Speeding up Column-Generation Algorithms by Using Reduced Costs of Paths to Eliminate Arcs

Stefan Irnich [a,*]

[a] *Deutsche Post Endowed Chair of Optimization of Distribution Networks, RWTH Aachen University, Templergraben 64, D-52056 Aachen, Germany.*

Abstract

In many column-generation algorithms, the pricing problem consists of computing feasible s-t-paths in a network. Examples stem from routing and scheduling applications in transportation, as well as from production and telecommunication. The contribution of this paper is to show how reduced costs of paths can be used to remove some arcs from the underlying network, where shortest-path subproblems are iteratively solved. This can lead to a substantial speedup of the pricing process and the overall branch-and-price algorithm, for which the optimality of the solution is still guaranteed. Special attention is given to variants of shortest-path problems with resource constraints and path-structural constraints. A by-product of the analysis is several new insights into reduced costs and dual solutions of some column-generation reformulations which are valuable in the context of *robust* branch-and-price-and-cut algorithms. Computational results show the usefulness of the proposed methods.

Key words: column generation, shortest paths, variable fixing, variable elimination

1 Introduction

Extensive formulations with many variables are widely used in integer programming models for hard combinatorial optimization problems. The elimination of many variables from a formulation is attractive, since smaller problem instances can often be solved significantly faster. One technique for variable elimination/fixing is based on reduced cost arguments and works as follows: Given an integer linear program, if the reduced cost of a non-negative integer variable in the linear-programming relaxation exceeds the optimality gap, the variable is zero in every optimal integer solution. It can therefore be fixed to zero or, equivalently, eliminated from the problem formulation.

* Corresponding author.
Email address: sirnich@or.rwth-aachen.de (Stefan Irnich).

Column-generation models (see Lübbecke and Desrosiers, 2006; Desaulniers *et al.*, 2005) use extensive formulations, often with millions or even billions of variables. At first glance, however, column-generation formulations are *not* well-suited to applying variable fixing directly: The solution process starts with an already small subset of variables, the number of newly generated variables is small relative to the number of feasible variables, and newly generated variables have negative reduced costs, but only w.r.t. the current dual variables which are an approximation of the optimal dual variables. Nevertheless, variable elimination cannot only be applied to the column-generation master program, but also to the pricing subproblem. In this paper, we apply it to formulations where the subproblems constitute variants of shortest-path problems. The elimination of arcs from the underlying network can lead to a substantial speedup of the pricing subproblem and, therefore, of the overall branch-and-price algorithm.

The technique proposed here is widely applicable because, in the majority of column-generation algorithms presented in the literature thus far, pricing problems consist of computing feasible s-t-paths in networks. Additional constraints limit scarce resources along the path or imply constraints on the structure of paths (cf. Irnich and Desaulniers, 2005). Examples stem from routing and scheduling applications in transportation, as well as from production and telecommunication, e.g., vehicle routing (Desrochers *et al.*, 1992), vehicle scheduling (Ribeiro and Soumis, 1994), crew scheduling (Desaulniers *et al.*, 1997; Vance *et al.*, 1997; Desaulniers *et al.*, 1998a; Gamache and Soumis, 1998), and network design (Barnhart and Schneur, 1996; Barnhart *et al.*, 2000). A unifying formulation and column-generation solution approach for these and other applications has been presented by Desaulniers *et al.* (1998b).

The paper is structured as follows: Section 2 is intended to establish basic notation in order to describe column-generation algorithms with well-structured subproblems in general. Next, Section 3 contains the main theoretical results. Three different methods for the computation of reduced costs are discussed along with algorithmic procedures. Computational results with an empirical comparison of two methods are reported in Section 4. Final conclusions are drawn in Section 5.

2 Column Generation

We start with the *original* or *compact formulation* (cf. Lübbecke and Desrosiers, 2006) of an integer linear programming problem (IP), to which Dantzig-Wolfe decomposition is applied later on:

$$
\begin{aligned}
z^*_{IP} = \min \quad & c^\top x \\
\text{s.t.} \quad & Ax \geq b && \text{(1a)} \\
& Dx = d && \text{(1b)} \\
& x \in \mathbb{Z}^n_+ && \text{(1c)}
\end{aligned}
$$

In the applications we have in mind, the decision variables x are arc-flow variables x_{ij}, $(i,j) \in \mathcal{A}$ of an underlying network $\mathcal{N} = (V, \mathcal{A})$. Moreover, x might contain additional slack or surplus variables, resource variables, and others (cf. unified model of Desaulniers *et al.*, 1998b). (Note that some of the variables can be continuous variables but this option is left out in order not to overload the notation.) Constraints (1a) are referred to as *covering constraints*, while (1b) are constraints forming the well-structured domain of the subproblem. The LP-relaxation obtained by relaxing (1c) to $x \geq \mathbf{0}$ will be referred to as problem (P).

Let $X = \{x \in \mathbb{Z}^n_+ : Dx = d\}$ be the domain of the subproblem. For the sake of simplicity, we assume that X is bounded and therefore finite. Let Q be the $(n \times p)$-matrix whose columns uniquely correspond to elements of X. (Alternatively, one can define Q over the set of extreme points of the convex hull of X, i.e., using convexification of X instead of discretization of X (see Lübbecke and Desrosiers, 2006, p. 1011f, for details)). The *extensive formulation* of (IP) utilizes the equality $X = \{Q\lambda : \mathbf{1}^\top\lambda = 1, \lambda \in \{0,1\}^p\}$ and replaces x by $Q\lambda$:

$$
\begin{aligned}
z^*_{IP} = \min \quad & (c^\top Q)\lambda && \text{(2a)}\\
\text{s.t.} \quad & (AQ)\lambda \geq b && \text{(2b)}\\
& \mathbf{1}^\top\lambda = 1 && \text{(2c)}\\
& \lambda \geq \mathbf{0} && \text{(2d)}\\
& Q\lambda - x = \mathbf{0} && \text{(2e)}\\
& x \in \mathbb{Z}^n_+ && \text{(2f)}
\end{aligned}
$$

Since column-generation can only handle linear programs, one has to solve the LP-relaxation of (2) and embed the approach into a branch-and-bound scheme (called *branch-and-price*, see Barnhart *et al.* (1998)). For solving the LP-relaxation of (2), there is no need to keep the coupling constraints (2e) in the master program. Hence, the Dantzig-Wolfe master program (DWM) is given by (2a)–(2d). However, Poggi de Aragão and Uchoa (2003) suggested using the coupling constraint (2e) to directly retrieve reduced costs of variables x from (DWM) or some further reformulation, called *explicit master* (EM), see Section 3.2.

Recall that the column-generation principle always starts with a subset of the variables λ, the so-called *restricted master program* (RMP), and iteratively generates new variables as they are needed. The *pricing problem* $\mathrm{PP}(\pi, \mu)$ is defined for dual variables π and μ of the covering constraints (2b) and convexity constraints (2c) of (RMP). (Here and in the following, we assume that dual variables are represented by *row* vectors.) The solution to the pricing problem either delivers new negative reduced-cost variables (columns) or enables us to stop the iterative column generation process:

$$
\begin{aligned}
z^*_{PP(\pi,\mu)} = \min \quad & (c^\top - \pi A)x - \mu \\
\text{s.t.} \quad & Dx = d && \text{(3a)}\\
& x \in \mathbb{Z}^n_+
\end{aligned}
$$

Efficient column-generation approaches need 'nicely' structured domains $X = \{x \in \mathbb{Z}_+^n : Dx = d\}$ in order to keep the effort of solving the pricing problem manageable. One well-structured type of problem considered in the following is the shortest-path problem with resource constraints (SPPRC) (see Irnich and Desaulniers, 2005). Often, D has a diagonal structure so that X decomposes into several SPPRCs over identical or different networks $\mathcal{N}^k = (V^k, \mathcal{A}^k)$. This occurs when different groups k of identical vehicles or crews (or crew members) have to be routed or scheduled. For our analysis, we will assume a *single* network $\mathcal{N} = (V, \mathcal{A})$ with n vehicles/crews so that D decomposes into n identical blocks (of SPPRCs). The reformulation with groups of (path) variables λ must replace the convexity constraint (2c) by $\mathbf{1}^\top \lambda = n$, for which we still assume that the dual variable is denoted by μ. Multiple networks can be handled similarly: The only difference is that the convexity constraint (2c) has to be replaced by constraints for each group k of vehicles/crews, leading to constraints of the form $\mathbf{1}^{k\top} \lambda^k = n_k$ for all $k = 1, \ldots, K$ (see Barnhart *et al.*, 1998, Sect. 2.1). Note that, in this case, the dual variable μ is the sum $\mu_1 + \mu_2 + \ldots + \mu_K$ of the corresponding duals to these generalized convexity constraints. Note also that K pricing problems can—formally and algorithmically—be handled as a single problem, when the K subnetworks are merged (as disjoint or partially joint networks) by adding super-nodes s and t connected to the individual sources s_k and sinks t_k for $k \in \{1, \ldots, K\}$ (e.g., Kallehauge *et al.*, 2005, Sect. 8.2).

3 Variable Elimination

Given an integer-linear program (IP) with an upper bound u_{IP}, with objective $\min c^\top x$ and constraints $Ax \geq b, x \in \mathbb{Z}_+^n$, let $\bar{\alpha}$ be a feasible solution to the dual of the linear-programming relaxation of (IP). For the eth non-negative integer variable x_e, $(c - \bar{\alpha}A)_e > u_{IP} - \bar{\alpha}b$ implies $x_e = 0$ in any optimal solution to IP. Here, the term $c - \bar{\alpha}A$ is the vector of *reduced costs* depending on the dual solution $\bar{\alpha}$, denoted in the following by $r = r(\alpha)$. Concluding, if the reduced cost of a non-negative integer variable exceeds the optimality gap, the variable must be zero in every optimal integer solution.

The central question we address in this paper is the following: *Can we use information from the solution of the column-generation master (DWM) and PP(π, μ) to eliminate some of the variables x of the original formulation (IP)?* The answer to this can either be based on reduced costs of variables or any primal and dual information from the master and subproblem together with additional information generated by some algorithmic procedures. For arc-flow variables x of the original formulation (IP), their elimination imposes sparser networks in the pricing subproblem. SPPRCs on reduced networks can typically be solved faster, causing a speedup of the entire branch-and-price process. This is particularly true for large branch-and-bound trees, where several modified master problems with additional branching or cutting constraints have to be solved. However, methods for eliminating variables cause an additional effort on top of the column-generation solution procedure. One goal of this paper is to identify such methods for variable elimination, to clarify conditions under which they are applicable, and to analyze their

computational effort. We will distinguish between three methods:

(i) If $\mathrm{PP}(\pi, \mu)$ can be solved as a pure *linear* program, i.e., $X = \{x \geq \mathbf{0} : Dx = d\}$, the method of Walker (1969) is applicable: The optimal dual solution to (P) is (π^*, ρ^*), where (π^*, μ^*) are optimal dual solutions to the constraints (2b) and (2c) of (DWM), and ρ^* is an optimal dual solution to $\mathrm{PP}(\pi^*, \mu^*)$, i.e., to $\min(c^\top - \pi^* A)x, Dx = d, x \geq \mathbf{0}$. Reduced costs of x are then given by $r(\pi^*, \rho^*)^\top = c^\top - \pi^* A - \rho^* D$.

(ii) If (DWM) is solved with additional coupling constraints (2e), reduced costs of x can be directly retrieved from the solution of the restricted master program, as suggested by Poggi de Aragão and Uchoa (2003).

(iii) The new method we propose in this paper is the following: In order to eliminate an arc $(i, j) \in \mathcal{A}$, it is not necessary to compute a reduced cost for its corresponding arc-flow variable x_{ij} in the original formulation or some equivalent reformulation. Instead, we compute minimum reduced costs of *all* path variables of (DWM) containing the arc (i, j). If this minimum exceeds a given optimality gap, no path that contains the arc (i, j) can be used in an optimal solution. Hence, the arc (i, j) can be eliminated.

The following sections describe the three methods in detail and—in particular—discuss advantages and drawbacks when these methods are applied to different types of SPPRC subproblems. We assume that the reader is familiar with SPPRC and dynamic-programming solutions methods for SPPRC as, e.g., described in (Irnich and Desaulniers, 2005). We also use the terminology established there.

3.1 The Method of Walker (1969)

The application of Walker's method directly implies the following question: Which types of subproblems (=SPPRCs) can be formulated as pure linear programs and how can we retrieve the optimal dual solution?

