A Branch-Price-and-Cut Algorithm for the Capacitated Multiple Vehicle Traveling Purchaser Problem with Unitary Demand

Nicola Bianchessi[a,b,*], Stefan Irnich[a], Christian Tilk[a]

[a] *Chair of Logistics Management, Gutenberg School of Management and Economics, Johannes Gutenberg University, Jakob-Welder-Weg 9, 55128 Mainz, Germany.*
[b] *Dipartimento di Informatica, Università degli Studi di Milano, Via Celoria 18, I-20122 Milan, Italy.*

Abstract

The multiple vehicle traveling purchaser problem (MVTPP) consists of simultaneously selecting suppliers and routing a fleet of homogeneous vehicles to purchase different products at the selected suppliers so that all product demands are fulfilled and traveling and purchasing costs are minimized. We consider variants of the MVTPP in which the capacity of the vehicles can become binding and the demand for each product is one unit. Corresponding solution algorithms from the literature are either branch-and-cut or branch-and-price algorithms, where in the latter case the route-generation subproblem is solved on an expanded graph by applying standard dynamic-programming techniques. Our branch-price-and-cut algorithm employs a novel labeling algorithm that works directly on the original network and postpones the purchasing decisions until the route has been completely defined. Moreover, we define a new branching rule generally applicable in case of unitary product demands, introduce a new family of valid inequalities to apply when suppliers can be visited at most once, and show how product incompatibilities can be handled without considering additional resources in the pricing problem. In comprehensive computational experiments with standard benchmark sets we prove that the new branch-price-and-cut approach is highly competitive.

Keywords: vehicle routing, multiple vehicle traveling purchaser problem, unitary demand, incompatible products, column generation, dynamic-programming labeling algorithm

1. Introduction

In the context of procurement problems, the *multiple vehicle traveling purchaser problem* (MVTTP) can be seen as the problem faced by a company which needs to collect given amounts of different products from several suppliers. To this aim, the company operates a fleet of homogeneous vehicles. All vehicles are based at a common depot. Each supplier offers a subset of the products at possibly different prices (purchasing costs) and availabilities. The company has to select a subset of suppliers and construct a set of routes to visit them and collect products so that all product demands are fulfilled and traveling and purchasing costs are minimized.

Formally, MVTPP variants can be defined on a directed graph $G = (V, A)$ with vertex set $V = M \cup \{0, m+1\}$ and arc set A. The vertices $M = \{1, \dots, m\}$ represent the set of possible suppliers and vertices 0 and $m+1$ represent the depot where vehicle routes start and end, respectively. For each arc $(i, j) \in A$ with $i \neq m+1$ and $j \neq 0$, the vehicles can travel from i to j with non-negative travel costs c_{ij}. In the following, it is assumed that the triangle inequality holds for the routing costs (c_{ij}).

Let K be the set of products to purchase. A non-negative quantity q_{ki} of product $k \in K$ is available at supplier $i \in M$. There are no products available at the depot so that we can define $q_{k0} = q_{k,m+1} = 0$. Additionally, let $K_i = \{k \in K : q_{ki} > 0\} \subseteq K$ be the subset of products sold by supplier $i \in M$, and let $M_k = \{i \in M : q_{ki} > 0\} \subseteq M$ be the set of suppliers offering product $k \in K$. The non-negative purchasing cost of product k at supplier i is denoted by p_{ki}. Moreover, for each product k a positive demand d_k has to be fulfilled, i.e., collected by one or several vehicles.

Suppliers are visited by means of a fleet F of homogeneous vehicles each with capacity Q. In the following, a *route* r is an elementary 0-$(m+1)$-path in G together with, for each $k \in K$ and $i \in M_k$, a quantity δ^r_{ki}

*Corresponding author.
Email address: nbianche@uni-mainz.de, bianchessi@di.unimi.it (Nicola Bianchessi)

indicating how much of product k is purchased at supplier i. The elementary 0-$(m+1)$-path can be described by $(i_0, i_1, i_2, \ldots, i_{L-1}, i_L)$ with the requirements that all vertices must be different, $i_0 = 0$, $i_L = m+1$, and $(i_{\ell-1}, i_\ell) \in A$ for all $\ell = 1, 2, \ldots, L$. A route r is *feasible* if

(i) the total amount of products purchased at the visited suppliers does not exceed the vehicle capacity, i.e., $\sum_\ell \sum_k \delta^r_{ki_\ell} \leq Q$,

(ii) the amount of product k purchased at supplier i_ℓ does not exceed the available supply, i.e., $\delta^r_{ki_\ell} \leq q_{ki_\ell}$ for all $\ell = 1, 2, \ldots, L$ and all $k \in K_{i_\ell}$, and

(iii) all purchases are feasible, i.e., $\delta^r_{ki_\ell} = 0$ for all $\ell = 1, 2, \ldots, L$ and all $k \in K \setminus K_{i_\ell}$.

MVTPPs can be classified according to the following four categories referring to the available supply, demand, vehicle capacity, and purchasing policy. First, the available supply can be

- *restricted*, if the available quantity is less than the product demand for at least one product available at a supplier;
- *unrestricted*, if the available quantity is not less than the corresponding demand for all products available at the suppliers.

Second, the demand can be classified as

- *unitary*, if the demand for all products is one;
- *general*, otherwise.

Third, the vehicles can be

- *capacitated*, meaning that the given capacity can become binding;
- *uncapacitated*, otherwise.

Fourth and finally, with respect to the purchasing policy we distinguish between

- *split (purchases)*, if the same product can be purchased multiple times and more than one purchase can be smaller than the demand and the product availability at a supplier, i.e., $\delta^r_{ki} < \min\{d_k, q_{ki}\}$. To fulfil the demand of product k, it can be that different suppliers are visited by the same vehicle or that several vehicles purchase k with visits to the same supplier or to different suppliers;
- *non-split (purchases)*, otherwise. Note that, with restricted supply and general demand, it may be possible to visit different suppliers to fulfil the demand of a product when non-split purchases are allowed. However, at most one purchase is allowed to be smaller than the demand and the product availability.

Table 1 summarizes the valid combinations for the four categories. Note that unitary demand implies an unrestricted supply and non-split purchases. Gray entries in a row of the table correspond to combinations giving rise to equivalent MVTPP variants. For a valid entry, there may still exist different MVTPP variants when additional side constraints are considered. One important class of side constraints is the *single-visit constraints* (SVC) imposing that no supplier is visited more than once. Without this requirement it is probably not possible to develop strong two-index compact models for the problem (for the related discussion on strong formulations for the split delivery vehicle-routing problem we refer to Bianchessi and Irnich, 2019). SVCs imply that all products that are available only at one specific supplier have to be purchased when this supplier is visited. Thus, in case of unitary demand and capacitated vehicles, imposing SVCs may cause the problem to become infeasible if the number of products that are available only at one specific supplier is greater than the capacity of the vehicles. Hence, when SVCs have to be fulfilled in this case, it is usually assumed that the problem remains feasible. Finally, it is worth noting that SVCs do not prevent split purchases, but impose that purchases of the same product must be done at different suppliers. In general, in case no further side constraint has to be considered, allowing or forbidding multiple visits to the same supplier is independent of the underlying purchasing policy.

	non-split	non-split	split	split	
unrestricted	[1,2,3,4,5,8]		invalid		capacited
unrestricted	[6]		invalid		uncapacited
restricted	invalid		invalid	[2,7]	capacited
restricted	invalid	[9]	invalid	[6]	uncapacited
	unitary	general	unitary	general	

Table 1: MVTPP variants — [1]: Baldacci *et al.* (2007); [2]: Choi and Lee (2010); [3]: Hoshino and De Souza (2012); [4]: Riera-Ledesma and Salazar-González (2012); [5]: Riera-Ledesma and Salazar-González (2013); [6]: Bianchessi *et al.* (2014); [7]: Manerba and Mansini (2015); [8]: Gendreau *et al.* (2016); [9]: Manerba and Mansini (2016).

The focus of this paper at hand is the unrestricted, capacitated, unitary MVTPP with non-split purchases

(left-upper corner variant in Table 1). Before highlighting our contribution, we briefly review the MVTPP literature according to the above categories.

1.1. Literature

The MVTPP is NP-hard as it generalizes the well-known *traveling purchaser problem* (TPP, Manerba *et al.*, 2017) to the multiple vehicles case. For an in-depth overview also on related procurement problems, the reader is referred to the cited survey. Here, we summarize the literature on the multi-vehicle case.

The MVTPP has been introduced by Choi and Lee (2010) who proposed Miller-Tucker-Zemlin formulations (Miller *et al.*, 1960) for both the unrestricted and the restricted version, considering unitary and general demand, respectively, and capacitated vehicles. Instances with up to 40 suppliers, 40 products, and 4 vehicles or up to 30 suppliers, 30 products, and 3 vehicles were solved by means of CPLEX 11.1, respectively.

Riera-Ledesma and Salazar-González (2012) used the capacitated, unitary MVTPP to model a school bus routing problem. They presented a *branch-and-cut* (BC) algorithm based on a two-index single-commodity flow formulation, solving symmetric and asymmetric instances with up to 125 stops (suppliers), 125 students (products), and 6 buses (vehicles). Later, the same authors (Riera-Ledesma and Salazar-González, 2013) addressed an extended version of the problem considering upper bounds on the length/duration of each route and on the number of possible stops of each bus as well as lower bounds on the number of students served by each bus. The MVTTP formulation was augmented by additional constraints and finally solved with a *branch-price-and-cut* (BPC) algorithm (Barnhart *et al.*, 1998; Lübbecke and Desrosiers, 2005; Desaulniers *et al.*, 2005). The pricing problem is modeled as an elementary *shortest path problem with resource constraints* (SPPRC, Irnich and Desaulniers, 2005) on an expanded graph, in which vertices correspond to all the potential assignments of a product to a supplier, and then solved through an adaptation of the *q-routes* dynamic-programming labeling algorithm (Christofides *et al.*, 1981). Instances with up to 125 stops and 125 students were solved to optimality considering different combinations of active bounds.

Bianchessi *et al.* (2014) introduced unrestricted and restricted versions of the MVTPP that consider unitary and general demand, respectively. In both versions, the length of each route is bounded, whereas no limit is imposed on the capacity of the vehicles. The authors proposed a *branch-and-price* (BP) algorithm (Barnhart *et al.*, 1998; Lübbecke and Desrosiers, 2005; Desaulniers *et al.*, 2005) in which, thanks to the unlimited capacity of the vehicles, the pricing problem is directly modeled and solved as an elementary SPPRC on the original graph. The algorithm is able to solve instances with up to 100 suppliers, 200 products, and 8 vehicles.

Manerba and Mansini (2015) introduced a variant of the restricted, capacitated, general MVTPP, named MVTPP-PIC, involving possible incompatibilities among products that forbid to load two incompatible products into the same vehicle. Incompatibilities may require several visits to the same supplier, even if this is not beneficial with respect to traveling costs. Note that SVCs are typically not imposed for the MVTPP-PIC, because they may quickly lead to infeasibility. The problem was addressed by means of a BC algorithm that is based on a three-index formulation. The BC also makes use of an ad-hoc primal heuristic. The algorithm solves instances with up to 50 suppliers, 100 products, 20 percent of cross-incompatibilities among them, and 16 vehicles.

For the unitary demand version of the MVTPP-PIC, a BP algorithm was then presented by Gendreau *et al.* (2016). At each column-generation iteration, the pricing problem is tentatively solved heuristically as an elementary SPPRC defined on an expanded graph as in Riera-Ledesma and Salazar-González (2013) by means of a labeling algorithm. To finally prove optimality, a BC algorithm is applied directly to a MIP formulation of the pricing problem resulting from the proposed Dantzig-Wolfe decomposition. The algorithm is able to solve instances up to 50 suppliers, 70 products, with 70 percent of cross-incompatibilities among them.

Finally, Manerba and Mansini (2016) introduced a further variant of the restricted, capacitated, general MVTPP to model a nurse routing problem with inter-route incompatibilities constraints and bounds on the duration of the routes. The BP algorithm proposed by the authors to solve the problem is similar to the one devised by Gendreau *et al.* (2016).

A closely related problem is the *capacitated m-ring-star problem* (CmRSP, Baldacci *et al.*, 2007) that can also be modelled as a capacitated, unitary MVTPP. The CmRSP is the problem of designing a set of routes starting and ending at a given depot, visiting some customers in between, and assigning each non-visited customer to a visited point or customer. Baldacci *et al.* (2007) presented a sophisticated BC algorithm and solved instances with up to 137 customers to optimality. Hoshino and De Souza (2012) presented a BPC algorithm that solves the subproblem as an elementary SPPRC on a expanded graph using a complex

dominance rule based on a deterministic finite automaton. Riera-Ledesma and Salazar-González (2012) used the instances of Baldacci *et al.* (2007) to computationally evaluate the performance of their algorithm.

	single-visit constraints	restricted route length	capacitated	with incompatibilities
Baldacci *et al.* (2007)	•		•	
Choi and Lee (2010)	•		•	
Hoshino and De Souza (2012)	•		•	
Riera-Ledesma and Salazar-González (2012)	•		•	
Riera-Ledesma and Salazar-González (2013)	•	•	•	
Bianchessi *et al.* (2014)	•	•		
Gendreau *et al.* (2016)			•	•

Table 2: Characteristics of UMVTPP variants in the literature

Table 2 provides a synopsis of the *unitary MVTPP* (UMVTPP) variants that we surveyed. For the pairwise incompatibilities between different products, binary parameters $w_{kk'}$ indicate whether products k and k' are *incompatible*. The set of incompatible product pairs is denoted by $B = \{(k, k') \in K \times K : w_{kk'} = 1\}$ (see Manerba and Mansini, 2015; Gendreau *et al.*, 2016). For the route duration constraints, non-negative travel times d_{ij} are given, for all arcs $(i, j) \in A$, and the *maximum route duration* is denoted by D^{max}. A route is feasible if $(k, k') \notin B$ holds for all pairs of products (k, k') purchased at the visited suppliers and the accumulated travel time $\sum_{\ell=1}^{L} d_{i_{\ell-1} i_\ell}$ for the path $(i_0, i_1, i_2, \ldots, i_{L-1}, i_L)$ does not exceed D^{max}.

1.2. Contribution

The overall contribution of the paper at hand is the presentation of a new BPC algorithm that can deal with the basic version of the capacitated UMVTPP considering additional SVCs (as previously addressed by five of the nine articles reviewed, see Table 1 and 2), as well as with the extensions arising from incompatibility constraints and route duration constraints. Particular contributions are:

1. All previous BP algorithms for the capacitated UMVTPP or one of its extensions solve the route-generation subproblem on an expanded graph with standard dynamic-programming labeling algorithms (Riera-Ledesma and Salazar-González, 2013; Gendreau *et al.*, 2016) or BC algorithms (Gendreau *et al.*, 2016; Manerba and Mansini, 2016). Our new labeling algorithm works directly on the original network, i.e., the one with vertices for the suppliers and the depot. The key idea of the new approach is to postpone the purchasing decision until the route has been completely defined. On the positive side, this drastically reduces the size of the pricing network and thus accelerating the solution process. On the downside, new and more complex dominance rules have to be defined.
2. For the UMVTPP and its extensions, we introduce a new generally applicable branching rule.
3. We introduce a new family of valid inequalities for strengthening SVCs.
4. Finally, we show how incompatibilities can be handled without additional resources in the pricing problem by considering multiple pricing problems.

Our later presented computational results indicate that the new subproblem solution strategies are beneficial, especially for instances in which many products are available at a supplier and routes are relatively short.

The remainder of this paper is organized as follows: Section 2 describes our BPC algorithm with subsections on the extensive formulation, the pricing subproblem, valid inequalities, and branching rules. In Section 3, we computationally evaluate the BPC using instances from different benchmarks. Final conclusions are drawn in Section 4.

2. Branch-Price-and-Cut Algorithm

This section describes the new BPC algorithm for the solution of different unitary MVTPP variants. Section 2.1 presents the straightforward set-partitioning formulation that provides the basis for the column-generation master program. The route-generation subproblem is discussed in Section 2.2. The valid inequalities used to strengthen the linear relaxation of the master program and a dynamic neighborhood extension are presented in Sections 2.3 and 2.4. Branching rules to finally obtain integer solutions are discussed in Section 2.5.

2.1. Route-based Formulation

We now propose a general route-based formulation that can model all the addressed unitary MVTPP variants, and on which our BPC algorithm is based on. Recall that a route r is defined as an elementary 0-$(m+1)$-path together with the purchasing decisions (δ^r_{ki}). Let Ω denote the set of all feasible routes. Feasibility takes into account all variant-specific intra-route constraints as described in Section 1. For a route $r \in \Omega$, let the integer coefficient b^r_{ij} give the number of times that arc $(i,j) \in A$ is traversed (0 or 1 for elementary routes). Finally, the *cost* of the route is defined as $c^r = \sum_{(i,j)\in A} c_{ij} b^r_{ij} + \sum_{k\in K} \sum_{i\in M_k} p_{ki} \delta^r_{ki}$ including routing and purchasing costs.

