}$, $q \in S^k \setminus l$;
15         $y_{j_l} \leftarrow \min\left\{w'_{j_l}, Q - \sum_{q \in S^k / l} w'_{j_q}\right\}$;
16         $w'_{j_q} \leftarrow 0$, $q \in S^k \setminus l$;
17         $w'_{j_l} \leftarrow w'_{j_l} - y_{j_l}$;
18         if $w'_{j_l} > 0$ then
19             $r \leftarrow l$;
20         else
21             $r \leftarrow l + 1$;
22         end
23         $k \leftarrow k + 1$;
24     else
25         stop $\leftarrow$ true ;
26     end
27 end
```

4. Branch-and-price algorithm

To obtain integer solutions of guaranteed optimality, the column generation algorithm is embedded into a branch-and-bound search tree. A sketch of the resulting branch-and-price implementation is presented in Algorithm 2. In order to obtain an initial feasible solution for the RMP^c, we apply an *initialization* phase using an adaptation of the greedy heuristic proposed in [13]. Then, at each iteration, after solving the current $LRMP^c$ we solve the pricing problem P_c^t with Algorithm 1 for all $t \in T$. For each $t \in T$ with $z_c^t < 0$ we add the corresponding new column and re-optimize. When no new columns can be found we branch. If the $LRMP^c$ resulting after branching is unfeasible, we apply Farkas pricing [7] in order to recover feasibility at the current node. Furthermore, we apply a stabilization procedure in order to improve the convergence of the overall algorithm. Below we give the details for the greedy heuristic, branching rules, Farkas pricing and stabilization procedure which are included in the branch-and-price algorithm.

Algorithm 2: Generic branch-and-price for the MP^c

```
1  Initialization;
2  stop ← false ;
3  while not stop do
4      solve ($LRMP^c$);
5      if ($LRMP^c$) feasible then
6          solve ($P_c^t$) $\forall t \in T$;
7          if no new variables then
8              if integer solution then
9                  eliminate current node;
10                 if unexplored nodes then
11                     select a new node;
12                 else
13                     stop ← true;
14                 end
15             else
16                 do branching;
17             end
18         end
19     else
20         solve Farkas's pricing problem;
21     end
22 end
```

4.1. Initialization

In the greedy procedure proposed in [13] we iteratively build a solution by selecting calendar-less customers one by one and finding a best calendar for each. To find such a calendar efficiently, we reformulate the problem for a customer as a shortest path problem in an auxiliary network which contains the calendars of all already scheduled customers.For further details the interested reader is referred to [13]. As we are now dealing with classes of customers instead of individual customers, we adapt the heuristic as follows. We first sort the classes of customers by non decreasing values of their service intervals, i.e., $s_{j_1} \leq s_{j_2} \leq \ldots \leq s_{j_{|J|}}$, then we pick the classes one by one, starting with the class j_1. For class j_r, we iteratively pick the customers and find the best calendar for each, as in [13]. An initial set of calendars now consists of all calendars used in the greedy solution, i.e., calendars $c = (a_1^c, \ldots, a_j^c, \ldots, a_{|J|}^c)$ where a_j^c is the number of customers of class j scheduled in period t. In addition, we included the set of calendars where customers of just one class are scheduled, i.e., calendars $c = (0, \ldots, a_j^c, \ldots, 0)$. (Computational results show that including these partial calendars speeds up the exploration of the search trees in 19% of the computing time.) The initial set of columns for the RMP^c then consists of all variables $x_c^t = 1$, $t \in T$, where c is a calendar in the initial set of calendars.

4.2. Branching strategies

Several branching rules can be used to define the search tree of any branch-and-price algorithm. General branching schemes can be found in [5, 18]. In this work, branching rules are applied taking advantage of the specific structure of the RMP^c. Observe that the optimality cut defined by Constraints (14) can be rewritten as:

$$\sum_{c \in C} x_c^t + v^t = 1 \qquad t \in T, \tag{24}$$

with $v^t \in \{0,1\}$, $t \in T$.

Let, $(\overline{x}, \overline{f}, \overline{v})$ denote an optimal solution to $LRMP^c$. We apply three different branching strategies:

BS1: Branching on variables v^t. Branching on variable v^t forces to decide whether or not to schedule visits in period t. In particular, we apply the most fractional variable rule, i.e., we choose to branch on variable v^{t^*} such that:

$$t^* = \arg\min_{t \in T} \left\{ \left| \overline{v}^t - 0.5 \right| \right\} \tag{25}$$

BS2: Branching on variables f_j^{th}. Since RMP^c inherits the original variables f_j^{th}, we use them in one of the branching strategies. We again apply the most fractional variable branching rule.

BS3: Branching on x_c^t variables. We use again the most fractional branching rule. When branching on an x_c^t variable, the structure of the pricing problem associated with time periods $t' \neq t$ remains the same. Moreover, if we fix $x_c^t = 1$ then the pricing problem associated with time period t does not have to be solved again in this branch. On the contrary, in the $x_c^t = 0$ branch it could happen that the pricing problem for time period t produces again the same variable x_c^t. In such a case, additional precautions have to be taken to avoid such a behavior. One possibility is to solve the pricing problem using its MILP formulation (19)-(22), extended with additional constraints excluding the respective column x_c^t. This can be done by (i) defining additional binary variables δ_j, $j \in J$ to indicate whether or not $y_j = a_j^c$; (ii) relating each δ_j to $y_j - a_j^c$ in order to guarantee that $\delta_j = 1$ if and only if $y_j - a_j^c \neq 0$; and, (iii) adding the constraint $\sum_{j \in J} \delta_j \geq 1$, to impose that at least one component differs from that of x_c^t. Since the y_j are general integer variables, (ii) requires, in its turn, introducing additional variables and constraints. Indeed, the extended MILP is considerably more involved, not only than Algorithm 1, but also than the MILP (19)-(22) without the additional binary variables and constraints. To reduce the number of calls to the extended MILP, we first apply the usual pricing via Algorithm 1, and only if for time period t it produces a negative reduced cost column associated with a branching variable x_c^t that we fixed to zero we solve the extended MILP.

