

Reachability Cuts for the Vehicle Routing Problem with Time Windows

Jens Lysgaard

Department of Accounting, Finance and Logistics

Aarhus School of Business

Fuglesangs Allé 4

DK-8210 Aarhus V

Denmark

e-mail: lys@asb.dk

November 2004

Revised March 2005

Abstract

This paper introduces a class of cuts, called *reachability cuts*, for the Vehicle Routing Problem with Time Windows (VRPTW). Reachability cuts are closely related to cuts derived from precedence constraints in the Asymmetric Traveling Salesman Problem with Time Windows and to k -path cuts for the VRPTW. In particular, any reachability cut dominates one or more k -path cuts. The paper presents separation procedures for reachability cuts and reports computational experiments on well-known VRPTW instances. The computational results suggest that reachability cuts can be highly useful as cutting planes for certain VRPTW instances.

Key words: Routing, time windows, precedence constraints.

1 Introduction

The Vehicle Routing Problem with Time Windows (VRPTW) can be defined as follows. Let $G = (V, A)$ be a directed graph, with vertex set $V = \{0, \dots, n\}$ and arc set A . Vertex 0 represents a depot, whereas each of the vertices in $V_c = \{1, \dots, n\}$ represents a customer. Each customer i has a time window $[r_i; d_i]$ within which service at customer i must begin. A vehicle is permitted to arrive earlier than r_i to a customer i , in which case it will wait until service begins at time r_i . Late arrivals, i.e., after d_i , are not permitted. The depot has a time window $[r_0; d_0]$ within which each route must begin and end at the depot. With each arc $(i, j) \in A$ is associated a travel cost c_{ij} and a travel time t_{ij} , which includes any service time at vertex i . No assumptions are made with respect to the triangle inequality on travel times, but it is assumed that all travel times are nonnegative. Each customer i demands a quantity of $q_i > 0$ units of a common product to be delivered from the depot. A fleet of vehicles, each with a capacity Q is available for making the deliveries. No vehicle can serve a set of customers whose total demand exceeds Q . The objective is to find a collection of routes, of minimum total travel cost, each beginning and ending at the depot, such that each customer is serviced exactly once, subject to the time window restrictions and capacity constraints.

The VRPTW can be formulated mathematically in various ways. One possibility is to use a Set Covering (SC) or Set Partitioning (SP) formulation, in which variables represent routes. In the literature this appears to be the dominating approach to the VRPTW, in particular on instances with tight time windows. Indeed, in [7] the SP formulation is presented as being the basis of the most successful algorithms for the VRPTW. Approaches based on SC and SP formulations are discussed in [8].

Alternatively, a polyhedral approach based on a two-index vehicle-flow formulation, where a 0-1 decision variable is associated with each arc in the graph, can be adopted. This formulation underlies the branch-and-cut algorithm in [5], designed for minimizing the number of routes in the VRPTW. The two-index formulation and related projection results are recently considered

in [12].

This paper also considers the two-index vehicle-flow formulation, for which we introduce a new class of cuts called *reachability cuts*. Intuitively, they can be viewed as a strengthening of k -path inequalities, where the strengthening results from the fact that—due to the time windows—only certain arcs can be traversed on a route which services a given customer. In addition, the paper also presents procedures for the separation of these cuts.

In order to investigate the potential of the new class of cuts, we have implemented a cutting-plane algorithm for which we present computational results obtained on the well-known 87 instances of Solomon [15].

2 Notation and modelling

To simplify notation throughout the paper, we make the following definitions for any $S \subset V$:

$$\begin{aligned}\delta^+(S) &= \{(i, j) \in A \mid i \in S, j \in V \setminus S\}, \\ \delta^-(S) &= \{(i, j) \in A \mid i \in V \setminus S, j \in S\},\end{aligned}$$

where for further notational simplicity we write $\delta^+(i)$ and $\delta^-(i)$ instead of $\delta^+(\{i\})$ and $\delta^-(\{i\})$, respectively.

Further, for any $S \subseteq V_c$ we let $r(S)$ denote the minimum number of vehicles required to load the demands of the customers in S . That is, $r(S)$ is the optimum solution to the Bin Packing Problem (BPP) with bin capacity Q and item sizes given by the demands of the customers in S .

Moreover, a path $P = (v_1, \dots, v_k)$, with arc set $A_P = \{(v_i, v_{i+1}) \mid i = 1, \dots, k-1\}$, is called infeasible if it is not possible to respect all its time windows while traversing the path, otherwise it is called feasible. In the paths that we consider, all vertices $\{v_1, \dots, v_k\}$ are assumed to be different, with the exception that both v_1 and v_k may be the depot. We call an infeasible path *minimal* if and only if both subpaths obtained by removing its first and last vertex, respectively,

are feasible. Note that our distinction between feasible and infeasible paths is concerned only with temporal aspects.

We consider a *two-index vehicle flow* formulation of the VRPTW, i.e., we define a decision variable x_{ij} for each $(i, j) \in A$, where $x_{ij} = 1$ if a vehicle travels along the arc (i, j) , and $x_{ij} = 0$ otherwise. For any arc set F we let $x(F)$ denote $\sum_{(i,j) \in F} x_{ij}$. The VRPTW can then be formulated as follows:

$$\min \quad \sum_{(i,j) \in A} c_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{(i,j) \in \delta^-(k)} x_{ij} = 1 \quad \forall k \in V_c \tag{2}$$

$$\sum_{(i,j) \in \delta^+(k)} x_{ij} = 1 \quad \forall k \in V_c \tag{3}$$

$$x(\delta^-(S)) \geq r(S) \quad \forall S \subseteq V_c \tag{4}$$

$$x(A_P) \leq |P| - 2 \quad \text{for each minimal infeasible path } P \tag{5}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A \tag{6}$$

The objective (1) expresses the total travel cost. The degree equations (2) and (3) ensure that each customer is visited exactly once. The capacity constraints (4) ensure that any subset of customers is serviced by sufficiently many vehicles to load the demands of the customers in the subset. The *infeasible path constraints* (5) say that not all $|P| - 1$ arcs in any minimal infeasible path P can be traversed. Finally, restrictions (6) ensure integrality of the solution.

For ease of reference, we define the *VRPTW polytope* as the convex hull of the set of points x satisfying (2) - (6). We note that the number of routes is not fixed in this definition. As is customary, a cut (or inequality) is said to be valid for the VRPTW polytope if and only if it is satisfied by every point in the VRPTW polytope.