The route-based formulation uses two types of variables: Binary variables λ^r for all $r \in \Omega$ are equal to 1 if route r is performed. The non-negative integer variable f describes the number of vehicles used.

$$\min \sum_{r\in\Omega} c^r \lambda^r \tag{1a}$$

$$\text{subject to} \quad \sum_{r\in\Omega} \sum_{i\in M_k} \delta^r_{ki} \lambda^r = 1 \qquad \forall\, k \in K \tag{1b}$$

$$\sum_{r\in\Omega} \lambda^r = f \tag{1c}$$

$$0 \le f \le |F| \text{ and integer} \tag{1d}$$

$$\lambda^r \in \{0,1\} \qquad \forall\, r \in \Omega \tag{1e}$$

The objective (1a) calls for the minimization of the total traveling and purchasing costs. The set-partitioning constraints (1b) ensure that all products are purchased exactly once. Constraints (1c) and (1d) limit the number of vehicles to use. Binary constraints (1e) are stated for all route variables.

In the following, the linear relaxation of formulation (1) in which the set of all feasible routes Ω is replaced by a subset $\bar{\Omega}$ is denoted as *restricted master program* (RMP). We initialize the RMP with one big-M variable that covers all product demands and has no impact on other constraints. For solving the linear relaxation of (1), a column-generation algorithm is employed (Desaulniers *et al.*, 2005). Branching is required to finally ensure integer solutions.

2.2. Column Generation

In this section, we consider an arbitrary column-generation iteration and thus assume that the dual prices $(\rho_k)_{k\in K}$ associated with constraints (1b) and the dual price μ of constraint (1c) are given.

We define the *reduced purchasing cost* $\bar{p}_{ki}$ of product $k \in K$ at supplier $i \in M$ as $\bar{p}_{ki} = p_{ki} - \rho_k$. The pricing problem asks for a route r with negative reduced cost $\bar{c}_r$, if any:

$$\bar{c}_r = c_r - \sum_{k\in K} \rho_k \left(\sum_{i\in M_k} \delta^r_{ki} \right) - \mu = c_r - \sum_{k\in K} \sum_{i\in M_k} \rho_k \delta^r_{ki} - \mu = \sum_{(i,j)\in A} c_{ij} b^r_{ij} + \sum_{k\in K} \sum_{i\in M_k} \bar{p}_{ki} \delta^r_{ki} - \mu.$$

For the sake of clarity, we first state this pricing problem as a binary program. For a vertex set $S \subset V$, we use the standard notation $A(S) = \{(i,j) \in A : i \in S, j \in S\}$ for the internal, $\Gamma^+(S) = \{(i,j) \in A : i \in S, j \notin S\}$ the outgoing, and $\Gamma^-(S) = \{(i,j) \in A : i \notin S, j \in S\}$ the ingoing arcs of the set. For a singleton set $S = \{i\}$, we write $\Gamma^+(i)$ and $\Gamma^-(i)$. The binary program used to find a negative reduced cost route r, defined by an elementary path and purchased quantities, comprises binary variables x_{ij} and z_i that are equal to 1 if and only if arc $(i,j) \in A$ is traversed and supplier $i \in M$ is visited, respectively. Moreover, binary variables δ_{ki} indicate if product $k \in K$ is purchased at supplier $i \in M_k$. The subproblem can now be

stated as follows:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{k\in K}\sum_{i\in M_k} \bar{p}_{ki}\delta_{ki} \; - \mu \tag{2a}$$

$$\text{subject to} \quad \sum_{(0,j)\in\Gamma^+(0)} x_{0j} = 1 \tag{2b}$$

$$\sum_{(j,i)\in\Gamma^-(i)} x_{ji} = \sum_{(i,j)\in\Gamma^+(i)} x_{ij} = z_i \qquad \forall\, i \in M \tag{2c}$$

$$\sum_{(i,j)\in\Gamma^+(S)} x_{ij} \geq z_i \qquad \forall\, S \subseteq M,\ |S| \geq 2, i \in S \tag{2d}$$

$$\sum_{i\in M_k} \delta_{ki} \leq 1 \qquad \forall\, k \in K \tag{2e}$$

$$\sum_{k\in K}\sum_{i\in M_k} \delta_{ki} \leq Q \tag{2f}$$

$$\delta_{ki} \leq z_i \qquad \forall\, k \in K,\ i \in M_k \tag{2g}$$

$$\delta_{ki} \in \{0,1\} \qquad \forall\, k \in K,\ i \in M_k \tag{2h}$$

$$z_i \in \{0,1\} \qquad \forall\, i \in M \tag{2i}$$

$$x_{ij} \in \{0,1\} \qquad \forall\, (i,j) \in A \tag{2j}$$

The objective (2a) is the minimization of the reduced cost of a route. Constraints (2b)–(2c) are the flow conservation implying also that $\sum_{(i,m+1)\in\Gamma^-(m+1)} x_{i,m+1} = 1$. Constraints (2d) together with (2i) ensure the route to be an elementary 0-$(m+1)$-path. Constraints (2e) guarantee that every product is purchased at most once, i.e., the demand in unitary. Inequality (2f) imposes the capacity constraint. Consistency between variables δ_{ki} and z_i is ensured by (2g). Binary domains for all variables are stated by (2h)–(2j).

Property 1. *If the triangle inequality for travel costs (c_{ij}) holds, then there exists an optimal solution to the UMVTPP in which at least one product is purchased at every visited supplier $i \in M$.*

Proof. See (Riera-Ledesma and Salazar-González, 2012). □

For each supplier $i \in M$, let $K_i^* = \{k \in K_i : \{i\} = M_k\}$ denote the set of *unique products* that can only be purchased at supplier i. Assuming that the routing costs fulfil the triangle inequality (as stated in Section 1), we can strengthen formulation (2) of the pricing problem by

$$\sum_{i\in M} z_i \leq Q \tag{3a}$$

$$\sum_{k\in K_i} \delta_{ki} \geq z_i \qquad \forall\, i \in M \tag{3b}$$

$$\delta_{ki} = z_i \qquad \forall\, i \in M, k \in K_i^* \tag{3c}$$

where constraint (3a) implies that the number of visited suppliers does not exceed the vehicle capacity, constraints (3b) guarantee at least one purchase per visited supplier, and constraints (3c) guarantee that all products that are only available at supplier i are bought when this supplier is visited.

2.2.1. Dynamic-Programming Labeling Algorithm

We now present the new labeling algorithm that works directly on the original network. The handling of side constraints concerning single visits, incompatibilities, and the maximum route duration is discussed in Section 2.2.2.

The pricing problem (2) is a variant of the SPPRC, for which dynamic programming-based labeling algorithms are commonly used in BP algorithms when solving vehicle-routing and crew-scheduling problems (Irnich and Desaulniers, 2005). In simple SPPRC variants, resource consumptions along arcs are known. In MVTPP variants, however, the products to purchase at each visited supplier are decision variables, yielding different reduced costs and resource consumptions. To circumvent additional decisions per arc traversed, the MVTPP can be modeled over an expanded graph, in which vertices correspond to all potential combinations of a product and a supplier (see, e.g., Riera-Ledesma and Salazar-González, 2013; Gendreau *et al.*, 2016).

This modeling approach drastically increases the size of the pricing network. Hence, previous works using the expanded network may suffer from long computation times required for pricing.

Our idea of solving the pricing problem directly on the original network strives for accelerated pricing using relatively small networks. Moreover, we postpone all unclear decisions concerning the potential purchase of products at visited suppliers until the end of the route, i.e., when reaching the destination depot $m+1$.

We show first how to compute the smallest reduced purchasing cost when a path is given. Let a partial path $P = (0, i_1, \ldots, i_n)$ be given, i.e., a path that does not necessarily end at the destination $m+1$. For simplification purposes, we introduce the shorthand notations $S(P) = M \cap V(P) = \{i_1, \ldots, i_n\}$ for the set of visited suppliers and $K(P) = K_{i_1} \cup K_{i_2} \cup \cdots \cup K_{i_n}$ for the set of products that can be purchased on path P. For each product $k \in K(P)$, we define the *minimum reduced purchasing cost* (MRPC) given by

$$\bar{p}^*_k(S(P)) = \min_{i \in S(P)} \bar{p}_{ki}. \tag{4a}$$

If the context is clear, we may lighten the notation and write $\bar{p}^*_k$ instead of $\bar{p}^*_k(S(P))$.

Note first that only those products that have a strictly negative MRPC are relevant for an optimal purchasing decision. Let h be the number of these products so that the h products can be sorted by non-decreasing MRPCs. Accordingly, we define

$$O(S(P)) = ((k_1, \bar{p}^*_{k_1}), \ldots, (k_h, \bar{p}^*_{k_h})) \tag{4b}$$

as the sorted *sequence* of pairs of products and their negative MRPCs. Note that absolute values are non-increasing, i.e., $|\bar{p}^*_{k_1}| \geq |\bar{p}^*_{k_2}| \geq \ldots \geq |\bar{p}^*_{k_{\bar{h}}}|$.

It is optimal to purchase the first $\bar{h} = \min\{Q, |O(S(P))|\} = \min\{Q, h\}$ products $k_1, k_2, \ldots, k_{\bar{h}}$ of the sequence. As Q is constant, the overall reduced purchasing cost can now be described by the *function*

$$f(S(P)) = \sum_{q=1}^{\bar{h}} \bar{p}^*_{k_q}. \tag{4c}$$

As a result, the minimum reduced cost of any route r using path $P = (0 = i_0, i_1, \ldots, i_L, i_{L+1} = m+1)$ with routing cost $c(P) = \sum_{\ell=1}^{L+1} c_{i_{\ell-1}, i_\ell}$ is

$$\bar{c}_r = c(P) + f(S(P)) - \mu \tag{5}$$

Note that such a sequence and function can also be defined independently of a path P. One only has to know the subset of suppliers (for the path P the suppliers $S = S(P)$). Therefore, for any subset $S \subseteq M$ of suppliers, we define $O(S)$ and $f(S)$ accordingly via (4a)–(4c).

Proposition 1. *The function $f : 2^M \to \mathbb{R}, S \mapsto f(S)$ defined by* (4a)–(4c) *has the following properties:*
(i) f is non-positive, i.e., $f(S) \leq 0$ for all $\varnothing \subseteq S \subseteq M$;
(ii) f is non-increasing, i.e., for any $R \subseteq S \subseteq M$ it follows $f(R) \geq f(S)$;
(iii) f is supermodular ($-f$ is submodular), i.e., for any $R, S \subseteq M$ it follows $f(R) + f(S) \leq f(R \cup S) + f(R \cap S)$.

Proof. (i): By definition of f because all values $\bar{p}^*_{k_q}$ are negative for $q = 1, \ldots, \bar{h}$.

(ii): $R \subseteq S$ implies that the minima fulfill $\bar{p}^*_{k_q}(R) \geq \bar{p}^*_{k_q}(S)$. Hence, the sequence $O(R)$ may contain less elements compared to $O(S)$ (this happens if $K(R) = \bigcup_{i \in R} K_i \subsetneq K(S) = \bigcup_{i \in S} K_i$). Moreover, for elements in the sequences with identical first component, i.e., $(k, \bar{p}^*_k(R)) \in O(R)$ and $(k, \bar{p}^*_k(S)) \in O(S)$, it follows $\bar{p}^*_k(R) \geq \bar{p}^*_k(S)$. Hence, $\bar{h}(R) \leq \bar{h}(S)$ and therefore $f(R) \geq f(S)$ holds true.

(iii): To prove the submodularity of $-f$ it is equivalent to prove that for any $S \subseteq S'$ and $i \in M$ the inequality

$$f(S) - f(S \cup \{i\}) \geq f(S') - f(S' \cup \{i\}) \tag{6}$$

holds (see, e.g., Schrijver, 2003, § 44, p. 766).

Note that the only interesting cases are those when $i \notin S'$, because otherwise $S' = S' \cup \{i\}$ so that the right-hand side in (6) is zero and the statement directly follows then from part (ii), i.e., since f is non-increasing.

We can therefore assume that $i \notin S'$ holds. If $f(S') = f(S' \cup \{i\})$, then (6) follows directly from part (ii). We can therefore assume that $f(S') > f(S' \cup \{i\})$ holds.

To simplify the analysis, we assume the sequences $O(S)$ always comprise Q entries. This is not restrictive because shorter sequences can be filled up with dummy products k_{dummy} with MRPC $\bar{p}^*_{k_{\text{dummy}}} = 0$. In

particular, the Qth element of $O(S)$ is then the product with non-positive but worst MRPC compared to all other products in the sequence $O(S)$.

Moreover, it is convenient to write $k \in O(S)$ if and only if there exists a pair $(k_h, \bar{p}^*_{k_h})$ in the sequence $O(S)$ with $k = k_h$ for some index $h \in \{1, \ldots, Q\}$. We use this convention in the following.

Another simplification is that we assume (for the moment) that $K_i = \{k\}$ holds, i.e., the supplier i offers a single product k only. Then, due to $f(S') > f(S' \cup \{i\})$,

$$k \in O(S' \cup \{i\}) \text{ with } p_{ki} = \bar{p}^*_k(S' \cup \{i\}). \tag{7}$$

Moreover, $S \subseteq S'$ implies that

$$k \in O(S \cup \{i\}) \text{ with } p_{ki} = \bar{p}^*_k(S \cup \{i\}). \tag{8}$$

We can now distinguish four cases:

- Case $k \in O(S)$ and $k \in O(S')$: Note first that the precondition of this case implies, together with (7) and (8), that both $O(S)$ and $O(S \cup \{i\})$ as well as $O(S')$ and $O(S' \cup \{i\})$ differ in exactly one pair, i.e., the pair $(k_h, \bar{p}^*_{k_h})$ with $k_h = k$ has differing values $\bar{p}^*_{k_h}$. Therefore,

$$f(S) - f(S \cup \{i\}) \overset{(8)}{=} \bar{p}^*_k(S) - p_{ki} \overset{S \subseteq S'}{\geq} \bar{p}^*_k(S') - p_{ki} \overset{(7)}{=} f(S') - f(S' \cup \{i\}).$$

  Note that the inequality directly results from the definition of $\bar{p}^*_k(\cdot)$ as a minimum of the subsets $K(S) \subseteq K(S')$, respectively, see equation (4a).
- Case $k \notin O(S)$ and $k \in O(S')$: The precondition of this case, together with $S \subseteq S'$, implies that $\bar{p}^*_k(S') \leq \bar{p}^*_{k_Q}(S)$, where k_Q is the product of the last pair $(k_Q, \bar{p}^*_{k_Q})$ in the sequence $O(S)$. According to (8), $O(S \cup \{i\})$ results from $O(S)$ by removing its last pair $(k_Q, \bar{p}^*_{k_Q})$ and inserting the pair (k, p_{ki}). It follows

$$f(S) - f(S \cup \{i\}) \overset{(8)}{=} \bar{p}^*_{k_Q}(S) - p_{ki} \overset{k \notin O(S) \wedge k \in O(S')}{\geq} \bar{p}^*_k(S') - p_{ki} \overset{(7)}{=} f(S') - f(S' \cup \{i\}).$$

- Case $k \in O(S)$ and $k \notin O(S')$: With arguments similar to those from the previous case we get

$$f(S) - f(S \cup \{i\}) \overset{(8)}{=} \bar{p}^*_k(S) - p_{ki} \overset{k \in O(S) \wedge k \notin O(S')}{\geq} \bar{p}^*_{k_Q}(S') - p_{ki} \overset{(7)}{=} f(S') - f(S' \cup \{i\}).$$

- Case $k \notin O(S)$ and $k \notin O(S')$: With the arguments from the two previous cases, the remaining last case is solved via

$$f(S) - f(S \cup \{i\}) \overset{(8)}{=} \bar{p}^*_{k_Q}(S) - p_{ki} \overset{S \subseteq S'}{\geq} \bar{p}^*_{k_Q}(S') - p_{ki} \overset{(7)}{=} f(S') - f(S' \cup \{i\}).$$

What remains to discuss is the case that supplier i offers more than just one product, i.e., $|K_i| > 1$. If $K_i = \{k_1, k_2, \ldots, k_q\}$ for $q \geq 2$, we can, just for the purpose of the proof, replace the one supplier i by q different copies named suppliers $i_1, i_2, \ldots, i_q$ with the product sets $K_{i_j} = \{k_j\}$. These copies are single-product suppliers for which the above-discussed four cases hold true. With $f(S \cup \{i\}) = f(S \cup \{i_1, i_2, \ldots, i_q\})$ and likewise $f(S' \cup \{i\}) = f(S' \cup \{i_1, i_2, \ldots, i_q\})$ we get

$$\begin{aligned}
f(S) - f(S \cup \{i\}) &= f(S) - f(S \cup \{i_1, i_2, \ldots, i_q\}) \\
&= \big(f(S\}) - f(S \cup \{i_1\})\big) + \sum_{j=2}^{q} \big(f(S \cup \{i_1, \ldots, i_{j-1}\}) - f(S \cup \{i_1, i_2, \ldots, i_{j-1}, i_j\})\big) \\
&\geq \big(f(S'\}) - f(S' \cup \{i_1\})\big) + \sum_{j=2}^{q} \big(f(S' \cup \{i_1, \ldots, i_{j-1}\}) - f(S' \cup \{i_1, i_2, \ldots, i_{j-1}, i_j\})\big) \\
&= f(S') - f(S' \cup \{i_1, i_2, \ldots, i_q\}) = f(S') - f(S' \cup \{i\})
\end{aligned}$$

where the inequality results from the four cases discussed above for single-product suppliers. This completes the proof. □

Forward Labeling. Let P be the partial path starting at the depot 0 and ending at vertex $i \in V$. We associate a label L with P (to highlight the association we write $L = L(P)$ and reversely $P = P(L)$) that includes the following attributes:

- i: The last vertex of P;
- S: The set $S(P)$ of visited suppliers;
- T^{cost}: The accumulated routing costs $c(P)$ including also (parts of) the value $-\mu$.
- T^{num} : The (minimum) number of collected products;
- O: The sequence of pairs (product, MRPC value), available at the visited suppliers, sorted by non-decreasing MRPC values as defined in (4b), i.e., $O(S(P)) = O(S)$.