3.1.1 Shortest-Path Problems without Resource Constraints

Shortest-path problems (SPP) without any resource and path-structural constraints can be formulated and solved as LPs as long as the underlying network does not have cycles of negative length. In column generation, negative cycles result from reduced arc costs $\tilde{c}_{ij} = \tilde{c}_{ij}(\pi, \mu)$ which have to be assigned to the arcs $(i, j) \in \mathcal{A}$ when solving the SPP on $\mathcal{N} = (V, \mathcal{A}, \tilde{c}(\pi, \mu))$. Acyclic networks, e.g., arising as pricing problems in (pure) multi-depot vehicle-scheduling problems (Ribeiro and Soumis, 1994), can be solved as LPs. The LP has the incidence matrix $I^{\mathcal{N}}$ of the network $\mathcal{N}$ as its constraint matrix D and the right hand side is $d = e_s - e_t$, where e_s and e_t are unit vectors corresponding to the source node s and the sink node t. The LP formulation of the subproblem is

$$\min \tilde{c}^\top x, \quad \text{s.t. } I^{\mathcal{N}} x = e_s - e_t, x \geq \mathbf{0}. \tag{4}$$

Instead of solving the SPP with general-purpose LP solvers, one wants to use more efficient algorithms, such as dynamic-programming labeling algorithms. When solving an s-t-shortest path problem with a labeling algorithm, the negative shortest path distances, i.e., the negative labels $(-\ell_i)_{i\in V}$ are identical to the dual variables $\rho = (\rho_i)_{i\in V}$ to (3a) (see Ahuja *et al.*, 1993, p. 136). Note that for pricing subproblems, the shortest-path distances ℓ depend on π, because the distances in the network $\mathcal{N} = (V, \mathcal{A}, \tilde{c})$ are defined by $\tilde{c}^\top = c^\top - \pi A$; they do not depend on μ.

The consequence for a column-generation approach is the following: Whenever (π, μ) is a feasible dual solution to (DWM), and (ℓ_i) are distance labels of the subproblem, reduced costs of variables x can be computed by $r^\top = r(\pi, \ell)^\top = c^\top - \pi A + \ell D$. The interpretation of the reduced cost

$$r_{ij} = r_{ij}(\pi, \ell) = c_{ij} - (\pi A)_{ij} + \ell_i - \ell_j \tag{5}$$

for a single arc-flow variable x_{ij} is that the network cost $\tilde{c}_{ij} = c_{ij} - (\pi A)_{ij}$ (used in the subproblem) is perturbed by $\ell_i - \ell_j$ (note that ℓD is a vector of size $|\mathcal{A}|$ with entries $\ell_i - \ell_j$ corresponding to the arcs $(i, j) \in \mathcal{A}$). There are several possibilities to compute feasible dual solutions (π, μ) to (DWM): One is to wait until the end of the column-generation process (at a node of the branch-and-bound-tree), where the optimal dual solution (π^*, μ^*) to the restricted master program is optimal to the dual of (DWM). Another possibility is to use an arbitrary optimal dual (π, μ) to any restricted master program together with the objective value of $\mathrm{PP}(\pi, \mu)$. It is straightforward to prove that $(\pi, \mu + nz^*_{PP(\pi,\mu)})$ is a feasible dual solution to (DWM). As a result, reduced costs can be computed without an additional effort in every iteration of the column-generation method, provided that the pricing problem is an SPP and is solved exactly. The reason for this is that optimal dual variables ρ to $\mathrm{PP}(\pi, \mu + nz^*_{PP(\pi,\mu)})$ only depend on π but not on μ or on $z^*_{PP(\pi,\mu)}$ and that reduced costs of x only depend on π and $\rho = -\ell$.

Note that the optimality conditions for shortest paths guarantee $\ell_j - \ell_i \leq \tilde{c}_{ij}$, implying $r_{ij} \geq 0$. If the $(\ell_i)_{i\in V}$ are shortest-path distance labels, the reduced costs r_{ij} are zero for all arcs $(i, j) \in \mathcal{A}$ that are used by non-dominated labels, i.e., all arcs in the shortest-path tree.

3.1.2 SPPRC without Path-Structural Constraints

We briefly recall the basic definitions of shortest-path problems with resource constraints (SPPRC); for a comprehensive introduction and analysis, we refer to (Irnich and Desaulniers, 2005; Irnich, 2006). For this section, any kind of path-structural constraints, such as precedence, pairing, elementarity, or k-cycle elimination constraints, are *not* present in the SPPRC formulation. Such 'pure' subproblems arise frequently, for instance, in some vehicle and crew scheduling and crew rostering applications (see, e.g., Desaulniers *et al.*, 1997; Gamache *et al.*, 1999).

Let $\mathcal{N} = (V, \mathcal{A})$ be a simple digraph. A *path* $P = (e_1, \ldots, e_p)$ is a finite sequence of arcs (some arcs may occur more than once), where the head node of $e_i \in \mathcal{A}$ is

identical to the tail node of $e_{i+1} \in \mathcal{A}$ for all $i \in \{1, \ldots, p-1\}$. Since $\mathcal{N}$ is assumed to be simple, a path can be written as $P = (v_0, v_1, \ldots, v_p)$, where $(v_{i-1}, v_i) \in \mathcal{A}$ holds for all $i \in \{1, \ldots, p\}$. Resource constraints can be formulated by means of *(minimal) resource consumptions* and *resource intervals* (e.g., the travel times t_{ij} and time windows $[a_i, b_i]$ in the case of the shortest-path problem with time windows (Desrochers and Soumis, 1988)). Let R be the number of resources. A vector $T = (T^1, \ldots, T^R)^\top \in \mathbb{R}^R$ is called a *resource vector* and its components *resource variables*. T is said to be *not greater* than S if the inequality $T^i \leq S^i$ holds for all components $i \in \{1, \ldots, R\}$. We denote this by $T \leq S$. For two resource vectors a and b, the interval $[a, b]$ is defined as the set $\{T \in \mathbb{R}^R : a \leq T \leq b\}$. Resource intervals, also called *resource windows*, are associated with nodes $i \in V$ and are denoted by $[a_i, b_i]$ with $a_i, b_i \in \mathbb{R}^R$. The changes in the resource consumptions associated with an arc $(i, j) \in \mathcal{A}$ are given by a vector $f_{ij} = (f_{ij}^r)_{r=1}^R$ of *resource extension functions* (REFs). An REF $f_{ij}^r : \mathbb{R}^R \to \mathbb{R}$ depends on a resource vector $T_i \in \mathbb{R}^R$, which corresponds to the resource consumption accumulated along a path $(s, \ldots, i)$ from s to i, i.e., up to the tail node i of arc (i, j). The result $f_{ij}(T_i) \in \mathbb{R}^R$ can be interpreted as a resource consumption accumulated along the path $(s, \ldots, i, j)$.

Let P be any path in $\mathcal{N}$. $P = (v_0, v_1, \ldots, v_p)$ is *resource-feasible* if resource vectors $T_i \in [a_{v_i}, b_{v_i}]$ exist for all $i \in \{0, 1, \ldots, p\}$ such that $f_{v_{i-1}, v_i}(T_{i-1}) \leq T_i$ holds for all $i \in \{1, \ldots, p\}$. Desaulniers *et al.* (1998b) were the first to point out that *non-decreasing* REFs, i.e., functions f_{ij} where $T \leq T'$ imposes $f_{ij}(T) \leq f_{ij}(T')$, allow efficient feasibility checks: P is resource-feasible, if and only if $T_0^P := a_{v_0} \leq b_{v_0}$ and $T_i^P := \max\{a_{v_i}, f_{v_{i-1}, v_i}(T_{i-1}^P)\} \leq b_{v_i}$ holds for all $i \in \{1, \ldots, p\}$. This can be interpreted as the forward propagation of minimum resource consumptions and their check against the upper bound of the resource consumption along the path.

Let $R = \mathit{cost}$ be the (reduced) cost resource and $1, \ldots, R-1$ be the other resources (such as time, accumulated load etc.). The SPPRC is a problem of the form

$$z^*_{SPPRC} = \min_{P=(v_0=s, v_1, \ldots, v_p=t) \in \mathcal{F}^{st}} (T_p^P)^{\mathit{cost}}, \tag{6}$$

where $\mathcal{F}^{st}$ is the set of all resource-feasible paths from source node s to sink node t. It can be solved using dynamic programming labeling algorithms (Irnich and Desaulniers, 2005, Sect. 4.1). As long as the resource windows and REFs are integer w.r.t. the resources $r \in \{1, \ldots, R-1\}$, SPPRCs can be solved in pseudo-polynomial time. In the following, we assume integer values for all non-cost resources.

Note that there exists no *pure* linear programming formulation of SPPRCs using variables x_{ij}, $(i, j) \in \mathcal{A}$ and resource variables only. First, the resource update along a path may involve non-linear REFs f_{ij}. Second, negative cost cycles render it impossible to associate a single resource vector (variable) with a node. Third, even in acyclic networks with linear REFs, the coupling between arc flow variables x_{ij}, $(i, j) \in \mathcal{A}$ and resource variables T_i, $i \in V$ requires non-linear constraints, such as $x_{ij}(T_j^r - T_i^r - t_{ij}^r) \geq 0$ for all $(i, j) \in \mathcal{A}$ and $r \in \{1, 2, \ldots, R-1\}$, or some linearization of these constraints imposing integer variables x_{ij}. The consequence

of the missing compact LP formulation is that additional variables and constraints are needed.

Next, we will show that SPPRCs defined by (6) can be solved as ordinary SPPs over the so-called *state space*. (In the following and in contrast to the use of the word in Dynamic Programming theory, we refer to a *state space* as a digraph, i.e., the set of states/nodes together with a set of transformations/arcs.) It also means that SPPRCs can be modeled and solved as pure LPs, but with a much larger set of variables and (flow conservation) constraints. A necessary assumption is, however, that REFs decouple resource updates of the non-cost resources from cost-computations (for details, see below). More formally, for any node $i \in V$ and any $T \in [a_i, b_i]$, define $\sigma(T) = (T^1, T^2, \ldots, T^{R-1})$ as the projection to the non-cost resource values. The set of *possible states* is defined by $\mathcal{S}_i = \{\sigma(T) \in \mathbb{Z}^{R-1} : T \in [a_i, b_i]\}$. The disjoint union $\mathcal{S} = \bigcup_{i \in V} \mathcal{S}_i$ of states forms the node set of the state space. Next, we define the arcs $\mathcal{A}^{\#}$ of the state space. Let $(i, j) \in \mathcal{A}$ be an arc of the original network. There are multiple 'parallel' arcs which connect states $\sigma \in \mathcal{S}_i$ with states $\sigma' \in \mathcal{S}_j$. We refer to them as $\mathcal{A}^{\#}_{ij}$. Moreover, let $T \in [a_i, b_i]$, $T' \in [a_j, b_j]$ be resource variables with $f_{ij}(T) = T'$, $\sigma := \sigma(T) \in \mathcal{S}_i$, and $\sigma' := \sigma(T') \in \mathcal{S}_j$. We want to assign a unique cost value to the arc $(\sigma, \sigma') \in \mathcal{A}^{\#}_{ij}$. Therefore, the cost difference $T'^{cost} - T^{cost}$ is assumed to be identical for all T and T' with $(\sigma(T), \sigma(T')) = (\sigma, \sigma')$ and $f_{ij}(T) = T'$. By defining $d_{\sigma\sigma'} = T'^{cost} - T^{cost}$ one gets the discretization of the SPPRC in the form of the network $\mathcal{N}' := (\mathcal{S}, \bigcup_{(i,j)} \mathcal{A}^{\#}_{ij}, d)$.

The solution to (6) can be obtained by computing a shortest path $P^{\#}$ in $\mathcal{N}'$ starting at state $\sigma(a_s) \in \mathcal{S}_s$ and ending at any state of $\mathcal{S}_t$. The nodes associated with the states visited by $P^{\#}$ yield the optimum s-t-path of the SPPRC. Note that, typically, the state space is acyclic because some resource consumptions are strictly increasing as, e.g., for the resource time.

The solution of an SPPRC instance by dynamic-programming labeling algorithms is performed by propagating labels through the *original* network $\mathcal{N}$. A *label* consists of the minimum resource consumption $T \in \mathbb{Z}^R$ accumulated along its associated path $P = P(T)$ and a link to a predecessor label, such that several paths sharing a common prefix are efficiently stored in a tree data-structure. The process starts with the initial label $T = a_s$ at the source s and is continued by extending this label and its successors by means of REFs. The two main components of such a labeling procedure are the *path extension step* and the *dominance algorithm* (Irnich and Desaulniers, 2005, Sect. 4.1). In the shortest-path computation on the state space, the path extension step is recreated exactly by the propagation of costs from one state to the next. However, there is no full equivalent in $\mathcal{N}' = (\mathcal{S}, \bigcup_{(i,j)} \mathcal{A}^{\#}_{ij}, d)$ to the dominance algorithm: SPPRC dominance algorithms can eliminate labels with non-identical states, i.e., when the resource consumption of one label is greater or equal to the resource consumption of another label. This allows the omittance of non-useful paths in the path extension step. In $\mathcal{N}'$, one can interpret this dominance algorithm as an *implicit removal* of all unreachable and dominated states. (A dominated state $\sigma_1 \in \mathcal{S}_i$ is one for which there exists another

state $\sigma_2 \in \mathcal{S}_i$ with $\sigma_2 \leq \sigma_1$ and $\ell_{\sigma_2} \leq \ell_{\sigma_1}$.) Contrary, a shortest-path algorithm in $\mathcal{N}'$ just compares costs. The equivalence in the SPPRC dynamic programming labeling algorithm would be the use of a very restricted dominance rule which only compares SPPRC labels which have identical states. In order to have a stronger dominance, we use additional zero-cost arcs $(\sigma, \sigma') \in \mathcal{A}_i^{\#}$ connecting pairs of states with $\sigma \leq \sigma'$ belonging to the same node $i \in V$. In the following, we assume that the state space is defined by $\mathcal{N}^{\#} = (\mathcal{S}, \mathcal{A}^{\#}, d) = (\mathcal{S}, \bigcup_{(i,j)\in\mathcal{A}} \mathcal{A}_{ij}^{\#} \cup \bigcup_{i\in V} \mathcal{A}_i^{\#}, d)$. Note that we have introduced $\mathcal{N}^{\#}$ for reasons of explanation and not for efficient computation purposes.