The linear programming (LP) relaxation of (1) - (6) can be strengthened by including other classes of cuts. For example, an infeasible path constraint for a path (v_1, \dots, v_k) can be strength-

ened to the *tournament inequality*

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^k x_{v_i v_j} \leq k - 2, \tag{7}$$

see [3]. Moreover, k -path inequalities as introduced in [11] (for $k=2$) may further strengthen the above formulation. Other classes of cuts are used in [5], where the objective of minimizing the number of routes is considered. Yet other classes of cuts derived from projection are recently presented in [12]. In this paper we introduce a new class of cuts, called *reachability cuts*, which may be useful for further strengthening the LP relaxation, in particular for instances with tight temporal constraints.

3 Reachability cuts

Basically, reachability cuts are derived by considering for each customer i which arcs can possibly—taking time windows into account—be traversed on a path from the depot to customer i , and on a path from customer i to the depot. Reachability cuts are presented formally in the following subsections. Throughout we will denote *reachability cuts* by *R-cuts*.

3.1 R-cuts for single customers

The basic constructs underlying R-cuts are the following two particular subsets of arcs, a *reaching arc set* and *reachable arc set*, respectively, for each customer.

Definition 1 For any customer $i \in V_c$, the *reaching arc set* $A_i^- \subset A$ is defined as the minimum arc set such that any feasible path $(0, \dots, i)$ lies inside A_i^- , i.e., $(j, k) \in A_i^-$ for each arc (j, k) on the feasible path.

Definition 2 For any customer $i \in V_c$, the *reachable arc set* $A_i^+ \subset A$ is defined as the minimum arc set such that any feasible path $(i, \dots, 0)$ lies inside A_i^+ , i.e., $(j, k) \in A_i^+$ for each arc (j, k) on the feasible path.

In any feasible solution, by definitions 1 and 2, the vehicle servicing customer i traverses only arcs in A_i^- from the depot to customer i and only arcs in A_i^+ from customer i to the depot.

The arc set A_i^- can be found by checking, for each arc $(j, k) \in A$, whether there exists a feasible path of the form $(0, \dots, j, k, \dots, i)$. Similarly, finding A_i^+ corresponds to checking paths of the form $(i, \dots, j, k, \dots, 0)$.

We are now ready to define R-cuts for single customers.

Definition 3 For any $S \subseteq V_c$ and any customer $i \in S$, we define the following $R_i^-(S)$ cut:

$$x(\delta^-(S) \cap A_i^-) \geq 1. \quad (8)$$

Proposition 1 The cut (8) is valid for the VRPTW polytope.

Proof: By definition (1), the vehicle servicing customer i must follow a path in A_i^- from the depot to customer i . Any such path must cross the cutset $(V \setminus S : S)$. \square

We define $R_i^+(S)$ cuts perfectly similar to $R_i^-(S)$ cuts.

Definition 4 For any $S \subseteq V_c$ and any customer $i \in S$, we define the following $R_i^+(S)$ cut:

$$x(\delta^+(S) \cap A_i^+) \geq 1. \quad (9)$$

The proof of validity of (9) follows that of (8).

For comparison with *subtour elimination constraints* (SECs), we first note that, given the degree equations (2)-(3), the following two inequalities are equivalent forms of a SEC for a customer set S :

$$x(\delta^-(S)) \geq 1, \quad (10)$$

$$x(\delta^+(S)) \geq 1. \quad (11)$$

It is obvious that (8) is a strengthening of (10) and that (9) is a strengthening of (11). Hence each of (8) and (9) dominates the SEC for S . Generally, the two cuts (8) and (9) are not equivalent, so they may both be useful simultaneously in the LP relaxation.

We note that an R-cut for a single customer to some extent corresponds to the *simple* (π, σ) -inequality in [4], which is also a strengthening of a SEC. Intuitively, the simple (π, σ) -inequality expresses, for a given ordered pair of vertices i and j in the precedence-constrained Asymmetric Traveling Salesman Problem, that a path from i to j in any feasible solution can traverse only a certain subset of arcs, which is derived from the given precedence constraints. The simple (π, σ) -inequality is also used in [3] for the Asymmetric Traveling Salesman Problem with Time Windows (ATSPTW), where feasible arc sets are derived from the time windows. In the ATSPTW, any pair of vertices i and j can be used to form a simple (π, σ) -inequality, since all vertices must be visited on the same route. In contrast, for the VRPTW it is generally not known in advance whether any two customers must be visited on the same route, i.e., it is not feasible to require a path between them. However, since all routes pass through the depot in the VRPTW, it is feasible to require a path in each direction between the depot and any other vertex in the VRPTW. Basically, this is the observation that leads to the simplest form of R-cuts, i.e., R-cuts for single customers.

Finally we consider the relation to SP based approaches to the VRPTW. In particular, if all columns in the SP problem represent elementary paths, then all SECs are satisfied by the solution which can be obtained by projecting the SP solution onto a two-dimensional x -space (see [11]). Moreover, all paths in the SP are feasible, provided that the time windows are taken into account in the SP column generation stage (which is done invariably in SC and SP based approaches). It is then easy to show that all R-cuts for single customers are implied by the column generation approach, provided that only elementary paths are generated. Note, however, that the strategy of generating only *elementary* paths is generally not computationally attractive, due to the NP-hardness of the column generation subproblem [9].

To conclude this subsection, we illustrate an R-cut for a single customer in Figure 1. The two pictures display the location of the depot and customers of the instance R102.25, with the depot represented by the solid square and customers represented by circles. The customer number is shown at each circle.