Note that the sequence O can be computed via (4a) and (4b) directly from the subset S. We store the sequence O within the label for ease of convenience and speed (one can directly see which products are bought at which visited supplier). Hence, for any partial path P, we can write the associated label $L = L(P) = (i, S, T)$, where T is the vector of attributes (T^{cost}, T^{num}). Later, this latter vector will be extended to take additional constraints into account and to implement acceleration techniques.

We can now describe the labeling in more detail, i.e., the initial label, label extension, feasibility conditions, and dominance between labels. The initial label at vertex 0 is defined as $L_0 = (0, \varnothing, \mathbf{0})$.

When a feasible path $P = (0, \ldots, i)$ with associated label $L_i = (i, S_i, T_i)$ is extended along arc $(i, j) \in A$, the new label $L_j = (j, S_j, T_j)$ for $P' = (0, \ldots, i, j)$ results from the following update rules:

$$S_j = \begin{cases} S_i \cup \{j\}, & \text{if } j \in M \\ S_i & \text{otherwise} \end{cases} \tag{9a}$$

$$T_j^{cost} = T_i^{cost} + c_{ij} + \begin{cases} -\mu/2, & \text{if } i = 0 \text{ or } j = m+1 \\ 0, & \text{otherwise} \end{cases} \tag{9b}$$

$$T_j^{num} = T_i^{num} + \begin{cases} 1, & \text{if } j \in M \\ 0, & \text{otherwise} \end{cases} \tag{9c}$$

The (partial) path P' is feasible if

$$T_j^{num} \leq Q. \tag{10}$$

This feasibility condition is a direct consequence of Property 1 exploiting (3a).

Note that up to now we did not require the paths to be elementary. Indeed, with the following dominance rule all non-elementary paths are dominated by their cycle-free subpaths whenever the triangle inequality for the routing costs (c_{ij}) holds.

Proposition 2. *Let label $L_1 = (i, S_1, T_1)$ and label $L_2 = (i, S_2, T_2)$ be two labels that represent two different paths, i.e., $P(L_1) \neq P(L_2)$, that end at the same vertex $i \in V$.*

Label L_1 dominates label L_2 if the following conditions both hold true:

$$T_1^{cost} + f(S_1) \leq T_2^{cost} + f(S_1 \cup S_2) \tag{11a}$$

$$T_1^{num} \leq T_2^{num} \tag{11b}$$

Proof. Note first that any feasible extension of $P_2 = P(L_2)$ along a path $\bar{P}$ is certainly also a feasible extension of $P_1 = P(L_1)$ because of (11b) and the feasibility condition (10).

We assume that the extension along path $\bar{P}$ visits the suppliers $\bar{S}$. We have to show that inequality (11a) implies

$$T_1^{cost} + c(\bar{P}) + f(S_1 \cup \bar{S}) \leq T_2^{cost} + c(\bar{P}) + f(S_1 \cup S_2 \cup \bar{S})$$

which is equivalent to the implication

$$T_1^{cost} - T_2^{cost} \leq f(S_1 \cup S_2 \cup \bar{S}) - f(S_1 \cup \bar{S}).$$

We write

$$\begin{aligned} T_1^{cost} - T_2^{cost} &\overset{(11a)}{\leq} f(S_1 \cup S_2) - f(S_1) \\ &\overset{(6)}{\leq} f(S_1 \cup S_2 \cup \bar{S}) - f(S_1 \cup \bar{S}) \end{aligned}$$

where the last inequality holds for $S_1 = S \subseteq S' = S_1 \cup S_2$ and $\bar{S} \equiv \{i\}$, with $K_i = K(\bar{P})$ arbitrary large. This completes the proof. □

If a path P_2 is not elementary, the removal of suppliers visited two or more times (from the top) creates an elementary path P_1 ending at the same vertex i. Now P_1 with label $L_1 = (i, S_1, T_1)$ dominates P_2 with label $L_2 = (i, S_2, T_2)$, because the triangle inequality for the routing costs ensures $T_1^{cost} \leq T_2^{cost}$ and by construction $T_1^{num} < T_2^{num}$ as well as $S_1 = S_2$ so that both conditions (11) are fulfilled.

ng-Path Relaxation. Cutting planes (Section 2.3) and branching constraints (Section 2.5) introduce additional dual prices on arcs that can make the associated reduced cost negative. In this situation, negative reduced-cost cycles can occur so that non-elementary partial paths are no longer dominated by elementary paths constructed by leaving out suppliers that are visited multiple times. It is known for a long time that the linear relaxation of path-based formulations can be strengthened by eliminating all or at least some non-elementary routes and their variables from the model. In this case, the pricing subproblem must prohibit the generation of non-elementary partial paths.

Baldacci *et al.* (2011) invented *ng*-path relaxations, parameterized by neighborhoods $N_i \subset M$ for all $i \in V$, to effectively eliminate many non-elementary paths. For given neighborhoods N_i, we enforce *ng*-path constraints for the UMVTPP as follows. Each label $L = (j, S, T)$ is complemented by binary attributes $T^{ng,k}$ for all suppliers $k \in N_j$. The initial label at vertex 0 has attributes $T^{ng} = \mathbf{0}$. When extending a partial path P_i associated with label $L_i = L(P_i)$ along the arc $(i, j) \in A$, $j \in N_i$, the additional feasibility condition $T_i^{ng,j} = 0$ is checked. If the extension is feasible the new attribute is updated via $T_j^{ng,j} = 1$ and $T_j^{ng,k} = T_i^{ng,k}$ if $k \in N_j \cap N_i$, and $T_j^{ng,k} = 0$ otherwise. Finally, the dominance rules (11) have to be supplemented by the (componentwise) check $T_1^{ng} \leq T_2^{ng}$ when comparing two labels $L_1 = (i, S_1, T_1)$ and $L_2 = (i, S_2, T_2)$. Section 2.4 explains the strategy that we apply to build initial neighborhoods N_i and how we enlarge the neighborhoods in the course of the algorithm in order to strengthen the linear relaxations.

Bidirectional Labeling. Bidirectional labeling has become a quasi-standard for solving SPPRCs (Righini and Salani, 2006; Tilk *et al.*, 2017). Defining backward labels for the UMVTPP is trivial, because one can swap 0 and $m + 1$ and consider the transposed digraph $(V, A^\top)$ where $A^\top$ comprises all arcs (j, i) for $(i, j) \in A$. We use the resource T^{num} as the monotone resource: only labels that fulfill the half-way condition $T^{num} \leq Q/2$ are extended. If the routing costs are symmetric (i.e., $c_{ij} = c_{ji}$ for all $i, j \in V$), it becomes even simpler: all forward labels can be directly interpreted as backward labels (such an implicit bidirectional labeling was, e.g., described in the works of Bode and Irnich, 2012; Goeke *et al.*, 2019; Gschwind *et al.*, 2019).

A forward label $L_{fw} = (i, S_{fw}, T_{fw})$ and a backward label $L_{bw} = (i, S_{bw}, T_{bw})$ can be merged, i.e., the concatenation $P = (P_{fw}, inv(P_{bw}))$ of the two associated paths $P_{fw} = P(L_{fw})$ and $P_{bw} = P(L_{bw})$ is feasible if the following conditions hold:

$$T_{fw}^{num} + T_{bw}^{num} \leq Q + 1 \tag{12a}$$

$$T_{fw}^{ng,k} + T_{bw}^{ng,k} \leq 1 \qquad \forall k \in N_i \setminus \{i\} \tag{12b}$$

Condition (12a) reflects feasibility condition (10), whereas condition (12b) refers to the *ng*-path relaxation. The reduced cost of the merged path P can be obtained as $c^P = T_{fw}^{cost} + T_{bw}^{cost} + f(S_{fw} \cup S_{bw})$.

2.2.2. Handling of Side Constraints

We now describe the handling of side constraints, i.e., SVCs, incompatibilities between products, and tour duration/length constraints in the column-generation algorithm that we propose.

Single-Visit Constraints. For the route-based formulation, the SVCs can directly be incorporated in the master problem by adding the following set of constraints.

$$\sum_{r \in \Omega} \sum_{j:(i,j) \in A} b_{ij}^r \lambda^r \leq 1 \qquad \forall i \in M \tag{13}$$

with dual prices $(\pi_i)_{i \in M}$.

Moreover, by exploiting the SVCs, we can reduce the solution space of the subproblem by imposing that all products K_i^* are purchased directly at the point when supplier $i \in M$ is visited because of Property 2.

Property 2. *If the triangle inequality holds for travel costs (c_{ij}) then there exists an optimal solution to UMVTPP with SVCs in which at least* $\max\{1, |K_i^*|\}$ *products are purchased at every visited supplier $i \in M$.*

Proof. See (Riera-Ledesma and Salazar-González, 2012). □

To this end, we divide all products into *unique* and *common products*. Unique products $\bigcup_{i \in M} K_i^*$ have to be purchased and can be directly incorporated into the reduced cost of a partial path by using routing costs

$$c_{ij}^{req} = c_{ij} + \frac{1}{2} \cdot \left(\sum_{k \in K_i^*} \bar{p}_{ki} + \sum_{k \in K_j^*} \bar{p}_{kj} \right) + \frac{1}{2} \cdot (\pi_i + \pi_j)$$

in (9b) instead of c_{ij} (we define $\pi_0 = \pi_{m+1} = 0$ for the depot vertices 0 and $m+1$). Note that the new reduced routing costs c_{ij}^{req} can be negative. Moreover, for symmetric c_{ij} the given definition of c_{ij}^{req} implies that these are then also symmetric.

The labeling algorithm takes the unique and common products into account as follows: For all unique products $k \in K_i^*, i \in M$, we set $q_{ki} = 0$ and $\bar{p}_{ki} = \infty$ to exclude unique products from being selected as common products, i.e., they do not occur in the sequence O of a label.

To keep track of the number of unique products purchased, we introduce a new attribute T^{uniq} for the labels. When propagating a label over an arc $(i,j) \in A$, the new attribute is updated via $T_j^{uniq} = T_i^{uniq} + |K_j^*|$. The minimum number of collected common products is $T_j^{num} = T_i^{num} + 1$ if $j \neq m+1$ and $K_j^* = \varnothing$, otherwise $T_j^{num} = T_i^{num}$ (this update rule replaces (9c)). The new label has the following additional feasibility conditions (in addition to $T_j^{num} \leq Q$, i.e., condition (10)): Not more than Q unique products can be purchased, i.e., $T_j^{uniq} \leq Q$, and the maximum number of common products purchased is now bounded by $T_j^{num} \leq Q - T_j^{uniq} = Q - T_i^{uniq} - |K_j^*|$.

Regarding domination, a label L_1 dominates another label L_2, if in addition to (11a) and (11b) the condition $T_1^{uniq} \leq T_2^{uniq}$ holds.

For the bidirectional labeling, the monotone resource is redefined as $T^{num} + T^{uniq}$ with identical half-way point $Q/2$ as before. Moreover, the merge conditions

$$T_{fw}^{num} + T_{bw}^{num} \leq \begin{cases} Q+1, \text{ if } |K_i^*| = 0 \\ Q, \text{ otherwise} \end{cases} \tag{14a}$$

$$T_{fw}^{uniq} + T_{bw}^{uniq} \leq Q + |K_i^*| \tag{14b}$$

have to be tested instead of (12a). Condition (14a) and (14b) refers respectively to the common and unique products. Note that, in the second constraint (14b), both the forward label and the backward label already take the unique products K_i^* into account that are only available at the merge vertex i.

Incompatibilities between Products. Gendreau *et al.* (2016) deal with incompatibilities between products directly in the labeling algorithm. They introduce additional resources that guarantee that only compatible products are selected along the partial path.

We propose another approach to cope with incompatibilities: We enumerate all maximal subsets of products that are pairwise compatible. These are the maximal independent sets I of the incompatibility graph (K, E_w), where an edge $\{k, k'\} \in E_w$ exists between products if and only if $(k, k') \in B$; see Section 1. Let $\mathcal{I} = \{I_1, I_2, \ldots, I_b\}$ be the set of all maximum independent sets. The computation of the set $\mathcal{I}$ can be done with a straightforward enumeration algorithm.

The pricing subproblem can now be decomposed into b subproblems, where in the subproblem to I_j for $j \in \{1, 2, \ldots, b\}$ only the products $I_j \subset K$ are available. For all suppliers, their product set K_i is reduced to $K_i \cap I_j$. If the set is empty and the triangle inequality for routing costs holds, the supplier becomes obsolete. Note that in the presence of incompatibility constraints, the same supplier may be visited by multiple vehicles and it is therefore not possible to impose SVCs and distinguish between unique and common products.

On the positive side, the b different column-generation subproblems are of the same structure as those discussed before. They can be solved with the labeling algorithm presented in Section 2.2.1. Due to the reduced product and supplier sets, one can expect that these subproblems can be solved faster than one subproblem with additional resources to cope with incompatibilities. On the downside, instead of one subproblem, there are $|\mathcal{I}|$ different subproblems. To avoid solving $|\mathcal{I}|$ subproblems in each iteration, partial pricing (Gamache *et al.*, 1999) can be used to accelerate the solution process. We terminate the pricing when a certain number of negative reduced-cost columns is found. However, all b subproblems must be solved in the final column-generation iteration to prove optimality of the linear relaxation (per branch-and-bound node).

Maximum Route Duration Constraints. The integration of a maximum route duration constraint is simple. We add the attribute T^{length} to the attributes vector T of each label $L = (i, S, T)$. Initially the attribute is set to $T^{length} = 0$. Extensions along an arc $(i, j) \in A$ increase the attribute by d_{ij}, i.e., $T_j^{length} = T_i^{length} + d_{ij}$. The extension is feasible if $T_j^{length} \leq D^{max}$. Dominance between two labels $L = (i, S_1, T_1)$ and $L = (i, S_2, T_2)$ has to additionally check $T_1^{length} \leq T_2^{length}$. The merge of a forward label $L_{fw} = (i, S_{fw}, T_{fw})$ and $L_{bw} = (i, S_{bw}, T_{bw})$ must additionally test $T_{fw}^{length} + T_{bw}^{length} \leq D^{max}$.

2.3. Valid Inequalities and Cutting Strategy

For the UMVTPP with SVCs, the initial RMP that we use does not contain the SVCs (13). Instead, we add SVCs dynamically, i.e., when we find them violated.

In addition, four classes of valid inequalities are dynamically separated and added to the RMP in order to strengthen the linear relaxation. These inequalities require different properties of the problem that are summarized in Table 3.

	Prerequisite		
Valid inequalities	Unitary demand	Single-visit constraints (SVCs)	Triangle Ineq. for costs c_{ij}
(Limited memory) subset-row inequalities, (lm)SRIs		•	
Rounded ring-capacity inequalities, RRCIs	•		
Ci-ring-capacity inequalities, CiRCIs	•		
Purchase-visit inequalities, PVIs	•	•	•

Table 3: Prerequisites for the validity of inequalities

Subset-row inequalities (SRIs) were originally introduced by Jepsen *et al.* (2008) for the vehicle routing problem with time windows. This class of inequalities can only be used for UMVTPP variants with SVCs. In the UMVTPP, each SRI is defined for a subset $U \subset M$ of the suppliers. As proposed by Jepsen *et al.* (2008), we restrict ourselves to SRIs defined on three suppliers, i.e., $|U| = 3$, because they can be separated by straightforward enumeration. The corresponding SRI is

$$\sum_{r \in \Omega} \left\lfloor \frac{g_r^U}{2} \right\rfloor \lambda^r \leq 1 \tag{SRI}$$

where $g_r^U = \sum_{i \in U} a_{ir}$ is the number of times route $r \in \Omega$ visits suppliers in U.