The view on the state space $\mathcal{N}^{\#}$ enables us to apply variable elimination to the original formulation and, thus, to the pricing network. We can consider two equivalent original formulations both using variables $y = (y_{\sigma\sigma'})$.

Compact Formulation with Variables $y = (y_{\sigma\sigma'})$ Only For the moment, we assume that the original model (1) is formulated in terms of routing variables $y_{\sigma\sigma'}$ for $(\sigma, \sigma') \in \mathcal{A}^{\#}$ instead of variables x_{ij} for $(i, j) \in \mathcal{A}$. This assumption is no serious restriction, since every variable x_{ij} is the sum of the $y_{\sigma\sigma'}$ with $(\sigma, \sigma') \in \mathcal{A}_{ij}^{\#}$. Now, the method of Walker—as explained in Section 3.1.1—is applicable to the compact formulation with variables $y_{\sigma\sigma'}$ and enables us to eliminate some of the variables $y_{\sigma\sigma'}$. Formally, we replace $x = (x_{ij})$ by $Q'y = Q'(y_{\sigma\sigma'})$, where the matrix $Q' \in \{0,1\}^{|\mathcal{A}^{\#}|\times|\mathcal{A}|}$ shows which of the arcs (σ, σ') correspond to an original arc (i, j). The compact formulation now is

$$\min c^{\top} Q'y, \qquad \text{s.t.} \quad AQ'y \geq b, DQ'y = d, y \in \mathbb{Z}_+^{|\mathcal{A}^{\#}|}. \tag{7a}$$

Since Q' is a 0-1-matrix, the condition $Q'y \in \mathbb{Z}_+^{|\mathcal{A}|}$ is equivalent to $y \in \mathbb{Z}_+^{|\mathcal{A}^{\#}|}$. Hence, the domain $Y = \{Q'y : DQ'y = d, y \in \mathbb{Z}_+^{|\mathcal{A}|}\}$ of the subproblem is identical to $X = \{Q\lambda : \mathbf{1}\lambda = 1, \lambda \in \{0,1\}^p\}$. Consequently, the master program (DWM) is still given by (2a)–(2d). The pricing problem $\mathrm{PP}(\pi, \mu)$ becomes $\min(c^{\top} - \pi A)Q'y$, subject to $y \in Y$. The point is that this pricing problem is identical to the pure LP, i.e., $\min(c^{\top} - \pi A)Q'y$ subject to $I^{\mathcal{N}^{\#}} y = e_{\sigma(a_s)} - e_{\sigma(b_t)}, y \geq \mathbf{0}$.

In order to eliminate arcs (σ, σ'), we do *not* need to solve the pricing problem in variables $y_{\sigma\sigma'}$ as LPs on the huge network $\mathcal{N}^{\#}$. Instead, we can simply apply standard dynamic programming labeling procedures to $\mathcal{N} = (V, \mathcal{A})$ and interpret the (Pareto-optimal) final labels T_i in the right way: A final non-dominated label T associated with node i corresponds to a cost label $\ell_\sigma = T^{cost}$ of state $\sigma = \sigma(T) \in \mathcal{S}_i$. With the reduced cost arguments given in Section 3.1.1, we are able to eliminate those arcs from $\mathcal{N}^{\#}$ whose reduced costs exceed the optimality gap. For a more precise algorithmic description, assume that Γ_i is the set of labels associated with node i and that each label $T \in \Gamma_i$ has state $\sigma(T) \in \mathcal{S}_i$ and cost $T^{cost} = \ell(T)$:

Algorithm 1 Reduced Cost Computation for State Space

1: **Input:** Sets Γ_i of labels at all nodes $i \in V$.

2: FORALL $(i,j) \in \mathcal{A}$ DO
3: FORALL $(\sigma,\sigma') \in \mathcal{A}_{ij}^{\#}$ DO
4: LET $r_{\sigma\sigma'} := \tilde{c}_{ij} + (\min_{T\in\Gamma_i:\sigma(T)\leq\sigma} T^{cost}) - (\min_{T'\in\Gamma_j:\sigma(T')\leq\sigma'} T'^{cost})$
5: **Output:** Reduced costs $r_{\sigma\sigma'}$.

We can use the reduced costs $r_{\sigma\sigma'}$ not only to eliminate arcs $(\sigma,\sigma') \in \mathcal{A}^{\#}$. If all arcs $(\sigma,\sigma') \in \mathcal{A}_{ij}^{\#}$ can be removed, the original arc (i,j) can also be removed from the pricing network $\mathcal{N}$. Note that this happens if the minimum reduced cost $r_{\sigma\sigma'}$ of the arcs $(\sigma,\sigma') \in \mathcal{A}_{ij}^{\#}$ exceeds the optimality gap.

Finally note that the addition of arcs $\mathcal{A}_i^{\#}$ to the state space $\mathcal{N}^{\#}$ is crucial for the correctness of the Formula in Step 4 of Algorithm 1. Without these arcs, we would have $r_{\sigma\sigma'} = \tilde{c}_{ij} - T^{cost} + T'^{cost}$ only if labels $T \in \Gamma_i$ and $T' \in \Gamma_j$ with $\sigma = \sigma(T)$ and $\sigma' = \sigma(T')$ exists and $r_{\sigma\sigma'} = 0$ otherwise. The more arcs in the state space, the smaller the chance that arcs $(\sigma,\sigma') \in \mathcal{A}_{ij}^{\#}$ are in the shortest-path tree (which means a reduced cost of 0).

Compact Formulation with a Mix of Variables $x = (x_{ij})$ and $y = (y_{\sigma\sigma'})$ One can utilize the fact that each arc variable x_{ij} can be expressed as the sum of variables $y_{\sigma\sigma'}$, $(\sigma,\sigma') \in \mathcal{A}_{ij}^{\#}$. With the results for reformulations and their impact on dual solutions, as given in the Appendix, we first extend (7a) by

$$x_{ij} = \sum_{(\sigma,\sigma')\in\mathcal{A}_{ij}^{\#}} y_{\sigma\sigma'} \quad \text{for all } (i,j) \in \mathcal{A}. \tag{7b}$$

Proposition 1 (in the Appendix) implies that the addition of (7b) leads to a fully equivalent model (P'_{ext}), in which the variables x_{ij} always have reduced costs of 0. This extension is, therefore, uninteresting for our purposes. Amazingly, the (formal) addition of the non-negativity constraints

$$x_{ij} \geq \mathbf{0} \quad \text{for all } (i,j) \in \mathcal{A} \tag{7c}$$

leads to the compact formulation (7) that corresponds to model (P_{ext}) of the Appendix. Here, Proposition 2 implies that the added variables x_{ij} can have positive reduced costs. For any dual feasible solution to the LP-relaxation of (7) or (7a), the reduced cost r_{ij} of x_{ij} can be chosen such that it is the minimum of all reduced costs $r_{\sigma\sigma'}$ of variables $y_{\sigma\sigma'}$, $(\sigma,\sigma') \in \mathcal{A}_{ij}^{\#}$. This obviously coincides with the arc-elimination criterion given in the last paragraph.

Moreover, Proposition 3 makes clear that, w.r.t. dual solutions and reduced costs, it is fully equivalent to formulating the covering constraints and objective either in variables x_{ij} or variables $y_{\sigma\sigma'}$, i.e., to minimize $c^\top x$ or $c^\top Q'y$ over $Ax \geq b$ or $AQ'y \geq b$, respectively. Because of Proposition 3(iii), these reformulations have no impact on the reduced costs of the variables x_{ij}. In all cases, the Dantzig-Wolfe decomposition of the models leads the model (DWM) with path variables. Its

subproblem can, again, be solved on the original network $\mathcal{N}$ with SPPRC labeling algorithms. This means that neither the master nor the subproblem have to work explicitly on variables y. The consideration of variables $y_{\sigma\sigma'}$ is, therefore, nothing other than a formal device to derive reduced costs of the original variables x. We achieve the following procedure for the computation of reduced costs of variables x_{ij} by means of SPPRC labels:

Algorithm 2 Reduced Cost Computation for SPPRC

1: **Input:** Sets Γ_i of labels at all nodes $i \in V$.
2: FORALL $(i,j) \in \mathcal{A}$ DO
3: LET $r_{ij} := \tilde{c}_{ij} + (\min_{T \in \Gamma_i} T^{cost}) - (\min_{T' \in \Gamma_j : \sigma(T') \leq \sigma(f_{ij}(T))} T'^{cost})$
4: **Output:** Reduced costs r_{ij} of variables $x = (x_{ij})$.

Resource Window Reduction If only a subset of the variables $y_{\sigma\sigma'}$ with $(\sigma, \sigma') \in \mathcal{A}_{ij}^{\#}$ can be eliminated, but not all of them to also eliminate the original arc $(i,j) \in \mathcal{A}$, the consideration of eliminated arcs in $\mathcal{N}^{\#}$ is still useful: The remaining (not eliminated) arcs are denoted by $\mathcal{A}_{ij}^{\#,rem}$ for $(i,j) \in \mathcal{A}$. For every node i, one can compute the maximum resource vector $\bar{b}_i$ with $\sigma(\bar{b}_i) \leq \sigma_i$ for all (i,j) in the forward star of i and all $(\sigma_i, \sigma_j) \in \mathcal{A}_{ij}^{\#,rem}$. Similarly, for any node j let $\bar{a}_j$ be the minimum resource vector with $\sigma(\bar{a}_j) \geq \sigma_j$ for all (i,j) in the backward star of j and all $(\sigma_i, \sigma_j) \in \mathcal{A}_{ij}^{\#,rem}$. One can now replace the resource window $[a_i^r, b_i^r]$ by $[\bar{a}_i^r, \bar{b}_i^r]$ for all nodes $i \in V$ and all non-cost resources $r \in \{1, \ldots, R-1\}$. This is a procedure to tighten resource windows. Similar ideas were suggested for the MDVSP by Hadjar *et al.* (2001).

3.1.3 SPPRC with Path-Structural Constraints

We cannot directly apply the same techniques of discretization to SPPRCs with path-structural constraints. The reason for this is that path-structural constraints, such as precedence and pairing constraints, k-cycle freeness, and elementarity, cannot be formulated directly, so that a pure LP represents the problem (e.g., by extending the above mentioned network flow formulation $I^{\mathcal{N}^{\#}} x = e_s - e_t$, $x \geq \mathbf{0}$). The word 'directly' means that it is, however, possible to model path-structural constraints by means of additional resources. For instance, Feillet *et al.* (2004); Salani (2005); Dell'Amico *et al.* (2006) use $|V|$ additional binary resources in the *elementary* SPPRC (ESPPRC) in order to keep track of the nodes that a path has visited or cannot visit anymore. Due to such a vast extension of the state space, we can expect a combinatorial explosion in the number of states and arcs as well as ESPPRC labels. The impact on the reduced costs is that, for every original arc $(i,j) \in \mathcal{A}$, in most of the cases at least one arc $(\sigma, \sigma') \in \mathcal{A}_{ij}^{\#}$ of the *extended* state space has reduced cost 0. Then, (i,j) cannot be eliminated on the basis of reduced costs r_{ij}.