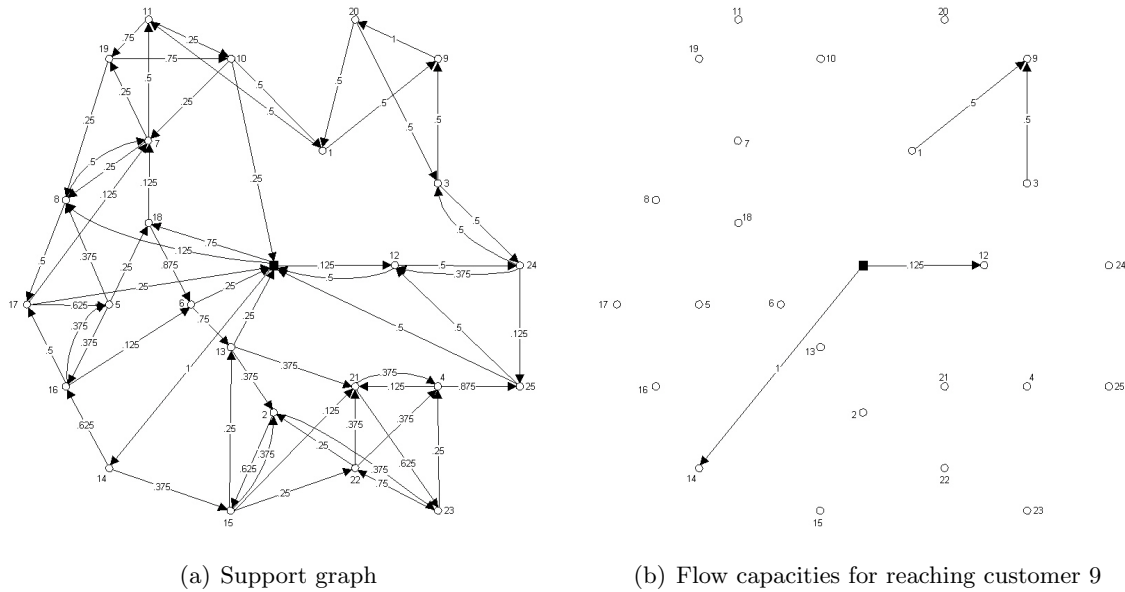


Figure 1: Illustration of a violated R-cut for a single customer

Figure 1(a) displays the support graph for the solution with objective value 408.36, obtained when separation of both capacity and tournament inequalities fail (as reported in Table 1). The weight on each arc is the value of the corresponding variable in the given LP solution.

Figure 1(b) displays those arcs in the support graph that also are contained in the reaching arc set of customer 9, whose time window closes relatively early. In a feasible solution, it is possible to send a flow of one unit from the depot to customer 9, using only arcs in the reaching arc set of customer 9, and with the current values of the decision variables as arc capacities. Since customer 9 is disconnected from the depot in Figure 1(b), a maximally violated $R_{\bar{q}}(S)$ is obtained by setting $S = \{1, 3, 9\}$, which is the component containing customer 9 in Figure 1(b).

3.2 R-cuts for multiple customers

When the VRPTW instance is sufficiently tightly constrained for *conflicting* customer sets to be identified, it is possible to derive R-cuts for multiple customers.

Definition 5 *A customer set $T \subseteq V_c$ is said to be conflicting if and only if the customers in T must be serviced on $|T|$ separate routes in any feasible solution.*

Further, we define the *reaching* and *reachable* arc set, respectively, for any conflicting customer set.

Definition 6 *For any conflicting customer set $T \subseteq V_c$, the reaching arc set A_T^- is defined as the union of the customers' reaching arc sets, i.e., $A_T^- := \bigcup_{i \in T} A_i^-$.*

Definition 7 *For any conflicting customer set $T \subseteq V_c$, the reachable arc set A_T^+ is defined as the union of the customers' reachable arc sets, i.e., $A_T^+ := \bigcup_{i \in T} A_i^+$.*

We can now define R-cuts for multiple customers.

Definition 8 *For any customer set $S \subseteq V_c$ and any conflicting customer set $T \subseteq S$, we define the following $R_T^-(S)$ cut:*

$$x(\delta^-(S) \cap A_T^-) \geq |T|. \quad (12)$$

Proposition 2 *The cut (12) is valid for the VRPTW polytope.*

Proof: By definition of A_T^- , each of the $|T|$ paths from the depot to the customers in T lies in A_T^- . These paths must use $|T|$ separate arcs to cross the cutset $(V \setminus S : S)$. \square

Definition 9 *For any customer set $S \subseteq V_c$ and any conflicting customer set $T \subseteq S$, we define the following $R_T^+(S)$ cut:*

$$x(\delta^+(S) \cap A_T^+) \geq |T|. \quad (13)$$

The proof of validity of (13) follows that of (12).

For comparison with k -path inequalities, we first note that the k -path inequality for a customer set S can be written in the following equivalent forms:

$$x(\delta^-(S)) \geq k, \tag{14}$$

$$x(\delta^+(S)) \geq k. \tag{15}$$

The k -path inequality expresses that at least k vehicles are required to service the customers in S , given that both capacity and time window constraints are imposed.

An important theoretical aspect of R-cuts is their strength over k -path inequalities, as described in Proposition 3.

Proposition 3 *For any conflicting set T , there exists one or more sets S such that the $R_T^-(S)$ cut dominates the k -path inequality for S .*

Proof: Create $|T|$ vertex-disjoint paths of customers such that i) each path is feasible, ii) each path contains exactly one of the vertices in T , and iii) the demand on each path does not exceed the vehicle capacity. Note that such paths can be created for any T . Let S denote the union of the vertices on these paths. The k -path inequality for S is $x(\delta^-(S)) \geq |T|$, which is obviously dominated by the $R_T^-(S)$ cut. \square

Most frequently, many such sets S can be generated. It is noted that the dominance relation also holds for $R_T^+(S)$ cuts over k -path inequalities.

Moreover, $R_T^-(S)$ cuts also have some similarity to the *Generalized Cut Constraints* (GCUTs) which are proposed in [10] for a class of scheduling problems, in which beginning of service is fixed to a given point in time for each customer. Indeed, the GCUTs are also concerned with sets of conflicting vertices, i.e., vertices that must be serviced on separate routes. The arguments

leading to GCUTs in [10] could be used to derive similar cuts for the VRPTW. As shown in the following, these cuts would in fact be a weakened version of $R_T^-(S)$ cuts.

To illustrate this idea, we adopt here the notation in [10] and introduce for any ordered pair of customers i, j a compatibility flag γ_{ij} which equals 1 if and only if there exists a feasible route of the form $(0, \dots, i, \dots, j, \dots, 0)$, otherwise $\gamma_{ij} = 0$. Further, for any $S \subset V_c$, we define $\Pi(S) = \{i \in V_c \setminus S \mid \gamma_{ij} = 0 \forall j \in S\}$. That is, a feasible path from 0 to any customer in S cannot visit any customer in $\Pi(S)$. Now consider a customer set S and a conflicting set $T \subseteq S$, and let $x(S_1, S_2)$ denote $\sum_{(i,j) \mid i \in S_1, j \in S_2} x_{ij}$. The GCUT is ([10, Theorem 4]):

$$x((V \setminus S) \setminus \Pi(T), S \setminus \Pi(T)) \geq |T|. \quad (16)$$

The inequality expresses that the cutset from $V \setminus S$ to S must be crossed at least $|T|$ times using arcs (i, j) for each of which $i, j \notin \Pi(T)$.

The GCUT (16) is, however, dominated by the $R_T^-(S)$ cut (12). Indeed, the two right hand sides are identical, and any variable occurring on the left hand side of (12) also occurs on the left hand side of (16). To see this, note that $(i, j) \in A_T^-$ implies $i, j \notin \Pi(T)$, for any arc (i, j) .