SRIs comprise a family of non-robust cuts meaning that for each active SRI, i.e., with non-zero dual price, one binary attribute $T^{SR,U}$ must be added to the labels. The initial value of $T^{SR,U}$ is zero and it is flipped (from zero to one, and vice versa) every time a supplier $i \in U$ is visited. The (non-positive) dual price of the SRI is subtracted when the attribute flips from one to zero. A higher value $T_1^{SR,U} = 1$ in a first label $L_1 = (i, S_1, T_1)$ compared to $T_2^{SR,U} = 0$ in a second label $L_2 = (i, S_2, T_1)$ can be compensated by replacing the cost comparison (11a) by $T_1^{cost} + f(S_1) - \sum_{U: T_1^{SR,U} > T_2^{SR,U}} \alpha^U \leq T_2^{cost} + f(S_1 \cup S_2)$, where $\alpha^U \leq 0$ is the dual price of the SRI related to the subset U. Even with this clever dominance rule of Jepsen *et al.* (2008), the presence of many SRIs for different subsets U often drastically increases the practical difficulty of the pricing problem.

To alleviate these negative effects, Pecin *et al.* (2017) introduced *limited memory SRIs* (lmSRIs) that are a generalization of SRIs and whose impact on the difficulty of the pricing subproblem is typically reduced by using a U-specific *memory* to store a given subset $M_U \subseteq M$ of the suppliers. The role of the memory is very similar to the neighborhoods in the *ng*-path relaxation of Baldacci *et al.* (2011). The attribute $T^{SR,U}$ is reset to zero when a vertex i outside the memory, i.e., $i \notin M_U$, is visited.

Rounded ring-capacity inequalities (RRCIs) and *Ci-ring-capacity inequalities* (CiRCIs) were introduced in the work (Baldacci *et al.*, 2007) for the CmRSP. They are defined for an arbitrary subset $S \subset M$ of suppliers. Let $K(S) = \bigcup_{i \in S} K_i$ be the union of all products available at all suppliers in S. As for other capacity cuts, RRCIs and CiRCIs impose a lower bound on the number of times the subset S must be exited:

$$\sum_{r \in \Omega} \sum_{(i,j) \in \Gamma^+(S)} b_{ij}^r \lambda^r + \sum_{r \in \Omega} \frac{\sum_{i \notin S} \sum_{k \in K(S)} \delta_{ki}^r}{|K(S)| \pmod Q} \lambda^r \geq \left\lceil \frac{|K(S)|}{Q} \right\rceil \tag{RRCI}$$

and

$$\sum_{r\in\Omega}\sum_{(i,j)\in\Gamma^+(S)} b_{ij}^r\lambda^r \geq \left\lceil \frac{|\{k\in K : M_k\subseteq S\}|}{Q}\right\rceil. \tag{CiRCI}$$

Note that RRCIs result from some non-trivial inequalities that round fractions of integers. Hence they are difficult to interpret and we refer to (Baldacci *et al.*, 2007) for the description and proof. In contrast, CiRCIs are simple to interpret: the left-hand side counts the number of times the subset S is exited, while the nominator on the right-hand side counts the number of products that can only be purchased at suppliers inside S so that the right-hand side term is the minimum number of vehicles needed to collect these products. For the separation of violated RRCIs and CiRCIs, we use the heuristic procedure suggested by Baldacci *et al.* (2007).

Finally, for UMVTPP with SVCs, in which the triangle inequality for the routing costs holds, we introduce a new class of valid inequalities that we denote as *purchase-visit inequalities* (PVIs). This class of inequalities exploits Property 2, i.e., that at each visited supplier $i \in M$ at least $\max\{1, |K_i^*|\}$ products must be purchased. The inequality for supplier $i \in M$ is then given by

$$\sum_{r\in\Omega}\left(\max\{1,|K_i^*|\}\sum_{(i,j)\in\Gamma^+(i)} b_{ij}^r - \sum_{k\in K_i}\delta_{ki}^r\right)\lambda^r \leq 0. \tag{PVI}$$

For lmSRIs, RRCIs, CiRCIs, and PVIs, we use the following separation strategies. All inequalities are separated in hierarchical order, and the next class is only separated if we have found no violated inequalities of the previous class. The search for violated inequalities is limited to search-tree nodes of the overall branch-and-bound tree up to a certain level $\max^{level}$. For each round of separation, only the $\max^{round}$ most violated inequalities found are added before the RMP is re-optimized. The total number of separated inequalities per class is limited to $\max^{total}$. Table 4 summarizes the values we used in our experiments.

	PVIs	CiRCIs	RRCIs	lmSRIs
Order	1	2	3	4
Separated up to level $\max^{level}$	∞	10	10	1
Maximum number $\max^{round}$ per round	$\|M\|$	30	30	20
Maximum number $\max^{total}$ in total	$\|M\|$	300	300	60

Table 4: Parameters of the separation strategy for the four classes of valid inequalities

2.4. Dynamic Neighborhood Extension

Dynamic neighborhood extension (DNE) is a strategy to initialize and modify *ng*-neighborhoods N_i depending on the solutions obtained at branch-and-bound nodes. DNE was first proposed by Roberti and Mingozzi (2014). The enlargement of neighborhoods can be interpreted as a tool to strengthen the linear relaxations. Note that a basic tradeoff exists between the strength of the lower bounds resulting from *ng*-neighborhoods N_i of given size and the required computation time for the solution of the corresponding subproblems.

We implement DNE with the help of two integer parameters sz_{init}^{ng} and sz_{max}^{ng}. Recall that cycles are the result of considering SVCs (and distinguishing between unique and common products) and adding cutting planes and branching constraints. As a consequence, negative reduced cost cycles at the root node certainly include some supplier i at least twice that offers a unique product, i.e., $K_i^* \neq \varnothing$. Therefore, we add to the initial neighborhood of $j \in M$ the sz_{init}^{ng} closest suppliers i with $K_i^* \neq \varnothing$, so that $|N_i| \leq sz_{init}^{ng}$ holds for all $i \in M$.

These initial neighborhoods are systematically enlarged considering the solution of linear relaxations at each node of the search tree. If a fractional solution contains a route with a cycle $D = (i, \ldots, j, \ldots, i)$, we try to eliminate this cycle by adding i to all neighborhoods of vertices j in the cycle. Before actually adding supplier i, we check that all neighborhoods N_j admit the addition, i.e., we check $|N_j| < sz_{max}^{ng}$. Given the solution of the current RMP defined over $\bar{\Omega}$, let $\bar{\lambda}_r$, $r \in \bar{\Omega}$, be the value of the corresponding variable λ_r. Cycles in routes r are considered in decreasing order of the values $\bar{\lambda}_r$. Our implementation uses $sz_{init}^{ng} = 2$ and $sz_{max}^{ng} = 8$.

2.5. Branching

Let $\bar{\lambda}_r$ for $r \in \bar{\Omega}$ and $\bar{f}$ be the values of the corresponding variables λ_r and f in the current solution of the RMP defined over $\bar{\Omega}$. If some of these values are fractional, branching is required.

We apply a six-stage hierarchical branching scheme to ensure integer solutions of formulation (1). For the UMVTPP with SVCs, the third stage is obsolete. Table 5 provides an overview.

Stage	SVCs with	SVCs w/o	Description	Branching on	Variable selection	Branches
1	•	•	Number of vehicles	$\bar{f}$	—	$\leq \lfloor \bar{f} \rfloor$ and $\geq \lceil \bar{f} \rceil$
2	•	•	Selection of a supplier	$\alpha_{ki}, \quad k \in K, i \in M_k$	closest to 0.5	$= 0$ and $= 1$
3		•	Number of visits	$\gamma_i, \quad i \in M$	$\gamma_i - \lfloor \gamma_i \rfloor$ closest to 0.5	$\leq \lfloor \gamma_i \rfloor$ and $\geq \lceil \gamma_i \rceil$
4	•	•	Slack of CiRCI	$\beta_S, \quad S \subset M$	< 1, closest to 0.5	$= 0$ and ≥ 1
5	•	•	Arc Flow	$\eta_{ij}, \quad (i,j) \in A$	$\eta_{ij} - \lfloor \eta_{ij} \rfloor$ closest to 0.5	$\leq \lfloor \eta_{ij} \rfloor$ and $\geq \lceil \eta_{ij} \rceil$
6	•	•	Flow splitting	see (Feillet *et al.*, 2005) for details, not necessary in our experiments		

Table 5: Six-stage branching strategy for the UMVTPP

First, if the number of vehicles $\bar{f}$ in use is fractional, we create two branches enforcing either $f \leq \lfloor \bar{f} \rfloor$ or $f \geq \lceil \bar{f} \rceil$.

Second, we introduce a new branching rule that decides whether product $k \in K$ is purchased at supplier $i \in M$, or not (the use of this branching rule is new compared to Hoshino and De Souza, 2012; Gendreau *et al.*, 2016). For all products $k \in K$ with $|M_k| > 1$ and all suppliers $i \in M_k$, we consider fractional values α_{ki} defined as $\sum_{r \in \Omega} \delta_{ki} \bar{\lambda}_r$. We choose a pair (k^*, i^*) with value $\alpha_{k^* i^*}$ closest to 0.5 and create two branches $\sum_{r \in \Omega} \delta_{ki} \lambda_r = 0$ and $\sum_{r \in \Omega} \delta_{ki} \lambda_r = 1$. Instead of adding constraints, we directly implement the branching constraints by manipulating the product set of the suppliers. In the first branch, product k^* is removed from K_{i^*}. In the second branch, product k^* is removed from K_i for all other suppliers $i \in M_k \setminus \{i^*\}$; this ensures that product k^* is purchased at supplier i^*. All route variables that do not comply with the branching decision are removed from the RMP.

Third, we branch on the number of times a supplier $i \in M$ is visited if this value is fractional. Accordingly, we define γ_i as $\sum_{r \in \Omega} \sum_{(i,j) \in \Gamma^+(i)} b_{ij}^r \bar{\lambda}^r$. Among all suppliers $i \in M$ with fractional γ_i, we choose a supplier i^* with $\gamma_{i^*} - \lfloor \gamma_{i^*} \rfloor$ closest to 0.5 and create two branches with either

$$\sum_{r \in \Omega} \sum_{(i,j) \in \Gamma^+(i)} b_{ij}^r \lambda^r \leq \lfloor \gamma_{i^*} \rfloor \qquad \text{or} \qquad \sum_{r \in \Omega} \sum_{(i,j) \in \Gamma^+(i)} b_{ij}^r \lambda^r \geq \lceil \gamma_{i^*} \rceil.$$

Note that this branching rule is obsolete for UMVTPP with SVCs when the second-stage branching rules are applied before.

Fourth, as suggested by Baldacci *et al.* (2007), we branch on the slack in CiRCIs. Recall that for any subset $S \subset M$, the corresponding CiRCI ensures that the subset S is exited sufficiently often. Defining β_S as $\sum_{r \in \Omega} \sum_{(i,j) \in \Gamma^+(S)} b_{ij}^r \bar{\lambda}^r - \left\lceil \frac{|\{k \in K : M_k \subseteq S\}|}{Q} \right\rceil$, we consider only those subsets S with a value β_S strictly between 0 and 1. We use the heuristic proposed by Baldacci *et al.* (2007) to identify candidate sets S. Among all candidate sets, we choose a subset S^* with a value β_{S^*} closest to 0.5 and create two branches by adding the additional constraints that either the slack is zero or at least one. The constraints defining these branches can be written as

$$\sum_{r \in \Omega} \sum_{(i,j) \in \Gamma^+(S^*)} b_{ij}^r \lambda^r - \left\lceil \frac{|\{k \in K : M_k \subseteq S^*\}|}{Q} \right\rceil = 0 \quad \text{or} \quad \geq 1.$$

Fifth, we branch on the number of times an arc $(i,j) \in A$ is traversed. Let η_{ij} be defined by $\sum_{r \in \Omega} b_{ij}^r \bar{\lambda}^r$, i.e., the number of times arc $(i,j) \in A$ is used in the current solution. Among all arcs with fractional value η_{ij}, we choose the arc (i^*, j^*) with $\eta_{i^* j^*}$ closest to 0.5. We create two branches by forcing $\sum_{r \in \Omega} b_{ij}^r \lambda^r \leq \lfloor \eta_{i^* j^*} \rfloor$ or $\sum_{r \in \Omega} b_{ij}^r \lambda^r \geq \lceil \eta_{i^* j^*} \rceil$. This branching rule reduces to standard binary arc branching for the UMVTPP with SVCs.

Note that integer flows on arcs may not guarantee that the route variables are integer for UMVTPPs without SVCs. Therefore, the sixth stage applies the additional branching rule based on the flow-splitting method introduced by Feillet *et al.* (2005) and later used also by Gendreau *et al.* (2016). In our computational experiments, it was never necessary to apply this rule because fractional route variables could always be convex combined into feasible integer solutions (see, Desaulniers *et al.*, 1998; Jans, 2010). We attribute this behavior to the use of the additional second-stage branching rule not used by Gendreau *et al.* (2016). We doubt however that the first five stages already guarantee, in general, that the branching is complete.

3. Computational Results

In this section, we report the results of computational experiments that were conducted on three standard benchmark sets (Section 3.1) for different UMVTPP variants. In Section 3.2, we introduce additional acceleration techniques based on partial pricing and compare the new labeling algorithm against a MIP-solver-based pricing algorithm. The impact of our new branching rule is evaluated in Section 3.3. Finally, we compare our BPC algorithm with algorithms from the literature in Section 3.4.

We have implemented the BPC algorithm in C++ and compiled into 64-bit single-thread code with MS Visual Studio 2015. The callable library of CPLEX 12.9.0 was used for (re-)optimizing the RMPs and for solving the pricing problem directly via MIP-solver by addressing model (2). All results were obtained using a standard PC with an Intel® Core™ i7-5930K processor clocked at 3.5 GHz and 64 GB RAM running Microsoft Windows 10 Education. In the following, all computation times are given in seconds.

3.1. UMVTPP Instances

We test our BPC algorithm on the CmRSP instances created by Baldacci *et al.* (2007) and Hoshino and De Souza (2012) as well as on the MVTPP-PIC instances of Gendreau *et al.* (2016). Several comments on these benchmarks sets and previous results presented in the literature are due:

1. The CmRSP instances of Baldacci *et al.* (2007) have been used in previous studies only in a restricted version, where a *canonical ring-star property* has been assumed. Hoshino and De Souza (2012, p. 2734) already mention that Baldacci *et al.* assume canonical ring stars in which a Steiner point appears in a ring only if there exists a connection arc in the star linking some customer to it. Only if routing costs satisfy the triangle inequality, there always exists an optimal solution composed exclusively of canonical ring stars. Otherwise, all optimal solutions may have non-canonical ring stars. Since the triangle inequality is not fulfilled in this benchmark set, the solutions reported in (Baldacci *et al.*, 2007) may not be optimal. Moreover, in order to speed up the convergence of the solution process, the authors provided their BC algorithm with initial feasible solutions (computed by applying at pre-processing time an initial stand-alone heuristic) whose values were, on average, less than 0.7 % away from the optimal (dual bound) values. Finally, it should also be noted that capacities have been chosen large for some of these instances, which makes them somewhat harder for column-generation based algorithms but simpler for BC algorithms.
2. Hoshino and De Souza (2012) also forced the ring stars to be canonical to be in conformity with Baldacci *et al.*. However, they adopt the instances of Baldacci *et al.*, such that the triangle inequality is satisfied in their new benchmark set. As their BC results are inferior to their BPC results, we do not include the BC results into our comparison in Section 3.4. Additionally, in order to further assess their BPC algorithm, the authors run a subset of experiments considering the instances proposed by Baldacci *et al.*. Their aggregated results show that the BPC algorithm solves less instances to optimality compared to the BC results of Baldacci *et al.*.
3. Riera-Ledesma and Salazar-González (2012, 2013) extended the original benchmark set comprising the two classes A and B of Baldacci *et al.* by three new classes C, D, and E. Unfortunately, these instances are not obtainable anymore (Riera-Ledesma, 2019).
4. Instances of the MVTPP-PIC benchmark set of Gendreau *et al.* (2016) do not fulfill the triangle inequality for routing costs. Therefore, Property 1 does not hold true, i.e., visits to suppliers without actually purchasing any product can be beneficial. In addition, their BP algorithm has been prematurely terminated whenever the optimality gap has been fallen below 0.2 percent. Hence, some non-optimal solutions are reported in their results.

Finally, we would like to mention that the instances of Bianchessi *et al.* (2014) were created and tested only for uncapacitated variants of the MVTPP. In this case, the purchasing decisions can be transferred to the master-problem level of a column-generation approach (as done by the authors). This makes these instances irrelevant for any approach that explicitly considers capacities on the subproblem level. We therefore disregard these instances.

3.2. Comparison of Labeling-based and MIP-Solver-based Pricing Algorithms

To accelerate the solution process, we use *partial pricing* (PP) with reduced networks. A reduced network is defined by a value $\alpha \in \mathbb{N}$ that limits the number of ingoing and outgoing arcs per vertex. Arcs are chosen according to the lowest reduced cost in the current pricing iteration.