Computational tests showed that the best performance is obtained when applying BS1, BS2 and BS3 sequentially, i.e., we explore the branching tree looking first for variables v^t to branch on, then for variables f_j^{th}, and finally for variables x_c^t. In our tests we observed that with this policy, Algorithm 1 never produced a negative reduced cost column associated with a branching variable x_c^t. Thus, in practice we never had to solve the extended MILP. This is probably due to the fact that BS3 follows the two previous branching rules BS1 and BS2, so when this rule is applied, solutions are almost integer.

4.3. Recovering infeasibility

When the $LRMP^c$ resulting after branching turns out to be infeasible, we apply Farkas pricing in order to recover feasibility by finding a new variable to add to the current RMP. Farkas pricing reduces therefore to the standard pricing with different objective function coefficients. In particular, the original cost coefficients are substituted by zero values and the components associated with an extreme dual ray are used instead of the dual variables vector. Thus, for formulation $LRMP^c$, the Farkas pricing problem is formulated as P_c^t with vector $(\gamma^t, \pi_j^t, \sigma_j^t)$ as the dual rays associated with MP^c's constraints (14) (15) and (16), respectively.

4.4. Stabilization

Since the convergence of column generation algorithms can be very slow when solving the $LRMP^c$ with the simplex algorithm, particularly for large and degenerate problems [12], several approaches have been developed to limit the erratic behaviour of the dual variables. In this work we apply the smoothing stabilization approach first introduced in [20] and further developed in [15], in which just a single parameter needs to be adjusted. Let $\lambda^* = (\gamma, \pi, \sigma)$ be the dual variables vector associated with the current $LMRP^c$. When solving the pricing problems, we use a convex combination of λ^* and the best dual multipliers found so far $\overline{\lambda}$, i.e., $\widehat{\lambda} = \Delta\lambda^* + (1-\Delta)\overline{\lambda}$, with $0 \leq \Delta \leq 1$.

When a promising column is found relative to vector $\widehat{\lambda}$, it is added to the $LRMP^c$ only if this column has a negative reduced cost with respect to λ^* as well. When the dual multipliers $\widehat{\lambda}$ improve the dual bound, we update the best known vector to $\overline{\lambda} = \widehat{\lambda}$. For defining and updating Δ, we have tried several strategies based on [19]. The updating of Δ strongly depends on the relative difference between the upper and lower bounds of the RMP (that we respectively denote by Z_{MP} and $L(\hat{\lambda})$), i.e.,

$$Gap = \frac{Z_{MP} - L(\hat{\lambda})}{L(\hat{\lambda})} \tag{26}$$

The strategies we have used are the following:

SS1. We set Δ equal to a fixed value Δ_{fix}. When $Gap < \varepsilon$, we stop the stabilization by setting $\Delta = 1$.

SS2. We use $\Delta_{init} \in (0, 1]$ as an initial value for Δ. Every time $Gap < 1 - \Delta_{init}$ we set $\Delta = 1 - Gap$. When $Gap < \varepsilon$, then $\Delta = 1$.

In both cases ε is a fixed parameter that is used to stop the stabilization.

5. Computational experience

To assess the effectiveness of the branch-and-price algorithm versus the class-based formulation, we ran a series of computational experiments. Since IBM ILOG CPLEX does not include callback routines for the implementation of branch-and-price algorithms (i.e., column generation within the exploration of an enumeration tree), we use SCIP for all experiments. This allows us to put all experiments on an equal footing for the comparison. Both formulations AS and AS^c, and the branch-and-price algorithm based on MP^c were implemented and run using SCIP 3.1.1 on a platform with a Linux 64 bit Ubuntu 12.04 operating system, a Intel® Core™ i7-2600 CPU @ 3.40GHz processor and 7.7 GB RAM. We use IBM ILOG CPLEX 12.5 to solve the linear programming relaxations. The computing time limit was set to one hour.

We randomly generated 360 problem instances with the following characteristics. For the number of customers we chose $|I| \in \{50, 100, 200\}$. The respective instances are denoted by "I50", "I100", and "I200". The number of periods is related to a time horizon of one month, i.e., $|T| = 30$. For the possible service intervals t_i of the customers, we considered two different settings corresponding to 6 and 10 different classes in each case. The first one considers the intervals $t_i \in \{3, 4, 5, 7, 11, 13\}$. The second one considers the intervals $t_i \in \{4, 5, \ldots, 15\}$. We abbreviate the two settings by A, and B, respectively. Different values $Q \in \{3, 5, 7, 10\}$ were used for the capacity of the operators. For each combination of values for $|I|$ and Q and for the two different settings for the service intervals, we generated five different problem instances by randomly determining service intervals for the customers according to a discrete uniform distribution over the respective set of service intervals. The resulting instances are denoted by "$\{A, B\}\_I < |I| > \_Q < Q > \_C < \#instance >$". For example, the first instance with 50 customers, an operator capacity of 3, and service intervals taken from $\{3, 4, 5, 7, 11, 13\}$ is denoted "A_I50_Q3_C1". Finally, we consider four different values for $\beta \in \{0.2, 0.5, 0.8, 1.0\}$. The extreme value of $\beta = 1.0$ was considered to analyze the effect of minimizing the total number of operators used over all periods without penalizing earliness.

First, we tested the effect of the stabilization. For this we solved $LRMP^c$, i.e. only the root node of the branch-and-price algorithm, with both stabilization strategies presented in Section 4.4. Tables 1 and 2 summarize the comparison between the stabilization strategies SS1 and SS2 and without any stabilization (NS). The tables report the average times (in seconds) and the average number of generated columns over ten instances (five instances for each setting), for every size and every value of β. For strategies SS1 and SS2 the results are presented for the best tested values for Δ_{fix} and Δ_{init}, respectively, which, for both types of instances, turned out to be 0.1 in both cases. For both strategies we set $\varepsilon = 0.01$. In both tables, results for the best strategy for each instance size are highlighted in bold.