To see that (12) *strictly* dominates (16), note that $i, j \notin \Pi(T)$ does not imply $(i, j) \in A_T^-$. For example, the route $(0, i, j, k, 0)$ may be infeasible although the routes $(0, i, j, 0)$, $(0, i, k, 0)$, and $(0, j, k, 0)$ are all feasible, in which case we would have $(i, j) \notin A_{\{k\}}^-$ and $i, j \notin \Pi(\{k\})$. For $T = \{k\}$, $i \notin S$, and $j \in S$, variable x_{ij} is then on the left hand side in (16) but not in (12).

Note also that if all three customers i, j, k had time windows of zero width (as in pure scheduling problems), the existence of both arcs (i, j) and (j, k) would imply feasibility wrt. time of the path (i, j, k) . Indeed, in the extreme case that we actually have a scheduling problem (i.e., $r_i = d_i \forall i \in V_c$), the $R_T^-(S)$ cuts would simplify to GCUTs.

It is noted that equivalent relations exist between $R_T^+(S)$ cuts and *transposed* GCUTs [10].

We finally consider the alternative of using a multicommodity (MC) flow formulation of the requirements relating to a given conflicting set. The relations to an MC flow formulation are important in connection with the separation procedure that is proposed in Section 4.

We consider a given conflicting set T . Since the customers in T must be visited on separate routes, a feasible solution permits a flow of one unit from the depot to each customer in T simultaneously for $|T|$ distinct commodities, using flow only along arcs (i, j) for which $x_{ij} = 1$. (The flow from T to the depot can be treated similarly and is as such not described further.) In terms of MC flows, let y_{ij}^k be the flow of commodity k (destined for customer k) along arc (i, j) . A solution x must then permit a feasible solution to the following MC flow constraints.

$$\sum_{k \in T} \sum_{(i,j) \in \delta^+(0)} y_{ij}^k = |T| \quad (17)$$

$$\sum_{(i,j) \in \delta^-(k)} y_{ij}^k = 1 \quad \forall k \in T \quad (18)$$

$$\sum_{(i,j) \in \delta^-(v)} y_{ij}^k = \sum_{(i,j) \in \delta^+(v)} y_{ij}^k \quad \forall k \in T, \forall v \in V_c \setminus T \quad (19)$$

$$\sum_{k \in T} y_{ij}^k \leq x_{ij} \quad \forall (i, j) \in A \quad (20)$$

$$y_{ij}^k = 0 \quad \forall k \in T, \forall (i, j) \notin A_k^- \quad (21)$$

It is important here to note the effect of (19) and (21). Constraint (19) ensures the balance of flow for *each* commodity, and constraint (21) ensures, for *each* commodity k , that it flows only along arcs in A_k^- .

On the other hand, R-cuts for this particular conflicting set T involves the flow of only one commodity, representing the total flow of the $|T|$ commodities in the MC flow formulation. As such, if the $R_T^-(S)$ cut is satisfied for any S for this fixed T , it is implied that there exists a feasible solution to the following single-commodity flow problem, where y_{ij} denotes the (aggregated) flow along arc (i, j) :

$$\sum_{(i,j) \in \delta^+(0)} y_{ij} = |T| \quad (22)$$

$$\sum_{(i,j) \in \delta^-(k)} y_{ij} = 1 \quad \forall k \in T \quad (23)$$

$$\sum_{(i,j) \in \delta^-(v)} y_{ij} = \sum_{(i,j) \in \delta^+(v)} y_{ij} \quad \forall v \in V_c \setminus T \quad (24)$$

$$y_{ij} \leq x_{ij} \quad \forall (i,j) \in A \quad (25)$$

$$y_{ij} = 0 \quad \forall (i,j) \notin A_T^- \quad (26)$$

A feasible solution to (17)-(21) can be converted into a feasible solution to (22)-(26) by setting $y_{ij} := \sum_{k \in T} y_{ij}^k$ for each $(i,j) \in A$. However, a feasible solution to (22)-(26) cannot necessarily be converted into a feasible solution to (17)-(21). As such, the set of R-cuts associated with a given T can be viewed as a weakening of the MC flow constraints for T . However, it would obviously be impractical to embed MC flow constraints for a large number of conflicting sets in the formulation. The important aspect of the above relations is that *if* a feasible solution x to (17)-(21) is known, it follows that all R-cuts for any conflicting set $T' \subseteq T$ are satisfied. The idea of maintaining a *flow pool* as described in Subsection 5.2 is based on this observation.

3.3 R-cuts for semi-conflicting pairs

We finally introduce a conditional form of the R-cut for two customers. The cut is obtained if the two customers are *semi-conflicting*.

Definition 10 *Two customers i and j are said to be semi-conflicting if and only if both of the following hold:*

1. *Customers i and j are not conflicting, and*
2. *The time windows do not allow the insertion of any customers on a path between i and j .*

That is, the only possibility of serving a pair $\{i,j\}$ of semi-conflicting customers on the same route is to visit them consecutively, i.e., to require that $x_{ij} + x_{ji} = 1$. This leads to the following definitions.

Definition 11 For any customer set $S \subseteq V_c$ and any pair of semi-conflicting customers $i, j \in S$, we define the following $R_{ij}^-(S)$ cut:

$$x(\delta^-(S) \cap (A_i^- \cup A_j^-)) + x_{ij} + x_{ji} \geq 2. \quad (27)$$

Proposition 4 The cut (27) is valid for the VRPTW polytope.

Proof: If $x_{ij} + x_{ji} = 0$, the cut simplifies to an $R_T^-(S)$ cut with $T = \{i, j\}$, which is valid since i and j are effectively conflicting under the condition $x_{ij} + x_{ji} = 0$. Alternatively, if $x_{ij} + x_{ji} = 1$, the inequality only implies a weakened version of the $R_i^-(S)$ cut. \square

Definition 12 For any customer set $S \subseteq V_c$ and any pair of semi-conflicting customers $i, j \in S$, we define the following $R_{ij}^+(S)$ cut:

$$x(\delta^+(S) \cap (A_i^+ \cup A_j^+)) + x_{ij} + x_{ji} \geq 2. \quad (28)$$

The proof of validity of (28) follows that of (27).

4 Separation

For a given LP solution vector x^* satisfying (2), (3), and the bounds implied by (6), we now consider some details related to the separation of R-cuts. We note that each customer in itself may be viewed as defining a conflicting set. In the following it is understood that a conflicting set may be a singleton.

4.1 Reaching and reachable arc sets

The first issue is to identify the reaching and reachable arc sets for each customer. In general, for each customer i and each arc (j, k) , we need to determine whether there exists a feasible

path of the form $(0, \dots, j, k, \dots, i)$. The arc set A_i^- contains exactly those arcs (j, k) for which such a path exists. (The identification of A_i^+ is similar and is not described further.)