Additionally, we use two heuristic dominance rule. In the first rule ('rule 1'), condition (11a) is replaced by

$$T_1^{cost} + f(S_1) \leq T_2^{cost} + f(S_2)$$

and in the second rule ('rule 2'), condition (11a) is replaced by

$$T_1^{cost} + f(S_1) \leq T_2^{cost} + \beta f(S_1 \cup S_2)$$

with some value β between 0 and 1. In our computational experiments, we used six heuristic pricers. Their configuration is summarized in Table 6.

Level	Number α of arcs	Dominance rule	Factor β in rule 2
1	20	rule 1	–
2	∞	rule 1	–
3	30	rule 2	0.85
4	30	rule 2	0.95
5	10	exact	1
6	20	exact	1

Table 6: Configuration of heuristic pricers

Due to the restricted and hence smaller set of products and suppliers per pricing subproblem in the MVTPP-PIC (see Section 2.2.2), we skip heuristic pricers at levels 1, 3, and 4.

Next, we evaluate the impact of the PP strategy on the performance of our new labeling algorithm compared to solving formulation (2) directly with CPLEX as the exact pricing method. Note that Gendreau *et al.* (2016) use a similar strategy that can be described as 'CPLEX combined with labeling-based partial pricing'.

It is non-trivial to design the computational experiment such that the comparison is fair. Note first that paths resulting from solving model (2) with a MIP-solver are always elementary 0-$(m+1)$-paths, while the labeling algorithm using the *ng*-path relaxation typically produces non-elementary paths. As a result, the corresponding root node lower bounds are different, making it also difficult to compare computation times.

Comparing different pricing schemes based on the performance of the entire BPC algorithm is also not a good option. The point here is that already slightly different trajectories of the column-generation algorithm at the root node can lead to completely different branching decisions (due to primal degeneracy) and herewith to substantially different branch-and-bound trees. As a result, observed computation times and related results have a higher variance and are therefore often less statistically significant. Hence, comparisons of computational setups on the basis of the linear-relaxation results are preferable.

For the UMVTPP, there are at least two possibilities to reach the elementary lower bound at the root node even when labeling and the *ng*-path relaxation are used: The first possibility is to consider all products as common products so that no product is unique, see Section 2.2.2. Then, all reduced costs of arcs are non-negative at the root node, ensuring that only elementary paths are Pareto-optimal in the labeling. The downside of this simple approach is that the maximum number of common products to purchase along a path increases. The consequence is more time-consuming labeling.

The second possibility is to dynamically extend the *ng*-neighborhoods, as suggested by Roberti and Mingozzi (2014). If the preliminary root-node solution contains cycles, the *ng*-neighborhoods of all vertices in one or several cycles are extended so that the cycles are eliminated from the next root-node solution. Iterating this process finally guarantees elementary paths, at the cost of large *ng*-neighborhoods and expectedly longer computation times for a single pricing problem.

In total, we compare seven different computational setups corresponding to different pricing algorithms and resulting from the combination of the following three options:

- either considering all products as common products (`CP`) or considering dynamic *ng*-neighborhoods (`DNG`) in order to ensure elementary routes;
- with or without labeling-based partial pricing (`PP`);
- exact pricing either MIP-solver-based (`CPLEX`) or labeling-based (`LABELING`).

Note that `CPLEX` without `PP` only produces elementary routes, making combinations with `CP` and `DNG` redundant, as they lead to identical results. Table 7 presents aggregated results for the seven setups. It contains, for each class of instances, the number of instances (#), the number of times the elementary linear relaxation is solved to optimality (#LP), and the average computation time (Time). For the latter, we only take into account only those instances that are solved by at least one setup, i.e., 57, 168, and 70 instances for the three benchmark sets, respectively. We use a time limit of 600 seconds for each instance.

		CPLEX		CP+LABELING		CP+PP+CPLEX		CP+PP+LABELING	
Instance class	#	#LP	Time	#LP	Time	#LP	Time	#LP	Time
Baldacci *et al.*	90	—	—	36	233.3	52	86.2	51	86.0
Hoshino and De Souza	234	—	—	120	188.9	131	168.8	159	54.3
Gendreau *et al.*	72	—	—	69	46.5	67	116.4	70	33.6
Total	396	—	—	225	163.7	250	140.4	280	55.5
		CPLEX		DNG+LABELING		DNG+PP+CPLEX		DNG+PP+LABELING	
Instance class	#	#LP	Time	#LP	Time	#LP	Time	#LP	Time
Baldacci *et al.*	90	42	214.0	42	164.4	56	81.9	56	29.5
Hoshino and De Souza	234	95	311.1	144	108.4	128	177.0	168	22.8
Gendreau *et al.*	72	44	298.4	69	43.5	67	120.5	70	33.6
Total	396	181	289.3	255	103.8	251	145.2	294	26.7

Table 7: Comparison of different pricing setups reaching the elementary lower bound at the root node

First of all, comparing the upper and lower part of Table 7 reveals that pricing algorithms using DNG outperforms the corresponding variants considering CP for reaching the elementary lower bound on most classes of instances. Exceptions are instances of Hoshino and De Souza, where DNG+PP+CPLEX solves only 128 linear relaxations compared to 131 solved by CP+PP+CPLEX. Moreover, on the instances of Gendreau *et al.* differences are relatively small, which can be explained by the fact that in these instances almost no unique products exist so that the majority of the routes generated by labeling are elementary. As a side note, we mention that results are very consistent in the sense that there is no single instance solved by considering CP that is not solved with DNG. We conclude that DNG should always be used in pure labeling-based pricing algorithms.

Moreover, we can see from the lower part of Table 7 that partial pricing significantly accelerates the solution process on all instance classes and for both CPLEX and DNG+LABELING. The results show that DNG+LABELING is superior to CPLEX on the instances of Hoshino and De Souza and Gendreau *et al.* For the instances of Baldacci *et al.*, DNG+LABELING performs only slightly better than CPLEX. Same considerations can be drawn by comparing DNG+PP+LABELING against DNG+PP+CPLEX. In particular, for the instances of Baldacci *et al.*, results are not clear cut, i.e., DNG+PP+CPLEX provides one solution not computed in 600 seconds by DNG+PP+LABELING, and vice versa. Besides, we have compared the average number of pricing iterations: Numbers are comparable for DNG+PP+CPLEX and DNG+PP+LABELING as well as for CPLEX and DNG+LABELING.

			DNG+PP+CPLEX		DNG+PP+LABELING	
Instance class	CPLEX	DNG+LABELING	Heuristic Pricing	Exact Pricing	Heuristic Pricing	Exact Pricing
Baldacci *et al.*	100 %	100 %	22 %	77 %	65 %	30 %
Hoshino and De Souza	100 %	98 %	9 %	91 %	57 %	38 %
Gendreau *et al.*	100 %	94 %	22 %	77 %	79 %	17 %
Total	100 %	97 %	15 %	84 %	65 %	30 %

Table 8: Percentage of the average time spent on labeling compared to the total computation time; averages are arithmetic means computed over instances consuming at least 1 second for the total computation and solved by at least one setup

Finally, we analyze the share of the total computation time (for solving the linear relaxation) spent on pricing for CPLEX and the three algorithms using DNG. Note that other algorithmic components such as the simplex algorithm for re-optimizing the RMP and the *ng*-neighborhoods extension consume only a small share of a BPC algorithm's total computation time. Table 8 shows the percentage of the total computation time spent on pricing grouped for the three classes of instances. For the most time-consuming algorithm, CPLEX, the MIP-solver consumes approximately (rounded by two digits) 100 % of the total time. We see for algorithm DNG+LABELING that the instances of Baldacci *et al.* are the hardest when solved with labeling. With partial pricing, the share of pricing time on the total computation time drops to 99 % (15 % + 84 %)

and 95 % (65 % + 30 %), respectively, i.e., other algorithmic components start consuming a noticeable share of the total time. Interestingly, the relationship between heuristic and exact pricing time is reversed when comparing `DNG+PP+CPLEX` and `DNG+PP+LABELING`. We interpret these numbers as a clear indication that exact pricing with our new labeling algorithm is highly effective.

As a consequence of the results provided in this section, we will conduct all following experiments by using the pricing algorithm corresponding to computational setup `PP+LABELING`.

3.3. Branching

In this section, we evaluate the impact of the new branching rule *Selection of a Supplier* (SoaS), i.e., the branching rule at level 2 in Table 5. Table 9 presents aggregated results for the three benchmark sets when either SoaS is used or omitted. For the comparison, we take into account only those instances that were solved to integer optimality by one of the two configurations and were branching was necessary. The table contains, for both configurations and all instance classes, the number of considered instances (#), the number of instances solved to optimality (#Opt), the average solution time (Time), and the number of evaluated branch-and-bound nodes (#B&B).

		without SoaS			with SoaS		
Instance class	#	#Opt	Time	#B&B	#Opt	Time	#B&B
Baldacci *et al.*	21	19	1337.2	123.2	21	755.7	166.4
Hoshino and De Souza	67	67	151.0	57.5	67	161.3	48.8
Gendreau *et al.*	35	34	523.2	92.4	35	411.3	113.6
Total	123	120	670.5	91.0	123	442.8	109.6

Table 9: Comparison of branching without and with the new rule SoaS that selects a supplier for the purchase of a product

In total, the configuration with the new branching rule SoaS performs better, because it solves more instances and the average computation time is smaller, in particular, for the classes Baldacci *et al.* and Gendreau *et al.* For the Hoshino and De Souza instances, both configurations perform almost similarly with identical optima and a slight difference in computation time. We observe that SoaS decisions on average lead to slightly simpler pricing problems as the number of products per supplier reduces. This seems to be the reason why branching with SoaS finally performs better.

3.4. Results for the CmRSP and the MVTPP-PIC Instances

In this section, we compare our BPC algorithm on instances for the CmRSP and the MVTPP-PIC with the state-of-the-art algorithms from the literature. Tables 10–12 present aggregated results for the instances of Baldacci *et al.* (2007), Hoshino and De Souza (2012), and Gendreau *et al.* (2016). Detailed instance-wise results can be found in the Appendix. We set the same time limit as in the literature: 7200 seconds for the instances of Baldacci *et al.* (2007) and Gendreau *et al.* (2016) as well as 1800 seconds for the instances of Hoshino and De Souza (2012).

According to the clock rate of the different processors used, the solution times reported in Tables 10 and 11 for the BC proposed Baldacci *et al.* (2007) and the BPC proposed by Hoshino and De Souza (2012), respectively, should be multiplied by a factor 0.63 and 0.97. (No measure of the clock rate has been provided by Gendreau *et al.*.) However, this kind of re-set of the solution times is quite risky and could lead to misleading comparisons. In order to fairly re-set values, it would be necessary to consider several other features of the computational environment such as, among the most important, the available RAM, the number of processor cores allowed to be used in parallel, the MIP solver used to solve linear programs, the version of the MIP solver, the setting used to run the MIP solver, the programming language, the compiler, and the operating system of the machine under which experiments are run.

Moreover, the use of primal heuristics can significantly reduce the solution times of exact algorithms. With this respect, Baldacci *et al.* (2007) use two primal heuristics in order to speed up the convergence of the proposed BC algorithm. The first heuristic is applied in a preprocessing phase. The second uses the information of the fractional LP solution to build a feasible primal solution, and it is applied at the first N nodes of the enumeration tree. Both heuristics (in particular, the second on the basis of the fractional LP solution at the root node) compute feasible upper bounds with an average errors less than 0.7% with respect to the corresponding optimal values (see Baldacci *et al.*, 2007, Table 4). Primal heuristics are also applied

in Gendreau *et al.* (2016). In a preprocessing phase, a simplified version of the four-step heuristic proposed in Manerba and Mansini (2015) is run. Then, a restricted master heuristic (Joncour *et al.*, 2010) is executed at each column-generation iteration. On the contrary, no primal heuristic is applied in our BPC algorithm and in the BPC algorithm proposed by Hoshino and De Souza (2012). To the sake of completeness, we have to mention that a primal heuristic is used in Hoshino and De Souza (2012) to initialize the RMP at least at the root node of the branch-and-bound tree. Then, the values of the primal solutions computed by the heuristic are somehow considered in the final results: "The value of the best integer solution found by all the algorithms, including that of the primal heuristic used to populate the initial basis, is presented in column z^*." (Hoshino and De Souza, 2012, p. 2735). For all this reasons, we report solution times in Tables 10–12 without applying any re-scaling factor.

Tables 10–12 contain for each instance class, the number of instances (#), the number of instances solved to optimality (#Opt), the average solution time in seconds, and the number of branch-and-bound nodes solved (#B&B).

		Baldacci *et al.*			Our algorithm		
Instance	#	#Opt	Time	#B&B	#Opt	Time	#B&B
grouped by class							
A25	9	9	0.7	1.1	9	0.3	1.0
A50	12	12	570.6	323.9	10	1659.2	27.3
A75	12	8	2942.1	590.8	4	4950.4	225.8
A100	12	6	4355.4	434.0	3	5424.9	1.8
B25	9	9	0.4	1.0	9	0.3	1.0
B50	12	12	613.1	230.3	12	655.8	22.7
B75	12	4	4976.5	597.1	3	5418.8	644.3
B100	12	3	5483.5	406.3	3	5458.8	17.7
grouped by vehicle capacity							
3–6	26	26	67.9	14.3	26	3.3	3.5
7–10	20	19	415.9	54.8	19	511.7	532.3
11–14	18	11	3866.2	581.3	7	5027.9	29.7
≥15	26	7	5678.0	733.8	1	7000.0	0.8
Total	90	63	2525.6	344.5	53	3142.4	125.5

Table 10: Comparison with the branch-and-cut (BC) algorithm of Baldacci *et al.* (2007)

The first part of Table 10 contains aggregated results in which the instances are grouped by class (A or B) and number of suppliers (25 to 100). Afterwards, in the second part of the table, we regroup the instances by vehicle capacity.

In the first part, we can see that our algorithm performs worse on the A instances, while it performs nearly equally on the B instances. From the second part we can see that both algorithms operate similarly on instances with a vehicle capacity up to 10. The BC algorithm of Baldacci *et al.* (2007) is slightly better on instances with vehicle capacity between 11 and 14. However, we can provide two new optimal solutions for that group. As expected, our algorithm performs poorly for instances that have a vehicle capacity larger than 15. All in all, we can solve 10 instances less and our solution time increases on average.

Compared with the BPC algorithm of Hoshino and De Souza (2012), our algorithm performs slightly better on average. For the small instance class, both algorithms work similarly. The two instances of class small C that we cannot solve (solved by the BPC of Hoshino and De Souza (2012)) have a large vehicle capacity of 14. For the large instance classes, our algorithm performs better, in particular for the large instances of class B and C. In total, we can solve five additional instances and the solution time decreases slightly on average. It is notable that we need significantly less branch-and-bound nodes to solve the instances to optimality. Moreover, the instance-wise results show that we can provide ten optimal solutions for previously unsolved instances.

Compared to the BPC algorithm of Gendreau *et al.* (2016), our BPC algorithm performs much better. We can solve all but one instance to optimality, thereby providing 19 new optimal solutions where six of them are resulting from the preliminary stop criterion in the BPC of Gendreau *et al.*. Moreover, our algorithm is on average more than six times faster, although the detailed results show (entry #Sub of Table 21 in the

		Hoshino and De Souza			Our algorithm		
Instance class	#	#Opt	Time	#B&B	#Opt	Time	#B&B
Small A	33	32	91.8	127.2	32	93.7	4.8
B	33	32	110.4	165.8	32	152.5	22.1
C	33	29	289.7	190.5	27	331.2	8.7
Large A	45	18	1162.5	483.3	18	1118.8	16.0
B	45	14	1269.1	828.3	16	1234.2	97.1
C	45	10	1464.4	384.7	15	1234.2	38.2
Total	234	135	818.6	394.4	140	771.3	34.1

Table 11: Comparison with the branch-price-and-cut (BPC) algorithm of Hoshino and De Souza (2012)

		Gendreau *et al.*✠			Our algorithm		
Instance size $\|M\|+1$	#	#Opt	Time	#B&B	#Opt	Time	#B&B
20	24	24	869.3	87.3	24	31.1	24.3
35	24	21	1906.9	8.5	24	93.0	20.1
50	24	13	4054.0	5.6	23	881.6	122.8
Total	72	58	2276.7	33.8	71	335.2	55.7

✠: The BPC algorithm of Gendreau *et al.* has been prematurely terminated whenever the optimality gap has been below 0.2 percent.

Table 12: Comparison with the BPC algorithm of Gendreau *et al.* (2016).

Online Appendix) that there are up to 286 different pricing subproblems per instance.

4. Conclusions

In this work, we have presented a new branch-price-and-cut (BPC) algorithm for the solution of different variants of the capacitated multiple vehicle traveling purchaser problem (MVTPP) with unitary demand. The main novelty is the way in which the column-generation subproblems are solved by a dynamic-programming labeling algorithm: The precise purchasing decisions are postponed until the route has been completely defined. The decisive point here is the definition of an effective dominance rule. The novel dominance rule uses some deep-seated properties of the optimal reduced purchasing cost function, which we have proved being supermodular.