The situation of efficient labeling algorithms for SPPRC with k-cycle elimination (SPPRC-k-cyc) is different. As for the ESPPRC, additional resources for each node $i \in V$ can be used to model k-cycle elimination on an extended state space:

Using Kronecker's symbol δ_{ab} (with $\delta_{ab} = 1$ if $a = b$ and $\delta_{ab} = 0$ otherwise), the REF of arc $(i,j) \in \mathcal{A}$ or $(\sigma,\sigma') \in \mathcal{A}_{ij}^{\#}$ is $f(T)^v = \max\{T^v - 1, k\delta_{iv}\}$ for resource (=node) $v \in V$. The resource window for resource v is $[0, k(1-\delta_{iv})]$ at node i or $\sigma_i \in \mathcal{S}_i$, respectively. This is a minor refinement of the simple SPPRC-k-cyc dominance rule discussed in (Irnich and Villeneuve, 2006, §5.1). Extending the 2-cycle elimination dominance rules of Kohl (1995), Irnich and Villeneuve were able to develop dominance rules for SPPRC-k-cyc with $k \geq 3$ which are stronger than the above straightforward approach of exhaustively extending the state space: Several labels $T_1, T_2, \ldots, T_q \in \Gamma_i$, which dominate a label $T \in \Gamma_i$ w.r.t. resources, allow the discarding of T even if all these labels come from (partially) different predecessor nodes. (Here, the labels $T_1, T_2, \ldots, T_q$ and T refer to the original resources and do *not* include additional resources for nodes!) We denote by $\mathcal{E}(T)$ the set of possible k-cycle free extensions of a label T. If $\bigcup_{p=1}^{q} \mathcal{E}(T_p) \supseteq \mathcal{E}(T)$ and dominance w.r.t. resources holds, then T can be discarded (see Irnich and Villeneuve, 2006, §6, for details). (Computationally efficient implementations encode these extensions by so-called *hole sets*.) The point is that such a stronger dominance rule cannot be modeled on any state-space graph. However, one can mimic the stronger dominance rule in SPPRC algorithms working on the state space $\mathcal{N}^{\#}$ (with the original resources) by adding additional artificial labels. Any subset of labels $\Gamma \subseteq \Gamma_j$ creates an *artificial label* at the same node j with resources $\max_{T\in\Gamma} T$ (componentwise) and possible extensions $\bigcup_{T\in\Gamma} \mathcal{E}(T)$. These artificial labels create additional connections between states of the same original node, and, therefore, more arcs $(\sigma,\sigma') \in \mathcal{A}_{ij}^{\#}$ have positive reduced costs. This finally imposes larger reduced costs on the original arcs $(i,j) \in \mathcal{A}$.

Algorithm 3 Reduced Cost Computation for SPPRC-k-cyc

1: **Input:** Sets Γ_i of labels at all nodes $i \in V$.
2: FORALL $(i,j) \in \mathcal{A}$ DO
3: LET $r_{ij} := \tilde{c}_{ij} + (\min_{T\in\Gamma_i} T^{cost}) - (\min_{\Gamma\subseteq\Gamma_j:\Gamma \text{ } dominates \text{ } f_{ij}(T)} (\max_{T'\in\Gamma} T'^{cost}))$
4: **Output:** Reduced costs r_{ij} of variables $x = (x_{ij})$.

In Step 3, a subset of labels $\Gamma \subseteq \Gamma_j$ *dominates* the label $T' := f_{ij}(T)$ corresponding to path $(P(T),(i,j))$, if $T \leq T'$ holds for all $T \in \Gamma$ and $\bigcup_{T\in\Gamma} \mathcal{E}(T) \supseteq \mathcal{E}(T')$ holds. The minimization over all subsets $\Gamma \subseteq \Gamma_j$ can easily be solved by simply sorting the labels in Γ_j by increasing costs and testing all subsets Γ with the first q labels for $q = 1, 2, \ldots, |\Gamma_j|$. Finally note that both types of labels (ordinary and artificial) together give a dual feasible solution to the shortest-path problem on $\mathcal{N}^{\#}$. Hence, the values r_{ij} are valid reduced costs for the variables x_{ij} of the original formulation.

3.2 The Method of Poggi de Aragão and Uchoa (2003)

Recall that we refer to (2a)–(2d) as (DWM). The LP-relaxation of the extensive formulation, i.e., (2a)–(2f), is denoted by (DWM$_{ext}$). The presence of the coupling constraints (2e) in the master program offers the possibility to directly retrieve reduced costs of variables x coming from the original formulation. This was first

suggested by Poggi de Aragão and Uchoa (2003). However, the coupling constraints in the master have important theoretical and algorithmic implications which are pointed out in the following.

First, Propositions 1 and 2 show that it is crucial to also keep $x \geq \mathbf{0}$ in the extensive formulation. Otherwise, the reduced costs are 0. However, even with the non-negativity constraints $x \geq \mathbf{0}$, there is no guarantee that any of the reduced costs of the variables x are positive and useful for variable elimination. Proposition 2 explains this: For every dual feasible solution (π, μ) to (DWM), there exists the dual feasible solution $(\pi, \mu, \mathbf{0})$ to the extensive formulation (DMW_{ext}), for which all x_{ij} have reduced cost 0. Beside this dual solution $(\pi, \mu, \mathbf{0})$, many other dual solutions typically exist, that imply different reduced costs on the x variables. Nevertheless, the result that (DWM_{ext}) can leave us with 'poor' reduced cost information simply means that we cannot *control* the output of (DWM_{ext}).

Second, Poggi de Aragão and Uchoa (2003) suggest to reformulating (DWM_{ext}) to the following *explicit master* (EM):

$$
\begin{aligned}
z^*_{EM} = \min \quad & c^\top x && \text{(8a)}\\
\text{s.t.} \quad & Ax \geq b && \text{(8b)}\\
& Q\lambda - x = \mathbf{0} && \text{(8c)}\\
& \mathbf{1}^\top \lambda = 1 && \text{(8d)}\\
& \lambda \geq \mathbf{0}, x \geq \mathbf{0} && \text{(8e)}
\end{aligned}
$$

(EM) differs from (DWM) by the formulation of the objective (8a) and the covering constraints (8b) in the original variables x. The advantage of (EM) is that it is as strong as (DWM) and (DWM_{ext}), but that its associated pricing problem is fully independent of the dual prices π of the constraints $Ax \geq b$. This allows, for instance, the addition of any inequality formulated in x to (8b)—for branching or cutting—without affecting the structure of the pricing problem for the generation of variables λ. Formulations with these properties lead to so-called *robust* branch-and-price(-and-cut) algorithm. With respect to reduced costs, (EM) is, however, no better than (DWM_{ext}), since degenerated solutions with reduced cost 0 are still possible. This is exactly the statement of Proposition 3 applied to (DWM_{ext}) and (EM) (i.e., models (P_{ext}) and (P^3_{ext}) in Proposition 3).

Third, Desrosiers and Lübbecke (2005, p. 11) suggest keeping the coupling constraints in the (restricted) master program and imposing the additional constraints $x \geq \boldsymbol{\varepsilon}$, for a small $\boldsymbol{\varepsilon} \geq \mathbf{0}$, at the end of the (column-generation) process. The shadow prices of these constraints are then the reduced costs of the original variables x. Here, the addition of $x \geq \boldsymbol{\varepsilon}$ changes the objective value and optimal solution of the master program. It is not clear to us as to whether we can deduce general statements about the computed reduced costs. Nevertheless, the addition of the constraints $x \geq \boldsymbol{\varepsilon}$ at the end of the column-generation process leaves several questions open: Should we add the constraints $x_{ij} \geq \varepsilon_{ij}$ for all arcs (i, j) simultaneously or consecutively? This makes either a single re-optimization of the master program

or several re-optimizations necessary. Are all values ε_{ij} identical? Do we have to perform additional iterations with pricing and re-optimizations? What is the quality of the computed reduced costs?

From an algorithmic point of view, there are also good arguments for not leaving the coupling constraints (2e) in a master program unless they are absolutely necessary: The main reason is that $Q\lambda - x = \mathbf{0}$ consists of $|\mathcal{A}|$ constraints, which substantially extends the restricted master program and can often make the LP too large to be solved iteratively.

Finally note that we do not negate the usefulness of considering (DWM$_{ext}$) or (EM) for theoretical purposes, such as the devising of *robust* branch-and-price algorithms with effective branching rules, allowing strong cutting planes to be added to the master, or for multiple column generation, as exemplified in the work of Poggi de Aragão and Uchoa (2003) and Lübbecke and Desrosiers (2006).

3.3 The Bidirectional Search Method

Proposition 2(iv) applied to (DMW$_{ext}$) gives us another way of expressing (maximum) reduced costs of variables x_{ij}. Let $\mathcal{F}_{ij}^{st}$ be the set of feasible s-t-paths containing arc (i,j). Moreover, let n_{ij}^P be the number of times arc (i,j) occurs in path $P \in \mathcal{F}_{ij}^{st}$. The path variables $\lambda = (\lambda_P)$ of (DWM) and (DWM$_{ext}$) have reduced costs $\tilde{c}(\pi,\mu) = c^\top Q - \pi A Q - \mu \mathbf{1}$ so that

$$r_{ij} := \min_{P \in \mathcal{F}_{ij}^{st}} \frac{\tilde{c}_P(\pi,\mu)}{n_{ij}^P} \leq \min_{P \in \mathcal{F}_{ij}^{st}} \tilde{c}_P(\pi,\mu) =: \bar{r}_{ij} \tag{9}$$

follows. Any value between 0 and r_{ij} is a valid reduced cost of x_{ij}. If $\bar{r}_{ij} > r_{ij}$, the value $\bar{r}_{ij}$ is not a valid reduced cost of a *variable* in an original compact formulation. However, we will show that $\bar{r}_{ij}$ has a meaningful interpretation that justifies its use instead of r_{ij} for variable elimination. The interesting algorithmic question now is how to compute these values.

The key observation is that $\bar{r}_{ij}$ is the minimum reduced cost of all paths containing arc (i,j). The proposed technique to determine $\bar{r}_{ij}$ is bidirectional shortest-path computation. Any path $P \in \mathcal{F}_{ij}^{st}$ can be decomposed into $P = (P_1, (i,j), P_2)$, where $P_1 \in \mathcal{F}^{si}$ and $P_2 \in \mathcal{F}^{jt}$. (The decomposition is not unique if $n_{ij}^P > 1$.) The values $\tilde{c}(P_1)$ and $\tilde{c}(P_2)$ can be bounded from above by results of two SPPRC computations: A standard *forward* SPPRC labeling procedure produces usual labels Γ_i^{fw} at nodes $i \in V$. Assuming that P_1 has an associated state $\sigma = \sigma(P_1) \in \mathcal{S}_i$, the path P_1 fulfills

$$\tilde{c}(P_1) \geq \min_{T \in \Gamma_i^{fw} : \sigma(T) \leq \sigma(P_1)} T^{cost}.$$

Similarly, a backward SPPRC labeling algorithm can be used to bound $\tilde{c}(P_2)$. One starts with the initial path (t) and extends partial paths ending at a node ℓ against

the arcs direction (k, ℓ) to a node k. Salani (2005) has shown that it is possible to extend resources (such as cost, time, and load) in the opposite direction. A unifying description of REFs and their inversion was presented in (Irnich, 2006): The idea here is that upper bounds on the resource consumption are propagated backward by means of inverse REFs. The paper also clarifies which types of REFs are invertible so that backward SPPRCs are well-defined and compatible with the forward SPPRCs. Assuming that backward REFs exist, the corresponding labels represent paths from a current node j to the sink t. The backward SPPRC labeling algorithm generates a set of backward labels Γ_j^{bw} for each node $j \in V$. Now, the cost of P_2 fulfills

$$\tilde{c}(P_2) \geq \min_{T' \in \Gamma_j^{bw} : \sigma(T') \geq \sigma(P_2)} T'^{cost}.$$

Putting the results together, one gets

$$\bar{r}_{ij} = \tilde{c}_{ij} + \min_{\substack{T \in \Gamma_i^{fw}, T' \in \Gamma_j^{bw} : \sigma(f_{ij}(T)) \leq \sigma(T'), \\ (P(T), i, j, P(T')) \in \mathcal{F}^{st}}} \left(T^{cost} + T'^{cost} \right) - \mu. \tag{10}$$

The interpretation of (10) is that one first has to solve the forward and then the corresponding backward SPPRC with arc costs $\tilde{c}_{ij}^{\top} = c^{\top} - (\pi A)_{ij}$. Subsequently, one must determine all matching pairs (T, T') of labels at node i and node j, respectively. Two labels $T \in \Gamma_i^{fw}$ and $T' \in \Gamma_j^{bw}$ match if they compose a feasible s-t-path $(P(T), (i, j), P(T')) \in \mathcal{F}^{st}$. Feasibility concerns two aspects: The path is resource-feasible if the lower bound T extended along the arc (i, j) does not exceed the upper bound of the resource consumption given by T'. Moreover, feasibility w.r.t. path-structural constraints, such as k-cycle freeness, elementarity, and precedence and pairing constraints, have to be tested. The formal description of the procedure reads as follows:

Algorithm 4 Computation of $\bar{r}$ for (E)SPPRC(-k-cyc)

Input: Sets Γ_i^{fw} of forward labels at all nodes $i \in V$.
Sets Γ_j^{bw} of backward labels at all nodes $j \in V$.
FORALL $(i, j) \in \mathcal{A}$ DO
  LET $\bar{r}_{ij} := \infty$
  FORALL $T \in \Gamma_i^{fw}$ DO
    FORALL $T' \in \Gamma_j^{bw}$ DO
      IF $(\sigma(f_{ij}(T)) \leq \sigma(T')$ AND $(P(T), (i, j), P(T')) \in \mathcal{F}^{st})$ THEN
        LET $r^P := T^{cost} + \tilde{c}_{ij} + T'^{cost}$
        LET $\bar{r}_{ij} := \min\{\bar{r}_{ij}, r^P\}$
Output: Values $\bar{r}_{ij}$ for all arcs $(i, j) \in \mathcal{A}$.