For identifying reaching sets, we have—for the sake of computational simplicity—restricted our implementation to a procedure which is heuristic in general, in the sense that it for each $i \in V_c$ produces a set $\hat{A}_i^- \supseteq A_i^-$ which we use instead of A_i^- throughout the code. Our procedure for identifying reaching arc sets works as follows:

1. Calculate, for each ordered pair $i, j \in V$, the fastest path from i to j subject to the restriction that the depot is not an intermediate vertex on the path. Time windows are disregarded in these calculations. This step is performed in $O(n^3)$ time by a single application of the *Floyd–Warshall* algorithm [2, Subsection 5.6], using the direct travel times as input. We let θ_{ij} denote the duration of the fastest path from i to j as calculated in this step.
2. Set $\hat{A}_i^- = \delta^-(i)$ for each $i \in V_c$.
3. Perform step 4 for each ordered set (j, k, i) of distinct vertices satisfying $(j, k) \in A$ and $k, i \in V_c$, then stop.
4. Let b_j , b_k , and b_i , respectively, be a lower bound on the earliest possible time at which service can begin at customer j , k , and i , respectively, on a path of the form $(0, \dots, j, k, \dots, i)$. These bounds can be computed as follows:

$$(a) \quad b_j = \max\{e_j, e_0 + \theta_{0j}\}$$

$$(b) \quad b_k = \max\{e_k, b_j + t_{jk}\}$$

$$(c) \quad b_i = \max\{e_i, b_k + \theta_{ki}\}$$

Add (j, k) to \hat{A}_i^- if and only if $b_k \leq d_k$ and $b_i \leq d_i$.

Provided that travel times satisfy the triangle inequality, the above procedure implies an exact determination of the reaching sets, i.e., $\hat{A}_i^- = A_i^-$ for each $i \in V_c$. Indeed, step 1 yields simply $\theta_{ij} = t_{ij} \forall (i, j) \in A$, so the paths considered are effectively of the form $(0, j, k, i)$.

In contrast, in cases where travel times do not satisfy the triangle inequality, the above procedure is only heuristic in the sense that $A_i^- \subset \hat{A}_i^-$ is possible for one or more customers i . In addition to omitted waiting times, a complicating reason for this is that the two path durations θ_{0j} and θ_{ki} in step 4 may correspond to two paths that are not vertex disjoint. However, for the reasons that all test instances considered in this paper (see Subsection 5.1) actually do satisfy the triangle inequality, and this also appears to be a usual assumption in the literature (as in, e.g., [5, 11]), we do not pursue this issue any further.

4.2 Identification of conflicts

The identification of conflicts is based on the reaching and reachable arc sets. In particular, we say that customer j is a *possible predecessor* of customer i if and only if there exists a $k \in V_c$ such that $(j, k) \in A_i^-$. On this basis, two customers i, j are conflicting if and only if i is not a possible predecessor of j , and j is not a possible predecessor of i . A customer set T is conflicting if and only if each pair $i, j \in T$ is conflicting.

In our implementation, the identification of conflicting sets is based on \hat{A}_i^- instead of A_i^- for each $i \in V_c$. Therefore, if travel times do not satisfy the triangle inequality, our procedure is heuristic in the sense that one or more conflicting pairs may not be identified.

Regarding the identification of semi-conflicting pairs, we simply check for each pair of customers i, j which is not identified as being conflicting, whether there exists a $k \in V_c \setminus \{i, j\}$ such that either $(i, k) \in A_j^-$ or $(j, k) \in A_i^-$. The customer pair i, j is semi-conflicting if no such k exists.

4.3 Separation for a conflicting set

For a given conflicting set T , exact separation of $R_T^-(S)$ cuts is done as follows (the procedure is similar for R_T^+ cuts). Construct a graph with vertex set $V \cup \{n+1\}$ and arc set A_T^- , and to that add an arc $(i, n+1)$ of infinite capacity for each $i \in T$. For each arc $(i, j) \in A_T^-$, let its capacity be x_{ij}^* . After solving the maximum flow problem on this graph with the depot as source and node $n+1$ as sink, the customers on the sink side of the minimum cut form the set S for which the $R_T^-(S)$ cut is maximally violated.

4.4 Separation for a semi-conflicting pair

Separation for a given semi-conflicting pair i, j is done by solving the maximum flow problem as if i, j form a conflicting set and subsequently including x_{ij}^* and x_{ji}^* when determining the violation.

5 Computational results

In order to investigate empirically the potential of R-cuts, we have implemented a cutting-plane algorithm in which various classes of cuts are included, specifically capacity inequalities, tournament inequalities, and R-cuts. Both tournament inequalities and R-cuts are based only on temporal aspects. The experiments make it possible to investigate the relative strength of these two classes of inequalities for dealing with the temporal aspects of a given instance.

In Subsection 5.1 we present the experiments that have been carried out in order to investigate the strength of R-cuts as a function of the largest cardinality of the conflicting sets considered. Given these experimental results, a particular separation algorithm is proposed in Subsection 5.2 based on a trade-off between bound quality and computing time.

5.1 Experiments

The cutting-plane algorithm has been coded in the C programming language using the Microsoft Visual C++ v. 6.0 compiler. All experiments were done on a PC with a 1.6 GHz Intel Pentium M processor and 512 MB of RAM running Microsoft Windows XP.

The algorithm has been applied to the instance classes R1, C1, and RC1 by Solomon [15]. For calculation of travel costs and travel times we have used the convention in [11].

The computational results are shown in tables 1, 2, and 3 (in which an LB is preceded by a ‘*’ if the corresponding LP solution is integer and feasible, i.e., optimal). For each instance is given the following information:

Name: The name of the instance;

Opt.: The cost of the optimal solution, if known. A ‘—’ indicates that the optimal solution is unknown. The data are taken from <http://web.cba.neu.edu/~msolomon/problems.htm>;

NV: The number of vehicles (routes) in the optimal solution, if known;

C: The cardinality of the largest set of conflicting customers, computed using the tree search algorithm in [6];

B: The minimum number of vehicles required to load the demands of all customers, i.e., the optimal solution to the BPP given by the vehicle capacity and customer demands;

Capacity: The lower bound (LB) obtained using only degree, nonnegativity, and capacity constraints, and the total time (in seconds) required to obtain it (T). For separation of capacity constraints we used the publicly available [13] separation routines from [14];

Tournament: The lower bound (LB) obtained by using tournament inequalities in addition to those used for the capacity bound, and the total time required to obtain it (T). A ‘—’ indicates that the bound is the same as the capacity bound. For the separation of tournament inequalities

we used an exact tree search procedure. The running time of this separation procedure is polynomially bounded, as only a polynomial number of paths need to be considered (see [3]). The separation algorithm for tournament inequalities is invoked whenever separation of capacity inequalities fails;

R1-R4: These four columns display the lower bounds obtained by using R-cuts in addition to those used for the capacity bound. Generally, in column Rr exact separation is done for each conflicting set T with $|T| \leq r$. A ‘—’ in column R1 indicates that the bound is the same as the capacity bound. A ‘—’ in columns R2-R4 indicates that the bound is the same as that in the previous (left) column;

R2.5: The lower bound (LB) obtained by adding R-cuts for semi-conflicting pairs to those used for the R2 bound, and the total time required to obtain it (T). Exact separation is done for each semi-conflicting pair. Details on obtaining this bound are given in Subsection 5.2. This separation algorithm is invoked whenever separation of capacity inequalities fails.