Additional components of the BPC algorithm that contributed to the overall effectiveness of the approach are implicit bidirectional and partial pricing, the incorporation of additional cutting planes, the dynamic extension of the *ng*-route neighborhoods, and new branching rules. For the UMVTPP with incompatibilities among products, the replacement of incompatibility constraints by multiple pricing subproblems has produced a BPC approach that outperforms former methods by one order of magnitude regarding computation times. In general, the computational results indicate that the new subproblem solution strategies are beneficial, especially for instances in which many products are available at a supplier and routes are relatively short.

Acknowledgement

This research was funded by the Deutsche Forschungsgemeinschaft (DFG) under grants no. IR 122/5-2 and IR 122/9-2 and by Regione Lombardia, grant agreement n. E97F17000000009, Project AD-COM.

References

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

Baldacci, R., Mingozzi, A., and Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, **59**(5), 1269–1283.

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

Bianchessi, N. and Irnich, S. (2019). Branch-and-cut for the split delivery vehicle routing problem with time windows. *Transportation Science*, **53**(2), 442–462.

Bianchessi, N., Mansini, R., and Speranza, M. (2014). The distance constrained multiple vehicle traveling purchaser problem. *European Journal of Operational Research*, **235**(1), 73–87.

Bode, C. and Irnich, S. (2012). Cut-first branch-and-price-second for the capacitated arc-routing problem. *Operations Research*, **60**(5), 1167–1182.

Choi, M. J. and Lee, S. H. (2010). The multiple traveling purchaser problem. In *The 40th International Conference on Computers Indutrial Engineering*, pages 1–5.

Christofides, N., Mingozzi, A., and Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, **20**(1), 255–282.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M., Soumis, F., and Villeneuve, D. (1998). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. G. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 57–93. Kluwer, Boston.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Springer, New York.

Feillet, D., Dejax, P., and Gendreau, M. (2005). The profitable arc tour problem: Solution with a branch-and-price algorithm. *Transportation Science*, **39**(4), 539–552.

Gamache, M., Soumis, F., Marquis, G., and Desrosiers, J. (1999). A column generation approach for large-scale aircrew rostering problems. *Operations Research*, **47**(2), 247–263.

Gendreau, M., Manerba, D., and Mansini, R. (2016). The multi-vehicle traveling purchaser problem with pairwise incompatibility constraints and unitary demands: A branch-and-price approach. *European Journal of Operational Research*, **248**(1), 59–71.

Goeke, D., Gschwind, T., and Schneider, M. (2019). Upper and lower bounds for the vehicle-routing problem with private fleet and common carrier. *Discrete Applied Mathematics*, **264**, 43–61.

Gschwind, T., Bianchessi, N., and Irnich, S. (2019). Stabilized branch-price-and-cut for the commodity-constrained split delivery vehicle routing problem. *European Journal of Operational Research*, **278**(1), 91–104.

Hoshino, E. A. and 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.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In G. Desaulniers, J. Desrosiers, and M. Solomon, editors, *Column Generation*, pages 33–65. Springer, New York.

Jans, R. (2010). Classification of Dantzig-Wolfe reformulations for binary mixed integer programming problems. *European Journal of Operational Research*, **204**(2), 251–254.

Jepsen, M., Petersen, B., Spoorendonk, S., and Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, **56**(2), 497–511.

Joncour, C., Michel, S., Sadykov, R., Sverdlov, D., and Vanderbeck, F. (2010). Column generation based primal heuristics. *Electronic Notes in Discrete Mathematics*, **36**, 695–702.

Lübbecke, M. and Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, **53**, 1007–1023.

Manerba, D. and Mansini, R. (2015). A branch-and-cut algorithm for the multi-vehicle traveling purchaser problem with pairwise incompatibility constraints. *Networks*, **65**(2), 139–154.

Manerba, D. and Mansini, R. (2016). The nurse routing problem with workload constraints and incompatible services. *IFAC-PapersOnLine*, **49**(12), 1192–1197. 8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016.

Manerba, D., Mansini, R., and Riera-Ledesma, J. (2017). The traveling purchaser problem and its variants. *European Journal of Operational Research*, **259**(1), 1–18.

Miller, C. E., Tucker, A. W., and Zemlin, R. A. (1960). Integer programming formulations and traveling salesman problem. *Journal of the ACM*, **7**, 326–329.

Pecin, D., Pessoa, A., Poggi, M., and Uchoa, E. (2017). Improved branch-cut-and-price for capacitated vehicle routing. *Mathematical Programming Computation*, **9**(1), 61–100.

Riera-Ledesma, J. (2019). Personal communication.

Riera-Ledesma, J. and 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. and 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.

Righini, G. and Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, **3**(3), 255–273.

Roberti, R. and Mingozzi, A. (2014). Dynamic ng-path relaxation for the delivery man problem. *Transportation Science*, **48**(3), 413–424.

Schrijver, A. (2003). *Combinatorial Optimization*. Springer, Berlin.

Tilk, C., Rothenbächer, A.-K., Gschwind, T., and Irnich, S. (2017). Asymmetry matters: Dynamic half-way points in bidirectional labeling for solving shortest path problems with resource constraints faster. *European Journal of Operational Research*, **261**(2), 530–539.

Online Appendix

Tables 13–21 present detailed computational results for all tested instances. The first column indicates the name of the instances, the second gives the number of suppliers and the third the number of products. Columns LB und UB contain the lower and upper bounds obtained at the end of the optimization. Columns *Time LP* and *Time IP* give the time the BPC algorithm takes for the solution of the root node and the complete IP, respectively. Column #B&B contains the number of branch-and-bound nodes solved, column #ExtNG the number of nodes at which the ng-neighborhood was extended, and column #Cuts the number of added cutting planes. Finally, for the instances with incompatibilities, column fr denotes the percentage of products that are free, i.e., the percentage of products that have no incompatibility restrictions, and column $\#Sub$ shows the number of subproblems for that instance.

Instance	$\|M\|$	$\|K\|$	UB	LB	Time LP	Time IP	#B&B	#ExtNG	#Cuts
eil26.tsp.3.12.5.A.BDS	25	12	242	242	0.1	0.1	1	1	8
eil26.tsp.4.12.4.A.BDS	25	12	261	261	0.1	0.1	1	1	7
eil26.tsp.5.12.3.A.BDS	25	12	292	292	0.1	0.1	1	1	12
eil26.tsp.3.18.7.A.BDS	25	18	301	301	0.1	0.8	1	7	57
eil26.tsp.4.18.5.A.BDS	25	18	339	339	0.1	0.1	1	1	10
eil26.tsp.5.18.4.A.BDS	25	18	375	375	0.1	0.1	1	1	17
eil26.tsp.3.25.10.A.BDS	25	25	325	325	0.2	0.8	1	8	38
eil26.tsp.4.25.7.A.BDS	25	25	362	362	0.1	0.4	1	5	26
eil26.tsp.5.25.6.A.BDS	25	25	382	382	0.1	0.1	1	1	18
eil51.tsp.3.12.5.A.BDS	50	12	242	242	0.1	0.1	1	1	8
eil51.tsp.4.12.4.A.BDS	50	12	261	261	0.1	0.1	1	1	7
eil51.tsp.5.12.3.A.BDS	50	12	286	286	0.1	0.1	1	1	8
eil51.tsp.3.25.10.A.BDS	50	25	322	322	0.5	2.0	1	4	38
eil51.tsp.4.25.7.A.BDS	50	25	360	360	0.3	1.3	1	5	53
eil51.tsp.5.25.6.A.BDS	50	25	379	379	0.2	0.3	1	1	17
eil51.tsp.3.37.14.A.BDS	50	37	373	373	7.3	491.7	18	19	71
eil51.tsp.4.37.11.A.BDS	50	37	405	405	1.8	1395.2	196	130	112
eil51.tsp.5.37.9.A.BDS	50	37	432	432	0.6	25.0	9	13	70
eil51.tsp.3.50.19.A.BDS	50	50	—	455	199.6	7200.0	8	18	119
eil51.tsp.4.50.14.A.BDS	50	50	490	490	39.1	3595.3	12	24	97
eil51.tsp.5.50.12.A.BDS	50	50	—	518	14.1	7200.0	78	72	93
eil76.tsp.3.18.7.A.BDS	75	18	330	330	0.4	1690.1	2695	227	135
eil76.tsp.4.18.5.A.BDS	75	18	385	385	0.2	4.7	8	5	62
eil76.tsp.5.18.4.A.BDS	75	18	448	448	0.2	4.9	5	9	68
eil76.tsp.3.37.14.A.BDS	75	37	—	394	2442.0	7200.0	0	2	27
eil76.tsp.4.37.11.A.BDS	75	37	—	440	178.0	7200.0	1	10	160
eil76.tsp.5.37.9.A.BDS	75	37	479	479	43.9	105.2	1	0	4
eil76.tsp.3.56.21.A.BDS	75	56	—	—	7200.0	7200.0	0	0	0
eil76.tsp.4.56.16.A.BDS	75	56	—	—	7200.0	7200.0	0	0	0
eil76.tsp.5.56.13.A.BDS	75	56	—	—	7200.0	7200.0	0	0	0
eil76.tsp.3.75.28.A.BDS	75	75	—	—	7200.0	7200.0	0	0	0
eil76.tsp.4.75.21.A.BDS	75	75	—	—	7200.0	7200.0	0	0	0
eil76.tsp.5.75.17.A.BDS	75	75	—	—	7200.0	7200.0	0	0	0
eil101.tsp.3.25.10.A.BDS	100	25	363	363	7.8	182.1	1	11	56
eil101.tsp.4.25.7.A.BDS	100	25	415	415	1.7	89.9	19	27	159
eil101.tsp.5.25.6.A.BDS	100	25	448	448	1.1	27.5	1	11	132
eil101.tsp.3.50.19.A.BDS	100	50	—	—	7200.0	7200.0	0	0	0
eil101.tsp.4.50.14.A.BDS	100	50	—	—	7200.0	7200.0	0	0	0
eil101.tsp.5.50.12.A.BDS	100	50	—	551	1540.7	7200.0	0	3	35
eil101.tsp.3.75.28.A.BDS	100	75	—	—	7200.0	7200.0	0	0	0
eil101.tsp.4.75.21.A.BDS	100	75	—	—	7200.0	7200.0	0	0	0
eil101.tsp.5.75.17.A.BDS	100	75	—	—	7200.0	7200.0	0	0	0
eil101.tsp.3.100.38.A.BDS	100	100	—	—	7200.0	7200.0	0	0	0
eil101.tsp.4.100.28.A.BDS	100	100	—	—	7200.0	7200.0	0	0	0
eil101.tsp.5.100.23.A.BDS	100	100	—	—	7200.0	7200.0	0	0	0

Table 13: Detailed Results for the instance class A of Baldacci *et al.* (2007)

Instance	$\|M\|$	$\|K\|$	UB	LB	Time LP	Time IP	#B&B	#ExtNG	#Cuts
eil26.tsp.3.12.5.B.BDS	25	12	1684	1684	0.1	0.1	1	3	8
eil26.tsp.4.12.4.B.BDS	25	12	1827	1827	0.1	0.1	1	1	9
eil26.tsp.5.12.3.B.BDS	25	12	2041	2041	0.1	0.0	1	1	12
eil26.tsp.3.18.7.B.BDS	25	18	2104	2104	0.1	1.0	1	11	74
eil26.tsp.4.18.5.B.BDS	25	18	2370	2370	0.1	0.1	1	2	14
eil26.tsp.5.18.4.B.BDS	25	18	2615	2615	0.1	0.1	1	1	16
eil26.tsp.3.25.10.B.BDS	25	25	2251	2251	0.1	0.5	1	3	30
eil26.tsp.4.25.7.B.BDS	25	25	2510	2510	0.1	0.4	1	4	42
eil26.tsp.5.25.6.B.BDS	25	25	2674	2674	0.1	0.2	1	2	23
eil51.tsp.3.12.5.B.BDS	50	12	1681	1681	0.1	0.4	3	4	29
eil51.tsp.4.12.4.B.BDS	50	12	1821	1821	0.1	0.2	2	4	18
eil51.tsp.5.12.3.B.BDS	50	12	1972	1972	0.1	0.1	1	2	8
eil51.tsp.3.25.10.B.BDS	50	25	2176	2176	0.6	0.9	1	1	21
eil51.tsp.4.25.7.B.BDS	50	25	2470	2470	0.2	5.6	3	13	112
eil51.tsp.5.25.6.B.BDS	50	25	2579	2579	0.2	0.4	1	1	25
eil51.tsp.3.37.14.B.BDS	50	37	2490	2490	6.3	117.8	5	7	85
eil51.tsp.4.37.11.B.BDS	50	37	2721	2721	1.2	978.5	192	97	103
eil51.tsp.5.37.9.B.BDS	50	37	2908	2908	0.6	41.6	20	23	86
eil51.tsp.3.50.19.B.BDS	50	50	3015	3015	104.3	2000.2	13	13	84
eil51.tsp.4.50.14.B.BDS	50	50	3260	3260	18.0	4709.0	30	28	94
eil51.tsp.5.50.12.B.BDS	50	50	3401	3401	6.7	14.7	1	2	17
eil76.tsp.3.18.7.B.BDS	75	18	2259	2248	0.6	7200.0	7677	685	156
eil76.tsp.4.18.5.B.BDS	75	18	2620	2620	0.3	2.8	5	5	50
eil76.tsp.5.18.4.B.BDS	75	18	3059	3059	0.2	11.5	47	12	115
eil76.tsp.3.37.14.B.BDS	75	37	—	2658	4803.2	7200.0	0	1	18
eil76.tsp.4.37.11.B.BDS	75	37	—	2999	335.1	7200.0	1	7	113
eil76.tsp.5.37.9.B.BDS	75	37	3284	3284	38.1	210.7	1	3	28
eil76.tsp.3.56.21.B.BDS	75	56	—	—	7200.0	7200.0	0	0	0
eil76.tsp.4.56.16.B.BDS	75	56	—	—	7200.0	7200.0	0	0	0
eil76.tsp.5.56.13.B.BDS	75	56	—	3580	7034.8	7200.0	0	1	9
eil76.tsp.3.75.28.B.BDS	75	75	—	—	7200.0	7200.0	0	0	0
eil76.tsp.4.75.21.B.BDS	75	75	—	—	7200.0	7200.0	0	0	0
eil76.tsp.5.75.17.B.BDS	75	75	—	—	7200.0	7200.0	0	0	0
eil101.tsp.3.25.10.B.BDS	100	25	2434	2434	8.5	329.1	15	12	88
eil101.tsp.4.25.7.B.BDS	100	25	2782	2782	1.8	345.7	195	67	141
eil101.tsp.5.25.6.B.BDS	100	25	3009	3009	1.2	30.7	2	11	121
eil101.tsp.3.50.19.B.BDS	100	50	—	—	7200.0	7200.0	0	0	0
eil101.tsp.4.50.14.B.BDS	100	50	—	—	7200.0	7200.0	0	0	0
eil101.tsp.5.50.12.B.BDS	100	50	—	3731	1017.3	7200.0	0	5	59
eil101.tsp.3.75.28.B.BDS	100	75	—	—	7200.0	7200.0	0	0	0
eil101.tsp.4.75.21.B.BDS	100	75	—	—	7200.0	7200.0	0	0	0
eil101.tsp.5.75.17.B.BDS	100	75	—	—	7200.0	7200.0	0	0	0
eil101.tsp.3.100.38.B.BDS	100	100	—	—	7200.0	7200.0	0	0	0
eil101.tsp.4.100.28.B.BDS	100	100	—	—	7200.0	7200.0	0	0	0
eil101.tsp.5.100.23.B.BDS	100	100	—	—	7200.0	7200.0	0	0	0

Table 14: Detailed Results for the instance class B of Baldacci *et al.* (2007)