The result is a value $\bar{r}_{ij}$ for each arc $(i, j) \in \mathcal{A}$. If $\bar{r}_{ij}$ exceeds the optimality gap, it means that every path containing the arc (i, j) is not part of any optimal solution to (IP). Hence, all paths $\mathcal{F}^{ij}$ can be removed from the master program (DWM).

More importantly, we know that the arc (i, j) can also be removed from the pricing network $\mathcal{N}$, since it can only produce non-optimal paths. Note that we have argued directly using the column-generation formulation.

If (DWM) is formulated solely with elementary paths, the values n_{ij}^P are all 0 or 1 and, hence, $\bar{r}_{ij}$ and r_{ij} are identical. In this case, $r_{ij} = \bar{r}_{ij}$ is a proper reduced cost of the original variable x_{ij}.

Concerning the computational complexity of Algorithm 4, we have to distinguish between different types of SPPRC labeling algorithms: Standard labeling algorithms compute the set of all undominated forward labels every time the pricing problem is solved exactly. An exact solution is required at least once for the solution of a branch-and-price tree node, i.e., when optimality of the corresponding restricted master program is proven. The additional effort of Algorithm 4 is the computation of all undominated backward labels (which is typically as hard as solving the forward SPPRC) and the comparison of forward and backward labels (Steps 5–9).

Also from a theoretical point of view, a comparison of r_{ij} and $\bar{r}_{ij}$ is interesting. In the state space $\mathcal{N}^{\#}$, forward and backward SPP is trivial to implement. Let $(\ell_\sigma^{fw})_{\sigma\in\mathcal{S}}$ be the forward labels and $(\ell_\sigma^{bw})_{\sigma\in\mathcal{S}}$ be the backward labels. Obviously, $\ell_{\sigma(a_s)}^{fw} = \ell_{\sigma(b_t)}^{bw} = 0$ and $\ell_{\sigma(b_t)}^{fw} = \ell_{\sigma(a_s)}^{bw} = z^*_{PP(\pi,\mu)}$. Any dual feasible solution (π, μ) to (DWM) guarantees $z^*_{PP(\pi,\mu)} \geq 0$. According to (5), for any arc $(\sigma, \sigma') \in \mathcal{A}_{ij}^{\#}$ the equation $r_{\sigma\sigma'} = \tilde{c}_{ij} + \ell_\sigma^{fw} - \ell_{\sigma'}^{fw}$ holds. For any state $\sigma \in \mathcal{S}$, the optimal solution of the SPP implies $\ell_\sigma^{fw} + \ell_\sigma^{bw} \geq z_{PP(\pi,\mu)} \geq 0$ and, therefore, $r_{\sigma\sigma'} = \tilde{c}_{ij} + \ell_\sigma^{fw} - \ell_{\sigma'}^{fw} \leq \ell_\sigma^{fw} + \tilde{c}_{ij} + \ell_{\sigma'}^{bw}$. Taking the minimum over all arcs $(\sigma, \sigma') \in \mathcal{A}_{ij}^{\#}$ yields

$$r_{ij} = \min_{(\sigma,\sigma')\in\mathcal{A}_{ij}^{\#}} r_{\sigma\sigma'} \leq \min_{(\sigma,\sigma')\in\mathcal{A}_{ij}^{\#}} \left(\ell_\sigma^{fw} + \tilde{c}_{ij} + \ell_{\sigma'}^{bw}\right) = \bar{r}_{ij}.$$

Concluding, for any given dual feasible solution (π, μ), the reduced cost r_{ij} computed by the method of Walker is not larger than the value $\bar{r}_{ij}$. Hence, the bidirectional method is superior to Walker's method, since it can eliminate at least the same arcs as Walker's method can.

A current and very successful trend for solving hard VRPs with time windows or other VRP variants with branch-and-price is to use the above mentioned elementary path formulations. Since ESPPRC is $\mathcal{NP}$-hard in the strong sense (Dror, 1994), the main difficulty lies in the development of effective labeling algorithms which can practically handle the elementarity constraints. Besides other techniques, such as extension of dominance rules (Feillet *et al.*, 2004) and state space relaxation/augmentation (Salani, 2005; Boland *et al.*, 2006), one of the most effective approaches is that of *bidirectional search with bounding*, which was successfully tested by Salani (2005) and Righini and Salani (2004). Inspired by the observation that the number of undominated ESPPRC paths typically grows exponentially with the length of these paths, one tries to bound the length of paths to half of the maximum path length: Thus, the first half of a path is computed by an s-to-all

labeling algorithm and the second half is computed by a backward t-to-all labeling algorithm. A so-called *critical* resource is used to control the maximum forward and backward path length (for details, see Salani, 2005). When this technique is used to solve ESPPRCs, Algorithm 4 is not directly applicable. The reason is that the computation of $\bar{r}_{ij}$ needs the *complete* set of all forward and backward labels. The half-way bounding technique does not give us a valid lower bound for either the first path P_1 or the second path P_2.

However, there is no need to use the same forward labels, as computed in the pricing problem together with backward labels of the inverse SPPRC. Any labeling solution, in particular a solution to an easier-to-solve ESPPRC relaxation, provides valid bounds. Possible relaxations are the (non-elementary) SPPRC (Irnich and Desaulniers, 2005), the SPPRC-k-cyc for $k \geq 2$ (Irnich and Villeneuve, 2006), and the SPPRC with forbidden (sub)paths (Villeneuve and Desaulniers, 2005). One can expect that the quality of the computed reduced costs depends significantly on the hardness of the relaxation used.

4 Computational Results

For the empirical evaluation of Walker's method (shortly denoted by W) and the bidirectional method (B), we examine a branch-and-price-and-cut algorithm for the vehicle-routing problem with time windows (VRPTW) tested on the well-known benchmark set of Solomon (1987). The implementation used for the following analysis is the one previously used in (Irnich and Villeneuve, 2006). In order to keep the computations verifiable, we leave out several other well-known acceleration techniques, such as massive heuristic pricing, complex branching rules and strong branching, sophisticated non-robust cutting planes, stabilization etc. (see Jepsen *et al.*, 2006; Desaulniers *et al.*, 2006, and literature cited there). The solver for the pricing problem only makes use of a monodirectional SPPRC-k-cyc labeling algorithm and tries heuristic pricing in a 5-nearest neighbor subnetwork. 1-path cuts and 2-path cuts (Kohl *et al.*, 1999) are the only cutting planes which are added to the root node of the branch-and-bound tree. Branching is performed with a best-node-first strategy, first on the number of vehicles (if fractional) and then on the arc $(i, j) \in \mathcal{A}$, where the product $c_{ij} \cdot \min\{\bar{x}_{ij}, 1 - \bar{x}_{ij}\}$ of cost and deviation from the next integer is maximum.

We first evaluate the two arc-elimination methods w.r.t. the percentage of arcs that can be eliminated using different algorithmic setups. Second, we briefly compare the computational effort. Third, we analyze the acceleration of the entire branch-and-price-and-cut algorithm caused by arc elimination. All algorithms were coded in C++, compiled in release mode with MS-Visual C++ version 6.0; all runs were performed on a standard PC (Intel x86 family 15 model 2) with 2.8 GHz, 1GB main memory, on MS-Windows 2000.

4.1 Percentage of Eliminated Arcs

The number of different possible setups for the branch-and-price-and-cut algorithm is huge. The most important parameters are summarized in Table 1: k-cycle elim-

Parameter	Values used here	Description
Method	W, B	W=Walker's method (Algorithm 2/3), B=Bidirectional method (Algorithm 4)
k	2, 3 and 4	k-cycle elimination
cuts	1-pc, 2-pc	use of 1-path cuts alone/with 2-path cuts
UB	$opt + x\%$ with $x =$ $0\%, 0.1\%, 0.5\%$ and 1%	quality of the upper bound supplied to W and B

Table 1
Parameters Controlling the Branch-and-Price-and-Cut Algorithm and Arc Elimination

ination leads to tighter relaxations of the master problem if k is increased. With increasing $k \in \{2, 3, 4\}$, the effort of solving the subproblem grows, the integrality gap decreases, and one can expect smaller branch-and-bound trees. The use of 1-path cuts (i.e., subtour-elimination constraints) is standard, since these cuts are efficiently separable. Additional 2-path cuts require more sophisticated separations procedures, in which the solution of TSPTW is an algorithmic component. They often help to substantially decrease the remaining integrality gap, so that one can also expect a reduction of the tree size. Finally, the quality of the upper bound UB provided to both methods, W and B, directly determines how many arcs can be eliminated. In order to be able to use computed reduced costs r_{ij} and $\bar{r}_{ij}$ not only at the moment when they are computed, but also when new improved upper bounds UB become available, we store the following information: For all arcs $(i, j) \in \mathcal{A}$, the objective of the dual feasible solution $\pi b + \mu$ plus the reduced cost r_{ij} or $\bar{r}_{ij}$ is stored as lower bounds $lb_{ij} = \pi b + \mu + r_{ij}$ or $\overline{lb}_{ij} = \pi b + \mu + \bar{r}_{ij}$. These lower bounds can be compared with any upper bound UB and allow the elimination of all arcs with $lb_{ij} > UB$ for method W and $\overline{lb}_{ij} > UB$ for method D. The following analysis supplies different upper bounds of $UB = (1 + x) \cdot opt$ to both methods, where opt is the cost of an optimal solution and x the deviation from it. Values for x between 0% and 1%, as given in Table 1, seem realistic, since good (meta)heuristics often produce high-quality solutions very close to the optimum.

For each VRPTW instance, the variation of the parameters given in Table 1 leaves us with 48 different setups to analyze. In order to keep the computational experiments concise and the amount of data to be displayed small, we decided to first choose four instances which reflect a 'typical behavior' of a group of instances. Later, we summarize results from a larger data set. A meaningful way to visualize the 48 data points is to compare the percentage of eliminated arcs with the

following gap

$$gap = \frac{UB - lb_{[1,2]}(k)}{opt} = 1 + x - \frac{lb_{[1,2]}(k)}{opt} \ . \tag{11}$$

Herein, $lb(k)$, $lb_1(k)$ and $lb_2(k)$ are the lower bounds $\pi b + \mu$ provided by the LP-relaxation of (DWM) using k-cycle elimination and no cuts, 1-path cuts, and 1-path cuts and 2-path cuts, respectively.

Figure 1 depicts the relationship between *gap* (based on $lb_1(k)$ or $lb_2(k)$ depending on 1-cp/2-cp) and the percentage of arcs eliminated for four representative instances. The obvious result of all computations is that method B can always eliminate a significantly larger portion of the arcs than method W can. This *qualitative* result is *not* surprising, since we have proven in Section 3.3 that $r_{ij} \leq \bar{r}_{ij}$ holds, and, therefore, that every arc eliminated by method W can also be eliminated by method D. The empirical and *quantitative* result is interesting: B eliminates up to 20% more arcs than W (100% is the number of arc after resource window reduction (see Desrosiers *et al.*, 1995)).

The first instance R103.100, depicted in Figure 1, is an example of an instance where (DWM) produces tight lower bounds for *opt* independent of the k-cycle elimination and cutting-plane approach chosen. For all setups, the integrality gap $(opt - lb_{[1,2]}(k))/opt$ is about 0.2%. This means that this particular instance does not tend to produce bad fractional solutions with many cycles and insufficient covered subsets of customers. Here, the number of eliminated arcs primarily depends on the quality of upper bounds. W and D differ (for otherwise identical parameters), by about 10%.

Instance RC107.100 shows that the parameters k and 1-cp/2-cp can also have an impact on the *gap*. One can clearly see that the most important determining factor for the percentage of eliminated arcs is *gap*, likewise for both methods W and B. There is a nearly linear dependency between *gap* and the portion of arcs eliminated. The presence of additional 2-path cuts clearly decreases the gap and (ceteris paribus) leads to the elimination of more arcs. However, 2-path cuts seemingly make both methods behave less effectively when gap and eliminated arcs are compared (a parallel shift of the points to the left/bottom). We interpret this behavior as follows: The additional 2-path cuts mean more constraints in (DWM) and, hence, more path variables λ_P in basis. The result is more arcs with reduced cost 0.

