INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 21 (2014) 453–478
DOI: 10.1111/itor.12080

School bus routing—a column generation approach

J. Kinable[a,b], F.C.R. Spieksma[a] and G. Vanden Berghe[b]

[a]*ORSTAT, KU Leuven, Belgium*
[b]*KU Leuven, Department of Computer Science, CODeS & iMinds-ITEC, Belgium*
E-mail: joris.kinable@kuleuven.be [Kinable]; frits.spieksma@kuleuven.be [Spieksma]; greet.vandenberghe@cs.kuleuven.be [Vanden Berghe]

Received 30 January 2013; received in revised form 10 January 2014; accepted 24 January 2014

Abstract

The School Bus Routing Problem (SBRP), a generalization of the well-known Vehicle Routing Problem, involves the routing, planning, and scheduling of public school bus transportation. The problem can be divided into several subproblems, including bus stop selection, assigning students to buses, and determining the bus routes. This work presents an exact branch-and-price framework for the SBRP, with a strong emphasis on efficiency issues inherently related to column generation (CG). Experiments are conducted on a set of 128 SBRP instances. Many of these instances are solved optimally; for the remaining instances, strong lower bounds have been derived. Furthermore, better integer solutions were found for a number of instances reported in the literature. Both lower bounds computed on the optimum solution and stabilization added to the CG procedure significantly improved computation times.

Keywords: vehicle routing; column generation; branch-and-price

1. Introduction

Many primary and secondary schools in Europe organize school bus transportation services for commuting students. The organization constitutes a challenging task from both a planning and budgetary perspective. A typical school bus planning problem entails selecting appropriate bus stops reachable by the students, assigning students to the available buses, and determining the necessary bus routes. Possible locations for the bus stops are usually restricted by local policies and legislation, such as the maximum walking distance to the stop or safety regulations. From an optimization point of view, different goals can be aspired to, including minimizing the total travel distance or the number of buses, balancing bus loads, or keeping the travel time spent by the students to a minimum. This Vehicle Routing Problem is commonly referred to as the School Bus Routing Problem (SBRP); see Section 2 for a precise description.

The SBRP is part of the class of Capacitated Vehicle Routing Problems in which a set of bus tours has to be designed, each passing through a number of bus stops. The tours have to be disjoint

except in the depot node. Students are assigned to stops on the tours; students can only be assigned to stops they can reach, and the total number of students assigned to stops on a single route cannot exceed the bus capacity.

Among the many variants that exist in the domain of school bus routing (for an overview, see Park and Kim, 2010, and Section 2), we focus here on a single-school SBRP, without time windows. The main contribution of this paper is to present a branch-and-price framework based on a set covering formulation of the SBRP. We provide an in-depth discussion on the design of the branch-and-price framework, thereby focusing on a number of choices made in the implementation in order to improve the efficiency of the framework. We discuss the following:

- Lower bounds on the optimal integer solutions.
- A comparison of two distinct stabilization approaches to reduce degeneracy.
- Effective pruning mechanisms.
- A column pool manager (CPM).

We demonstrate the performance of our branch-and-price algorithm on two benchmark sets: one set containing traditional SBRP instances and a newly generated set of instances.

The remainder of this paper is structured as follows: Section 2 provides an overview of related work on the SBRP. Sections 3 and 4 introduce the column generation (CG) procedure. The latter procedure is then integrated in a branch-and-price framework, which we discuss in Section 5. To improve the framework's efficiency, several extensions are implemented, including stabilization (Section 4.2), a column manager (Section 4.3), and a pruning mechanism (Sections 5.3 and 5.4). Finally, the resulting algorithm is tested on a series of 128 SBRP instances. The results are presented in Section 6. Section 7 provides the conclusions.

2. Problem description and related research

The SBRP can be defined as follows. We are given a set of bus stops V (including the school) with a distance for each ordered pair of stops, as well as a set of students S. For each student $s \in S$, a set $V_s \subseteq V$ is given that represents the set of stops to which the student can be assigned. Assigning student s to a stop in V_s is called a *feasible* assignment. There is a fleet of identical vehicles available, each with capacity Q. A *route* is a sequence of stops ending with the school. Students assigned to stops in a route are picked up by the vehicle traveling on that route. The problem is to find a feasible assignment of students to stops, and find routes for the vehicles, such that (a) the capacity of each vehicle is respected, (b) each student is picked up, and (c) total length of the routes is minimized.

Note that the description of our problem does not provide the locations of the students. Indeed, we only know for each student the set of stops to which this student can be assigned. This feature distinguishes our problem from the more general Multiple Vehicle Traveling Purchaser problem (MV-TPP; see Riera-Ledesma and Salazar-González, 2012) where a location for a student, and its induced distance to a stop, is used to include assignment costs in the objective of the problem. These assignment costs capture the cost of assigning a student to a stop, and may represent walking distance. Of course, MV-TPP is more general since, by having assignment costs in $\{0, \infty\}$, an instance of SBRP arises. However, the setting without assignment costs conforms with the situation faced by a bus company designing routes (for more details, see Schittekat et al., 2013), where it is stipulated

that any assignment of students to stops should satisfy a maximum walking distance. Thus, from the point of view of the bus company, no optimization of the walking distances is required; it is only required that they do not exceed this maximum walking distance.

Obviously, SBRP is not a new problem. Indeed, many variations have been proposed in the literature. Here, we do not aim to give an overview, instead we restrict ourselves to discussing the main solution approaches. For a recent overview, we refer to Park and Kim (2010); a discussion of the MV-TPP and related models can be found in Riera-Ledesma and Salazar-González (2013).

Due to its composite nature, the earliest papers discussing school bus routing attempted to solve the problem via decomposition. The selection, assignment, and routing problems are solved independently, and the results are then merged into a feasible SBRP solution. These attempts can be roughly classified into two groups (Park and Kim, 2010): Location Allocation Routing (LAR) and Allocation Routing Location (ARL). In the LAR class (e.g. Bodin and Berman, 1979; Desrosiers et al., 1986; Dulac et al., 1980), first the stops are determined and students are assigned to those stops, after which the necessary bus routes are generated. A disadvantage of this approach is that the first two subproblems, selection and allocation, are solved independently of the routing problem, often resulting in excessive and suboptimal routes (Park and Kim, 2010). To counter this problem, the ARL strategy has been proposed effectively changing the order in which the subproblems are treated (Bowerman et al., 1995; Chapleau et al., 1985). First the students are assigned to buses, thereby taking capacity constraints into consideration. Then the bus stops are selected, and the bus routes are created. Although this approach resolves some of the issues inherent to LAR (Bowerman et al., 1995), assigning students to buses before the stop locations have been decided upon may still lead to suboptimal schedules. The problems surrounding the decomposition of the SBRP as demonstrated by the discussions on LAR and ARL strategies motivate the use of a more integrated approach that treats the SBRP as an integral problem instead of the sum of several subproblems.

Schittekat et al. (2013) describe a metaheuristic for the SBRP. By comparing their results with a lower bound, they show that the metaheuristic is capable of efficiently producing high-quality solutions for the instances generated.

Riera-Ledesma and Salazar-González (2012) propose a cutting plane algorithm for MV-TPP. The method is based on an integer programming formulation that uses, among other variables, a binary variable for each pair of stops. They report extensive computational results solving instances with up to 125 stops and 125 students. In a recent follow-up paper, Riera-Ledesma and Salazar-González (2013) use a set covering formulation, together with cuts from Riera-Ledesma and Salazar-González (2012) to construct a branch-and-cut-and-price algorithm for the MV-TPP. Although their work represents a formidable step forward in our ability to solve instances of MV-TPP, it is good to note that in most of their instances the number of students does not exceed the number of stops. In our experience, however (see Park et al., 2012; Schittekat et al., 2013), typical instances of the SBRP feature many more students than stops. One goal of our work is to find out how a branch-and-price approach fares upon such instances.

Another problem related to SBRP is the m-Capacitated Ring-Star Problem (m-CRSP; Baldacci et al., 2007). In m-CRSP, one has to find m paths (rings), starting and ending in a depot, and traversing through a number of customers and Steiner nodes. The paths have to be disjoint, except for the depot node. Each customer in the graph needs to be either part of a ring, or must be assigned to a node that is part of a ring. The total number of customers in a single ring, plus the number of customers assigned to it, cannot exceed a predefined capacity Q. The m-CRSP considers both

assignment and routing costs. Assignment costs are incurred whenever a customer is assigned to another node in a ring. Routing costs are incurred for the edges that are part of a ring.

Let us clarify the relation between MV-TPP and its special case SBRP, and, on the other hand, m-CRSP. Clearly, as described in Riera-Ledesma and Salazar-González (2012), an algorithm for MV-TPP can be used to solve instances of m-CRSP. To see this, imagine that each customer in m-CRSP becomes a student plus a stop in MV-TPP, while a Steiner node in m-CRSP becomes a stop. Next, a solution to the resulting instance of MV-TPP that consists of routes that visit to stops to which students have been assigned is easily casted as a solution to m-CRSP. The reverse is true as well: MV-TPP is a special case of m-CRSP. For each student in MV-TPP, a customer is created, and for every stop a Steiner node is created. Routing costs between the Steiner nodes are identical to the routing costs between the stops in MV-TPP. Routing costs of edges incident to customers are set to infinity. In a similar fashion, assignment costs are determined. All assignment costs are set to infinity, except for certain customer–Steiner node pairs: for a student $s \in S$ and V_s, the assignment costs are set to 0 for the corresponding customer–Steiner node pairs. When the optimal solution to the constructed m-CRSP yields an objective value smaller than infinity, a feasible solution to MV-TPP follows directly. Note that m-CRSP requires m, the number of rings, as input. This value is not known for MV-TPP, but is bounded from above. Hence, an algorithm for m-CRSP can be used to solve the MV-TPP by means of binary search on the number of vehicles. Exact methods for m-CRSP based on integer programming formulations have been presented in Baldacci et al. (2007) (branch and cut) and Hoshino and de Souza (2009) (branch and cut and price).