Instance	$\|M\|$	$\|K\|$	UB	LB	Time LP	Time IP	#B&B	#ExtNG	#Cuts
eil26.tsp.3.6.3.A.CEIL_2D	25	6	178	178	0.1	0.1	1	0	0
eil26.tsp.3.12.5.A.CEIL_2D	25	12	254	254	0.1	0.1	1	1	6
eil26.tsp.4.12.4.A.CEIL_2D	25	12	271	271	0.1	0.1	1	0	0
eil26.tsp.5.12.3.A.CEIL_2D	25	12	304	304	0.1	0.1	1	0	0
eil26.tsp.3.18.7.A.CEIL_2D	25	18	312	312	0.1	0.4	1	7	48
eil26.tsp.4.18.5.A.CEIL_2D	25	18	349	349	0.1	0.2	1	2	24
eil26.tsp.5.18.4.A.CEIL_2D	25	18	387	387	0.1	0.1	1	2	40
eil26.tsp.7.18.3.A.CEIL_2D	25	18	462	462	0.1	0.1	1	1	20
eil26.tsp.3.25.10.A.CEIL_2D	25	25	346	346	0.1	0.8	1	8	56
eil26.tsp.4.25.7.A.CEIL_2D	25	25	379	379	0.1	0.3	1	4	39
eil26.tsp.5.25.6.A.CEIL_2D	25	25	398	398	0.1	0.1	1	0	0
eil26.tsp.7.25.4.A.CEIL_2D	25	25	490	490	0.1	0.1	1	1	10
eil26.tsp.10.25.3.A.CEIL_2D	25	25	601	601	0.1	0.1	1	3	37
eil51.tsp.3.12.5.A.CEIL_2D	50	12	254	254	0.1	0.2	1	2	30
eil51.tsp.4.12.4.A.CEIL_2D	50	12	271	271	0.1	0.1	1	1	3
eil51.tsp.5.12.3.A.CEIL_2D	50	12	303	303	0.1	0.1	1	1	4
eil51.tsp.3.25.10.A.CEIL_2D	50	25	343	343	0.4	2.0	1	6	65
eil51.tsp.4.25.7.A.CEIL_2D	50	25	378	378	0.2	1.9	1	10	82
eil51.tsp.5.25.6.A.CEIL_2D	50	25	395	395	0.1	0.2	1	0	0
eil51.tsp.7.25.4.A.CEIL_2D	50	25	489	489	0.1	0.2	1	1	26
eil51.tsp.10.25.3.A.CEIL_2D	50	25	595	595	0.1	0.2	1	1	49
eil51.tsp.3.37.14.A.CEIL_2D	50	37	406	406	3.4	37.3	6	16	108
eil51.tsp.4.37.11.A.CEIL_2D	50	37	437	437	0.9	28.7	5	20	115
eil51.tsp.5.37.9.A.CEIL_2D	50	37	467	467	0.4	12.2	9	17	99
eil51.tsp.7.37.6.A.CEIL_2D	50	37	547	547	0.3	2.4	1	11	97
eil51.tsp.10.37.5.A.CEIL_2D	50	37	626	626	0.1	3.7	4	15	156
eil51.tsp.14.37.3.A.CEIL_2D	50	37	829	829	0.1	1.2	2	7	122
eil51.tsp.3.50.19.A.CEIL_2D	50	50	—	495	27.2	1800.0	29	40	110
eil51.tsp.4.50.14.A.CEIL_2D	50	50	530	530	7.8	1075.4	35	61	116
eil51.tsp.5.50.12.A.CEIL_2D	50	50	560	560	3.0	35.7	2	11	108
eil51.tsp.7.50.8.A.CEIL_2D	50	50	648	648	1.2	67.6	26	34	162
eil51.tsp.10.50.6.A.CEIL_2D	50	50	754	754	0.5	13.2	10	16	125
eil51.tsp.14.50.4.A.CEIL_2D	50	50	954	954	0.2	5.9	7	10	234

Table 15: Detailed Results for instance class small A of Hoshino and De Souza (2012)

Instance	$\|M\|$	$\|K\|$	UB	LB	Time LP	Time IP	#B&B	#ExtNG	#Cuts
eil26.tsp.3.6.3.B.CEIL_2D	25	6	1246	1246	0.1	0.1	1	0	0
eil26.tsp.3.12.5.B.CEIL_2D	25	12	1778	1778	0.1	0.1	1	1	6
eil26.tsp.4.12.4.B.CEIL_2D	25	12	1897	1897	0.1	0.1	1	0	0
eil26.tsp.5.12.3.B.CEIL_2D	25	12	2128	2128	0.1	0.1	1	0	0
eil26.tsp.3.18.7.B.CEIL_2D	25	18	2184	2184	0.1	0.3	1	5	50
eil26.tsp.4.18.5.B.CEIL_2D	25	18	2443	2443	0.1	0.1	1	2	24
eil26.tsp.5.18.4.B.CEIL_2D	25	18	2709	2709	0.1	0.1	1	2	40
eil26.tsp.7.18.3.B.CEIL_2D	25	18	3234	3234	0.1	0.1	1	1	20
eil26.tsp.3.25.10.B.CEIL_2D	25	25	2422	2422	0.1	0.7	1	8	58
eil26.tsp.4.25.7.B.CEIL_2D	25	25	2653	2653	0.1	0.3	1	4	39
eil26.tsp.5.25.6.B.CEIL_2D	25	25	2786	2786	0.1	0.1	1	0	0
eil26.tsp.7.25.4.B.CEIL_2D	25	25	3430	3430	0.1	0.1	1	1	10
eil26.tsp.10.25.3.B.CEIL_2D	25	25	4207	4207	0.1	0.1	1	3	37
eil51.tsp.3.12.5.B.CEIL_2D	50	12	1778	1778	0.1	0.2	1	3	34
eil51.tsp.4.12.4.B.CEIL_2D	50	12	1897	1897	0.1	0.1	1	1	3
eil51.tsp.5.12.3.B.CEIL_2D	50	12	2121	2121	0.1	0.1	1	1	4
eil51.tsp.3.25.10.B.CEIL_2D	50	25	2354	2354	0.4	1.4	1	4	56
eil51.tsp.4.25.7.B.CEIL_2D	50	25	2606	2606	0.2	1.2	1	7	65
eil51.tsp.5.25.6.B.CEIL_2D	50	25	2718	2718	0.1	0.1	1	0	0
eil51.tsp.7.25.4.B.CEIL_2D	50	25	3400	3400	0.1	0.2	1	2	32
eil51.tsp.10.25.3.B.CEIL_2D	50	25	4111	4111	0.1	0.1	1	1	26
eil51.tsp.3.37.14.B.CEIL_2D	50	37	2795	2795	2.3	20.4	2	19	96
eil51.tsp.4.37.11.B.CEIL_2D	50	37	3022	3022	0.6	45.7	13	35	116
eil51.tsp.5.37.9.B.CEIL_2D	50	37	3236	3236	0.3	11.9	5	17	109
eil51.tsp.7.37.6.B.CEIL_2D	50	37	3775	3775	0.3	0.7	1	3	61
eil51.tsp.10.37.5.B.CEIL_2D	50	37	4335	4335	0.2	2.9	2	13	128
eil51.tsp.14.37.3.B.CEIL_2D	50	37	5759	5759	0.1	1.0	2	7	137
eil51.tsp.3.50.19.B.CEIL_2D	50	50	—	3385	34.4	1800.0	57	77	101
eil51.tsp.4.50.14.B.CEIL_2D	50	50	3624	3624	6.4	617.2	46	40	88
eil51.tsp.5.50.12.B.CEIL_2D	50	50	3853	3853	2.4	1770.0	274	146	127
eil51.tsp.7.50.8.B.CEIL_2D	50	50	4474	4474	0.8	719.6	286	90	162
eil51.tsp.10.50.6.B.CEIL_2D	50	50	5216	5216	0.5	4.8	3	11	83
eil51.tsp.14.50.4.B.CEIL_2D	50	50	6612	6612	0.2	32.2	18	19	234

Table 16: Detailed Results for instance class small B of Hoshino and De Souza (2012)

Instance	$\|M\|$	$\|K\|$	UB	LB	Time LP	Time IP	#B&B	#ExtNG	#Cuts
eil26.tsp.3.6.3.C.CEIL_2D	25	6	159	159	0.1	0.1	1	0	0
eil26.tsp.3.12.5.C.CEIL_2D	25	12	226	226	0.1	0.1	1	0	0
eil26.tsp.4.12.4.C.CEIL_2D	25	12	243	243	0.1	0.1	1	0	0
eil26.tsp.5.12.3.C.CEIL_2D	25	12	270	270	0.1	0.1	2	0	1
eil26.tsp.3.18.7.C.CEIL_2D	25	18	289	289	0.2	6.6	7	0	56
eil26.tsp.4.18.5.C.CEIL_2D	25	18	324	324	0.1	0.2	3	0	0
eil26.tsp.5.18.4.C.CEIL_2D	25	18	353	353	0.1	0.1	1	0	0
eil26.tsp.7.18.3.C.CEIL_2D	25	18	410	410	0.1	0.1	1	0	0
eil26.tsp.3.25.10.C.CEIL_2D	25	25	327	327	1.2	8.4	1	0	22
eil26.tsp.4.25.7.C.CEIL_2D	25	25	362	362	0.2	1.7	6	0	28
eil26.tsp.5.25.6.C.CEIL_2D	25	25	385	385	0.1	0.1	1	0	0
eil26.tsp.7.25.4.C.CEIL_2D	25	25	460	460	0.1	0.1	1	0	0
eil26.tsp.10.25.3.C.CEIL_2D	25	25	545	545	0.1	0.1	1	0	1
eil51.tsp.3.12.5.C.CEIL_2D	50	12	226	226	0.1	0.1	1	0	0
eil51.tsp.4.12.4.C.CEIL_2D	50	12	241	241	0.1	0.1	1	0	0
eil51.tsp.5.12.3.C.CEIL_2D	50	12	270	270	0.1	0.2	5	0	0
eil51.tsp.3.25.10.C.CEIL_2D	50	25	325	325	8.6	50.3	1	0	16
eil51.tsp.4.25.7.C.CEIL_2D	50	25	359	359	0.7	14.4	9	0	73
eil51.tsp.5.25.6.C.CEIL_2D	50	25	383	383	0.1	0.3	1	0	0
eil51.tsp.7.25.4.C.CEIL_2D	50	25	457	457	0.1	0.1	1	0	0
eil51.tsp.10.25.3.C.CEIL_2D	50	25	539	539	0.2	0.5	3	0	4
eil51.tsp.3.37.14.C.CEIL_2D	50	37	—	394	1800.0	1800.0	0	0	21
eil51.tsp.4.37.11.C.CEIL_2D	50	37	—	423	45.1	1800.0	7	0	72
eil51.tsp.5.37.9.C.CEIL_2D	50	37	446	446	7.2	18.0	1	0	12
eil51.tsp.7.37.6.C.CEIL_2D	50	37	530	530	0.7	10.7	10	0	45
eil51.tsp.10.37.5.C.CEIL_2D	50	37	598	598	0.1	0.2	1	0	0
eil51.tsp.14.37.3.C.CEIL_2D	50	37	765	765	0.2	1.7	14	0	7
eil51.tsp.3.50.19.C.CEIL_2D	50	50	—	—	1800.0	1800.0	0	0	0
eil51.tsp.4.50.14.C.CEIL_2D	50	50	—	—	1800.0	1800.0	0	0	0
eil51.tsp.5.50.12.C.CEIL_2D	50	50	—	530	1800.0	1800.0	0	0	4
eil51.tsp.7.50.8.C.CEIL_2D	50	50	—	620	5.0	1800.0	185	0	81
eil51.tsp.10.50.6.C.CEIL_2D	50	50	721	721	0.9	8.7	6	0	42
eil51.tsp.14.50.4.C.CEIL_2D	50	50	905	905	0.5	6.1	15	0	15

Table 17: Detailed Results for instance class small C of Hoshino and De Souza (2012)

Instance	$\|M\|$	$\|K\|$	UB	LB	Time LP	Time IP	#B&B	#ExtNG	#Cuts
eil76.tsp.3.18.7.A.CEIL_2D	75	18	338	338	0.4	367.9	341	109	253
eil76.tsp.4.18.5.A.CEIL_2D	75	18	399	399	0.2	17.5	39	15	184
eil76.tsp.5.18.4.A.CEIL_2D	75	18	460	460	0.2	6.4	3	10	165
eil76.tsp.7.18.3.A.CEIL_2D	75	18	558	558	0.1	0.3	1	3	12
eil76.tsp.3.37.14.A.CEIL_2D	75	37	—	422	1800.0	1800.0	0	4	40
eil76.tsp.4.37.11.A.CEIL_2D	75	37	—	463	1800.0	1800.0	0	13	149
eil76.tsp.5.37.9.A.CEIL_2D	75	37	501	501	0.1	16.3	1	0	0
eil76.tsp.7.37.6.A.CEIL_2D	75	37	641	641	2.1	3.2	1	1	1
eil76.tsp.10.37.5.A.CEIL_2D	75	37	748	748	0.6	10.8	2	13	105
eil76.tsp.14.37.3.A.CEIL_2D	75	37	1045	1045	0.3	1.6	1	3	39
eil76.tsp.3.56.21.A.CEIL_2D	75	56	—	—	1800.0	1800.0	0	0	0
eil76.tsp.4.56.16.A.CEIL_2D	75	56	—	—	1800.0	1800.0	0	0	0
eil76.tsp.5.56.13.A.CEIL_2D	75	56	—	—	1800.0	1800.0	0	0	0
eil76.tsp.7.56.9.A.CEIL_2D	75	56	—	703	40.9	1800.0	3	23	79
eil76.tsp.10.56.7.A.CEIL_2D	75	56	824	824	3.6	67.0	2	12	91
eil76.tsp.14.56.5.A.CEIL_2D	75	56	1030	1030	0.7	20.9	5	18	122
eil76.tsp.3.75.28.A.CEIL_2D	75	75	—	—	1800.0	1800.0	0	0	0
eil76.tsp.4.75.21.A.CEIL_2D	75	75	—	—	1800.0	1800.0	0	0	0
eil76.tsp.5.75.17.A.CEIL_2D	75	75	—	—	1800.0	1800.0	0	0	0
eil76.tsp.7.75.12.A.CEIL_2D	75	75	—	—	1800.0	1800.0	0	0	0
eil76.tsp.10.75.9.A.CEIL_2D	75	75	—	931	1800.0	1800.0	0	9	71
eil76.tsp.14.75.6.A.CEIL_2D	75	75	1180	1180	4.1	71.0	9	15	115
eil101.tsp.3.25.10.A.CEIL_2D	100	25	381	381	1.6	138.7	1	22	371
eil101.tsp.4.25.7.A.CEIL_2D	100	25	433	433	0.9	50.9	1	22	356
eil101.tsp.5.25.6.A.CEIL_2D	100	25	469	469	0.8	45.6	5	21	388
eil101.tsp.7.25.4.A.CEIL_2D	100	25	576	576	0.6	124.7	128	38	383
eil101.tsp.10.25.3.A.CEIL_2D	100	25	693	693	0.6	12.2	3	8	263
eil101.tsp.3.50.19.A.CEIL_2D	100	50	—	—	1800.0	1800.0	0	0	0
eil101.tsp.4.50.14.A.CEIL_2D	100	50	—	—	1800.0	1800.0	0	0	0
eil101.tsp.5.50.12.A.CEIL_2D	100	50	—	—	1800.0	1800.0	0	0	0
eil101.tsp.7.50.8.A.CEIL_2D	100	50	—	692	1800.0	1800.0	0	9	136
eil101.tsp.10.50.6.A.CEIL_2D	100	50	819	819	6.0	273.8	9	19	193
eil101.tsp.14.50.4.A.CEIL_2D	100	50	1042	1042	1.7	517.1	100	39	392
eil101.tsp.3.75.28.A.CEIL_2D	100	75	—	—	1800.0	1800.0	0	0	0
eil101.tsp.4.75.21.A.CEIL_2D	100	75	—	—	1800.0	1800.0	0	0	0
eil101.tsp.5.75.17.A.CEIL_2D	100	75	—	—	1800.0	1800.0	0	0	0
eil101.tsp.7.75.12.A.CEIL_2D	100	75	—	—	1800.0	1800.0	0	0	0
eil101.tsp.10.75.9.A.CEIL_2D	100	75	—	897	1800.0	1800.0	0	4	51
eil101.tsp.14.75.6.A.CEIL_2D	100	75	—	1126	10.1	1800.0	57	35	338
eil101.tsp.3.100.38.A.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.4.100.28.A.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.5.100.23.A.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.7.100.16.A.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.10.100.12.A.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.14.100.8.A.CEIL_2D	100	100	—	1176	96.0	1800.0	6	15	130

Table 18: Detailed Results for instance class large A of Hoshino and De Souza (2012)