For the instance R112.50, the integrality gap is between 2.2% and 3.7% for different choices of k and 1-pc/2-pc. For the bidirectional method, the dependency between *gap* and the percentage of eliminated arcs is nearly linear (linear regression yields $\%\ arcs\ eliminated = 89,1\% - 8.59 \cdot gap$ with error $R^2 = 0.97$). The method W does not show such a direct dependency. Instead, for different values of k, the dependency seem to be linear, but different 'stripes' result from variations of k. The interesting observation is here that increasing values of k, on the one hand,

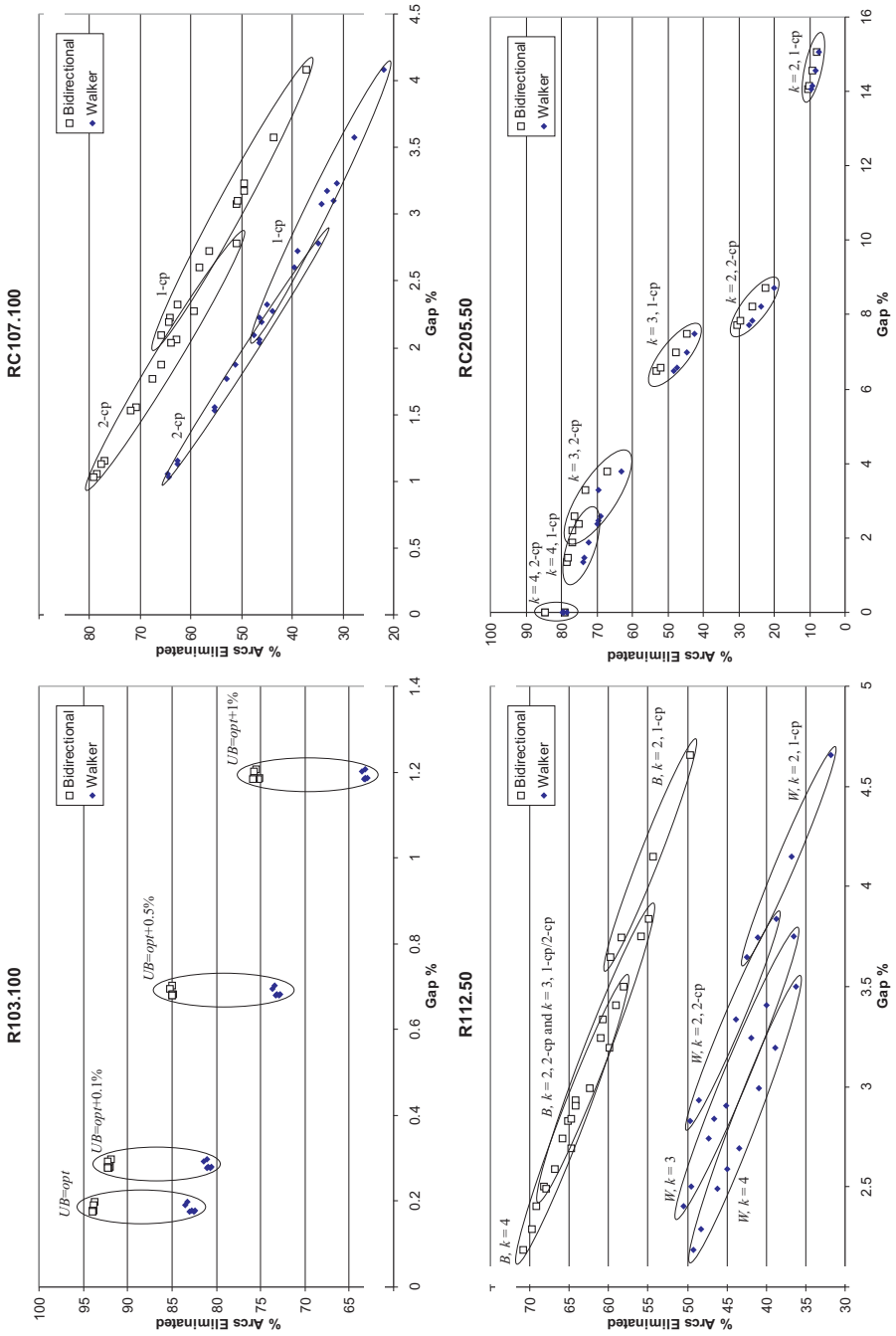

Fig. 1. Percentage of Eliminated Arcs relative to Gap *gap*; Note the different Scales of the Axes and the different Groupings

improve the lower bound of (DWM), and on the other hand, lead to a larger number of labels generated in SPPRC-k-cyc, which, in turn, makes method W work less effectively. We interpret this result as follows: Both effects are concurring and, therefore, runs of the branch-and-price-and-cut algorithm, just differing in parameter k, roughly eliminate the same number of arcs.

The instance RC205.50 shows a different behavior. The integrality gap substantially

depends on the chosen relaxation, i.e., on k and 1-cp/2-cp, and varies from 0% for $k = 4$ and 2-cp to about 14% for $k = 2$ and 1-pc. Hence, the ability of the two arc-elimination methods mainly depends on the *gap*. The difference between W and D (up to about 5%) is smaller than has been observed for the other instances. The smaller the *gap*, the larger the difference between W and D.

Figure 2 depicts the difference between the lower bounds lb_{ij} and $\overline{lb}_{ij}$ for the instance R205.100. The pricing network $\mathcal{N} = (V, \mathcal{A})$ consists of 7,327 arcs (=100%),

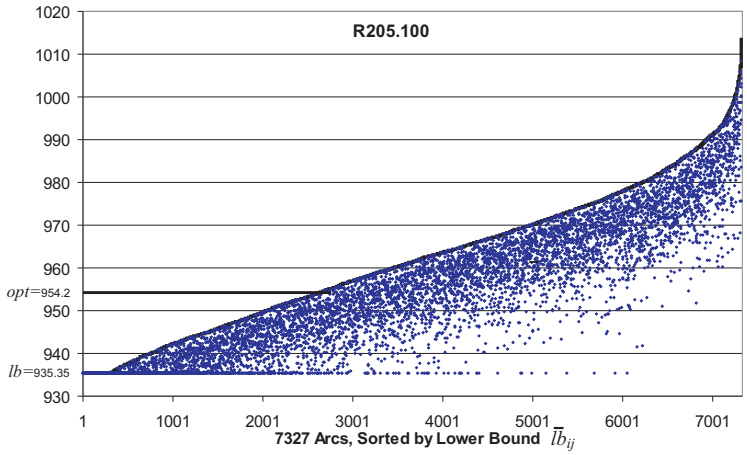

Fig. 2. Values lb_{ij} and $\overline{lb}_{ij}$ computed by Walker's Method and Bidirectional Method; Line=Values $\overline{lb}_{ij}$; Dots=Values lb_{ij} for the Same Arc (i, j)

which could not be eliminated by resource windows strengthening techniques during the pre-processing phase. The LP-relaxation of the master program is solved with 4-cycle elimination and 1-path cuts. At the root node of the branch-and-bound tree (when all 1-path cuts are added), the 7,327 arcs $(i, j) \in \mathcal{A}$ are sorted according to the values $\overline{lb}_{ij}$. In this ordering, the two series of lower bounds, i.e., lb_{ij} and $\overline{lb}_{ij}$, are shown in Figure 2. The bottom level at $lb = 935.35$ corresponds to all arcs with reduced cost 0. These arcs get a lower bound lb_{ij} identical to the objective of (DWM). Walker's method yields 1,218 arcs (= 16.6%) with $r_{ij} = 0$ but only 303 arcs (=4.1%) have $\bar{r}_{ij} = 0$ in method B. With the upper bound $UB = opt$, method W eliminates 3,518 arcs (=48%) while method B eliminates 4,696 arcs (=64%).

4.2 Computational Effort

The computational effort of arc elimination by Walker's method and by the bidirectional method is compared next. Both methods use dual feasible solutions (π, μ) of (DWM), which are available and globally valid whenever a root node of the branch-and-bound tree is solved to optimality. By the addition of cutting planes, different (improved) root nodes are solved such that methods W and B are, in general, invoked more than once. (In principle, it would be possible to use both methods in the entire branch-and-bound tree, but then lower bounds lb_{ij} and $\overline{lb}_{ij}$ for each arc would have to be stored at each node of the tree.) The input data for method W is a monodirectional solution to the SPPRC-k-cyc subproblem applied to PP(π, μ), which is available without any additional effort at the end of the

column-generation process at each node. Contrary, method B also needs a solution to the backward SPPRC-k-cyc, which must be computed on top of what is freely available. Hence, we can expect that method B is computationally more costly than method W. Both Algorithms 2/3 and 4 then roughly do the same: They loop over all arcs $(i, j) \in \mathcal{A}$ and compare (extended) labels of node i with other labels of node j.

For the selection of a reasonable set of test instances, we have chosen the following rules. The best combination of the parameters k for k-cycle elimination and 1-cp/2-cp are taken from (Irnich and Villeneuve, 2006). Moreover, we only consider an instance if its branch-and-bound tree contains more than 10 nodes and the overall computation time does not exceed 3,600 seconds. Note that the setup in (Irnich and Villeneuve, 2006) was slightly different, since additional techniques for the acceleration of the pricing were used. Using these rules, 33 instances result. They are given in the first column of Table 2 and will be used in this and in the next subsection.

Besides the detailed results of Table 2, we have the following overall results: The computation times for method B are, by factors of between 1.7 and 7.5 (with an average of 3.5), longer than those for method W. For method B, the portion of its time spent on solving the inverse SPPRC widely varies from about 9% to 85%, with an average of 32%. A *single* call of method B takes between less than 0.1% and up to about 12% (avg. 2.8%) of the time needed to solve the root nodes.

The results show that the computing power necessary to use reduced cost information of paths from the master to eliminate arcs is not negligible, but makes up only a fraction of the overall computing time. Only if the speedup for the entire branch-and-bound tree overcompensates the additional effort, do arc-elimination methods pay off. This issue will be analyzed next.

4.3 Impact on the Branch-and-Bound Tree—Overall Acceleration

The preceding sections have shown that the bidirectional method outperforms the method of Walker w.r.t. the number of eliminated arcs, while its computational effort seems still comparable to that of Walker's method. The relevance of the additional computational effort vanishes when hard-to-solve instances with a large branch-and-bound tree are considered. Therefore, we analyze solely the bidirectional method and its impact on the computation times of the branch-and-price algorithm.

Table 2 summarizes the computational analysis and contains the following information. The name of the instance and the best parameters k and 1-cp/2-cp are given in the first two columns (.25, .50, and .100 refers to the number of customers in the respective instance). The column *gap* refers to the gap as defined by (11) with $UB = opt$ and with $lb(k)$ ($lb_f(k)$) the lower bound computed by (DWM) before (after) adding cutting planes. The number of arcs before and after multiple calls of the bidirectional method is given in the fourth column. In order to make the standard branch-and-price algorithm (std) comparable with the one using

method B, we first compute the branch-and-bound tree with (std) and record the branching decisions. Subsequently, the branch-and-price algorithm with method B is run, using the recorded branching decisions. This guarantees that the tree has the same structure (otherwise, due to degeneracy, trees could become different). However, method B can lead to tighter bounds in the tree so that some nodes can be pruned in addition to those pruned by (std) ([+std] is the number of these nodes given in column *Tree*). The next two columns compare the times of (std) and branch-and-price with method B which are needed to solve the *root nodes* of the branch-and-bound tree, i.e., to finish the first node with lower bound $lb(k)$ and the last root node with lower bound $lb_1(k)$ or $lb_2(k)$. For the first root node, the computation times differ exactly in the time method B needs to compute the reduced costs of all arcs (the absolute difference is shown as +B and the relative difference as %). After the elimination of some arcs, cutting planes are iteratively added to (DWM) which is then re-optimized and method B is invoked again. On the one hand, these additional runs of method B increase the computation time for the last root node. On the other hand, method B iteratively reduces the size of the pricing network, which can lead to a speedup of the re-optimization. Hence, the values ±B and % for the seventh column, referring to the last root node, can be positive or negative. Similarly, the remaining columns show the overall time (*Time all*) and the time spent in the branch-and-bound tree (*Time tree*) for solving non-root nodes. The latter time does not include the time spent on solving the root nodes. This information is significant, since it shows the impact of the full arc-elimination method B on the computing times. All values −B and % are negative because solving master programs (with additional branching constraints) always takes less time when the pricing networks $\mathcal{N}$ are smaller. The *speedup* factors in the ninth and last column are the quotients of the times for (std) and branch-and-price with method B.

The results can be summarized as follows: The gap after adding cutting planes is between 0.05% and 8.9%, with an average of 1.3%. Branch-and-bound tree sizes vary from 11 to 370 nodes (multiple re-optimizations after adding 1-path or 2-path cuts are counted as additional (root) nodes of the tree). Conform with the results reported in Section 4.1, the percentage of arcs that method B can eliminate mainly depends on the *gap*. As a rule of thumb we state: With a gap of 1% one can eliminate approximately 80% of the arcs.

The additional effort of method B relative to the time necessary to solve the first root node (column *Time root 1*) is relevant and can take up to about 50% (avg. 28%) for 25 customer instances, up to 34% (avg. 17%) for 50 customers, and up to 6% (avg. 4%) for 100 customers. The larger the instances, the smaller the additional effort of a single call of method B. The additional effort of method B for solving the root nodes *with* cutting planes (column *Time root 2*) can increase to 85% of the time that (std) consumes. However, for some instances, the effort of applying Algorithm 4 is overcompensated by the faster pricing, so that the difference of the computing times can decrease. For three instances, C207.25, RC202.50, and RC105.100, method B already pays off only for solving the root nodes.