3. Set covering formulation of SBRP

We use the following Mixed Integer Programming (MIP) formulation of the SBRP, which we will denote as the master problem (MP). Table 1 describes the necessary variables and parameters. For a traditional three-index MIP formulation of the SBRP, we refer to Schittekat et al. (2013) and Riera-Ledesma and Salazar-González (2013):

$$MP: \min \quad \sum_{p \in P} \delta_p z_p \tag{1}$$

$$\text{s.t.} \quad \sum_{p \in P} t_{sp} z_p \geq 1 \quad \forall s \in S \tag{2}$$

$$\sum_{p \in P} r_{vp} z_p \leq 1 \quad \forall v \in V \tag{3}$$

$$\sum_{p \in P} z_p \leq U \tag{4}$$

$$\sum_{p \in P} z_p \geq L \tag{5}$$

$$z_p \in \{0, 1\} \quad \forall p \in P. \tag{6}$$

Table 1
Notation used throughout the paper

Variable/parameter	Description
z_p	1 if bus schedule $p \in P$ is used, 0 otherwise
V	Set of bus stops (including the school v_0)
S	Set of students
V_s	$V_s \subseteq V \backslash \{school\}$: the set of stops student $s \in S$ can reach
P	Set of all bus schedules
t_{sp}	1 if student s is picked up in bus schedule p, 0 otherwise
r_{vp}	1 if stop v is part of bus schedule p, 0 otherwise
Q	Maximum capacity of the buses
δ_p	Cost induced by schedule p
L, U	Lower and upper bound on the number of buses

In this formulation, p is an index corresponding to a bus route. More precisely, p defines a complete, valid, *bus schedule*: an ordered sequence of stops the bus driver should visit (ending at the school), and the specific students who should be picked up at the corresponding stops. The set of all feasible bus schedules is denoted by P. As mentioned before, we assume that all buses are identical and have a capacity Q. The cost δ_p associated with each bus schedule is the travel distance required to visit all stops on the schedule (note that these travel distances do not necessarily satisfy the triangle inequality). Constraints (2) and (3), respectively, ensure that each student is picked up at some stop, and no stop is visited more than once. Constraints (2) are commonly referred to as set cover constraints. Constraint (4) enforces bounds on the number of buses used in the solution. Observe that formulation (1)–(6) is quite flexible in the sense that it can accommodate all kinds of potential constraints on individual routes. For instance, upper bounds on the length of a route, or on the number of stops within a route, are easily incorporated.

4. Column generation

Solving a problem MP (Section 3) requires an exponentially large set of columns P. When the integrality constraints are replaced by the weaker constraints $z_p \geq 0$, we obtain a relaxation, denoted LPM, which we will solve via a CG procedure (for details on CG, see Chvátal, 1983; Vanderbeck, 1994; Wolsey, 1998). Instead of solving LPM directly, a reduced version called the restricted MP (RMP) is solved, where the set of columns P is replaced by a subset $P' \subseteq P$, $|P'| \ll |P|$. Subset P' contains an initial feasible solution that is produced by the heuristic proposed in Schittekat et al. (2013). Section 4.1 describes the pricing problem and provides two algorithms to solve it. The performance of the CG procedure is improved by using stabilization and adding a CPM. Section 4.2 compares two popular stabilization approaches, whereas Section 4.3 elaborates on the CPM.

4.1. Pricing problem

The pricing problem, obtained from the dual formulation of problem MP (1)–(6), is as follows:

$$\exists p : q_p < 0 \tag{7}$$

$$q_p = \delta_p + \sum_{v \in V} r_{vp} w_v - \sum_{s \in S} t_{sp} u_s + m - n, \tag{8}$$

where w_v, u_s, m, and n are the dual variables associated with constraints (3), (2), (4), and (5), respectively.

Throughout the paper, p^* denotes the column that yields the most negative value for q_p, that is, $p^* = \text{argmin}\{q_p\}$, and $q^* = q_{p^*}$ is the corresponding optimal objective value of the pricing problem.

Informally, the pricing problem involves finding a path ending at the school such that q_p yields a negative value. When the school is decoupled into a start and end node, the problem becomes the well-known elementary shortest path problem with resource constraints (ESPPRC; Desaulniers et al., 2005). Since ESPPRC is NP-hard, we use two different procedures to solve the pricing problem: a fast heuristic and a slower but exact labeling algorithm. First, the heuristic attempts to quickly find several columns with a negative reduced cost. When this attempt fails, the exact labeling algorithm takes over. Due to its exact nature, the labeling algorithm is guaranteed to find a negative reduced cost column if such a column exists. Ideally, the number of times the exact algorithm is invoked is kept to a minimum as it is relatively expensive to execute.

4.1.1. A local search heuristic

Our local search algorithm is initialized with a randomly generated feasible bus route, that is, an ordered sequence of stops. The algorithm iteratively attempts to improve the solution by exploring neighboring solutions. The set of feasible neighbor solutions is defined as the union of the following three neighborhoods:

1. Insert neighborhood—an unvisited stop is inserted in the route.
2. Remove neighborhood—a stop is removed from the route.
3. Swap neighborhood—two stops in the route are swapped, thereby changing the order in which stops are visited.

For each route, a student assignment problem has to be solved. Due to the presence of constraint (3) in the MP, certain students are forced to be part of the route; a student $s \in S$ must be picked up whenever the route visits all stops in V_s. When there is residual bus capacity, additional students are added to the schedule, thereby maximizing the total dual price collected by the students.

The objective value of a solution is calculated via Equation (8). A neighboring solution is selected over another solution if its objective value is better; only improving moves are allowed. The algorithm terminates when there are no more improving moves available, that is, when a local optimum is reached.

It is known that the CG procedure can be accelerated when the pricing problem returns multiple columns at once. Therefore, the heuristic is executed several times to obtain multiple solutions. To prevent the heuristic from returning the same solution twice, we initialize it with different routes, and only accept neighborhood moves that are sufficiently distinct from earlier returned solutions.

4.1.2. Labeling algorithm

To prove optimality of the CG procedure, an exact pricing algorithm is required. We solve the pricing problem to optimality using a dynamic programming based approach: a labeling algorithm. In related works, such approaches have been successfully applied to problems such as the pickup and delivery problem with time windows (Dell'Amico et al., 2006; Ropke and Cordeau, 2009), the vehicle routing problem with time windows (Desaulniers et al., 2005), and the capacitated location-routing problem (Albareda Sambola, 2003). Alternatively, one may use the q-route approach presented in Fukasawa et al. (2006), as has been demonstrated for the m-CRSP in Hoshino and De Souza (2012).

Consider the weighted, undirected, graph $G = (V', E)$, $V' = (V \cup \{t\})$, where V represents the set of bus stops; t, a copy of the school vertex v_0, and $E = (V \times V) \cup (V \backslash v_0 \times \{t\})$ the set of edges. The weight on an edge $(i, j) \in V \times V$ equals c_{ij}, whereas the weights on the edges (i, t), $i \in (V \backslash v_0)$ are equal to c_{iv_0}. To solve the pricing problem, the algorithm searches for a simple path from v_0 to t with a negative reduced cost.

The labeling algorithm starts by generating the shortest possible partial path, $[v_0]$. At each consecutive iteration, a partial path p' is extended to each stop in V not already present in p'. For efficiency reasons, partial paths are not stored in their entirety; instead, labels are used. Each label ℓ is associated with a vertex $v(\ell) \in V$, and has a reference (pointer) to a preceding label $p(\ell)$, except if the label is associated with vertex v_0. Each label ℓ uniquely identifies a partial path from v_0 to $v(\ell)$, that is, a path $p_\ell = [v(\ell), v(p(\ell)), v(p(p(\ell))) \ldots, v_0]$ can be reconstructed simply by following the pointers to the preceding labels. Let $S(\ell)$ be a set of students who are assigned to a stop in path p_ℓ, and let $V(\ell)$ be the set of stops in p_ℓ, that is, $V(\ell) = \{v(\ell), v(p(\ell)), v(p(p(\ell))) \ldots, v_0\}$. The set $S(\ell)$ is calculated by selecting at most Q students, having the highest dual price; of course, a student $s \in S$ can only be selected if $V_s \cap V(\ell) \neq \emptyset$. Moreover, if $V_s \backslash V(\ell) = \emptyset$, then student s must be in $S(\ell)$ due to constraint (3). Finally, for each partial path denoted by label ℓ, $c(\ell)$ is the accumulated cost defined by Equation (8).

When a partial path would be extended to all its unvisited neighbors, the algorithm would simply be an inefficient enumeration approach. To circumvent this issue, several restrictions are applicable:

1. A path can only be extended to the school if the resulting path is of a negative reduced cost.
2. A path may not be extended to a vertex if this would result in a cycle.
3. A path is not extend to a vertex if a lower bound proves that it can never become a negative reduced cost path. For any path identified by label ℓ, a lower bound on the reduced cost can be computed by taking into account: (a) $c(\ell)$, (b) a lower bound on cost required to complete the path to vertex t, and (c) an upper bound on the dual prices that can be collected by students on the complete route.

