2015 ACP Summer School Competition in Constraint Programming

Inventory Rebalancing for Bike-Sharing Systems

Joris Kinable[1]

[1]Robotics Institute & Tepper School of Business, Carnegie Mellon University, jkinable@cs.cmu.edu

August 6, 2015

1 Introduction

The city of Toronto runs a bike-sharing system in which bicycles are made available for shared use to individuals. The bikes are kept at self-service terminals (stations) throughout the city. Individuals can rent a bike at a station and return it, after a certain amount of time, to the same or any another station. Each station has a limited number of *docks* (places where bikes are positioned inside the station). Due to the fact that bikes are not necessarily returned to the station they originated from, certain stations may run out of bikes, or may have no empty docks left. Consequently, inventory rebalancing has to be performed periodically, thereby transporting bikes from stations with an excess of bikes to stations with a shortage of bikes. This problem, solved for a homogeneous fleet of vehicles, is known as the Bicycle Rebalancing Problem (BRP).

As part of the ACP Summer School Competition in Constraint Programming, we were asked to design and implement an algorithm for a variation of the BRP, where each station has to be serviced within a given time window by a heterogeneous fleet of vehicles. We will denote this variant as BRP-TW. In this article we will describe the winning implementation which is based on a Constraint Programming (CP) model, and compare its performance against another, general-purpose Mixed Integer Programming (MIP) approach.

Given is a directed, weighted graph $G(V, A)$, with vertex set $V = \{0, 1, \ldots, n\}$ and a set of arcs $A \subset V \times V$. Vertex 0 represent a depot where a fleet of trucks is parked; the remaining vertices represent bicycle stations. Each station $i \in V \setminus \{0\}$ has a positive or negative demand d_i for bikes: $d_i > 0$ represents an excess of bikes, $d_i < 0$ a shortage of bikes at station i. Bikes may be redistributed over the stations by a set of heterogeneous trucks K. Each truck has a capacity q_k and a driving cost per time unit c_k. Each station i has an associated time window $[a_i, b_i]$ during which the requested bikes have to be delivered or removed. A positive driving time t_{ij} is associated with each arc $(i, j) \in A$; the time required to service a station is negligible. Travel times t_{ij}, t_{ji}, are symmetrical, but do not necessary comply with the triangle inequality (e.g. $t_{ij} + t_{jk}$ may be smaller than t_{ik}). The objective is to calculate routes of minimum cost for each truck such that the demand of each station is met within its respective time window. The cost of a route is computed by the total number of time units the truck travels (excluding waiting time), multiplied by the cost of the truck c_k. A truck's route must start and end at the depot. To facilitate operations, a station may only be serviced once by a single truck. When the truck enters or leaves the depot, it may carry a positive number of bikes [1].

In what follows, let $\delta^+(i)$, $\delta^-(i)$ denote resp. the set of outgoing, incoming arcs into/from a node $i \in V$. Furthermore, it is assumed that there are no stations with a demand equal to zero. If such a station would exist, it can be removed as part of a pre-processing step. Similarly, any arc in $(i, j) \in A$ is removed if $a_i + t_{ij} > b_j$.

[1]This problem description deviates slightly from the one posted during the competition, where trucks were assumed to carry 0 bikes once they left the depot, but this is uncommon in practice.

Constraint	Description
presenceOf(α)	Returns 1 if interval α is present, 0 otherwise.
noOverlapSequence($B, dist$)	Sequences the intervals in the set B. Ensures that the intervals in B do not overlap. Furthermore, the two-dimensional distance matrix $dist$ specifies for each pair of intervals a sequence dependent setup time. Absent intervals are ignored. Returns a sequence of the intervals in B.
first(α, seq)	If interval α is present in sequence seq, it must be scheduled before any other interval in the sequence.
last(α, seq)	If interval α is present in sequence seq, it must scheduled after any of the intervals in the sequence.
isSucc($\alpha, \beta, seq, \gamma$)	Returns the value γ if α is the immediate successor of β in sequence seq, 0 otherwise.
pred(α, seq)	Returns the interval immediately preceding the interval α in the sequence seq.
startOf(α)	Returns an expression representing the start time of interval α.
endOf(α)	Returns an expression representing the end time of interval α.
stepAtStart(α, h_{min}, h_{max})	Function in time t which returns a value between h_{min} and h_{max}, starting from time $t =$ **startOf**(α). The function returns 0 when t is absent, or before the start of α.
reservoirConstr(. . .)	Custom global constraint, see Section 2.1 for details.