Each of the two values in the ‘C’ and ‘B’ columns can be taken as a valid lower bound on the number of routes. On most R1 and RC1 instances the ‘NV’ value is strictly greater than both the ‘C’ and the ‘B’ value. However, the ‘B’ value equals the ‘NV’ value on all C1 instances, which might suggest that the capacity bound would be relatively tight for this class. On the other hand, temporal constraints might be expected to be relatively more important on instances where ‘C’ greatly exceeds ‘B’.

As suggested, the capacity bound is indeed particularly tight on the C1 instances, among which the optimum is obtained as lower bound on several instances.

Tournament inequalities give an improvement in the lower bound on several instances, although the bound increase over the capacity bound tends to be relatively small.

The R1 bound is significantly greater than the capacity bound on many instances. The first LP solution on which capacity separation fails does in fact satisfy all SECs (this was verified

using exact min-cut computations), so any difference between R1 and capacity bounds is due to the increased strength of R1 cuts over SECs.

Among those instances that are not solved by using only capacity cuts, the R1 bound is greater than the tournament bound on almost all instances. Indeed, among the instances in tables 1, 2, and 3, the R1 bound is greater than the tournament bound on 32 out of 35, 14 out of 15, and 22 out of 24 instances, respectively. Across all three tables, the tournament bound is better than the R1 bound on only two instances (R101.100 and C104.100). We also note that the gap remaining after capacity cuts is in fact closed by R1 on five C-instances. In comparison, tournament inequalities were not able to close any of the gaps. To summarize, it is clear that R-cuts for single customers compares favourably with tournament inequalities in terms of the obtained lower bounds.

Regarding R2, R3, and R4, the bound increase diminishes as the maximum cardinality of conflicting sets increases. We have not reported the computing times for columns R1–R4; it suffices to say that the running times become excessive when all conflicting sets of cardinality three or more are considered.

The R2.5 column shows that semi-conflicting pairs in several cases can give a bound improvement beyond that obtained by R3 or R4. Moreover, the computational complexity of obtaining R2.5 is smaller than that for obtaining R3.

In the light of these observations, we propose the R2.5 bound as a trade-off between bound quality and running time. Concludingly, we note that the proposed R2.5 bound on the vast majority of instances is better, and in many cases significantly better, than the bound obtained by tournament inequalities.

5.2 Obtaining the R2.5 bound

In a straightforward implementation for obtaining the R2.5 bound, the required number of max-flow computations is two for each customer, two for each conflicting pair, and two for

each semi-conflicting pair, i.e., a total of $O(n^2)$ max-flow computations. In order to reduce this number we keep a pool of flows, by which a solution to one maximum flow problem can contribute to a faster solution to other, related maximum flow problems. The basis for this is described in Subsection 3.2.

The idea is to maintain a pool which contains two flow vectors for each customer. Generally, whenever a maximum flow problem has been solved for a given set T , we decompose it into path flows using standard flow decomposition (see [2, Subsection 3.5]). Say that the direction is from 0 to T . For each $i \in T$, we check if each of the path flows from 0 to i uses only arcs in A_i^- . If so, and if the path flows into i add up to one unit, the union of paths from 0 to i defines a *feasible 1-flow* from 0 to i .

A feasible 1-flow from 0 to i is an optimal solution to the maximum flow problem associated with the separation of $R_i^-(S)$ cuts. In general, whenever we wish to solve a maximum flow problem for separating $R_T^-(S)$ cuts, we first check if a feasible 1-flow is available in the pool for each $i \in T$. If so and the sum of the $|T|$ feasible 1-flows is feasible for the maximum flow problem it is also a maximum flow. Generally this approach can significantly reduce the number of times that a maximum flow algorithm must be called.

We first search for up to $n/2$ vertex-disjoint pairs of conflicting customers, for each of which we compute the two max-flows. Hopefully this provides a reasonably large number of feasible 1-flows for the flow pool. Subsequently we proceed through all conflicting pairs, then all semi-conflicting pairs, and finally all individual customers. The cut generation stops if a limit of at most $3n$ cuts between any two consecutive LP reoptimizations is reached.

In addition to the flow pool, another partial explanation for the reasonable running times is that for solving maximum flow problems we use an implementation of the *highest-label preflow-push algorithm*, which has shown excellent running times in practice. Although its worst-case running time is only $O(n^2\sqrt{m})$ on a graph with n vertices and m arcs, in practice it has shown even better performance than that, down to an average of $O(n^{1.5})$ running time, see [1].

6 Conclusion

This paper has introduced a new class of cuts called reachability cuts (R-cuts). The relations between R-cuts and previously known classes of cuts have been described. In particular, a dominance relation exists between R-cuts and k -path inequalities such that any R-cut dominates one or more k -path inequalities.

Computational experiments were done with a cutting plane algorithm which involved capacity inequalities, tournament inequalities, and R-cuts. The computational results showed that capacity inequalities and R-cuts generally provides better bounds than those obtained by capacity inequalities and tournament inequalities. As such, the new class of cuts has shown to be competitive in dealing with the temporal aspects of VRPTW instances.

Finally, a particular separation procedure with reasonable running time has been proposed for the new class of cuts.

Based on the findings in this paper, we conclude that R-cuts constitute a promising class of cuts for a full branch-and-cut algorithm which we intend to develop in the near future.

Acknowledgement

The author wishes to thank two anonymous referees for their helpful comments which led to an improved paper.

References

- [1] R. K. Ahuja, M. Kodialam, A. K. Mishra, and J. B. Orlin. Computational investigations of maximum flow algorithms. *European Journal of Operational Research*, 97:509–542, 1997.
- [2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: Theory, Algorithms, and Applications*. Prentice-Hall, 1993.