4. A path is not extended if it is dominated by another path, that is, if there exists a more cost-effective route.

To determine whether a path having label ℓ_1 dominates another path having label ℓ_2, the following domination criteria are used:

$$v(\ell_1) = v(\ell_2) \tag{9}$$

$$c(\ell_1) \leq c(\ell_2) \tag{10}$$

$$V(\ell_1) \subseteq V(\ell_2). \tag{11}$$

The above conditions intuitively state that for two paths leading to the same stop, the path identified by label ℓ_1 dominates the path having label ℓ_2 if its reduced cost is better, and if path ℓ_2 has access to the same students as path ℓ_1, that is, $S(\ell_1) \subseteq \{s \in S : V_s \cap V(\ell_2) \neq \emptyset\}$.

Clearly, it is pointless to extend a dominated label. Moreover, as a logical consequence, all successors of a dominated label are also suboptimal. For this reason, in our implementation, we also maintain pointers from a label to its direct successors such that we can delete or modify successors in case one of their predecessors turns out to be suboptimal (for more details, we refer to a discussion on label correcting algorithms, for example, Ahuja et al., 1993).

The labeling algorithm can be accelerated by adding an admissible heuristic. Given the sequence of stops that make up the partial route, the heuristic first computes an upper bound on the maximal dual price that can be incurred by the students. Next, the heuristic computes a lower bound on the total distance of the complete route, that is, the distance traveled so far plus the minimum distance required to extend the route to node t. Finally, via Equation (8), the heuristic assesses whether extending the label to a path with a negative reduced cost is attainable.

The labeling algorithm terminates as soon as there are no more labels to extend. If, during the course of the algorithm, no path has been found, we can positively attest that no negative reduced cost path exists.

It should be pointed out that the order in which labels are being processed is of importance. Preferably, labels that at a later point in time appear to be dominated by some other label are not extended. The latter would require that the algorithm recomputes a substantial part of the search tree. In our implementation, we experimented with three orderings in which the labels are processed: Breadth First Search (BFS), Depth First Search (DFS), and Best First Search (BEFS). In BFS, the oldest unextended label gets extended first. In DFS, the newest generated label is always extended first. Finally, BEFS extends the label representing the path with the best objective value. Our experiments showed that BFS slightly outperformed DFS, which in turn outperformed BEFS. Several extensions and improvements for the labeling algorithm have been proposed in related works, some of which we have implemented to accelerate the pricing procedure.

The labeling algorithm should not necessarily search for a column with the best objective value. Instead of performing an exhaustive search, the algorithm could be easily modified such that it returns a negative reduced cost column as soon as it finds one. Note that this modification still enables us to solve the CG procedure to optimality.

A second modification amounts to changing the nodes to which a label can be extended. In the previous section, the exact labeling procedure always extends a partial path that ends in a node $v \in V$ to all its unvisited neighbors. Similar to Dell'Amico et al. (2006), we also implemented a k-nearest neighbor (k-NN) variant where a node is only extended to its k-NNs. Naturally, for $k = |V|$, the search neighborhood becomes exact. We start the algorithm for $k = 2$. When no solution has been found, we expand the search to $k = 4$ and finally to $k = |V|$. This procedure is motivated by the fact that the total travel distance is minimized for each bus tour. As a consequence of this approach, the average number of labels generated during the labeling procedure decreased for a number of instances.

4.2. Stabilization

CG is susceptible to degeneracy (Desaulniers et al., 2005), a process that decreases the convergence speed of the CG algorithm. During several iterations, new columns are added to the MP, but no improvements are being made in terms of the objective function (plateau effect). Also, once the objective function gets closer to its optimal value, the convergence speed decreases drastically. Sometimes many additional iterations are needed to complete the process (tailing-off effect; Desaulniers et al., 2005). The slow convergence speed is partially attributed to degeneracy in the primal. In addition, it is known that strong oscillations in the dual variables play an important role. In a typical CG process, it is frequently observed that some dual variables pick up most of the dual price, whereas the remaining variables have a near-zero value. The unbalanced distribution of dual prices regularly causes the pricing problem to generate redundant columns that will never be part of an optimal solution; an example of which is given in Rousseau et al. (2007).

To counter these issues, stabilization procedures have been developed in an attempt to guide the search faster toward a global optimum. Many different stabilization approaches exist, proximal-type stabilization (PTS; Amor and Desrosiers, 2006; Merle et al., 1997), bundle-type (Briant et al., 2008), and interior point stabilization (IPS; Rousseau et al., 2007) are the most prominent ones; detailed discussions and comparisons of these methods can be found in other studies (Amor et al., 2009; Lübbecke, 2010; Lübbecke and Desrosiers, 2002). In this work, we compare two of the most popular stabilization approaches, namely, Du Merle's 3-piecewise PTS (Merle et al., 1997) and IPS (Rousseau et al., 2007). In the next two subsections, we apply both methods to the SBRP formulation. Numerical experiments are reported in Section 6.

4.2.1. 3-Piecewise stabilization

In PTS methods, fluctuations in the dual solutions are suppressed by limiting the Euclidean distance between consecutive dual solutions. In general terms, the idea can be described as follows. First a good dual solution is estimated, around which a hypercube is centered, thereby marking a trust area in which dual points generated by the RMP must lie. Penalties are incurred when the points fall outside the allotted trust region. Next, the stability center is readjusted whenever a better estimate of the optimal dual solution is available. Similarly, the hypercube can be resized and the penalty functions adjusted.

In this work, we will focus on a specific PTS approach proposed by Merle et al. (1997): 3-piecewise stabilization. To obtain a 3-piecewise stabilized CG formulation, we have added stabilization

variables y^- and y^+ to constraints (2), (3), and the objective function. The resulting stabilized formulation becomes

$$\min \quad \sum_{p\in P}\delta_p z_p - \sum_{s\in S}\delta_s^- y_s^- + \sum_{s\in S}\delta_s^+ y_s^+ - \sum_{v\in V}\delta_v^- y_v^- + \sum_{v\in V}\delta_v^+ y_v^+ \tag{12}$$

$$\text{s.t.} \quad \sum_{p\in P} t_{sp} z_p - y_s^- + y_s^+ \geq 1 \quad \forall s \in S \tag{13}$$

$$\sum_{p\in P} r_{vp} z_p + y_v^- - y_v^+ \leq 1 \quad \forall v \in V \tag{14}$$

$$L \leq \sum_{p\in P} z_p \leq U \tag{15}$$

$$y_i^- \leq \epsilon_i \qquad \forall i \in V \cup S \tag{16}$$

$$y_i^+ \leq \epsilon_i \qquad \forall i \in V \cup S \tag{17}$$

$$z_p, y_i^+, y_i^- \geq 0 \qquad \forall p \in P, i \in V \cup S, \tag{18}$$

where δ^+, δ^-, ϵ are predefined positive parameters. Note that when $\delta^+ = \delta^- = \epsilon = 0$, the original LPM (Section 3) appears. Writing down the dual gives us

$$\max \sum_{s\in S}(u_s - \epsilon_s t_s^- - \epsilon_s t_s^+) - \sum_{v\in V}(w_v + \epsilon_v t_v^- + \epsilon_v t_v^+) - Um + Ln \tag{19}$$

$$\text{s.t.} - \sum_{v\in V} r_{vp} w_v + \sum_{s\in S} t_{sp} u_s - m + n \leq \delta_p \quad \forall p \in P \tag{20}$$

$$-u_s - t_s^- \leq -\delta_s^- \qquad \forall s \in S \tag{21}$$

$$u_s - t_s^+ \leq \delta_s^+ \qquad \forall s \in S \tag{22}$$

$$-w_v - t_v^- \leq -\delta_v^- \qquad \forall v \in V \tag{23}$$

$$w_v - t_v^+ \leq \delta_v^+ \qquad \forall v \in V \tag{24}$$

$$t^+, t^-, u, w \geq 0, \tag{25}$$

where u, w, m, n, t^+, and t^- are the dual variables associated with constraints (13)–(17).

As can be observed from the dual formulation, the addition of stabilization parameters restricts the domains of the dual variables u_s, w_v to $\delta_s^- \leq u_s \leq \delta_s^+, s \in S$, resp. $\delta_v^- \leq w_v \leq \delta_v^+, v \in V$; deviation of these intervals is penalized by $\epsilon_i t_i$, $i \in V \cup S$ in the dual objective. Note that the stabilized formulation does not change the pricing problem (Equation (8)). However, an optimal solution to the stabilized LPM is only obtained if there are no more columns with negative reduced cost, and $y_i^- = y_i^+ = 0, \forall i \in V \cup S$. Several different update procedures for δ and ϵ are proposed in Merle et al. (1997) and Oukil et al. (2007). However, during our experiments, we obtained the best results with the following update procedure. At iteration $j+1$ set:

$$\delta_s^- := u_s^j - \delta_s \quad \forall s \in S \tag{26}$$

$$\delta_s^+ := u_s^j + \delta_s \quad \forall s \in S \tag{27}$$

$$\delta_v^- := w_v^j - \delta_v \quad \forall v \in V \tag{28}$$

$$\delta_v^+ := w_v^j + \delta_v \quad \forall v \in V, \tag{29}$$

where u_s^j resp. w_v^j are the dual values, of variables u_s, w_v, obtained at iteration j. At iteration $j = 0$, we use the dual values obtained from Equations (2) resp. (3). Initially, we set $\delta_i = 0.5$, $\epsilon_i = 0.1$, $\forall i \in S \cup V$. Whenever, at iteration j, the pricing problem does not find any more negative reduced cost columns, we update the values of δ and ϵ as follows:

$$\epsilon_i := \frac{\epsilon_i}{2} \qquad \forall i \in U \cup V \tag{30}$$

$$\delta_i := \begin{cases} \max\{0.1, \frac{1}{\delta_i}\} & \text{if } \delta_i^- \leq u_s^j(w_v^j) \leq \delta_i^+ \\ \min\{1, 2\delta_i\} & \text{otherwise} \end{cases} \qquad \forall i \in U \cup V. \tag{31}$$

Since ϵ is decreased at every update cycle, the y variables gradually dissipate.