As can be seen in Table 1, the effect of stabilization is not clear for the A-instances, where the computing times indicate that it would be preferable not to apply any stabilization technique. On the contrary, Table 2 shows that for the B-instances stabilization helps, in general, to reduce not only the number of added columns but also the computing times. In addition, SS2 outperforms SS1 in most cases across all types of problem instances and parameter settings. On average, SS2 obtains optimal solutions 62.6% faster than without stabilization, generating 29.6% fewer columns. Overall, it seems that the number of classes affects stabilization more than the number of customers. Moreover, particularly for B-instances, the performance of stabilization decreases as the number of customers per class increases.

Tables 3 and 4 give a summary of the results for the formulation AS proposed in [13], as well as the new formulation AS^c, and the branch-and-price algorithm for the formulation MP^c proposed in this paper, for A- and B-instances, respectively. Results are presented for every value of β. Each table displays the average values over the five instances in each group of z_{LP}, the initial lower bounds (LP-relaxation for formulations AS and AS^c and lower bound at the root node for MP^c), and UB, the values of the best solutions found. The columns labeled *Gap* show the maximum values among the five instances in the group for the percentage relative deviations of the best-known solutions with respect to the lower bounds at termination (LB); that is, $Gap = \frac{UB-LB}{LB}100$. The average of computing times in seconds, the number of nodes in the branch-and-bound/branch-and-price trees, and the total number of optimally solved instances are given in columns *Time*, *Nodes*, and *Solved*, respectively.

β	Strategy		*I*50				*I*100				*I*200				Total
			*Q*3	*Q*5	*Q*7	*Q*10	*Q*3	*Q*5	*Q*7	*Q*10	*Q*3	*Q*5	*Q*7	*Q*10	
0.2	NS	Time	**3.4**	**2.8**	**2.7**	2.8	**3.9**	363.1	**3.1**	**2.7**	**7.4**	**5.5**	**4.5**	**4.0**	406.0
		Cols	**148.6**	155.5	171.5	176.0	**144.4**	174.6	164.3	164.1	**149.9**	**164.4**	168.6	187.1	**1969.0**
	SS1	Time	7.0	4.4	3.6	**2.3**	9.0	6.5	5.5	4.3	12.0	9.5	7.4	5.9	77.6
		Cols	156.2	**144.8**	**144.5**	**134.0**	176.7	**161.8**	**151.8**	**154.3**	173.5	176.5	**162.8**	176.8	1913.6
	SS2	Time	5.1	4.2	3.5	2.4	6.0	**6.4**	4.0	3.8	10.6	6.6	6.1	4.7	**63.4**
		Cols	158.2	154.5	144.7	**134.0**	160.0	177.0	152.1	156.1	192.0	170.7	174.1	**160.9**	1934.6
0.5	NS	Time	**3.0**	**2.8**	**2.8**	2.6	**4.6**	**3.7**	**3.5**	**3.4**	**8.5**	**5.8**	**5.0**	**3.9**	**49.6**
		Cols	**148.6**	157.6	165.8	173.2	**164.7**	177.5	186.6	199.7	**166.3**	**180.5**	191.1	189.1	2100.7
	SS1	Time	6.9	4.6	3.4	**2.3**	10.1	7.0	5.4	4.5	19.5	12.7	11.0	6.7	94.2
		Cols	162.7	**151.3**	**133.2**	**138.8**	164.8	175.9	164.5	**153.0**	199.3	188.6	**178.9**	**169.9**	**1980.9**
	SS2	Time	6.2	4.6	3.5	2.4	8.3	5.9	3.9	4.0	12.4	9.5	7.6	5.1	73.5
		Cols	159.4	163.4	136.8	140.0	186.7	**161.0**	**162.2**	160.3	187.3	185.1	184.1	171.8	1998.1
0.8	NS	Time	**3.2**	**2.6**	**2.8**	2.7	**4.3**	**3.3**	**3.3**	**3.1**	**8.1**	**5.8**	**4.4**	**3.9**	**47.4**
		Cols	**141.1**	147.3	167.5	174.1	160.2	**159.0**	175.4	185.4	**160.9**	179.2	**169.5**	173.8	1993.4
	SS1	Time	6.3	4.2	3.2	**2.2**	10.9	7.6 1.0 6.8	4.8	20.5	14.4	12.0	8.5	101.4	
		Cols	146.5	**143.2**	**134.3**	**127.5**	181.5	165.6	167.0	151.3	200.3	182.7	177.7	174.8	1952.4
	SS2	Time	5.9	3.9	3.2	2.3	5.7	6.3	4.8	3.9	10.6	8.6	7.4	6.0	68.7
		Cols	157.7	147.1	139.6	128.3	**158.9**	165.3	**164.0**	**151.1**	169.7	**178.7**	184.9	**165.6**	**1910.9**
1.0	NS	Time	1.8	1.2	1.4	1.2	**2.1**	**1.8**	1.5	1.5	**4.8**	**2.9**	**2.5**	2.3	**24.8**
		Cols	**81.1**	68.7	**79.0**	81.4	**84.5**	**87.5**	78.2	**90.4**	**96.7**	95.5	**98.0**	109.5	1050.5
	SS1	Time	2.0	1.3	1.2	0.9	3.1	2.5	2.1	1.6	5.7	4.3	3.1	2.5	30.3
		Cols	83.2	**71.4**	82.1	**76.3**	95.6	96.7	80.3	92.9	112.2	96.2	110.0	100.6	1097.6
	SS2	Time	**1.7**	**1.1**	**1.1**	**0.8**	3.2	2.0	**1.4**	**1.4**	7.1	4.0	3.1	**2.1**	29.1
		Cols	82.5	71.7	82.1	**76.3**	101.5	94.0	**76.3**	94.0	108.1	**95.4**	107.0	**93.1**	1081.9
	NS	Time	**11.4**	**9.4**	**9.7**	9.3	**14.9**	371.9	**11.4**	**10.7**	**28.8**	**20.0**	**16.4**	**14.0**	527.9
		Cols	**519.4**	529.1	583.8	604.7	**553.8**	598.6	604.5	639.6	**573.8**	**619.6**	**627.2**	659.5	7113.6
	SS1	Time	22.3	14.5	11.3	**7.8**	33.2	23.6	19.9	15.2	57.7	41.0	33.5	23.6	303.5
		Cols	548.6	**510.7**	**494.1**	**476.6**	618.6	600.1	563.5	**551.5**	685.3	644.0	629.4	622.1	6944.5
	SS2	Time	18.9	13.8	11.4	7.9	23.2	**20.5**	14.2	13.1	40.8	28.7	24.2	17.9	**234.7**
		Cols	557.8	536.7	503.2	478.6	607.1	**597.3**	**554.6**	561.5	657.1	630.0	650.1	**591.4**	**6925.4**

Table 1: Summary of the stabilization results for $LRMP^c$ (A-instances).