- [3] N. Ascheuer, M. Fischetti, and M. Grötschel. Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. *Mathematical Programming*, 90:475–506, 2001.
- [4] E. Balas, M. Fischetti, and W. R. Pulleyblank. The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68:241–265, 1995.
- [5] J. F. Bard, G. Kontoravdis, and G. Yu. A branch-and-cut procedure for the vehicle routing problem with time windows. *Transportation Science*, 36:250–269, 2002.
- [6] R. Carraghan and P. M. Pardalos. An exact algorithm for the maximum clique problem. *Operations Research Letters*, 9:375–382, 1990.
- [7] W. Cook and J. L. Rich. A parallel cutting-plane algorithm for the vehicle routing problem with time windows. Technical report, Computational and Applied Mathematics, Rice University, 1999.
- [8] J.-F. Cordeau, G. Desaulniers, J. Desrosiers, M. M. Solomon, and F. Soumis. VRP with time windows. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, chapter 7. SIAM Monographs on Discrete Mathematics and Applications, 2002.
- [9] M. Desrochers, J. Desrosiers, and M. Solomon. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40:342–354, 1992.
- [10] M. Fischetti, A. Lodi, S. Martello, and P. Toth. A polyhedral approach to simplified crew scheduling and vehicle scheduling problems. *Management Science*, 47:833–850, 2001.
- [11] N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, and F. Soumis. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33:101–116, 1999.
- [12] A. N. Letchford and J.-J. Salazar-González. Projection results for vehicle routing. To appear in *Mathematical Programming*, 2004.

- [13] J. Lysgaard. CVRPSEP: A package of separation routines for the capacitated vehicle routing problem. Working Paper 03-04, Department of Management Science and Logistics, Aarhus School of Business. Available at www.asb.dk/~lys, 2003.
- [14] J. Lysgaard, A. N. Letchford, and R. W. Eglese. A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, 100:423–445, 2004.
- [15] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987.

Name	Opt.	NV	C	B	Capacity		Tournament				R2.5				
					LB	T	LB	T	R1	R2	R3	R4	LB	T	
R101.25	617.1	8	8	2	*617.10	0.02	—	—	—	—	—	—	—	—	—
R101.50	1044.0	12	11	4	1029.30	0.03	1042.17	0.05	1043.37	—	—	—	1043.37	0.09	
R101.100	1637.7	20	18	8	1611.90	0.10	1631.70	0.16	1622.30	—	—	—	1622.30	0.20	
R102.25	547.1	7	6	2	389.70	0.02	408.36	0.03	498.45	536.36	536.75	—	540.20	0.24	
R102.50	909.0	11	9	4	667.75	0.07	682.38	0.24	812.31	871.09	896.56	909.00	871.89	2.03	
R102.100	1466.6	18	16	8	1071.12	0.78	1090.33	1.55	1240.99	1327.51	1365.45	1393.46	1329.41	25.79	
R103.25	454.6	5	4	2	346.00	0.01	350.87	0.03	413.47	443.36	444.81	—	448.95	0.27	
R103.50	772.9	9	8	4	582.78	0.13	592.42	0.28	678.17	723.47	741.55	745.72	725.15	2.25	
R103.100	1208.7	14	13	8	871.97	1.65	885.46	2.74	990.73	1054.42	1087.36	1115.37	1055.14	35.61	
R104.25	416.9	4	4	2	339.88	0.02	341.43	0.04	368.30	403.74	406.99	—	404.00	0.26	
R104.50	625.4	6	5	4	530.20	0.08	531.31	0.24	538.38	560.47	573.25	577.11	561.93	1.60	
R104.100	—	—	8	8	808.76	2.45	812.48	3.07	833.23	842.16	846.58	848.75	843.09	20.19	
R105.25	530.5	6	4	2	483.18	0.01	498.00	0.04	526.05	526.06	—	—	526.55	0.04	
R105.50	899.3	9	5	4	836.70	0.01	855.30	0.04	875.61	—	—	—	875.95	0.12	
R105.100	1355.3	15	7	8	1159.00	0.32	1189.72	0.65	1273.83	1274.46	—	—	1275.90	3.47	
R106.25	465.4	3	3	2	366.40	0.02	371.85	0.05	436.76	447.05	447.64	—	457.30	0.19	
R106.50	793.0	5	5	4	603.95	0.16	613.36	0.24	700.05	725.22	726.68	—	752.19	2.56	
R106.100	1234.6	13	7	8	933.77	1.62	949.01	2.93	1036.95	1062.33	1066.16	1067.22	1076.26	30.74	
R107.25	424.3	4	3	2	334.10	0.01	339.25	0.03	393.96	407.87	409.04	—	409.80	0.25	
R107.50	711.1	7	5	4	538.60	0.10	544.88	0.18	610.81	626.88	629.78	630.42	654.45	2.54	
R107.100	1064.6	11	6	8	826.52	2.39	836.19	3.73	904.85	914.54	915.91	916.13	930.56	40.80	
R108.25	397.3	4	3	2	328.97	0.01	332.53	0.06	351.00	365.63	367.60	—	367.67	0.21	
R108.50	617.7	6	4	4	520.67	0.14	521.53	0.21	535.47	538.70	538.77	—	543.88	0.84	
R108.100	—	—	5	8	804.48	3.53	805.15	4.37	819.71	820.69	820.79	—	823.35	16.42	
R109.25	441.3	5	3	2	374.30	0.01	380.08	0.02	419.60	—	—	—	419.60	0.03	
R109.50	786.8	8	3	4	588.23	0.08	605.38	0.24	698.76	699.31	—	—	701.87	1.29	
R109.100	1146.9	13	4	8	861.76	1.72	884.60	3.02	952.45	—	—	—	953.22	12.55	
R110.25	444.1	4	1	2	328.97	0.01	331.19	0.02	374.04	—	—	—	379.25	0.19	
R110.50	697.0	7	3	4	535.83	0.17	537.61	0.24	582.76	583.47	583.48	—	594.00	1.76	
R110.100	1068.0	12	3	8	809.50	2.56	812.37	3.54	867.18	867.23	—	—	868.06	16.40	
R111.25	428.8	5	3	2	336.70	0.01	343.27	0.02	395.48	405.22	406.07	—	409.14	0.28	
R111.50	707.2	7	4	4	539.04	0.12	546.77	0.22	600.02	609.87	611.03	611.16	623.91	2.32	
R111.100	1048.7	12	6	8	818.23	2.47	828.31	3.77	884.50	885.73	886.18	886.48	896.24	22.18	
R112.25	393.0	4	1	2	328.97	0.02	—	0.06	329.17	—	—	—	329.17	0.03	
R112.50	630.2	6	1	4	520.67	0.15	—	0.16	—	—	—	—	520.67	0.23	
R112.100	—	—	1	8	800.38	3.77	—	3.94	—	—	—	—	800.38	4.40	

Table 1: Results for the R1 instances.