The above update procedure yielded significantly better results than a procedure that updates δ and ϵ during every CG iteration.

4.2.2. IPS

A major disadvantage of PTS methods is the extensive number of parameters that have to be tuned. To our knowledge, there is no consensus on how to select the initial parameters or how to update them. Furthermore, not every choice of parameters works well for each instance.

A different stabilization approach, which circumvents the issue of parameter selection, is IPS (Rousseau et al., 2007). IPS attempts to achieve a better dual value distribution by generating a dual solution of the RMP that is an interior point of the optimal dual face rather than an extreme point. The latter is achieved by taking a convex combination of several different optimal dual solutions. Experiments have shown that this approach decreases the amount of columns needed to prove optimality. However, the main argument against the use of IPS is that it requires the RMP to be solved multiple times during each CG iteration to obtain the interior points. Nevertheless, on some occasions, better results are reported compared to proximal approaches, for example, in Dell'Amico et al. (2006).

Let P^* be the set of columns for which $z_p > 0$, that is, a subset of columns that are in the basis. Furthermore, let V^* and S^* be the sets of students resp. stops for which constraints (3) resp. (2) are *not* tight. By complementary slackness conditions, the optimal face of the dual polyhedron (D^*) corresponding to Equations (1)–(5) is given by the following constraints (Rousseau et al., 2007):

$$\sum_{s\in S} t_{sp}u_s - \sum_{v\in V} r_{vp}w_v - m + n \leq \delta_p \quad \forall p \in P\backslash P^* \tag{32}$$

$$\sum_{s\in S} t_{sp}u_s - \sum_{v\in V} r_{vp}w_v - m + n = \delta_p \quad \forall p \in P^* \tag{33}$$

$$u_s = 0 \qquad \forall s \in S^* \tag{34}$$

$$u_s \geq 0 \qquad \forall s \in S\backslash S^* \tag{35}$$

$$w_v = 0 \qquad \forall v \in V^* \tag{36}$$

$$w_v \geq 0 \qquad \forall v \in V\backslash V^* \tag{37}$$

$$m = 0 \qquad \text{if Equation (4) is tight} \tag{38}$$

$$m \geq 0 \qquad \text{otherwise} \tag{39}$$

$$n = 0 \qquad \text{if Equation (5) is tight} \tag{40}$$

$$n \geq 0 \qquad \text{otherwise.} \tag{41}$$

An extreme point of this polyhedron can be obtained by using an arbitrary objective function

$$\max \sum_{s\in S} \mu_s u_s - \sum_{v\in V} \mu_v w_v - Um + Ln. \tag{42}$$

Here, the vector μ is randomly generated with each term uniformly chosen from the interval [0, 1]. Let D^μ denote the complete LP (Equations (32)–(42)). The dual of D^μ, P^μ becomes

$$\min \quad \sum_{p \in P} \delta_p z_p \tag{43}$$

$$\text{s.t.} \quad \sum_{p \in P} r_{vp} z_p \leq \mu_v \quad \forall v \in V \backslash V^* \tag{44}$$

$$\sum_{p \in P} t_{sp} z_p \geq \mu_s \quad \forall s \in S \backslash S^* \tag{45}$$

$$L \leq \sum_{p \in P} z_p \leq U \tag{46}$$

$$z_p \geq 0 \qquad \forall p \in P \backslash P^* \tag{47}$$

$$z_p \in \mathcal{R} \qquad \forall p \in P^*. \tag{48}$$

When solving P^μ for different μ, several extreme points are obtained. To obtain distant points, when we solve P^μ we also solve $P^{-\mu}$.

Experiments revealed that for many choices of μ, P^μ is infeasible, implying that D^μ is unbounded. The latter can be attributed to Equation (44); when compared to Equation (3) in the original formulation, Equation (44) is stronger when $\mu_v < 1$. Fixing $\mu_v = 1 \; \forall v \in V$ resolved the issue while still generating sufficiently distinct extreme points. We would like to point out that whenever D^μ is unbounded, it would be possible to generate useful points by taking any feasible extreme point and determining the recession direction. The result is a ray that can in turn be used to compute an interior point of D^*.

4.3. Column pool manager

During each iteration, the pricing problem returns one or more columns, thereby increasing the size of the MP. When the total number of columns in the RMP becomes too large, evaluating the MP becomes a time-consuming process. A CPM is used to reduce the number of *active* columns in the RMP (see Barnhart et al., 1998).

The CPM associates a timer with each column in the MP. in every CG iteration, the timers are increased by one. The timer of a column $p \in P$ is set to zero when the column is part of the basis, that is, when a nonzero value is associated with z_p. Once the number of active columns exceeds a certain threshold (2000 in our implementation), the patterns remained unused for the longest are removed from the MP and stored in a pool. In each iteration, before the pricing problem is invoked, the pool is queried to determine whether certain columns need to be moved back into the MP. The latter is necessary to prevent invoking an expensive pricing problem, which could yield the same columns. When the lookup procedure returns columns, the

pricing problem is skipped. Note that the initial columns are never removed from the active set of columns because they ensure feasibility that could otherwise not be assured in a stabilized MP.

To keep the pool size reasonable, similar to the active columns, a timer is associated with the columns in the pool. Once the size of the pool exceeds 3000 columns, the oldest columns are permanently deleted.

In practice, CPM is a powerful mechanism to limit the number of active columns. Obviously, it is desirable to choose the number of allowed columns in the active set as low as possible. However, when this number is set too low, patterns are frequently swapped in and out of the pool. When a column re-enters the active set via a lookup operation, the pricing problem is skipped and hence no new columns are generated. This reduces the amount of fresh information introduced to the MP. As a consequence, when too many lookups are performed, the convergence of the CG algorithm stalls. To counter this swapping behavior, the CPM keeps track of the number of successful lookups in a given time window. When this number exceeds a threshold, the number of allowed active columns is increased.

5. Branch and price

Since the integrality constraints of the MP are relaxed in the LPM, it is possible that the optimal solution to the LPM is fractional. A branch-and-price framework is needed to obtain integer solutions. In this section, the necessary branching rules are described, as well as bounds used to prune parts of the branch tree.

5.1. Branching rules

Our branching strategy is based on the following lemma.

Lemma 1. *Suppose that $z = \{z_1, z_2, \ldots, z_k\}$, is a feasible solution to LPM (Section 3), is fractional, and has cost $c(z)$. Then, either an integral solution exists with cost $c(z)$, or there exists an edge $e \in V \times V$ such that:*

$$0 < \sum_{p \in P: e \in p} z_p < 1.$$

Proof. We will show that if $\sum_{p \in P: e \in p} z_p \in \{0, 1\}$ for each edge $e \in V \times V$, then an integral solution with cost $c(z)$ exists, thereby proving the lemma. Recall that a bus schedule $p \in P$ specifies a set of stops, as well as, for each of these stops, a set of students. Consider two schedules i and j that have a positive value in our current solution (i.e. $0 < z_i, z_j < 1$), and that both visit a specific pair of stops consecutively (such a pair of schedules must exist, otherwise z is not fractional).

Consider the set of stops in schedule i (say stop set S_i) and the set of stops in schedule j (say stop-set S_j). Suppose S_i and S_j do not coincide, that is, suppose $S_i \neq S_j$. Then stops $t, u, v, w \in V$ exist s.t. (t, u) is a route segment of both schedule i and j, (u, v) is a route segment of schedule i,

and (u, w) is a route segment of schedule j. Recall that we assumed that for each edge $e \in V \times V$, $\sum_{p \in P: e \in p} z_p \in \{0, 1\}$ holds. Hence, for each route segment $e \in \{(t, u), (u, v), (u, w)\}$, we must have that $\sum_{p \in P: e \in p} z_p = 1$, which, however, is in violation to constraint (3) for node $u \in V$. It follows that the set of stops visited by schedule i and schedule j coincide. This implies that the fractional solution $z = \{z_1, z_2, \ldots, z_k\}$ consists of schedules whose stop-sets are either identical or disjoint. In other words, we can identify a partition of $\{1, \ldots, k\}$ into α subsets such that subset $K_j \subseteq \{1, \ldots, k\}$ contains schedules with an identical stop-set, $j = 1, \ldots, \alpha$. Now, define for each $s \in S$, $j = 1, \ldots, \alpha$ the *presence* of student s in subset K_j as

$$\sum_{\substack{l \in K_j \\ l \text{ contains } s}} z_l = \beta_{js}.$$

Since z is a feasible solution to LPM, and in particular satisfies constraint (2), we know that the β_{js} satisfy

$$\sum_{j=1}^{\alpha} \beta_{js} = 1 \quad \forall s \in S \tag{49}$$

$$\sum_{s \in S} \beta_{js} \leq Q \quad \forall j = 1, \ldots, \alpha. \tag{50}$$

Due to integrality of Q, it is well-known that an integral solution to this system must exist, thereby ensuring that a student is either present completely in some subset K_j, or not at all. The existence of an integral solution z follows with cost $c(z)$. □

In the following, we describe the two branching rules that we use. Branching rules are used to cut off the current fractional solution, and to partition the remaining solution space Z_u into two dichotomous partitions. After a finite number of branchings, either an optimal integral solution is obtained, or Z_u is exhausted, proving that no feasible integral solution exists. Note that the above lemma guarantees that the presence of a fractional solution either gives us a way to find an integral solution, or we can identify a fractional edge on which we can branch. The latter observation is used for the second branching rule described below.