β	Strategy		$I50$				$I100$				$I200$				Total
			$Q3$	$Q5$	$Q7$	$Q10$	$Q3$	$Q5$	$Q7$	$Q10$	$Q3$	$Q5$	$Q7$	$Q10$	
0.2	NS	Time	367.2	7.4	7.4	8.0	16.1	10.5	368.0	9.5	**26.5**	16.5	14.1	12.0	863.1
		Cols	373.9	386.8	418.0	464.8	444.8	404.8	457.5	473.3	352.0	387.2	420.3	462.3	5045.7
	SS1	Time	12.6	8.0	5.5	**5.4**	18.7	9.0	9.5	7.1	31.1	16.1	16.3	10.6	149.7
		Cols	273.2	315.5	290.2	**333.9**	339.1	248.8	294.3	314.7	**324.2**	274.6	326.1	322.0	3656.7
	SS2	Time	**10.2**	**7.3**	**5.1**	**5.4**	**15.9**	**8.2**	**8.8**	**6.6**	29.6	**15.2**	**13.8**	**9.6**	**135.7**
		Cols	**263.8**	**306.5**	**288.4**	334.3	**315.6**	**244.9**	**290.9**	**306.8**	325.5	**271.2**	**322.6**	**316.0**	**3586.2**
0.5	NS	Time	9.6	8.4	7.7	7.5	17.2	13.0	11.4	9.8	40.1	20.0	17.7	13.3	175.7
		Cols	365.1	402.2	397.0	430.4	464.9	470.8	478.7	481.3	463.9	433.9	498.4	490.1	5376.7
	SS1	Time	10.8	7.5	5.6	**5.1**	18.7	10.3	10.2	7.6	45.7	20.6	18.4	10.6	171.0
		Cols	242.5	**292.9**	287.4	310.2	295.2	290.0	288.5	306.7	**382.4**	335.0	349.2	303.0	3683.0
	SS2	Time	**9.3**	**6.8**	**5.0**	**5.1**	**16.4**	**9.5**	**9.1**	**6.3**	**38.6**	**18.5**	**14.0**	**9.3**	**147.8**
		Cols	**238.6**	293.0	**286.7**	**306.0**	**290.8**	**289.8**	**285.0**	**296.8**	386.6	**333.2**	**338.1**	**294.8**	**3639.4**
0.8	NS	Time	10.7	9.7	9.2	8.0	18.2	12.3	12.0	10.7	39.7	**21.0**	16.3	13.9	181.8
		Cols	375.2	422.0	433.5	430.4	443.3	432.7	481.8	499.4	460.9	445.5	457.6	501.6	5383.9
	SS1	Time	10.5	6.7	5.4	5.0	21.0	10.6	10.0	7.3	48.2	31.1	18.4	12.8	187.0
		Cols	**245.5**	271.3	277.6	299.5	**308.3**	244.7	**273.5**	305.7	373.3	**354.2**	304.7	317.6	3575.9
	SS2	Time	**7.9**	**6.1**	**5.1**	**4.9**	**16.8**	**7.9**	**8.7**	**6.5**	**35.7**	21.4	**14.6**	**11.0**	**146.5**
		Cols	248.2	**259.3**	**274.1**	**296.5**	315.7	**238.3**	275.0	**300.1**	**363.9**	356.8	**302.7**	**303.6**	**3534.2**
1.0	NS	Time	3.4	2.8	3.1	3.4	**5.5**	4.8	4.0	3.4	**10.1**	7.0	5.7	5.1	58.5
		Cols	147.3	145.3	170.6	177.5	**167.5**	186.9	182.0	177.8	**131.5**	**160.5**	174.2	182.5	2003.6
	SS1	Time	2.8	2.1	**1.8**	1.9	5.9	3.6	3.8	2.1	11.8	7.9	4.9	4.3	52.9
		Cols	**130.1**	132.1	**136.2**	**150.0**	183.5	141.7	173.8	146.1	157.9	169.1	**151.6**	**153.4**	**1825.4**
	SS2	Time	**2.7**	**2.0**	1.9	**1.7**	6.0	**3.0**	**3.2**	**2.2**	11.4	**6.8**	**4.5**	**3.4**	**48.7**
		Cols	133.1	**130.2**	**136.2**	**150.0**	186.1	**137.7**	**168.7**	**140.8**	173.2	164.4	156.2	155.5	1832.2
	NS	Time	390.9	28.3	27.4	26.9	57.0	40.6	395.4	33.4	116.4	64.5	53.8	44.4	1279.0
		Cols	1261.5	1356.3	1419.1	1503.1	1520.5	1495.2	1600.0	1631.8	1408.3	1427.1	1550.5	1636.5	17809.9
	SS1	Time	36.6	24.3	18.4	17.3	64.2	33.5	33.4	24.1	136.8	75.6	58.0	38.3	560.6
		Cols	891.3	1011.8	991.4	1093.6	1126.1	925.1	1030.2	1073.2	**1237.8**	1132.9	1131.6	1096.0	12741.0
	SS2	Time	**30.1**	**22.2**	**17.0**	**17.1**	**55.1**	**28.7**	**29.7**	**21.6**	**115.2**	**61.9**	**46.9**	**33.3**	**478.7**
		Cols	**883.7**	**989.0**	**985.4**	**1086.8**	**1108.2**	**910.7**	**1019.6**	**1044.5**	1249.2	**1125.5**	**1119.6**	**1069.9**	**12592.1**

Table 2: Summary of the stabilization results for $LRMP^c$ (B-instances).