The most important results are related to the speedup gained by method B on the overall solution process and the speedup within the branch-and-bound tree. There is only one instance, RC101.100, out of the 33 instances, for which method B does not accelerate the overall solution procedure (it takes 20% longer; 22 cuts are added to 13 root nodes, so that method B is invoked 13 times). However, the non-root nodes benefit from the elimination of arcs by an acceleration of factor 1.9. All other instances are solved faster when method B is integrated. The speedups vary from factors of between 1.05 and 18.9 (avg. 2.8). If the time for solving the root nodes is excluded, the acceleration factors in the tree are significantly higher and vary between 1.3 and 29.2 (avg. 5.1). The largest speedups can be observed for instances of the second series (C2, R2, RC2) of the Solomon (1987) benchmark problems, since these tend to have longer routes and harder-to-solve subproblems. Finally, we could not find a statistical correlation between the *gap* and the acceleration in percent or the speedup factors (linear regression gives $R^2 \approx 0.07$ and $R^2 \approx 0.02$, respectively). It remains unclear to us which properties of an instance and parameters of the solution method determine the overall speedup.

5 Conclusions

The paper has provided insights into the relationship between reduced costs of paths in extensive column-generation formulations and reduced costs of arcs in original compact formulations. Both types of reduced costs can be used for the elimination of arcs, and two practical methods are available: The adaption of a first technique, originally proposed by Walker (1969), allows the computing of reduced costs of original variables from an extensive column-generation model when dual feasible solutions to the master and the subproblem are known. This method is however restricted to subproblems which can be formulated as pure linear programs. The other technique is newly proposed in this paper and is based on solving the s-t SPPRC subproblem twice, i.e., with bidirectional methods as an s-to-all forward and as an t-to-all backward SPPRC. The paper has theoretically proven that the bidirectional method is superior in the sense that it always provides bounds for arc elimination that are at least as good as those computed by Walker's method.

Moreover, both methods were empirically tested on standard benchmark problems for the VRPTW. On the one hand, the computational effort of the bidirectional method is slightly higher than the effort needed for Walker's method. On the other hand, the bidirectional method can consistently remove more arcs (up to 20% more) and often leads to SPPRC subproblems from which 80%-90% of the arcs are eliminated: Roughly, with a gap of 1% one can eliminate 80% of the arcs. Empirically tested on 33 VRPTW instances, this caused a significant overall speedup with factors of between 1.3 and 29.2, with an average factor of 5.1.

A promising trend in column generation approaches for VRPs is that of solving subproblems with (good) heuristics and exactly solving the hard (E)SPPRC only a very few times, hopefully only to show optimality (see, e.g., Xu *et al.*, 2003; Jepsen *et al.*, 2006; Desaulniers *et al.*, 2006). The extensive use of heuristics is clearly

another way of speeding up pricing, and we expect that the combination of arc-elimination techniques with heuristics will still improve these highly sophisticated implementations, even if the speedups are probably smaller.

The proposed arc-elimination algorithm can be seen as a cooperative scheme, in which exact and heuristic algorithms can *both* benefit. Thus far, exact algorithms primarily benefit from good integer solutions for bounding. One of the few papers discussing cooperative approaches in exact vehicle-routing is that by Danna and Le Pape (2005). In the future, heuristics may provide (at an early point in time) good upper bounds, which are useful in the exact method for eliminating arcs and accelerating the exact approach. In turn, exact algorithms can provide sparser underlying networks for the heuristics, still guaranteeing that the heuristics can find an optimal solution, but faster, because of the smaller underlying networks. The latter approach is an exact intensification method.

Finally, the adaptation of the proposed variable-elimination techniques to other column-generation formulations is another interesting path of future research: Then, subproblems of different combinatorial structures such as, e.g., trees, selections, packings etc. have to be considered. The goal here would again be the devising of efficient variable-elimination techniques that lead to smaller instances of the pricing subproblems and to speedups for the overall column-generation approach.

References

Ahuja, R., Magnanti, T., and Orlin, J. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, New Jersey.

Barnhart, C. and Schneur, R. (1996). Air network design for express shipment service. *Operations Research*, **44**(6), 852–863.

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

Barnhart, C., Hane, C., and Vance, P. (2000). Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems. *Operations Research*, **48**(2), 318–326.

Boland, N., Dethridge, J., and Dumitrescu, I. (2006). Accelerated label setting algorithms for the elementary resource constrained shortest path problem. *Operations Research Letters*, **34**(1), 58–68.

Danna, E. and Le Pape, C. (2005). Branch-and-price heuristics: A case study on the vehicle routing problem with time windows. In Desaulniers *et al.* (2005), chapter 4, pages 99–129.

Dell'Amico, M., Righini, G., and Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, **40**(2), 235–247.

Desaulniers, G., Desrosiers, J., Dumas, Y., Marc, S., Rioux, B., Solomon, M. M., and Soumis, F. (1997). Crew pairing at Air France. *European Journal of Operational Research*, **97**, 245–259.

Desaulniers, G., Desrosiers, J., Gamache, M., and Soumis, F. (1998a). Crew scheduling in air transportation. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, chapter 1, pages 169–186. Kluwer Academic Publishers, Norwell, Massachusetts and Dordrecht, The Netherlands.

Desaulniers, G., Desrosiers, J., Ioachim, I., Solomon, M., Soumis, F., and Villeneuve, D. (1998b). A unified framework for deterministic time constrained vehicle routing and crew scheduling problems. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, chapter 3, pages 57–93. Kluwer Academic Publisher, Boston, Dordrecht, London.

Desaulniers, G., Desrosiers, J., and Solomon, M., editors (2005). *Column Generation*. Number 5 in GERAD 25th Anniversary Series. Springer.

Desaulniers, G., Lessard, F., and Hadjar, A. (2006). Tabu search, generalized k-path inequalities, and partial elementarity for the vehicle routing problem with time windows. Les Cahiers du GERAD G-2006-45, GERAD, École des Hautes Études Commerciales, Montréal, Canada.

Desrochers, M. and Soumis, F. (1988). A generalized permanent labelling algorithm for the shortest path problem with time windows. *Information Systems and Operations Research*, **26**(3), 191–212.

Desrochers, M., Desrosiers, J., and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, **40**(2), 342–354.

Desrosiers, J. and Lübbecke, M. (2005). A primer in column generation. In Desaulniers *et al.* (2005), chapter 1, pages 1–32.

Desrosiers, J., Dumas, Y., Solomon, M., and Soumis, F. (1995). Time constrained routing and scheduling. In M. Ball, T. Magnanti, C. Monma, and G. Nemhauser, editors, *Network Routing*, volume 8 of *Handbooks in Operations Research and Management Science*, chapter 2, pages 35–139. Elsevier, Amsterdam.

Dror, M. (1994). Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, **42**(5), 977–978.

Feillet, D., Dejax, P., Gendreau, M., and Guéguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, **44**(3), 216–229.

Gamache, M. and Soumis, F. (1998). A method for optimally solving the rostering problem. In G. Yu, editor, *Operations Research in the Airline Industy*, chapter 5, pages 124–157. Kluwer Academic Publishers, Boston, Dordrecht, London.

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

Hadjar, A., Marcotte, O., and Soumis, F. (2001). A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. Technical Report G-2001-25, GERAD, Université de Montréal, Quebec, Canada.

Irnich, S. and Desaulniers, G. (2005). Shortest path problems with resource constraints. In Desaulniers *et al.* (2005), chapter 2, pages 33–65.

Irnich, S. and Villeneuve, D. (2006). The shortest path problem with resource constraints and k-cycle elimination for $k \geq 3$. *INFORMS Journal on Computing*, **18**(3), 391–406.

Irnich, S. (2006). Resource extension functions: Properties, inversion, and generalization to segments. Technical Report 2006-01, Deutsche Post Endowed Chair of Optimization of Distribution Networks, RWTH Aachen University, Aachen, Germany. Available at `www.dpor.rwth-aachen.de`.

Jepsen, M., Spoorendonk, S., Petersen, B., and Pisinger, D. (2006). A non-robust branch-and-cut-and-price algorithm for the vehicle routing problem with time windows. DIKU Technical Report no. 06/03, Dept. of Computer Science, University of Copenhagen, Copenhagen, Denmark.

Kallehauge, B., Larsen, J., Madsen, O., and Solomon, M. (2005). Vehicle routing problem with time windows. In Desaulniers *et al.* (2005), chapter 3, pages 67–98.

Kohl, N., Desrosiers, J., Madsen, O., Solomon, M., and Soumis, F. (1999). 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, **33**(1), 101–116.

Kohl, N. (1995). *Exact methods for Time Constrained Routing and Related Scheduling Problems*. Dissertation, Department of Mathematical Modelling, Technical University of Denmark.

Lübbecke, M. and Desrosiers, J. (2006). Selected topics in column generation. *Operations Research*, **53**(6), 1007–1023.

Poggi de Aragão, M. and Uchoa, E. (2003). Integer program reformulation for robust branch-abd-price algorithms. Technical report, Departamento de Informatica, PUC-Rio.

Ribeiro, C. and Soumis, F. (1994). A column generation approach to the multiple-depot vehicle scheduling problem. *Operations Research*, **42**(1), 41–52.

Righini, G. and Salani, M. (2004). Bounded bidirectional dynamic programming for the elementary shortest path problem with resource constraints. Technical report, Dipartimento di Tecnologie dell'Informazione, Università degli Studi di Milano, Crema, Italy.

Salani, M. (2005). *Branch-and-price algorithms for Vehicle Routing Problems*. Dissertation, Università degli Studi di Milano, Facoltà die Scienze Matematiche, Fisiche e Naturali, Dipartimento di Tecnologie dell'Informatione, Milan, Italy.

Solomon, M. (1987). Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, **35**(2), 254–265.

Vance, P., Atamtürk, A., Barnhart, C., Gelman, E., Johnson, E., Krishna, A., Mahidhara, D., Nemhauser, G., and Rebello, R. (1997). A heuristic branch-and-price approach for the airline crew pairing problem. Technical report, Department of Industrial and Systems Engeneering, Auburn Univerity.

Villeneuve, D. and Desaulniers, G. (2005). The shortest path problem with forbidden paths. *European Journal of Operational Research*, **165**(1), 97–107.

Walker, W. (1969). A method for obtaining the optimal dual solution to a linear program unsing the Dantzig-Wolfe decomposition. *Operations Research*, **17**, 368–370.

Xu, H., Chen, Z., Rajagopal, S., and Arunapuram, S. (2003). Solving a practical pickup and delivery problem. *Transportation Science*, **37**(3), 347–364.

Appendix

A Results on the Dual of Some Extended and Reformulated Models

We consider a standard linear program (P) and an equivalent reformulation (P'_{ext}) of (P), where additional variables w are uniquely determined by the variables v of (P) and no other constraints on the new variables exist:

$$(P) \qquad \begin{aligned} \min \quad & g^\top v \\ \text{s.t.} \quad & Mv \geq m \quad (\alpha) \\ & v \geq \mathbf{0} \end{aligned} \qquad\qquad (P'_{ext}) \qquad \begin{aligned} \min \quad & g^\top v + \mathbf{0}^\top w \\ \text{s.t.} \quad & Mv + \mathbf{0}w \geq m \quad (\beta) \\ & Nv - w = \mathbf{0} \quad (\gamma) \\ & v \geq \mathbf{0}, w \in \mathbb{R}^q \end{aligned}$$

Proposition 1 Models (P) and (P'_{ext}) are equivalent in the sense of the following points (i) and (ii).

(i) Every primal feasible (optimal) solution v to (P) implies a primal feasible (optimal) solution $(v, w) = (v, Nv)$ to (P'_{ext}), and vice versa. [Feasible corresponds to feasible, optimal to optimal solution.]

(ii) Every dual feasible (optimal) solution α to (P) implies a dual feasible (optimal) solution $(\alpha, \mathbf{0})$ to (P'_{ext}), and vice versa. Every dual feasible solution to (P'_{ext}) has $\gamma = \mathbf{0}$.

(iii) For every dual feasible solution (β, γ) to (P'_{ext}), the reduced cost of the variables w are $\mathbf{0}$.

This Proposition is only partly intuitive when one interprets it in the following way: The equality $w = Nv$ is easy to fulfill since unbounded and unconstrained variables w can simply be set equal to Nv. Hence, the dual price of the constraints $Nv - w = \mathbf{0}$ is zero, since it can be fulfilled without imposing additional costs. However, the fact that the reduced costs of all variables w are zero is somehow surprising. Marginally increasing one of the variables w_e directly implies that some of the v_f (for $n_{ef} \neq 0$) have to be adapted also. The reduced cost of w_e should, therefore, be expressible in the reduced costs of the variables v_f, which is obviously here not the case.