Table 1: Description of CP constraints. All of these constraints, except the custom reservoirConstr, are available in IBM's CP Optimizer by default.

2 Constraint Programming model

To solve the BRP-TW during the ACP Competition, we implemented a Constraint Programming approach which relies on interval variables (Laborie and Rogerie, 2008; Laborie et al., 2009). An interval variable represents an interval during which an activity can be performed. More specifically, an interval variable α is a variable whose domain $dom(\alpha)$ is a subset of $\{\perp\} \cup \{[s, e) | s, e \in \mathbb{Z}, s \leq e\}$. An interval variable is fixed if its domain is reduced to a singleton, i.e. if α denotes a fixed interval variable:

- $\alpha = \perp$ if the interval is absent; the activity is not scheduled
- $\alpha = [s, e)$ if the interval is present

An absent interval variable is ignored by any constraint or expression it is involved in. Such a constraint or expression would treat the absent interval variable as if it had never been specified to the constraint. Each interval variable α has a start time $startOf(\alpha)$, an end time $endOf(\alpha)$, and a duration $dur(\alpha)$. Whenever an interval is present, it must hold that $endOf(\alpha) - startOf(\alpha) \geq dur(\alpha)$. As shorthand notation, an interval variable α is defined as a tuple: $\alpha = \{r, d, t, o\}$, specifying respectively the earliest start time of the interval, latest end time, minimum duration, and whether the interval is optional or obligatory. The constraints used in our model are summarized in table 1.
To model the BRP-TW as a CP Problem (Algorithm 1), three sets of interval variables are used: an obligatory interval s representing the start of the schedule, obligatory intervals t^k for all $k \in K$ representing the end of a schedule for vehicle k, and finally optional intervals v_i^k for all stations $i \in V \setminus \{0\}$, $k \in K$ representing the servicing of station i by vehicle k. Interval variables t^k are defined on the interval $[0, H]$, where H is some valid upper bound on the time horizon of the schedule.

Algorithm 1: CP model for BRP-TW.

`Variable definitions:`

1 $s = \{0, 0, 0, oblig.\}$
2 $t^k = \{0, H, 0, oblig.\}$ $\quad \forall k \in K$
3 $v_i^k = \{a_i, b_i, 0, opt.\}$ $\quad \forall i \in V \setminus \{0\}, k \in K$
4 $obj \in \{0, \infty\}$

`Objective:`

5 `Min` obj

`Constraints:`

6 **forall** $i \in V \setminus \{0\}$
7 $\quad \sum_{k \in K} \text{presenceOf}(v_i^k) = 1$
8 **forall** $k \in K$
9 $\quad obj \mathrel{+}= c^k \sum_{i \in V \setminus \{0\}} \left[\text{isSucc}(v_i^k, s, seq, t_{0i}) + \sum_{j \in V \setminus \{0\}} \text{isSucc}(v_j^k, v_i^k, seq, t_{ij}) + \text{isSucc}(t^k, v_i^k, seq, t_{i,0})\right]$
10 $\quad seq = \text{noOverlap}(\{s, \bigcup_{i \in V \setminus \{0\}} v_i^k, t^k\}, t_{ij})$
11 $\quad \text{first}(s, seq)$
12 $\quad \text{last}(t^k, seq)$
13 $\quad cumulFunc^k = \text{stepAtStart}(s, 0, q^k) + \sum_{i \in V \setminus \{0\}} \text{stepAtStart}(v_i^k, d_i, d_i)$
14 $\quad 0 \leq cumulFunc^k \leq q^k$
15 $\quad \text{reservoirConstr}(\{s, \bigcup_{i \in V \setminus \{0\}} v_i^k, t^k\}, seq, cumulFunc, q^k)$
16 $\quad$ **forall** $i \in V \setminus \{0\} \cup \{t^k\}$
17 $\quad\quad \text{startOf}(v_i^k) = \texttt{Max}\ \{a_i, \text{endOf}(\text{pred}(v_i^k, seq)) + t_{\text{pred}(v_i^k, seq), i}\}$

The constraint on line 7 ensures that each station is serviced exactly once by a single vehicle. The objective function is defined on line 9. The constraints on lines 10-12 sequence the visits to the stations for each vehicle, thereby ensuring that s and t^k are always resp. the first and last interval in the sequence. Constraints 12-13 are the capacity constraints which ensure that the inventory of the vehicle is always between 0 (empty) and q^k. The constraint on line 15 is a custom constraint, described in the next section. Its purpose is to provide a tighter coupling between the resource and time constraints. Finally, the (redundant) constraint on line 17 strengthens the model by coupling the start and end time of each interval in the schedule for each vehicle.