For formulations AS and AS^c we use as initial upper bounds the values of the best solutions obtained by the greedy heuristic. The beneficial effect of adding these initial upper bounds is not clear for AS, as it increases the computing times in 4% on average. However, the positive effect of these initial upper bounds is considerable for AS^c, as it speeds up the exploration of the search trees in 64% on average.

Formulations AS, AS^c and MP^c produce the same initial lower bounds, except for $\beta = 1.0$. For this value, MP^c produces slightly better lower bounds than formulations AS, and AS^c. This similarity ends, however, when looking at the upper bounds, maximum gaps, computing times and number of optimally solved instances for AS, AS^c, and the branch-and-price algorithm. For example, despite the upper bound provided in the initialization, formulation AS is not able to find a feasible integer solution for larger instances (I100 and I200). In contrast to the branch-and-price algorithm, which even finds optimal solutions for a large number of these instances. For all size of instances, average maximum gaps are around 1.33% and 0.77% (for formulation AS^c), and 0.37% and 0.16% (for formulation MP^c), for A- and B-instances, respectively. In general, proving optimality of the obtained solutions strongly depends on the instance type. For A-instances, proven optimal solutions can be found for 0.0%, 42.5% and 75.4% of the cases for formulations AS, AS^c and MP^c, respectively. For the B-instances, these percentages increase to 0.4%, 72.1% and 90.0%, respectively.

A remarkable feature of the branch-and-price algorithm for MP^c is the small number of nodes that have to be explored in the enumeration tree. Furthermore, the algorithm is not only able to obtain the largest number of optimal solutions in shorter computing times when compared to AS and AS^c, but it also provides the same or better upper bounds with considerable lower gaps for instances without optimal solutions, especially for the A-instances which are the most difficult ones to solve.

6. Conclusions

We have introduced two alternative formulations for the A-MSSP which are based on clusters of customers with identical service intervals. One of these formulations is suitable for column generation and has been taken as the basis for an exact branch-and-price algorithm. We have presented

	Instance		z_{LP}			UB			Gap			$Time$			$Nodes$			$Solved$		
			AS	AS^c	MP^c	AS	AS^c	MP^c	AS	AS^c	MP^c	AS	AS^c	MP^c	AS	AS^c	MP^c	AS	AS^c	MP^c
$\beta = 0.2$	$I50$	$Q3$	17.48	17.48	17.48	-	17.8	17.8	-	2.1	0.0	3600.0	3600.0	75.5	-	460114.6	434.2	0	0	5
		$Q5$	10.49	10.49	10.49	-	10.9	10.9	-	3.3	0.0	3600.0	3482.7	59.8	-	895724.0	184.4	0	2	5
		$Q7$	7.49	7.49	7.49	8.8	8.0	8.0	22.2	0.0	0.0	3600.0	641.8	55.3	8819.8	175455.6	396.8	0	5	5
		$Q10$	5.24	5.24	5.24	5.8	5.8	5.8	14.8	0.0	0.0	3600.0	172.7	44.0	12301.0	77700.8	265.2	0	5	5
	$I100$	$Q3$	31.95	31.95	31.95	-	32.3	32.2	-	1.4	0.6	3600.0	3600.0	2097.9	-	112853.8	301.2	0	0	3
		$Q5$	19.17	19.17	19.17	-	19.5	19.5	-	2.0	0.9	3600.0	3600.0	2584.0	-	300864.6	267.8	0	0	2
		$Q7$	13.69	13.69	13.69	-	14.0	14.0	-	2.8	1.4	3600.0	3600.0	770.3	-	489158.8	186.0	0	0	4
		$Q10$	9.58	9.58	9.58	-	9.9	9.9	-	0.0	0.0	3600.0	1583.7	52.8	-	355511.8	241.6	0	5	5
	$I200$	$Q3$	48.76	48.76	48.76	-	49.2	49.0	-	1.1	0.5	3600.0	3600.0	1541.8	-	26225.6	383.4	0	0	3
		$Q5$	29.26	29.26	29.26	-	29.6	29.5	-	1.8	0.9	3600.0	3600.0	2219.2	-	69787.0	319.8	0	0	2
		$Q7$	20.90	20.90	20.90	-	21.2	21.2	-	2.4	0.8	3600.0	3598.5	1484.4	-	158599.0	217.4	0	1	3