Given a fractional solution $z = \{z_1, z_2, \ldots, z_k\}$, where z_p, $p = 1, \ldots, k$ is the variable associated with bus schedule $p \in P$. Let $P' \subseteq P$ be the set of fractional bus schedules, that is, $P' = \{p \in P | 0 < z_p < 1\}$. If $b = \sum_{p \in P'} z_p$ is fractional, we can branch on the number of buses b, thereby creating two branches: one where the number of buses used is at least $\lceil b \rceil$ and one with at most $\lfloor b \rfloor$ buses. Implementing this rule is straightforward as it simply amounts to updating U and L. Furthermore, all columns generated at the parent node can be reused at the child nodes.

The first branching rule is not sufficient to guarantee an integral solution. Therefore, as a second branching rule, we *branch on an edge*: two stops are visited consecutively in a single route or not. Enforcing that stop $v_i \in V$ is not reachable from $v_j \in V$ and *vice versa* is achieved by removing the edge (v_i, v_j) from the underlying graph. Ensuring that the two stops occur consecutively in a route

requires some more effort. To accomplish this, we choose to modify the pricing problems. In case of the heuristic pricing method, the branching rules are easily accommodated in the neighborhood definitions. Similarly, in the labeling algorithm, a route cannot be extended if the resulting route violates the branching constraints.

Strictly speaking, the second branching rule renders the first rule redundant. However, computational experiments revealed that the use of the first branching rule has a positive effect on the size of the tree.

In contrast to branching on the number of buses, when branching on an edge, it is not possible to reuse all columns from the parent node. For each column generated in the parent node, one needs to check whether the column is in accordance with the newly created branching rule. Depending on the branching rule, the column can either be modified to meet the new branching constraints, or has to be removed altogether.

In our implementation, the branching rules are always employed in the order of description. When the branching framework branches on an edge, the most uncertain edge is selected, that is, $\text{argmin}_{e \in E}(|\sum_{p:e \in p} z_p - 0.5|)$. In case of ties, the edge that occurs in most patterns is used. For alternative, more sophisticated approaches to select a branching candidate, for example, strong branching or branch decisions based on pseudocosts, we refer the interested reader to the works of Martin (1999).

A final aspect of interest is the order in which branches are investigated. The branch-and-price tree is explored in a DFS manner, always starting with the most constrained branch, for example, when branching on the number of buses, the tightest bound is expanded first. In case of the remaining two branching rules, the algorithm always starts by investigating the branch which enforces the use of an edge (vertex). The choice for DFS is based on the observation by Martin (1999) that feasible solutions tend to lie deep in the branch tree.

5.2. *Pattern initialization*

At each node of the branch-and-price tree, the MP has to be initialized with a feasible set of columns. Alternatively, when such a set does not exist, one needs to prove infeasibility of the MP. The root node can be straightforwardly initialized by any feasible solution to the SBRP, but finding an initial feasible solution for any of its siblings tends to be difficult. Moreover, proving that such a solution does not exist is a cumbersome task. Hence it would be beneficial when the MP itself could be used to prove nonexistence of a feasible solution. With this goal in mind, artificial columns, which together meet all the constraints of the MP (constraints (3)–(6)), are introduced. Each artificial column has a cost strictly larger than the cost of the longest feasible route; hence, the MP favors cheaper nonartificial columns over the artificial ones. When the CG procedure terminates, and the resulting solution still contains artificial columns, one can indeed conclude that no feasible solution exists. To generate the artificial columns, first L stops are selected which are used to generate simple school-stop-school paths (recall that L is the lower bound on the required number of buses). Next, each student is assigned to one of those paths, independent of whether the student can reach the path, as the latter is not imposed by any of the constraints in the MP. Similarly, the bus capacity requirements can be ignored. Finally, a large cost is assigned to the artificial patterns. In our implementation, it usually only takes a few iterations before the artificial columns leave the basis of the RMP.

5.3. Bounds

The efficiency of a branch-and-price framework is determined by two factors: the size of the tree and the processing time required by individual nodes. As discussed in Section 4.2, the processing time of a node is strongly affected by the tailing-off effect, which, in a branch-and-price framework, could occur at every node. Consequently, to speed up the processing times and to reduce the size of the search tree, bounds are used. Let Z_{MP} denote the value of the optimal value of the MP (Equations (1)–(6)) and UB an upper bound on Z_{MP}. Further, for some node u in the tree, let Z^u_{LMP} denote the optimal solution of the *relaxed* MP at node u, $LB1^u$ a lower bound on Z^u_{LMP}, and Z^u_{RMP} the solution to the RMP at node u at any given iteration. By definition, the following relations must hold: $UB \geq Z_{MP}$ and $Z^u_{RMP} \geq Z^u_{LMP} \geq LB1^u$. Computations at a node u can be terminated as soon as the gap between the lower bound at node u and the upper bound is closed, that is, $LB1^u = UB$, independent of whether there still exist columns with a negative reduced cost. Additionally, a node is pruned when its lower bound exceeds the upper bound UB. Consequently, the availability of strong bounds has a significant impact on the efficiency of our framework. Any feasible integral solution to the MP discovered at some node u is an acceptable upper bound on Z_{MP}. For SBRP, an upper bound UB is readily available at the root node because it is initialized with a feasible solution. In a branch-and-price tree, the root node is by definition the least constrained; each subsequent branching will add extra constraints to the model. Hence, the optimal objective value of a parent node is always lower or equal to the objective of any of its siblings (in case of minimization problems), and hence the optimal objective value of a node serves as a lower bound to its siblings. The lower bound inherited from a parent serves as an initial lower bound for its siblings. This lower bound can be tightened during the processing of the node using the following equation (Lasdon, 1970):

$$LB1^u = Z^u_{RMP} + Uq^*. \tag{51}$$

Here, q^* is the optimal solution to the pricing problem as defined in Section 4 and U the upper bound on the number of buses (Table 1).

A second, stronger lower bound $LB2^u$ can be deduced via Langrangian relaxation as demonstrated by Vanderbeck (1994). Given the original MP LPM as presented in Section 4, we can relax complicating constraints (2) and (3) using the Lagrangian multipliers u_s and w_v, respectively, and drop the integrality constraints. The resulting problem becomes

$$Z^u_{LMP} = \min \sum_{p\in P} \delta_p z_p + \sum_{s\in S} u_s \left(1 - \sum_{p\in P} t_{sp} z_p\right) - \sum_{v\in V} w_v \left(1 - \sum_{p\in P} r_{vp} z_p\right) \tag{52}$$

$$\text{s.t. } L \leq \sum_{p\in P} z_p \leq U$$

$$= \sum_{s\in S} u_s - \sum_{v\in V} w_v + \min \sum_{p\in P} \left[\delta_p - \sum_{s\in S} t_{sp} u_s + \sum_{v\in V} r_{vp} w_v\right] z_p \tag{53}$$

$$\text{s.t. } L \leq \sum_{p\in P} z_p \leq U$$

$$\geq Z^u_{RMP} + Um - Ln + \min \sum_{p \in P} [q_p - m + n] z_p \tag{54}$$

$$\text{s.t. } L \leq \sum_{p \in P} z_p \leq U$$

$$\geq Z^u_{RMP} + Um - Ln + \min\{L(q^* - m + n), U(q^* - m + n)\} \tag{55}$$

$$= Z^u_{RMP} + Uq^* + (U - L)n \equiv LB2^u. \tag{56}$$

Equation (54) follows from the weak duality theorem and the fact that optimal Langrangian multipliers equal the dual variables of the dual problem (Wolsey and Nemhauser, 1988).

When compared, the second lower bound $LB2^u$ (Equation (56)) is stronger than the first bound $LB1^u$ because $U - L \geq 0$ and $n \geq 0$.

For both bounds, it holds that the stronger the bounds L and U on the number of buses are, the tighter the lower bound will be. The latter also motivates the use of a branching rule that branches on the number of buses. When solving LPM (Section 4), Z^u_{RMP} decreases monotonically toward its optimal value Z^u_{LMP}. The lower bound $LB2^u$ also converges toward Z^u_{LMP} but not monotonically as $q*$ does not change monotonically at each successive iteration. As a consequence, the computed lower bound at the latest iteration might be lower (less tight) than a lower bound computed at an earlier iteration. It is therefore important to store the tightest lower bound.

A clear downside to calculating the lower bounds is the need for the optimal value of the pricing problem q^*. As elaborated in Section 4.1, the pricing problem is preferably solved by a fast (local search) heuristic, and not by the more expensive exact algorithm. The latter is worsened by the fact that one might need to compute the lower bound at several iterations to obtain a tight bound on Z^u_{LMP}. Vanderbeck (1994) points out that instead of the optimal value of the pricing problem, a lower bound could also be used, but this usually leads to weaker bounds.

5.4. Branch and price implementation

This section briefly discusses the course of the branch-and-price algorithm. At each node of the branch-and-price tree, a MP is solved. The lower bound on the node is updated whenever the pricing problem is solved to optimality, which we enforce at least once every 100 iterations (Section 4.2). A node is pruned whenever the lower bound exceeds the upper bound. Once the node's MP is solved to optimality, a branching decision is made. The following three possible scenarios exist:

(1) The solution is fractional (and feasible). A branch is created.
(2) The solution is an integral solution. The upper bound on the global optimum is updated when possible.
(3) The solution contains an artificial column. The solution is infeasible and the node is pruned.