Instance	$\|M\|$	$\|K\|$	UB	LB	Time LP	Time IP	#B&B	#ExtNG	#Cuts
eil76.tsp.3.18.7.B.CEIL_2D	75	18	2319	2317	0.4	1800.0	2061	320	253
eil76.tsp.4.18.5.B.CEIL_2D	75	18	2729	2729	0.2	10.7	10	9	177
eil76.tsp.5.18.4.B.CEIL_2D	75	18	3172	3172	0.3	12.6	17	9	172
eil76.tsp.7.18.3.B.CEIL_2D	75	18	3824	3824	0.1	0.2	1	2	31
eil76.tsp.3.37.14.B.CEIL_2D	75	37	—	2906	1800.0	1800.0	0	5	43
eil76.tsp.4.37.11.B.CEIL_2D	75	37	—	3214	81.8	1800.0	1	13	145
eil76.tsp.5.37.9.B.CEIL_2D	75	37	3495	3495	0.1	27.0	1	0	0
eil76.tsp.7.37.6.B.CEIL_2D	75	37	4451	4451	2.0	4.5	1	2	12
eil76.tsp.10.37.5.B.CEIL_2D	75	37	5177	5177	0.7	19.5	7	16	124
eil76.tsp.14.37.3.B.CEIL_2D	75	37	7235	7235	0.3	7.2	37	16	95
eil76.tsp.3.56.21.B.CEIL_2D	75	56	—	—	1800.0	1800.0	0	0	0
eil76.tsp.4.56.16.B.CEIL_2D	75	56	—	—	1800.0	1800.0	0	0	0
eil76.tsp.5.56.13.B.CEIL_2D	75	56	—	—	1800.0	1800.0	0	0	0
eil76.tsp.7.56.9.B.CEIL_2D	75	56	—	4741	26.8	1800.0	11	29	112
eil76.tsp.10.56.7.B.CEIL_2D	75	56	5541	5541	2.4	13.0	1	4	47
eil76.tsp.14.56.5.B.CEIL_2D	75	56	7010	7010	0.7	800.2	536	144	178
eil76.tsp.3.75.28.B.CEIL_2D	75	75	—	—	1800.0	1800.0	0	0	0
eil76.tsp.4.75.21.B.CEIL_2D	75	75	—	—	1800.0	1800.0	0	0	0
eil76.tsp.5.75.17.B.CEIL_2D	75	75	—	—	1800.0	1800.0	0	0	0
eil76.tsp.7.75.12.B.CEIL_2D	75	75	—	5317	1800.0	1800.0	0	2	12
eil76.tsp.10.75.9.B.CEIL_2D	75	75	—	6241	20.0	1800.0	21	36	104
eil76.tsp.14.75.6.B.CEIL_2D	75	75	—	8000	2.2	1800.0	268	217	177
eil101.tsp.3.25.10.B.CEIL_2D	100	25	2629	2629	1.8	192.5	21	25	231
eil101.tsp.4.25.7.B.CEIL_2D	100	25	2972	2972	0.9	20.8	1	15	157
eil101.tsp.5.25.6.B.CEIL_2D	100	25	3237	3237	0.7	39.0	2	19	228
eil101.tsp.7.25.4.B.CEIL_2D	100	25	3986	3986	0.5	194.4	437	56	291
eil101.tsp.10.25.3.B.CEIL_2D	100	25	4803	4803	0.5	31.2	47	27	325
eil101.tsp.3.50.19.B.CEIL_2D	100	50	—	—	1800.0	1800.0	0	0	0
eil101.tsp.4.50.14.B.CEIL_2D	100	50	—	—	1800.0	1800.0	0	0	0
eil101.tsp.5.50.12.B.CEIL_2D	100	50	—	—	1800.0	1800.0	0	0	0
eil101.tsp.7.50.8.B.CEIL_2D	100	50	—	4729	1800.0	1800.0	0	10	360
eil101.tsp.10.50.6.B.CEIL_2D	100	50	5596	5596	5.1	771.1	137	34	180
eil101.tsp.14.50.4.B.CEIL_2D	100	50	7130	7127	1.3	1800.0	456	81	422
eil101.tsp.3.75.28.B.CEIL_2D	100	75	—	—	1800.0	1800.0	0	0	0
eil101.tsp.4.75.21.B.CEIL_2D	100	75	—	—	1800.0	1800.0	0	0	0
eil101.tsp.5.75.17.B.CEIL_2D	100	75	—	—	1800.0	1800.0	0	0	0
eil101.tsp.7.75.12.B.CEIL_2D	100	75	—	—	1800.0	1800.0	0	0	0
eil101.tsp.10.75.9.B.CEIL_2D	100	75	6012	6012	67.0	1192.9	6	6	144
eil101.tsp.14.75.6.B.CEIL_2D	100	75	7530	7518	5.6	1800.0	196	126	323
eil101.tsp.3.100.38.B.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.4.100.28.B.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.5.100.23.B.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.7.100.16.B.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.10.100.12.B.CEIL_2D	100	100	—	6404	1800.0	1800.0	0	3	34
eil101.tsp.14.100.8.B.CEIL_2D	100	100	—	7914	12.5	1800.0	95	57	182

Table 19: Detailed Results for instance class large B of Hoshino and De Souza (2012)

Instance	$\|M\|$	$\|K\|$	UB	LB	Time LP	Time IP	#B&B	#ExtNG	#Cuts
eil76.tsp.3.18.7.C.CEIL_2D	75	18	—	324	5.1	1800.0	71	0	404
eil76.tsp.4.18.5.C.CEIL_2D	75	18	385	385	0.5	22.2	5	2	367
eil76.tsp.5.18.4.C.CEIL_2D	75	18	439	439	0.4	30.1	17	5	232
eil76.tsp.7.18.3.C.CEIL_2D	75	18	527	527	0.2	1.6	12	0	12
eil76.tsp.3.37.14.C.CEIL_2D	75	37	—	—	1800.0	1800.0	0	0	0
eil76.tsp.4.37.11.C.CEIL_2D	75	37	—	—	1800.0	1800.0	0	0	0
eil76.tsp.5.37.9.C.CEIL_2D	75	37	491	491	46.1	86.0	1	0	2
eil76.tsp.7.37.6.C.CEIL_2D	75	37	626	626	2.1	145.9	64	0	87
eil76.tsp.10.37.5.C.CEIL_2D	75	37	723	723	1.2	20.8	13	3	45
eil76.tsp.14.37.3.C.CEIL_2D	75	37	969	969	0.7	2.1	2	0	19
eil76.tsp.3.56.21.C.CEIL_2D	75	56	—	—	1800.0	1800.0	0	0	0
eil76.tsp.4.56.16.C.CEIL_2D	75	56	—	—	1800.0	1800.0	0	0	0
eil76.tsp.5.56.13.C.CEIL_2D	75	56	—	—	1800.0	1800.0	0	0	0
eil76.tsp.7.56.9.C.CEIL_2D	75	56	—	683	60.9	1800.0	11	1	70
eil76.tsp.10.56.7.C.CEIL_2D	75	56	811	805	4.3	1800.0	300	0	95
eil76.tsp.14.56.5.C.CEIL_2D	75	56	1000	1000	2.0	105.2	304	1	79
eil76.tsp.3.75.28.C.CEIL_2D	75	75	—	—	1800.0	1800.0	0	0	0
eil76.tsp.4.75.21.C.CEIL_2D	75	75	—	—	1800.0	1800.0	0	0	0
eil76.tsp.5.75.17.C.CEIL_2D	75	75	—	—	1800.0	1800.0	0	0	0
eil76.tsp.7.75.12.C.CEIL_2D	75	75	—	—	1800.0	1800.0	0	0	0
eil76.tsp.10.75.9.C.CEIL_2D	75	75	—	903	132.0	1800.0	4	0	77
eil76.tsp.14.75.6.C.CEIL_2D	75	75	1145	1145	3.6	58.2	12	1	79
eil101.tsp.3.25.10.C.CEIL_2D	100	25	—	367	1800.0	1800.0	0	0	35
eil101.tsp.4.25.7.C.CEIL_2D	100	25	416	416	4.3	448.0	9	0	374
eil101.tsp.5.25.6.C.CEIL_2D	100	25	440	440	2.8	8.1	2	0	10
eil101.tsp.7.25.4.C.CEIL_2D	100	25	542	542	1.2	419.9	54	3	354
eil101.tsp.10.25.3.C.CEIL_2D	100	25	631	631	1.0	3.9	3	1	5
eil101.tsp.3.50.19.C.CEIL_2D	100	50	—	—	1800.0	1800.0	0	0	0
eil101.tsp.4.50.14.C.CEIL_2D	100	50	—	—	1800.0	1800.0	0	0	0
eil101.tsp.5.50.12.C.CEIL_2D	100	50	—	—	1800.0	1800.0	0	0	0
eil101.tsp.7.50.8.C.CEIL_2D	100	50	—	671	29.4	1800.0	18	1	121
eil101.tsp.10.50.6.C.CEIL_2D	100	50	793	793	5.9	179.7	31	0	97
eil101.tsp.14.50.4.C.CEIL_2D	100	50	993	993	3.8	7.2	1	0	11
eil101.tsp.3.75.28.C.CEIL_2D	100	75	—	—	1800.0	1800.0	0	0	0
eil101.tsp.4.75.21.C.CEIL_2D	100	75	—	—	1800.0	1800.0	0	0	0
eil101.tsp.5.75.17.C.CEIL_2D	100	75	—	—	1800.0	1800.0	0	0	0
eil101.tsp.7.75.12.C.CEIL_2D	100	75	—	—	1800.0	1800.0	0	0	0
eil101.tsp.10.75.9.C.CEIL_2D	100	75	—	876	327.7	1800.0	3	1	73
eil101.tsp.14.75.6.C.CEIL_2D	100	75	—	1096	9.5	1800.0	722	1	122
eil101.tsp.3.100.38.C.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.4.100.28.C.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.5.100.23.C.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.7.100.16.C.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.10.100.12.C.CEIL_2D	100	100	—	—	1800.0	1800.0	0	0	0
eil101.tsp.14.100.8.C.CEIL_2D	100	100	—	1147	42.0	1800.0	58	0	86

Table 20: Detailed Results for instance class large C of Hoshino and De Souza (2012)

Instance	$\|M\|+1$	$\|K\|$	fr	$\#Sub$	UB	LB	Time LP	Time IP	#B&B	#ExtNG	#Cuts
CapEuclideo.20.10.30.5	20	10	30	4	3286	3286	0.0	0.0	1	0	0
CapEuclideo.20.10.30.2	20	10	30	4	4267	4267	0.0	0.1	9	0	0
CapEuclideo.20.10.55.5	20	10	55	4	2784	2784	0.0	0.0	1	0	0
CapEuclideo.20.10.55.2	20	10	55	4	4036	4036	0.0	0.0	1	0	0
CapEuclideo.20.10.80.5	20	10	80	2	2422	2422	0.0	0.0	1	0	0
CapEuclideo.20.10.80.2	20	10	80	2	3928	3928	0.0	0.1	19	0	0
CapEuclideo.20.30.30.15	20	30	30	35	8419	8419	1.5	1.5	1	0	0
CapEuclideo.20.30.30.5	20	30	30	35	8419	8419	0.5	0.5	1	0	0
CapEuclideo.20.30.55.15	20	30	55	16	6359	6359	1.1	1.1	1	0	0
CapEuclideo.20.30.55.3	20	30	55	16	7899	7899	0.1	1.4	31	0	0
CapEuclideo.20.30.80.15	20	30	80	5	4655	4655	2.3	4.9	3	0	0
CapEuclideo.20.30.80.2	20	30	80	5	10521	10521	0.0	0.5	41	0	0
CapEuclideo.20.50.30.25	20	50	30	102	11264	11264	11.2	126.2	40	3	0
CapEuclideo.20.50.30.9	20	50	30	102	11264	11264	9.5	106.9	43	2	0
CapEuclideo.20.50.55.17	20	50	55	35	9392	9392	10.8	15.5	3	0	0
CapEuclideo.20.50.55.4	20	50	55	35	10523	10523	1.2	60.7	143	1	0
CapEuclideo.20.50.80.25	20	50	80	11	6181	6181	16.4	16.4	1	0	0
CapEuclideo.20.50.80.4	20	50	80	11	9613	9613	0.3	18.9	157	5	0
CapEuclideo.20.70.30.18	20	70	30	286	19460	19460	75.5	75.5	1	0	0
CapEuclideo.20.70.30.6	20	70	30	286	19536	19536	25.4	25.4	1	0	0
CapEuclideo.20.70.55.35	20	70	55	87	14133	14133	139.9	144.0	2	0	0
CapEuclideo.20.70.55.7	20	70	55	87	14209	14209	14.2	72.7	15	0	0
CapEuclideo.20.70.80.35	20	70	80	12	9480	9480	53.4	53.4	1	0	0
CapEuclideo.20.70.80.5	20	70	80	12	13086	13085	0.7	21.1	65	2	0
CapEuclideo.35.10.30.5	35	10	30	5	1351	1351	0.1	0.1	1	0	0
CapEuclideo.35.10.30.2	35	10	30	5	1596	1596	0.0	0.1	3	0	0
CapEuclideo.35.10.55.4	35	10	55	4	1342	1342	0.1	0.1	1	0	0
CapEuclideo.35.10.55.2	35	10	55	4	1538	1538	0.0	0.1	3	0	0
CapEuclideo.35.10.80.5	35	10	80	2	1187	1187	0.1	0.1	1	0	0
CapEuclideo.35.10.80.2	35	10	80	2	1538	1538	0.0	0.1	7	0	0
CapEuclideo.35.30.30.10	35	30	30	25	5236	5236	1.0	1.0	1	0	0
CapEuclideo.35.30.30.4	35	30	30	25	5245	5245	0.5	1.1	3	0	0
CapEuclideo.35.30.55.15	35	30	55	15	4097	4097	4.7	4.7	1	0	0
CapEuclideo.35.30.55.3	35	30	55	15	4703	4703	0.1	0.5	7	0	0
CapEuclideo.35.30.80.8	35	30	80	5	3501	3501	1.3	10.8	10	2	0
CapEuclideo.35.30.80.2	35	30	80	5	5713	5713	0.1	4.5	135	0	0
CapEuclideo.35.50.30.17	35	50	30	62	12907	12907	25.7	25.7	1	0	0
CapEuclideo.35.50.30.6	35	50	30	62	12907	12907	9.1	9.1	1	0	0
CapEuclideo.35.50.55.13	35	50	55	45	9324	9324	75.2	75.2	1	0	0
CapEuclideo.35.50.55.3	35	50	55	45	11160	11160	0.8	7.2	27	0	0
CapEuclideo.35.50.80.25	35	50	80	11	5640	5640	1508.3	1508.3	1	0	0
CapEuclideo.35.50.80.4	35	50	80	11	8723	8723	0.8	4.4	23	0	0
CapEuclideo.35.70.30.18	35	70	30	147	12133	12133	47.2	47.2	1	0	0
CapEuclideo.35.70.30.6	35	70	30	147	12308	12308	21.8	59.5	9	0	0
CapEuclideo.35.70.55.24	35	70	55	81	9521	9521	99.0	99.0	1	0	0
CapEuclideo.35.70.55.5	35	70	55	81	10195	10195	10.0	62.6	13	0	0
CapEuclideo.35.70.80.35	35	70	80	15	6911	6911	152.0	152.0	1	0	0
CapEuclideo.35.70.80.5	35	70	80	15	8944	8944	2.0	158.5	230	11	0
CapEuclideo.50.10.30.4	50	10	30	6	2869	2869	0.1	0.1	1	0	0
CapEuclideo.50.10.30.2	50	10	30	6	3045	3045	0.0	0.0	1	0	0
CapEuclideo.50.10.55.4	50	10	55	4	2371	2371	0.1	0.1	1	0	0
CapEuclideo.50.10.55.2	50	10	55	4	2803	2803	0.1	0.1	1	0	0
CapEuclideo.50.10.80.5	50	10	80	2	1779	1779	0.2	0.2	1	0	0
CapEuclideo.50.10.80.2	50	10	80	2	2803	2803	0.0	0.0	1	0	0
CapEuclideo.50.30.30.8	50	30	30	33	5708	5708	10.5	10.5	1	0	0
CapEuclideo.50.30.30.3	50	30	30	33	5803	5803	0.5	0.5	1	0	0
CapEuclideo.50.30.55.15	50	30	55	15	5008	5008	32.8	32.8	1	0	0
CapEuclideo.50.30.55.3	50	30	55	15	5487	5487	0.3	0.8	5	0	0
CapEuclideo.50.30.80.15	50	30	80	5	3601	3601	235.3	235.3	1	0	0
CapEuclideo.50.30.80.2	50	30	80	5	6448	6448	0.1	0.3	5	0	0
CapEuclideo.50.50.30.25	50	50	30	138	7919	7919	52.7	52.7	1	0	0
CapEuclideo.50.50.30.9	50	50	30	138	7919	7919	44.1	44.1	1	0	0
CapEuclideo.50.50.55.25	50	50	55	37	6807	6807	66.5	66.5	1	0	0
CapEuclideo.50.50.55.5	50	50	55	37	7271	7271	4.9	101.5	47	0	0
CapEuclideo.50.50.80.17	50	50	80	9	5222	5222	28.1	798.2	15	0	0
CapEuclideo.50.50.80.3	50	50	80	9	8425	8425	0.3	17.6	135	0	0
CapEuclideo.50.70.30.35	50	70	30	237	10132	10132	283.7	2941.0	17	0	0
CapEuclideo.50.70.30.12	50	70	30	237	10140	10140	251.8	1904.9	15	0	0
CapEuclideo.50.70.55.24	50	70	55	68	9167	9167	451.6	615.3	2	0	0
CapEuclideo.50.70.55.5	50	70	55	68	9725	9725	13.5	37.1	7	0	0
CapEuclideo.50.70.80.35	50	70	80	17	—	—	7200.0	7200.0	0	0	0
CapEuclideo.50.70.80.5	50	70	80	17	8360	8360	3.3	7095.1	2687	511	0

Table 21: Detailed Results for the instances of Gendreau *et al.* (2016)