		$Q10$	14.63	14.63	14.63	-	15.0	14.9	-	3.1	1.4	3600.0	3600.0	1477.3	-	326103.6	226.6	0	0	3
$\beta = 0.5$	$I50$	$Q3$	43.70	43.70	43.70	-	44.5	44.5	-	2.3	0.0	3600.0	3600.0	845.4	-	425984.4	612.0	0	0	5
		$Q5$	26.22	26.22	26.22	-	27.3	27.3	-	3.3	0.0	3600.0	3352.3	1157.0	-	828725.4	265.6	0	2	5
		$Q7$	18.73	18.73	18.73	22.8	19.9	19.9	32.0	0.0	0.0	3600.0	825.3	629.2	8630.2	212998.2	567.4	0	5	5
		$Q10$	13.11	13.11	13.11	15.1	14.6	14.5	16.4	0.0	0.0	3600.0	290.6	454.8	13506.6	130449.6	496.0	0	5	5
	$I100$	$Q3$	79.87	79.87	79.87	-	80.8	80.5	-	1.6	0.6	3600.0	3600.0	2684.2	-	97409.0	288.2	0	0	3
		$Q5$	47.92	47.92	47.92	-	48.8	48.7	-	2.6	1.0	3600.0	3600.0	1492.0	-	283981.0	200.6	0	0	4
		$Q7$	34.23	34.23	34.23	-	35.1	35.0	-	2.8	1.5	3600.0	3600.0	732.8	-	539176.8	201.2	0	0	5
		$Q10$	23.96	23.96	23.96	-	24.8	24.8	-	0.0	0.0	3600.0	1426.8	506.0	-	325025.2	370.4	0	5	5
	$I200$	$Q3$	121.90	121.90	121.90	-	122.8	122.6	-	1.0	0.4	3600.0	3600.0	3218.8	-	28497.4	173.2	0	0	1
		$Q5$	73.14	73.14	73.14	-	74.2	73.8	-	2.1	0.7	3600.0	3600.0	2997.1	-	71754.8	269.2	0	0	2
		$Q7$	52.24	52.24	52.24	-	53.2	52.9	-	2.3	0.8	3600.0	3600.0	1756.8	-	155458.4	253.6	0	0	4
		$Q10$	36.57	36.57	36.57	-	37.4	37.3	-	2.7	1.4	3600.0	3600.0	2002.9	-	302861.2	293.2	0	0	3
$\beta = 0.8$	$I50$	$Q3$	69.92	69.92	69.92	-	70.9	70.9	-	1.5	0.0	3600.0	3600.0	1210.0	-	343674.2	619.8	0	0	5
		$Q5$	41.95	41.95	41.95	-	43.4	43.1	-	2.9	0.0	3600.0	3600.0	596.2	-	697950.8	304.6	0	0	5
		$Q7$	29.97	29.97	29.97	-	31.4	31.4	-	1.7	0.0	3600.0	2411.1	550.6	-	640957.2	328.2	0	3	5
		$Q10$	20.98	20.98	20.98	28.8	23.2	23.2	109.5	0.0	0.0	3600.0	1147.6	419.4	12208.4	486889.4	313.2	0	5	5
	$I100$	$Q3$	127.79	127.79	127.79	-	128.8	128.4	-	1.0	0.4	3600.0	3600.0	3255.0	-	109725.8	270.8	0	0	1
		$Q5$	76.67	76.67	76.67	-	77.7	77.5	-	1.5	0.9	3600.0	3600.0	3014.0	-	252162.2	212.0	0	0	1
		$Q7$	54.77	54.77	54.77	-	56.1	55.6	-	2.8	1.1	3600.0	3600.0	2402.9	-	383808.4	331.0	0	0	2
		$Q10$	38.34	38.34	38.34	-	39.5	39.5	-	2.6	0.0	3600.0	3599.9	561.6	-	786505.8	198.8	0	1	5
	$I200$	$Q3$	195.04	195.04	195.04	-	196.5	195.9	-	1.1	0.4	3600.0	3600.0	3600.0	-	30923.2	163.6	0	0	0
		$Q5$	117.02	117.02	117.02	-	118.4	117.8	-	1.3	0.6	3600.0	3600.0	3600.0	-	82400.0	270.2	0	0	0
		$Q7$	83.59	83.59	83.59	-	84.8	84.5	-	1.6	0.7	3600.0	3600.0	3600.0	-	152987.2	226.2	0	0	0
		$Q10$	58.51	58.51	58.51	-	59.8	59.5	-	2.7	0.9	3600.0	3600.0	3600.0	-	270892.2	239.8	0	0	0
$\beta = 1.0$	$I50$	$Q3$	87.40	87.40	87.75	-	87.8	87.8	-	0.0	0.0	3600.0	64.2	3.3	-	4310.2	1.0	0	5	5
		$Q5$	52.44	52.44	52.60	-	52.6	52.6	-	0.0	0.0	3600.0	35.7	1.8	-	3593.4	1.0	0	5	5
		$Q7$	37.46	37.46	37.80	-	37.8	37.8	-	0.0	0.0	3600.0	21.8	1.8	-	2449.8	1.0	0	5	5
		$Q10$	26.22	26.22	26.60	-	26.6	26.6	-	0.0	0.0	3600.0	66.8	1.5	-	19733.6	1.0	0	5	5
	$I100$	$Q3$	159.73	159.73	160.20	-	160.2	160.2	-	0.0	0.0	3600.0	54.3	7.7	-	589.4	1.0	0	5	5
		$Q5$	95.84	95.84	96.16	-	96.2	96.2	-	0.0	0.0	3600.0	61.2	3.9	-	2449.4	1.0	0	5	5
		$Q7$	68.46	68.46	68.80	-	68.8	68.8	-	0.0	0.0	3600.0	65.1	2.4	-	3567.6	1.0	0	5	5
		$Q10$	47.92	47.92	48.40	-	48.4	48.4	-	0.0	0.0	3600.0	23.7	2.1	-	1243.2	1.0	0	5	5
	$I200$	$Q3$	243.80	243.80	244.30	-	244.4	244.4	-	0.0	0.0	3600.0	94.9	8.4	-	270.0	1.0	0	5	5
		$Q5$	146.28	146.28	146.73	-	146.8	146.8	-	0.0	0.0	3600.0	111.6	4.4	-	1888.6	1.0	0	5	5
		$Q7$	104.49	104.49	105.00	-	105.2	105.0	-	1.3	0.0	3600.0	760.2	3.3	-	22629.4	1.0	0	4	5
		$Q10$	73.14	73.14	73.68	-	73.8	73.8	-	1.4	0.0	3600.0	1443.1	3.0	-	79623.4	1.0	0	4	5

Table 3: Branch-and-price results comparison vs. formulations AS and AS^c (A-instances).