As discussed in Section 4.2, the occurrence of the tailing-off effect has a large impact on the overall convergence speed, especially if this effect occurs at every node of the branch-and-price tree. In an attempt to counter this behavior, tailing-off is detected by measuring the relative difference (improvement) in the objective value of the MP over a number of iterations. If the measured difference is less than a constant (0.1) during 20 iterations, we conclude that tailing-off occurs. Then a lower bound on the solution is computed and a branching decision is made. Note however that if the solution is integral or contains an artificial column, we cannot straightforwardly terminate computations. In such a case, we store the state of the algorithm, including all the generated columns. Solving the node to optimality is then postponed until all remaining nodes have been evaluated. The advantage of this approach is that when a new (better) integral solution is discovered in the meantime, it might be possible to prune the node based on the improved bounds, which potentially saves time.

Finally, when a new incumbent optimal solution is discovered, all nodes are pruned, and a new branch-and-price tree is grown starting from the latter solution and the columns already generated for this node. The philosophy behind this pruning decision is again based on the assumption that a faster converge of a branch-and-price algorithm is achieved with better initial solutions. In particular, this approach works well when the regular bound-based pruning is ineffective due to a relatively large gap between the optimal solutions to the MP and LPM, respectively.

6. Computational experiments

One aspect of instances of SBRP is that the number of students is typically much larger than the number of stops. This feature is not present in instances of the MV-TPP that have been used for computational testing (see Riera-Ledesma and Salazar-González, 2012); therefore, we used instances from Schittekat et al. (2013), and we generated new instances to focus on this property. Computational experiments are conducted on two data sets. The first data set (data set I) contains 94 of the original 112 SBRP instances[1] that have previously been used by Schittekat et al. (2013). The easiest instance has 5 stops and 25 students, whereas the hardest instances have 40 stops and 800 students. Each instance describes the distances between the stops, the maximum bus capacity and the maximum walking distance. The set of reachable stops for each student follows from its location and the allowed walking distance; the latter distance controls the average number of stops a student is able to reach. Routing occurs in the Euclidean plane. The second, newly generated data set (data set II) contains 32 instances. The problem sizes in this class range from 20 to 35 stops, and up to a maximum of 450 students. These instances are generated randomly as follows. For a given number of stops, a complete graph is created with distances on the edges uniformly selected from the interval [1, 21]. Similarly, the stops a student can reach are randomly selected such that the average number of reachable stops equals a predefined ratio. Note that the distances in this class may violate the triangle inequality. The exact instance characteristics for both sets are summarized in Tables 2 and 3: the instance ID (*ID*), the number of stops (*stop*), number of students (*stud*), bus capacity (*cap*), and the average number of stops in the vicinity of a student (v/s), that is, $\frac{\sum_s |V_s|}{|S|}$. For

[1]The instances are available online at http://antor.ua.ac.be/schoolbus-routing.

Table 2
Computational results for data set I

ID	*stop*	*stud*	*cap*	*v/s*	LB_{old}	LB_{new}	*MH*	*B&P*	*gap*	*nodes*	*t* (ms)	$t_{pricing}$ (ms)
1	5	25	25	1	141.01	141.01	141.01	141.01	0	1	1778	136
2	5	25	50	1	161.62	161.62	161.62	161.62	0	2	104	47
3	5	25	25	1.52	182.14	182.14	182.14	182.14	0	3	195	106
4	5	25	50	1	195.8	195.8	195.80	195.8	0	4	86	34
5	5	25	25	1.88	111.65	111.65	111.65	111.65	0	5	156	72
6	5	25	50	2.4	103.18	103.18	103.18	103.18	0	6	97	38
7	5	25	25	4.88	7.63	7.63	7.63	7.63	0	7	41	5
8	5	25	50	4.56	25.64	25.64	25.64	25.64	0	8	88	27
9	5	50	25	1	281.49	**286.68**	286.68	286.68	0	11	183	26
10	5	50	50	1.4	197.2	197.2	197.20	197.2	0	12	99	28
11	5	50	25	1.46	181.02	**193.55**	193.55	193.55	0	29	3035	493
12	5	50	50	1.4	215.86	215.86	215.86	215.86	0	30	72	21
13	5	50	25	1.96	130.53	130.53	130.53	130.53	0	31	245	61
14	5	50	50	2.74	96.26	96.26	96.26	96.26	0	32	110	40
15	5	50	25	4.8	12.89	12.89	12.89	12.89	0	33	260	65
16	5	50	50	4.42	30.24	30.24	30.24	30.24	0	34	145	52
17	5	100	25	1	360.35	360.35	360.35	360.35	0	35	94	1
18	5	100	50	1	290.67	**304.23**	304.23	304.23	0	38	111	12
19	5	100	25	1.55	255.93	**294.21**	294.21	294.21	0	41	1869	151
20	5	100	50	1.37	229.41	229.41	229.41	229.41	0	43	1627	226
21	5	100	25	2.8	134.95	134.95	134.95	134.95	0	44	535	84
22	5	100	50	1.84	139.87	**144.41**	144.41	144.41	0	47	12,974	958
23	5	100	25	4.77	58.95	58.95	58.95	58.95	0	48	1388	215
24	5	100	50	4.58	39.44	39.44	39.44	39.44	0	49	791	287
25	10	50	25	1.22	242.85	242.85	242.85	242.85	0	50	471	297
26	10	50	50	1.2	282.12	282.12	282.12	282.12	0	51	323	178
27	10	50	25	1.6	244.54	244.54	244.54	244.54	0	52	809	450
28	10	50	50	1.26	288.33	288.33	288.33	288.33	0	53	632	444
29	10	50	25	3.96	108.95	108.98	108.98	108.98	0	57	5680	2279
30	10	50	50	2.86	157.48	157.48	157.48	157.48	0	58	407	310
31	10	50	25	9.2	32.25	32.25	32.25	32.25	0	59	430	292
32	10	50	50	8.96	36.66	36.66	36.66	36.66	0	60	260	183
33	10	100	25	1.18	400.54	**403.18**	403.18	403.18	0	65	834	246
34	10	100	50	1.29	294.11	**296.53**	296.53	296.53	0	68	5398	2244
35	10	100	25	1.25	369.62	**388.87**	388.87	388.87	0	190	20,578	5704
36	10	100	50	1.33	294.8	294.8	294.80	294.8	0	192	3030	1640
37	10	100	25	3.98	178.28	178.28	178.28	178.28	0	194	29,681	5027
38	10	100	50	3.26	175.41	**175.96**	175.96	175.96	0	199	131,052	30,255
39	10	100	25	9.18	57.5	57.5	57.50	57.5	0	200	1819	1069
40	10	100	50	9.22	31.89	31.89	31.89	31.89	0	201	1522	941
41	10	200	25	1	735.27	735.27	735.27	735.27	0	202	21	1
42	10	200	50	1	506.06	506.06	512.16	**506.06**	0	204	73	14
43	10	200	25	1.85	463.78	**513**	513.00	513	0	261	124,693	9716
44	10	200	50	1.28	458.17	**475.21**	475.21	475.21	0	371	69,692	10,891
45	10	200	25	3.53	331.49	**347.29**	347.29	347.29	0	374	30,215	4061
46	10	200	50	3.66	194.66	194.66	217.46	217.46	11.71	136	3,600,000	155,777
47	10	200	25	9.22	102.93	102.93	102.93	102.93	0	137	5890	2262