Name	Opt.	NV	C	B	Capacity		Tournament				R2.5			
					LB	T	LB	T	R1	R2	R3	R4	LB	T
C101.25	191.3	3	3	3	*191.30	0.04	—	—	—	—	—	—	—	—
C101.50	362.4	5	5	5	*362.40	0.04	—	—	—	—	—	—	—	—
C101.100	827.3	10	10	10	*827.30	0.12	—	—	—	—	—	—	—	—
C102.25	190.3	3	2	3	186.90	0.05	190.28	0.10	*190.30	—	—	—	*190.30	0.06
C102.50	361.4	5	5	5	358.00	0.08	361.38	0.24	*361.40	—	—	—	*361.40	0.10
C102.100	827.3	10	9	10	819.85	0.83	826.85	1.16	827.15	—	—	—	827.15	0.93
C103.25	190.3	3	2	3	186.90	0.03	187.35	0.05	*190.30	—	—	—	*190.30	0.05
C103.50	361.4	5	4	5	358.00	0.14	358.77	0.17	*361.40	—	—	—	*361.40	0.24
C103.100	826.3	10	7	10	819.85	3.58	822.70	4.16	826.15	—	—	—	826.15	4.23
C104.25	186.9	3	2	3	186.45	0.05	186.46	0.06	*186.90	—	—	—	*186.90	0.07
C104.50	358.0	5	2	5	357.51	0.38	357.69	0.41	357.86	—	—	—	357.86	0.47
C104.100	822.9	10	4	10	817.36	6.43	817.76	6.58	817.66	817.67	—	—	817.77	6.84
C105.25	191.3	3	2	3	*191.30	0.04	—	—	—	—	—	—	—	—
C105.50	362.4	5	3	5	*362.40	0.03	—	—	—	—	—	—	—	—
C105.100	827.3	10	5	10	*827.30	0.08	—	—	—	—	—	—	—	—
C106.25	191.3	3	3	3	*191.30	0.03	—	—	—	—	—	—	—	—
C106.50	362.4	5	5	5	*362.40	0.02	—	—	—	—	—	—	—	—
C106.100	827.3	10	6	10	*827.30	0.45	—	—	—	—	—	—	—	—
C107.25	191.3	3	1	3	*191.30	0.01	—	—	—	—	—	—	—	—
C107.50	362.4	5	1	5	*362.40	0.03	—	—	—	—	—	—	—	—
C107.100	827.3	10	1	10	*827.30	0.08	—	—	—	—	—	—	—	—
C108.25	191.3	3	1	3	186.53	0.05	187.15	0.05	190.75	—	—	—	190.75	0.07
C108.50	362.4	5	1	5	358.30	0.15	358.78	0.18	361.85	—	—	—	361.85	0.21
C108.100	827.3	10	1	10	817.83	3.18	819.67	3.44	826.75	—	—	—	826.75	3.88
C109.25	191.3	3	1	3	185.87	0.06	186.35	0.06	187.55	—	—	—	187.55	0.10
C109.50	362.4	5	1	5	357.51	0.16	357.94	0.22	358.65	—	—	—	358.65	0.21
C109.100	827.3	10	1	10	816.65	3.98	818.73	4.29	823.55	—	—	—	823.55	4.31

Table 2: Results for the C1 instances.

Name	Opt.	NV	Capacity				Tournament				R2.5			
			C	B	LB	T	LB	T	R1	R2	R3	R4	LB	T
RC101.25	461.1	4	3	3	359.70	0.05	363.95	0.05	371.85	—	—	—	374.45	0.09
RC101.50	944.0	8	6	5	670.55	0.04	683.87	0.09	745.40	762.46	—	—	794.23	0.35
RC101.100	1619.8	15	8	9	1283.79	0.43	1319.13	0.86	1481.96	1491.58	1491.88	—	1508.39	5.18
RC102.25	351.8	3	3	3	313.60	0.01	319.25	0.04	342.73	—	—	—	344.63	0.03
RC102.50	822.5	7	5	5	557.80	0.03	570.94	0.06	621.69	—	—	—	622.67	0.20
RC102.100	1457.4	14	7	9	1025.20	5.78	1038.12	8.24	1182.63	1201.36	1209.18	1210.33	1225.70	43.16
RC103.25	322.8	3	3	3	296.90	0.01	300.45	0.04	318.37	—	—	—	318.37	0.10
RC103.50	710.9	6	5	5	527.60	0.03	534.74	0.05	583.00	—	—	—	583.00	0.37
RC103.100	1258.0	11	7	9	975.18	10.40	980.81	12.90	1055.60	1064.51	1066.49	1066.50	1076.03	44.97
RC104.25	306.6	3	3	3	296.00	0.03	297.15	0.03	299.70	—	—	—	299.70	0.05
RC104.50	545.8	5	3	5	519.20	0.04	520.87	0.04	522.90	—	—	—	522.90	0.05
RC104.100	—	—	6	9	965.03	11.90	966.32	12.74	983.70	985.64	986.08	—	992.50	34.86
RC105.25	411.3	4	4	3	317.55	0.04	325.63	0.04	354.32	390.27	401.47	—	391.87	0.14
RC105.50	855.3	8	6	5	578.80	0.02	592.22	0.06	637.58	684.30	—	—	684.30	0.31
RC105.100	1513.7	15	12	9	1087.88	2.60	1107.50	3.90	1262.71	1322.96	1345.92	1359.56	1330.73	22.54
RC106.25	345.5	3	2	3	309.10	0.02	316.40	0.03	325.20	—	—	—	325.20	0.05
RC106.50	723.2	6	4	5	552.90	0.03	559.47	0.05	589.76	—	—	—	589.76	0.36
RC106.100	—	—	4	9	1029.73	4.39	1044.52	6.36	1110.42	1110.51	—	—	1113.12	21.78
RC107.25	298.3	3	2	3	294.50	0.03	294.60	0.03	296.30	—	—	—	296.30	0.04
RC107.50	642.7	6	3	5	521.05	0.04	523.39	0.05	536.70	—	—	—	536.70	0.25
RC107.100	—	—	4	9	969.51	11.72	973.48	14.74	1011.86	1012.26	—	—	1012.80	36.72
RC108.25	294.5	3	2	3	294.50	0.03	—	0.04	—	—	—	—	294.50	0.04
RC108.50	598.1	6	2	5	517.70	0.04	—	0.04	—	—	—	—	517.70	0.05
RC108.100	—	—	3	9	959.06	14.24	959.13	14.61	963.10	963.17	—	—	963.41	26.15

Table 3: Results for the RC1 instances.