	Instance		z_{LP}			UB			Gap			$Time$			$Nodes$			$Solved$		
			AS	AS^c	MP^c	AS	AS^c	MP^c	AS	AS^c	MP^c	AS	AS^c	MP^c	AS	AS^c	MP^c	AS	AS^c	MP^c
	$I50$	$Q3$	12.09	12.09	12.09	-	12.2	12.2	-	0.0	0.0	3600.0	1146.8	278.5	-	26943.6	52.2	0	5	5
		$Q5$	7.26	7.26	7.26	-	7.4	7.3	-	2.8	0.0	3600.0	1841.1	303.9	-	113248.6	74.8	0	3	5
		$Q7$	5.18	5.18	5.18	5.6	5.3	5.3	11.1	0.0	0.0	3600.0	422.4	274.4	6157.2	45136.0	28.4	0	5	5
		$Q10$	3.63	3.63	3.63	4.0	4.0	4.0	12.3	0.0	0.0	3600.0	121.8	13.9	13911.4	23472.2	23.4	0	5	5
$\beta = 0.2$	$I100$	$Q3$	21.59	21.59	21.59	-	21.7	21.7	-	0.0	0.0	3600.0	685.7	266.0	-	3553.4	159.2	0	5	5
		$Q5$	12.95	12.95	12.95	-	13.0	13.0	-	2.0	0.0	3600.0	955.5	125.3	-	16401.4	99.8	0	4	5
		$Q7$	9.25	9.25	9.25	-	9.4	9.4	-	0.0	0.0	3600.0	153.7	141.8	-	3897.2	152.2	0	5	5
		$Q10$	6.48	6.48	6.48	-	6.6	6.6	-	0.0	0.0	3600.0	104.2	104.8	-	6025.2	114.2	0	5	5
	$I200$	$Q3$	45.04	45.04	45.04	-	45.3	45.1	-	0.9	0.0	3600.0	3374.3	280.8	-	8863.0	73.4	0	1	5
		$Q5$	27.02	27.02	27.02	-	27.2	27.1	-	1.0	0.0	3600.0	2798.4	607.7	-	15722.4	186.2	0	3	5
		$Q7$	19.30	19.30	19.30	-	19.4	19.4	-	1.2	0.0	3600.0	2273.3	472.5	-	18150.2	179.6	0	3	5
		$Q10$	13.51	13.51	13.51	-	13.6	13.6	-	1.4	0.0	3600.0	1038.5	229.0	-	16919.8	117.8	0	4	5
0.0	$I50$	$Q3$	30.23	30.23	30.23	-	30.5	30.4	-	1.6	0.0	3600.0	929.2	502.1	-	17912.6	59.4	0	4	5
		$Q5$	18.14	18.14	18.14	-	18.4	18.3	-	2.8	0.0	3600.0	976.8	517.2	-	56186.8	73.6	0	4	5
		$Q7$	12.96	12.96	12.96	14.2	13.3	13.3	16.0	0.0	0.0	3600.0	510.9	709.9	7640.6	49623.8	21.6	0	5	5
		$Q10$	9.07	9.07	9.07	10.0	9.9	9.9	12.3	0.0	0.0	3271.3	154.6	65.3	13904.2	28262.0	20.2	1	5	5
$\beta = 0.5$	$I100$	$Q3$	53.97	53.97	53.97	-	54.3	54.3	-	0.0	0.0	3600.0	514.5	285.7	-	2828.0	141.0	0	5	5
		$Q5$	32.38	32.38	32.38	-	32.5	32.5	-	0.0	0.0	3600.0	570.4	132.4	-	9494.2	110.4	0	5	5
		$Q7$	23.13	23.13	23.13	-	23.4	23.4	-	0.0	0.0	3600.0	207.2	155.5	-	5219.6	168.8	0	5	5
		$Q10$	16.19	16.19	16.19	-	16.5	16.5	-	0.0	0.0	3600.0	101.9	112.3	-	4914.6	137.4	0	5	5
	$I200$	$Q3$	112.60	112.60	112.60	-	113.4	112.7	-	2.2	0.0	3600.0	3118.5	874.9	-	5020.4	128.6	0	1	5
		$Q5$	67.56	67.56	67.56	-	67.7	67.7	-	0.0	0.0	3600.0	1729.6	225.0	-	7319.6	185.0	0	5	5
		$Q7$	48.26	48.26	48.26	-	48.6	48.4	-	1.4	0.0	3600.0	2251.5	241.0	-	18468.8	176.2	0	3	5
		$Q10$	33.78	33.78	33.78	-	34.2	34.0	-	1.8	0.0	3600.0	1637.1	139.6	-	25401.6	194.4	0	3	5
	$I50$	$Q3$	48.37	48.37	48.37	-	48.7	48.6	-	1.2	1.0	3600.0	3111.8	1565.8	-	66117.6	48.2	0	1	3
		$Q5$	29.02	29.02	29.02	-	29.3	29.3	-	2.4	2.0	3600.0	3070.5	1582.6	-	214242.6	46.8	0	2	3
		$Q7$	20.73	20.73	20.73	-	21.2	21.2	-	3.7	0.0	3600.0	2321.2	534.0	-	236307.8	25.4	0	3	5
		$Q10$	14.51	14.51	14.51	17.3	15.7	15.7	53.8	0.0	0.0	3600.0	567.7	372.3	12854.8	104821.6	20.0	0	5	5
$\beta = 0.8$	$I100$	$Q3$	86.35	86.35	86.35	-	86.9	86.9	-	0.7	0.5	3600.0	3600.0	3600.0	-	23069.0	139.4	0	0	0
		$Q5$	51.81	51.81	51.81	-	52.3	52.0	-	2.2	0.6	3600.0	2755.9	245.2	-	55499.4	77.0	0	2	5
		$Q7$	37.01	37.01	37.01	-	37.4	37.4	-	1.4	0.7	3600.0	3600.0	2959.6	-	128662.2	153.2	0	0	1
		$Q10$	25.90	25.90	25.90	-	26.4	26.4	-	2.3	1.8	3600.0	3600.0	3600.0	-	254851.2	136.0	0	0	0
	$I200$	$Q3$	180.16	180.16	180.16	-	181.2	180.4	-	1.6	0.2	3600.0	3600.0	1531.0	-	8091.2	117.8	0	0	4
		$Q5$	108.10	108.10	108.10	-	108.5	108.3	-	0.7	0.0	3600.0	3433.7	698.5	-	19508.4	161.0	0	1	5
		$Q7$	77.21	77.21	77.21	-	77.6	77.4	-	0.8	0.4	3600.0	3571.2	1065.9	-	47896.2	207.8	0	1	4
		$Q10$	54.05	54.05	54.05	-	54.6	54.4	-	1.1	0.6	3600.0	3600.0	2919.8	-	91843.8	232.4	0	0	1
0.0	$I50$	$Q3$	60.47	60.47	60.80	-	60.8	60.8	-	0.0	0.0	3600.0	95.7	1.3	-	2264.6	1.0	0	5	5
		$Q5$	36.28	36.28	36.60	-	36.6	36.6	-	0.0	0.0	3600.0	28.7	1.9	-	871.8	1.0	0	5	5
		$Q7$	25.91	25.91	26.20	-	26.2	26.2	-	0.0	0.0	3600.0	24.6	1.9	-	919.6	1.0	0	5	5
		$Q10$	18.14	18.14	18.40	-	18.4	18.4	-	0.0	0.0	3600.0	24.1	2.9	-	1530.0	1.0	0	5	5
$\beta = 1.0$	$I100$	$Q3$	107.93	107.93	108.55	-	109.5	108.6	-	0.0	0.0	3600.0	71.1	7.7	-	339.3	1.0	0	5	5
		$Q5$	64.76	64.76	65.00	-	65.0	65.0	-	0.0	0.0	3600.0	53.4	3.0	-	560.0	1.0	0	5	5
		$Q7$	46.26	46.26	46.78	-	46.8	46.8	-	0.0	0.0	3600.0	42.6	3.8	-	700.8	1.0	0	5	5
		$Q10$	32.38	32.38	32.96	-	33.0	33.0	-	0.0	0.0	3600.0	17.5	3.1	-	257.8	1.0	0	5	5
	$I200$	$Q3$	225.20	225.20	225.33	-	225.4	225.4	-	0.0	0.0	3600.0	170.8	11.5	-	373.4	1.0	0	5	5
		$Q5$	135.12	135.12	135.38	-	135.4	135.4	-	0.0	0.0	3600.0	126.2	8.7	-	436.0	1.0	0	5	5
		$Q7$	96.51	96.51	96.78	-	96.8	96.8	-	0.0	0.0	3600.0	82.6	7.2	-	440.4	1.0	0	5	5
		$Q10$	67.56	67.56	67.99	-	68.0	68.0	-	0.0	0.0	3600.0	49.5	5.4	-	534.8	1.0	0	5	5

Table 4: Branch-and-price results comparison vs. formulations AS and AS^c (B-instances).