Continued

Table 2
Continued

ID	*stop*	*stud*	*cap*	*v/s*	LB_{old}	LB_{new}	*MH*	*B&P*	*gap*	*nodes*	*t* (ms)	$t_{pricing}$ (ms)
48	10	200	50	8.93	55.05	55.05	55.05	55.05	0	138	8201	3694
49	20	100	25	1.15	507.81	507.81	520.24	**507.81**	0	140	6002	5544
50	20	100	50	1.17	406.65	406.65	420.64	**406.65**	0	142	2,920,371	2,918,926
51	20	100	25	2.08	404.78	**419.17**	422.21	**419.17**	0	211	708,642	478,578
52	20	100	50	1.73	356.52	356.52	360.86	360.86	1.22	456	3,600,000	3,560,533
53	20	100	25	5.09	245.17	245.17	245.17	245.17	0	458	71,459	37,360
54	20	100	50	6.13	181.3	181.3	185.06	185.06	2.08	462	3,600,000	848,080
55	20	100	25	17.36	52.52	52.52	52.52	52.52	0	463	16,494	13,825
56	20	100	50	18.46	19.05	19.05	19.05	19.05	0	464	2666	2334
57	20	200	25	1.2	851.98	**875.46**	903.84	**875.46**	0	734	1,564,365	99,575
58	20	200	50	1.15	473.89	**476.05**	485.65	**476.05**	0	766	472,158	317,436
59	20	200	25	1.76	589.89	**597.99**	616.93	**606.8**	1.47	615	3,600,000	577,259
60	20	200	50	1.78	451.09	451.09	462.31	462.31	2.49	809	3,600,000	819,042
61	20	200	25	5.38	366.1	366.1	373.21	373.21	1.94	1062	3,600,000	437,484
62	20	200	50	5.53	246.49	246.49	250.75	250.75	1.73	1389	3,600,000	625,320
63	20	200	25	18	93.01	93.01	93.01	93.01	0	1391	62,586	42,405
64	20	200	50	17.97	45.4	45.4	45.40	45.4	0	1392	28,623	18,384
65	20	400	25	1.46	1247.65	**1323.35**	1323.35	1323.35	0	1535	1,446,744	76,934
66	20	400	50	1.22	709.87	**720.83**	733.54	**720.83**	0	1663	1,058,957	165,084
67	20	400	25	2.48	911.06	911.06	975.12	975.12	7.03	1248	3,600,000	134,663
68	20	400	50	1.89	599.12	599.12	614.67	614.67	2.6	1867	3,600,000	260,149
69	20	400	25	4.58	756.04	**763.76**	763.76	763.76	0	1903	3,111,861	136,752
70	20	400	50	7.45	298.05	298.05	298.47	298.47	0.14	1915	3,600,000	384,688
71	20	400	25	18.15	239.58	239.58	239.58	239.58	0	1916	92,356	41,734
72	20	400	50	18.13	84.49	84.49	84.49	84.49	0	1917	190,202	62,401
73	40	200	25	1.4	787.14	787.14	831.94	831.94	5.69	1936	3,600,000	3,527,195
74	40	200	50	1.38		**549.64**	593.35	593.35	7.95	1463	3,600,000	3,574,732
75	40	200	25	2.57	696.04	696.04	728.44	728.44	4.65	1583	3,600,000	921,613
76	40	200	50	2.7		**474.14**	481.05	481.05	1.46	1584	3,600,000	3,543,283
77	40	200	25	9.57	328.19	328.19	339.75	339.75	3.52	59	3,600,000	2,415,227
78	40	200	50	10.72	273.05	273.05	273.88	273.88	0.31	60	3,600,000	3,568,533
79	40	200	25	36.42	76.77	76.77	76.77	76.77	0	2028	137,932	129,914
80	40	200	50	36.3	58.46	58.46	58.46	58.46	0	2029	330,240	322,539
81	40	400	25	1.51	1307.52	1307.52	1407.05	**1394.23**	6.63	2289	3,600,000	605,095
82	40	400	50	1.3		**820.52**	858.80	858.8	4.67	204	3,600,000	3,541,207
83	40	400	25	3.23	869.38	869.38	891.02	891.02	2.49	2341	3,600,000	671,259
84	40	400	50	2.57		**721.75**	757.42	757.42	4.94	247	3,600,000	779,834
85	40	400	25	10.02	575.66	575.66	586.29	586.29	1.85	294	3,600,000	789,361
86	40	400	50	9.88		**392.06**	395.95	395.95	0.99	2400	3,600,000	2,033,063
87	40	400	25	35.74	195.33	195.33	195.33	195.33	0	2401	333,985	304,871
88	40	400	50	36.59	70.77	70.77	70.77	70.77	0	2402	449,242	415,835
89	40	800	25	1.33	2801.05	2801.05	2900.14	2900.14	3.54	166	3,600,000	467,658
90	40	800	50	1.54	1280.51	1280.51	1345.70	1345.7	5.09	258	3,600,000	704,129
91	40	800	25	2.71	2153.76	2153.76	2200.57	2200.57	2.17	286	3,600,000	494,468
92	40	800	50	3.08	978.88	**978.9**	1025.16	1025.16	4.72	179	3,600,000	431,057
93	40	800	25	9.63	1404.16	1404.16	1404.16	1404.16	0	303	542,641	204,234
94	40	800	50	10.92	613.72	613.72	616.58	616.58	0.47	1	3,600,000	3,467,511
95	40	800	25	36.11		**396.92**	396.92	396.92	0	305	790,502	701,400

Table 3
Computational results for data set II

ID	*stop*	*stud*	*cap*	*v/s*	LB_{new}	*MH*	*B&P*	*gap*	*nodes*	*t* (ms)	$t_{pricing}$ (ms)
1	20	100	15	4.02	73	73	73	0	520	17,306	13,925
2	20	100	20	4.02	60	60	60	0	33	315,377	111,698
3	20	100	25	4.02	55	55	55	0	42	84,882	46,638
4	20	100	15	7.72	71	71	71	0	522	13,248	10,472
5	20	100	20	7.72	49	49	49	0	35	34,049	23,610
6	20	100	25	7.72	42	42	42	0	44	33,028	25,316
7	20	200	15	4.16	238	238	238	0	523	3758	2133
8	20	200	25	4.16	102	104	102	0	47	969,466	226,204
9	20	200	15	8.52	224	224	224	0	1	11,433	4487
10	20	200	25	8.52	86	86	86	0	48	15,689	10,673
11	25	150	15	4.13	115.2	120	120	4.17	464	3,600,000	974,827
12	25	150	25	4.13	72.47	81	**74**	2.12	203	3,600,000	1,113,312
13	25	150	15	8.13	111	115	**111**	0	466	164,844	148,875
14	25	150	25	8.13	60	62	**60**	0	260	1,838,698	1,140,206
15	25	250	15	4.14	277	277	277	0	467	12,768	7818
16	25	250	25	4.14	126.88	132	**131**	3.25	304	3,600,000	1,276,481
17	25	250	15	8.41	275	275	275	0	468	23,223	18,311
18	25	250	25	8.41	111	120	**111**	0	326	1,068,039	775,015
19	30	300	15	3.99	326	326	326	0	469	55,441	38,709
20	30	300	25	3.99	151.89	172	**156**	2.71	348	3,600,000	3,226,403
21	30	300	15	8.48	315	317	**315**	0	473	175,157	130,120
22	30	300	25	8.48	123	134	**123**	0	356	2,278,841	2,012,475
23	30	350	15	3.91	439	439	439	0	474	19,185	12,401
24	30	350	25	3.91	186.14	200	**188**	1	401	3,600,000	1,797,273
25	30	350	15	8.39	439	440	**439**	0	478	92,573	71,374
26	30	350	25	8.39	162	169	**162**	0	404	115,4063	924,603
27	35	400	15	3.92	521	521	521	0	479	49,677	35,519
28	35	400	25	3.92	233.07	244	**237**	1.69	429	3,600,000	2,282,498
29	35	400	15	8.63	498	498	498	0	480	67,755	53,351
30	35	400	25	8.63	206	217	**207**	0.49	451	3,600,000	3,026,446
31	35	450	15	3.93	613	613	613	0	481	33,458	17,080
32	35	450	25	3.93	277.85	290	**282**	1.49	509	3,600,000	639,639
33	35	450	15	8.63	608	608	608	0	482	78,742	51,914
34	35	450	25	8.63	256.14	260	**257**	0.33	516	3,600,000	3,160,984

instance, we performed a number of experiments, whose results are reported in Tables 2 and 3 and are compared against the results published in Schittekat et al. (2013).

- LB_{old}: The best bound published in Schittekat et al. (2013).
- LB_{new}: The lower bound on an optimal solution obtained while solving the branch-and-price tree. Bold face entries indicate an improvement compared to LB_{old}.
- MH: The best solution obtained by the metaheuristic as published in Schittekat et al. (2013).
- *B&P*: The branch-and-price solution. This column reports the best integer solutions that have been found within a time limit of one hour (3,600,000 milliseconds). Bold face entries indicate an improvement compared to MH.

- *gap*: The gap between *B&P* and LB_{new}, computed as $100\frac{B\&P - LB_{new}}{B\&P}$.
- *nodes*: Number of nodes explored in the branch-and-price tree.
- t(ms): Total computation time in milliseconds.
- $t_{pricing}$ (ms): Amount of time spent solving pricing problems in milliseconds.

For the majority of instances, our branch-and-price algorithm is able to find integer solutions, many of which have been proven to be optimal. Some of the largest instances we solved to optimality contain 40 stops and 800 students. Typically, the bounds computed are very strong, that is, the average gap is less than 1%. These bounds are much stronger than the bounds obtained by solving the LP relaxation of the SBRP MIP model reported in Schittekat et al. (2013). Instances with a high v/s ratio are usually solved faster compared to instances with a low v/s ratio. Solutions for these instances often exhibit routes with a small number of stops, resulting in faster pricing problems. Furthermore, the bounds for these instances are usually very strong. While comparing the effects of the number of stops and students in the instances, we noticed that instances with a large number of stops (and limited number of students) are significantly harder to solve than instances with a limited number of stops and a larger number of students; for example, doubling the number of stops has much more impact on the runtime than doubling the number of students.

When compared to Schittekat et al. (2013), 26 of the instances had their bounds improved, and better solutions were discovered for nine instances. In addition, note that in Schittekat et al. (2013), the exact methods to compute bounds and optimal integer solutions were allotted a runtime of two hours, whereas in this work the runtime is limited to one hour. Using a traditional MIP model in Schittekat et al. (2013), 43 instances were solved to optimality. When comparing their heuristic solutions to the lower bound they computed, three more instances were solved to optimality, resulting in a total of 46 instances. Using our branch-and-price approach, a total of 68 instances are solved to optimality.

Although no solutions for data set II (Table 2) were published in Schittekat et al. (2013), we were able to use their code to obtain heuristic solutions for these instances. Using the branch-and-price approach, better solutions were discovered for nine instances. Optimality is attested for 25 of the instances. We again observed that instances with shorter routes are generally solved faster. In particular, if we reduce the vehicle capacity, we automatically obtain shorter routes and, hence, better run times. The latter is indeed confirmed when we compare the instances with vehicle capacities of 15 against instances with larger vehicle capacities.