From now on we are only interested in the special case that $N \in \mathbb{Z}_+^{q \times p}$ holds, i.e., every variable w_e is a (non-negative) sum of some variables v_f. The assumption that all entries of N are non-negative implies that $w = Nv \geq \mathbf{0}$ holds. Let (P_{ext}) denote the model (P'$_{ext}$) together with the constraint $w \geq \mathbf{0}$ (and with the assumption $N \in \mathbb{Z}_+^{q \times p}$). The addition of the non-negativity constraints $w \geq \mathbf{0}$ is nothing but a formal device; it enables us to derive more interesting results than those stated for (P'_{ext}):

Proposition 2 Models (P) and (P_{ext}) are equivalent in the sense of the following points (i) and (ii). Statements about the relationship of dual solutions and reduced

costs are given by (iii)-(v):

(i) Every primal feasible (optimal) solution v to (P) implies a primal feasible (optimal) solution $(v, w) = (v, Nv)$ to (P$_{ext}$), and vice versa.

(ii) Every dual feasible (optimal) solution α to (P) implies a dual feasible (optimal) solution $(\alpha, \mathbf{0})$ to (P$_{ext}$), and vice versa.

(iii) Every dual feasible (optimal) solution α to (P) implies a set

$$\Delta(\alpha) = \left\{(\alpha, \gamma) : \gamma \geq \mathbf{0}, N\gamma \leq g^\top - \alpha M\right\}$$

of dual feasible (optimal) solutions to (P$_{ext}$). Moreover, any dual feasible (optimal solution (β, γ) to (P$_{ext}$) fulfills $(\beta, \gamma) \in \Delta(\beta)$.

(iv) Let a dual feasible (optimal) solution α to (P) (or, equivalently, $(\alpha, \mathbf{0})$ to (P$_{ext}$)) be given.

For any index $e^* \in \{1, \ldots, q\}$, a vector $\gamma = (\gamma_e)$ with

$$0 \leq \gamma_{e^*} \leq \min_{f \in \{1,\ldots,p\}: n_{e^*,f} \neq 0} \frac{g_f - (\alpha M)_f}{n_{e^*,f}} \leq \min_{f \in \{1,\ldots,p\}: n_{e^*,f} \neq 0} g_f - (\alpha M)_f \quad \text{(A.1)}$$

and $\gamma_e = 0$ for all $e \neq e^*$ fulfills $(\alpha, \gamma) \in \Delta(\alpha)$, i.e., implies another dual feasible (optimal) solution to (P$_{ext}$). For 0-1-matrices N, the second inequality holds as an equality.

Conversely, for any dual feasible (optimal) solution (β, γ) to (P$_{ext}$), the components of γ are all (individually) constrained by inequality (A.1).

For the original dual feasible (optimal) solution $(\alpha, \mathbf{0})$ to (P$_{ext}$), the rightmost term in (A.1) is the minimum reduced cost of all variables v_f involved in the equality for w_{e^*}.

(v) For every (β, γ) feasible dual solution to (P$_{ext}$), the value γ_e is the reduced cost of the variable w_e.

Next, we consider three different reformulations of (P$_{ext}$), where the cost vector g can be expressed as $g^\top = h^\top N$ and the coefficient matrix M can be expressed as $M = QN$. Because of $w = Nv$, it means that one can either express the objective by $g^\top v$ or $h^\top w$ and, similarly, the constraints either by $Mv \geq m$ or $Qw \geq m$.

$$\begin{array}{lrlrlrl}
 & \min & \boxed{\mathbf{0}^\top v + h^\top w} & \min & g^\top v + \mathbf{0}^\top w & \min & \boxed{\mathbf{0}^\top v + h^\top w} & \\
(P_{ext}^{1,2,3}) & \text{s.t.} & Mv + \mathbf{0}w \geq m & \text{s.t.} & \boxed{\mathbf{0}v + Qw} \geq m & \text{s.t.} & \boxed{\mathbf{0}v + Qw} \geq m & (\beta^{1,2,3}) \\
 & & Nv - w = \mathbf{0} & & Nv - w = \mathbf{0} & & Nv - w = \mathbf{0} & (\gamma^{1,2,3}) \\
 & & v \geq \mathbf{0}, w \geq \mathbf{0} & & v \geq \mathbf{0}, w \geq \mathbf{0} & & v \geq \mathbf{0}, w \geq \mathbf{0} &
\end{array}$$

Proposition 3 Models (P$_{ext}$) and (P$^k_{ext}$), $k = 1, 2, 3$ are equivalent in sense of the following point (i). The relationship between the dual solutions and reduced costs of the four models is given by (ii) and (iii):

(i) Every primal feasible (optimal) solution (v, w) to (P$_{ext}$) is a primal feasible (optimal) solution to (P$^k_{ext}$), $k = 1, 2, 3$ and vice versa.

(ii) Every dual feasible (optimal) solution (β, γ) to (P_{ext}) implies a dual feasible (optimal) solution (β^k, γ^k) to (P^k_{ext}), $k = 1, 2, 3$ and vice versa. The relationship between the dual solutions is

$$\begin{pmatrix}\beta^1\\ \gamma^1\end{pmatrix} = \begin{pmatrix}\beta\\ \gamma - h^\top\end{pmatrix} \qquad \begin{pmatrix}\beta^2\\ \gamma^2\end{pmatrix} = \begin{pmatrix}\beta\\ \gamma + \beta Q\end{pmatrix} \qquad \begin{pmatrix}\beta^3\\ \gamma^3\end{pmatrix} = \begin{pmatrix}\beta\\ \gamma + \beta Q - h^\top\end{pmatrix}.$$

(iii) Every dual feasible solution to (P_{ext}) or (P^k_{ext}), $k = 1, 2, 3$ as given in (ii) imposes the same reduced cost γ of the variables w.

Proofs of all three propositions are straightforward based on elementary linear programming theory.

Table 2
Impact of Bidirectional Method on Branch-and-Bound Tree

Instance	Param	*gap* w.r.t.	Arcs	Tree	Time root 1	Time root 2	Time all	Speed-	Time tree	Speed-
	$k=$, f-pc	$(lb(k)/lb_f(k))$	(std/B/% elim)	(B[+std])	(std/+B/%) [s]	(std/±B/%) [s]	(std/±B/%) [s]	up all	(std/−B/%) [s]	up tree
R110.25	2, 1-cp	1.5/1.5	526/136/74%	21	0.3/0.02/6%	0.3/0.02/6%	1.9/-0.6/-34%	1.5	1.6/-0.6/-40%	1.7
R112.25	2, 2-cp	2.2/1.9	646/157/76%	15	0.8/0.2/23%	3.2/0.2/6%	5.1/-0.9/-18%	1.2	1.9/-1.1/-57%	2.3
C204.25	2, 1-cp	1.0/1.0	622/113/82%	11	3.4/1.3/40%	5.4/0.2/4%	52.1/-37.1/-71%	3.5	46.8/-37.3/-80%	4.9
C207.25	3, 1-cp	1.4/0.05	463/71/85%	11[+1]	0.8/0.3/34%	2.8/-0.3/-11%	6.5/-3.7/-57%	2.3	3.7/-3.4/-91%	10.8
R203.25	3, 1-cp	1.1/1.1	596/126/79%	7	0.8/0.1/16%	0.8/0.1/16%	3.2/-1.6/-52%	2.1	2.4/-1.8/-73%	3.8
R204.25	4, 1-cp	1.7/1.7	627/141/78%	23[+8]	4.7/2.3/49%	4.7/2.3/49%	369.2/-349.7/-95%	18.9	364.5/-352.0/-97%	29.2
R207.25	3, 1-cp	1.2/0.8	609/119/80%	25	1.0/0.2/22%	1.7/0.06/3%	18.6/-12.9/-69%	3.3	16.9/-12.9/-76%	4.3
R208.25	3, 1-cp	1.5/1.5	631/139/78%	16	2.9/1.1/38%	4.1/0.5/13%	53.5/-35.8/-67%	3.0	49.4/-36.3/-73%	3.8
R211.25	4, 1-cp	3.1/3.1	647/244/62%	137[+2]	5.3/0.6/11%	5.3/0.6/11%	1038.4/-813.8/-78%	4.6	1033.1/-814.4/-79%	4.7
RC203.25	4, 1-cp	13.4/8.9	596/422/29%	370[+4]	4.6/2.1/46%	9.0/7.7/85%	3884.2/-1825.9/-47%	1.9	3875.2/-1833.7/-47%	1.9
RC207.25	4, 1-cp	11.3/6	561/300/47%	126[+8]	3.2/1.1/35%	7.1/0.4/6%	771.6/-352.8/-46%	1.8	764.4/-353.2/-46%	1.9
R103.50	3, 2-cp	0.6/0.5	1969/233/88%	17	3.8/0.6/15%	11.2/0.7/6%	17.5/-3.3/-19%	1.2	6.3/-4.0/-64%	2.8
R104.50	3, 2-cp	1.1/0.6	2397/295/88%	24[+2]	12.8/3.8/29%	32.6/0.0/0%	234.9/-189.9/-81%	5.2	202.3/-189.9/-94%	16.3
R107.50	3, 2-cp	0.9/0.7	2091/285/86%	22	6.9/0.9/13%	11.5/0.7/6%	34.5/-17/-49%	2.0	23.0/-17.7/-77%	4.3
R109.50	3, 2-cp	1.5/1.3	1533/385/75%	137[+2]	4.4/0.2/4%	11.8/0.0/0%	91.6/-35.5/-39%	1.6	79.8/-35.4/-44%	1.8
R110.50	4, 2-cp	0.4/0.1	2018/163/92%	11	6.3/0.9/14%	15.0/0.1/1%	19.6/-3.8/-19%	1.2	4.6/-3.8/-83%	5.9
R111.50	3, 2-cp	2.0/1.7	2034/574/72%	109[+2]	6.1/1.0/17%	22.5/1.1/5%	172.1/-78.1/-45%	1.8	149.6/-79.3/-53%	2.1
R112.50	3, 2-cp	2.7/2.4	2527/777/69%	976	10.8/2.4/22%	29.6/2.2/7%	6059.9/-3284.5/-54%	2.2	6030.3/-3286.7/-55%	2.2
RC102.50	3, 2-cp	12.4/1.1	1462/433/70%	139[+12]	5.4/0.4/7%	36.8/0.9/2%	137.5/-57/-41%	1.7	100.7/-57.9/-57%	2.4
RC106.50	4, 2-cp	8.1/0.4	1242/304/76%	16	3.1/1.1/34%	17.3/1.9/11%	24.4/-4.2/-17%	1.2	7.0/-6.1/-87%	8.0
RC107.50	4, 2-cp	6.3/0.5	1834/331/82%	19	26.5/7.2/27%	72.8/2.8/4%	257.5/-165.9/-64%	2.8	184.8/-168.6/-91%	11.5
R203.50	3, 1-cp	1.1/1.1	2295/406/82%	11	23.0/4.6/20%	23.0/4.6/20%	90.4/-48.2/-53%	2.1	67.5/-52.7/-78%	4.6
R205.50	4, 1-cp	1.1/1.1	1878/370/80%	141	13.1/0.4/3%	13.1/0.4/3%	991.9/-721.1/-73%	3.7	978.9/-721.5/-74%	3.8
RC202.50	4, 1-cp	3.9/1.5	1973/424/79%	23[+6]	16.9/2.5/15%	32.8/-4.0/-12%	375.5/-301.4/-80%	5.1	342.7/-297.3/-87%	7.6
RC206.50	4, 1-cp	2.5/2.1	1860/510/73%	62	16.8/1.6/9%	20.9/0.5/3%	533.4/-377.2/-71%	3.4	512.5/-377.8/-74%	3.8
R101.100	2, 2-cp	0.4/0.2	3243/309/90%	10[+2]	29.0/1.1/4%	56.0/1.4/3%	65.6/-3/-5%	1.05	9.6/-4.5/-47%	1.9
R103.100	2, 2-cp	0.2/0.2	7704/473/94%	37	105.8/4.9/5%	253.3/5.7/2%	602.1/-289.3/-48%	1.9	348.9/-295.0/-85%	6.5
R105.100	2, 2-cp	0.7/0.4	4260/563/87%	47	64.2/3.0/5%	133.7/3.9/3%	239.1/-18.6/-8%	1.1	105.3/-22.6/-21%	1.3
R106.100	2, 2-cp	0.7/0.6	6558/996/85%	208	82.5/4.8/6%	149.2/4.7/3%	2220.1/-1174.7/-53%	2.1	2070.9/-1179.4/-57%	2.3
RC101.100	2, 2-cp	2.2/0.1	3641/343/91%	13[+1]	72.0/1.4/2%	261.8/65.0/25%	278.0/56.1/20%	0.8	16.2/-9.0/-55%	2.2
RC105.100	2, 2-cp	2.8/0.3	5249/502/90%	16[+9]	76.0/3.5/5%	521.3/-57.1/-11%	563.6/-90.1/-16%	1.2	42.3/-33.0/-78%	4.5
R201.100	4, 1-cp	0.3/0.3	5917/435/93%	35	105.1/2.3/2%	105.1/2.3/2%	1183.6/-845.4/-71%	3.5	1078.6/-847.6/-79%	4.7
RC201.100	4, 1-cp	0.5/0.5	5918/585/90%	101[+6]	114.1/5.4/5%	114.1/5.4/5%	3686.8/-2602.9/-71%	3.4	3572.7/-2608.4/-73%	3.7