the master problem as well its associated pricing problem, for which a polynomial time exact algorithm has been proposed. The branch-and-price algorithm includes several features that improve its performance. Three different branching strategies are combined and Farkas pricing is applied when the subproblem resulting after branching becomes unfeasible. Additionally, a stabilization procedure has been included to avoid generating a large number of columns. Computational results show not only an improvement in efficiency over the class based formulations but also the outperformance of the branch-and-price algorithm.

Acknowledgements

This research has been partially supported by the Spanish Ministry of Economy and Competitiveness and EDRF funds through grant MTM2015-63779-R (MINECO/FEDER). The research of the third author has been partially through of the Mexican National Council for Science and Technology (CONACyT) through grant 310857. This support is gratefully acknowledged.

References

[1] Anily, S., Glass, C., Hassin, R., 1998. The scheduling of maintenance service. Discrete Applied Mathematics 82 (1-3), 27–42.

[2] Bar-Noy, A., Bhatia, R., Naor, J., Schieber, B., 2002. Minimizing service and operation costs of periodic scheduling. Mathematics of Operations Research 27 (3), 518–544.

[3] Bar-Noy, A., Ladner, R., 2003. Windows scheduling problems for broadcast systems. SIAM Journal on Computing 32 (4), 1091–1113.

[4] Bar-Noy, A., Ladner, R., Tamir, T., Van De Grift, T., 2012. Windows scheduling of arbitrary-length jobs on multiple machines. Journal of Scheduling 15 (2), 141–155.

[5] Barnhart, C., Johnson, E., Nemhauser, G., Savelsbergh, M., Vance, P. H., 1998. Branch-and-price: Column generation for solving huge integer programs. Operations research 46 (3), 316–329.

[6] Campbell, A., Hardin, J., 2005. Vehicle minimization for periodic deliveries. European Journal of Operational Research 165 (3), 668–684.

[7] Farkas, G., 1894. A fourier-féle mechanikai elv alkamazásai [hungarian]. Mathematikai és Természettudományi Értesítö 12, 457–472.

[8] Grigoriev, A., van de Klundert, J., Spieksma, F., 2006. Modeling and solving the periodic maintenance problem. European Journal of Operational Research 172 (3), 783–797.

[9] Han, C.-C., Lin, K.-J., Hou, C.-J., 1996. Distance-constrained scheduling and its applications to real-time systems. IEEE Transactions on Computers 45 (7), 814–826.

[10] Herrmann, J., 2011. Using aggregation to reduce response time variability in cyclic fair sequences. Journal of Scheduling 14 (1), 39–55.

[11] Korst, J., Aarts, E., Lenstra, J., Wessels, J., 1994. Periodic assignment and graph colouring. Discrete Applied Mathematics 51 (3), 291–305.

[12] Lübbecke, M., 2011. Column generation. In: Cochran, J. (Ed.), Encyclopedia of Operations Research and Management Science. John Wiley & Sons, Chichester.

[13] Núñez-del Toro, C., Fernández, E., Kalcsics, J., Nickel, S., 2016. Scheduling policies for multi-period services. European Journal of Operational Research 251 (3), 751–770.

[14] Pereira-Lopes, M., de Carvalho, J. V., 2007. A branch-and-price algorithm for scheduling parallel machines with sequence dependent setup times. European Journal of Operational Research 176 (3), 1508 – 1527.

[15] Pessoa, A., Uchoa, E., de Aragão, M. P., Rodrigues, R., 2010. Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. Mathematical Programming Computation 2 (3-4), 259–290.

[16] Solyalı, O., Özpeynirci, Ö., 2009. Operational fixed job scheduling problem under spread time constraints: a branch-and-price algorithm. International Journal of Production Research 47 (7), 1877–1893.

[17] Vance, P., Barnhart, C., Johnson, E., Nemhauser, G., 1997. Airline crew scheduling: A new formulation and decomposition algorithm. Operations Research 45 (2), 188–200.

[18] Vanderbeck, F., 2011. Branching in branch-and-price: a generic scheme. Mathematical Programming 130 (2), 249–294.

[19] Violin, A., 2014. Mathematical programming approaches to pricing problems. Ph.D. thesis, Universite Libre de Bruxelles.

[20] Wentges, P., 1997. Weighted dantzig-wolfe decomposition for linear mixed-integer programming. International Transactions in Operational Research 4 (2), 151–162.

[21] Xu, Y., Yu, H., Liu, K., Tang, L., 2011. Inventory replenishment scheduling to minimize the number of vehicles. Proceedings - 2011 International Joint Conference on Service Sciences, IJCSS 2011, 74–78.