As described in Section 4.2, degeneracy in CG significantly affects the time required to solve a branch-and-price node. To test the effects of the stabilization mechanisms, we solved the root nodes of instances 11–51 (data set I) to optimality, while measuring the required number of iterations as well as computation times. Figure 1 compares the different stabilization approaches: IPS stabilization with resp. 20 and 50 extreme points (IPS20, IPS50), Du Merle's PTS, or no stabilization at all (No Stab). Only the PTS approach was capable of solving all instances within a time limit of one hour. Both PTS and IPS reduced the number of iterations required to solve the MP. IPS20 needed 46% of the number of iterations required when no stabilization was used, while this number was 43% for IPS50, and 45% for PTS. However, looking from a time perspective, IPS20 required on average 314% of the computation time required when no stabilization was used, IPS50 needed 710%, whereas only 22% was required for PTS. Although IPS needs significantly more time to solve the instances, we must note that instance 50 could only be solved if either IPS or PTS was used. Concluding, IPS and

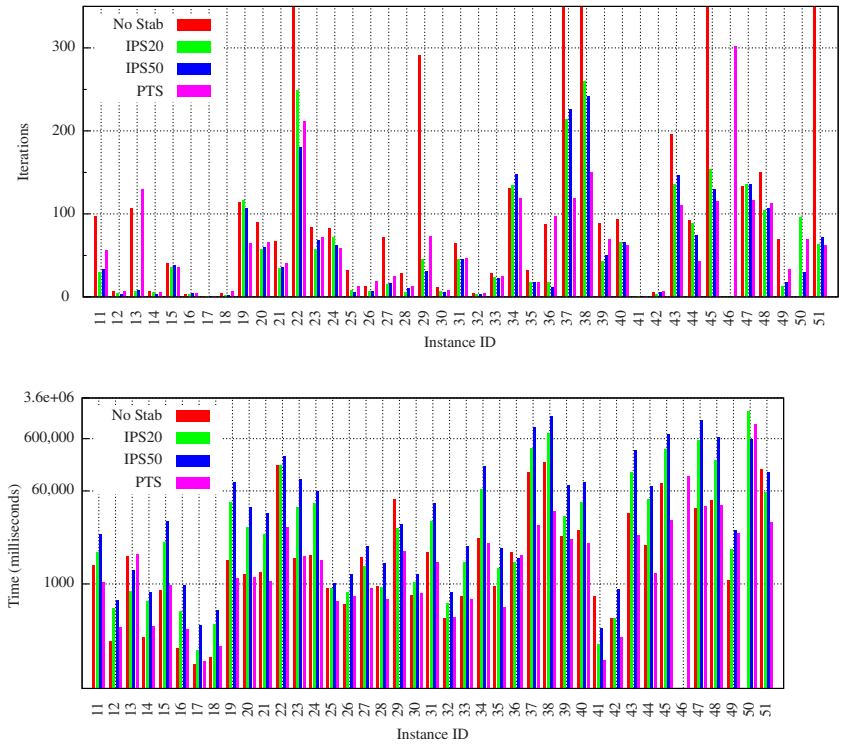

Fig. 1. Comparison of different stabilization approaches.

PTS both reduce the number of iterations required, but, from a time perspective, PTS significantly outperforms IPS.

7. Conclusions

In this paper, we studied the School Bus Routing Problem (SBRP), a vehicle routing problem that encompasses selecting bus stops, assigning students to the selected stops, and finally determining the necessary bus routes, while minimizing the routing costs. We developed an exact algorithm for the SBRP based on a set covering formulation. This formulation enables us to solve a large number of the SBRP instances of the benchmark released by Schittekat et al. (2013) to optimality. For the remaining instances, lower bounds as well as some of the integer solutions have been improved. Also, when comparing to Riera-Ledesma and Salazar-González (2013), we observe that set-partitioning formulations solve instances with large number of students more efficiently than instances with a large number of stops. However, when the set covering formulation is strengthened with valid inequalities, the number of stops that can be handled increases significantly (Riera-Ledesma and Salazar-González, 2013). Thus, one possible extension is to incorporate the separation of valid inequalities. In this work, we applied several techniques that have been reported to increase the performance of column generation, such as exact and heuristic pricing algorithms, bounding

procedures, a column pool manager, stabilization techniques, and a more rigid branching approach for the branch-and-price tree. Among these approaches, bounding procedures and stabilization cause the largest performance improvements. Two types of stabilization have been compared: interior point stabilization and Du Merle's proximal type stabilization. Although both reduce the number of required iterations considerably, the proximal type outperforms the interior point stabilization as the latter increased the computation time per iteration significantly.

Acknowledgment

This work is supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office.

References

Ahuja, R.K., Magnanti, T.L., Orlin, J.B., 1993. *Network Flows: Theory, Algorithms, and Applications*, Chapter 5. Prentice Hall, Upper Saddle River, NJ

Albareda Sambola, M., 2003. *Models and Algorithms for Location-Routing and Related Problems*. PhD thesis, Universitat Politècnica de Catalunya, Spain.

Amor, H.B., Desrosiers, J., 2006. A proximal trust-region algorithm for column generation stabilization. *Computers & Operations Research* 33, 910–927.

Amor, H.B., Desrosiers, J., Frangioni, A., 2009. On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics* 157, 6, 1167–1184.

Baldacci, R., Dell'Amico, M., González, J.S., 2007. The capacitated *m*-ring-star problem. *Operations Research* 55, 6, 1147–1162.

Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H, 1998. Branch-and-price: column generation for solving huge integer programs. *Operations Research* 46, 3, 316–329.

Bodin, L., Berman, L., 1979. Routing and scheduling of school buses by computer. *Transportation Science* 13, 2, 113–129.

Bowerman, R., Hall, B., Calami, P., 1995. A multiobjective optimization approach to urban school bus routing: formulation and solution method. *Transportation Research: Part A, Policy and Practice* 29, 107–123.

Briant, O., Lemaréchal, C., Meurdesoif, P., Michel, S., Perrot, N., Vanderbeck, F., 2008. Comparison of bundle and classical column generation. *Mathematical Programming* 113, 299–344.

Chapleau, L., Ferland, J.A., Rousseau, J.M., 1985. Clustering for routing in densely populated areas. *European Journal of Operational Research* 20, 1, 48–57.

Chvátal, V., 1983. *Linear Programming*, Chapter 13. W.H. Freeman, New York.

Dell'Amico, M., Righini, G., Salani, M., 2006. A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science* 40, 235–247.

Desaulniers, G., Desrosiers, J., Solomon, M. (ed.), 2005. *A Primer in Column Generation*, Chapters 2, 3, 12. Springer, Berlin.

Desrosiers, J., Ferland, J., Rousseau, J.M., Lapalme, G., Chapleau, L., 1986. TRANSCOL—a multiperiod school bus routing and scheduling system. *Delivery of Urban Services—TIMS Studies in the Management Sciences* 22, 47–71.

Dulac, G., Ferland, J.A., Forgues, P.A., 1980. School bus routes generator in urban surroundings. *Computers & Operations Research* 7, 3, 199–213.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M.P., Reis, M.L., Uchoa, E., Werneck, R.F.F., 2006. Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming* 106, 3, 491–511.

Hoshino, E.A., de Souza, C.C., 2009. A branch-and-cut-and-price approach for the capacitated m-ring-star problem. *Electronic Notes in Discrete Mathematics* 35, 103–108.

Hoshino, E.A., de Souza, C.C., 2012. A branch-and-cut-and-price approach for the capacitated m-ring-star problem. *Discrete Applied Mathematics* 160, 18, 2728–2741.

Lasdon, L.S., 1970. *Optimization Theory for Large Systems. Macmillan Series in Operations Research*. Macmillan, New York.

Lübbecke, M.E., 2010. *Column generation*. Wiley Encyclopedia of Operations Research and Management Science (EORMS). John Wiley & Sons, Chichester.

Lübbecke, M.E., Desrosiers, J., 2002. Selected topics in column generation. *Operations Research* 53, 6, 1007–1023.

Martin, A., 1999. Integer programs with block structure. Habilitation thesis. Preprint SC 99-03, Konrad-Zuse-Zentrum für Informationstechnik, Berlin.

Merle, O.D., Villeneuve, D., Desrosiers, J., Hansen, P., 1997. Stabilized column generation. *Discrete Math* 194, 229–237.

Oukil, A., Amor, H.B., Desrosiers, J., Gueddari, H.E., 2007. Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems. *Computers & Operations Research* 34, 3, 817–834.

Park, J., Kim, B.I., 2010. The school bus routing problem: a review. *European Journal of Operational Research* 202, 2, 311–319.

Park, J., Tae, H., Kim, B.I., 2012. A post-improvement procedure for the mixed load school bus routing problem. *European Journal of Operational Research* 217, 1, 204–213.

Riera-Ledesma, J., Salazar-González, J.J., 2012. Solving school bus routing using the multiple vehicle traveling purchaser problem: a branch-and-cut approach. *Computers & Operations Research* 39, 2, 391–404.

Riera-Ledesma, J., Salazar-González, J.J., 2013. A column generation approach for a school bus routing problem with resource constraints. *Computers & Operations Research* 40, 2, 566–583.

Ropke, S., Cordeau, J.F., August 2009. Branch and cut and price for the pickup and delivery problem with time windows. *Transportation Science* 43, 267–286.

Rousseau, L., Gendreau, M., Feillet, D., 2007. Interior point stabilization for column generation. *Operations Research Letters* 35, 5, 660–668.

Schittekat, P., Kinable, J., Sörensen, K., Sevaux, M., Spieksma, F., Springael, J., 2013. A metaheuristic for the school bus routing problem with bus stop selection. *European Journal of Operational Research* 229, 518–528.

Vanderbeck, F., 1994. Decomposition and column generation for integer programs. PhD thesis, Louvain-La-Neuve.

Wolsey, L.A., 1998. *Integer Programming*. Wiley-Interscience, New York.

Wolsey, L.A., Nemhauser, G.L., 1988. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, pp. 323–337.