2.1 Reservoir balancing constraint

The inventory of each vehicle in BRP-TW is known as a *reservoir resource*. Each reservoir has a minimum capacity of 0 and a maximum capacity of q^k, for all $k \in K$. Even though the constraints on lines 13-14 (Algorithm 1) are sufficient to manage the reservoir levels, they provide very little propagation because they do not consolidate at what *time*, how much of a particular resource is required to guarantee a feasible schedule. Building upon the work of Laborie (2003), we present a new reservoir constraint which connects the reservoir capacity constraints with the scheduling constraints.

Given are a set of resource events S which affect the capacity of a reservoir, and a precedence graph which provides a (partial) ordering of these events. The basic idea behind the reservoir constraint is to compute, for each event $x \in S$ in the precedence graph a lower and an upper bound on the reservoir level just before and just after x, and to compare these levels to the maximum and minimum capacities of the reservoir (Laborie, 2003). We extent upon the work by Laborie (2003) by incorporating *optional* resource events (i.e. the presence status of the events is not necessarily fixed to present or absent) into the constraint, and by applying it to BRP-TW.

Following the notation used by Laborie (2003), let P resp. C be the set of production, resp. consumption events, and let $S = P \cup C$. In case of BRP-TW, $P = \{i \in V | d_i > 0\}$, $C = \{i \in V | d_i < 0\}$. Furthermore, let $B(x) \subset S$ be the events that have to be completed *strictly before* the *start of* event $x \in S$, $U(x) \subset S$ the set of events who's precedence relation with respect to x is undecided, i.e. an event $y \in U(x)$ can occur either before or after x. The relative change of the reservoir resource due to an event $x \in S$ is denoted by $q(x)$. $q_{min}(x)$, $q_{max}(x)$ are respectively the smallest and largest values in the domain

of $q(x)$ [2]. In case of BRP-TW, the relative change of the vehicle's inventory is modeled through the **stepAtStart**$(\alpha, h_{min}, h_{max})$ constraints (line 13, Algorithm 1), which implement a resource event x at the start of interval α, with $q_{min}(x) = h_{min}$, and $q_{max}(x) = h_{max}$. Finally, we can define the sets O, $\overline{O}$, $\widetilde{O}$, containing resp. the events which are present, absent, and the events who's presence status is undetermined. Obviously, the following relation holds: $O \cap \overline{O} = O \cap \widetilde{O} = \emptyset$. For the sake of generality, we assume that 0 and Q are fixed, finite bounds on the reservoir resource.
For each event $x \in S$, we can now define an upper bound $L^{<}_{max}(x)$, and a lower bound $L^{<}_{min}(x)$ on the resource level just before x as follows:

$$L^{<}_{max}(x) = \sum_{y \in P \cap (B(x) \cup U(x)) \setminus \overline{O}} q_{max}(y) + \sum_{y \in C \cap B(x) \cap O} q_{max}(y) \tag{1}$$

$$L^{<}_{min}(x) = \sum_{y \in C \cap (B(x) \cup U(x)) \setminus \overline{O}} q_{min}(y) + \sum_{y \in P \cap B(x) \cap O} q_{min}(y) \tag{2}$$

2.1.1 Dead ends and presence relations

Using the definitions from the previous section, a number of conditions can be specified under which the propagator of the Reservoir Balancing Constraint fails, or under which additional presence relations can be deduced. The propagator fails if $L^{<}_{max}(x) < 0$ or $L^{<}_{min}(x) > Q$, resulting in a backtrack. If $x \in O$, the propagator also fails if $L^{<}_{max}(x) + q_{max}(x) < 0$ or $L^{<}_{min}(x) + q_{min}(x) > Q$. Finally, if $x \in \widetilde{O}$, we can post a constraint stating that x must be set to absent if $L^{<}_{max}(x) + q_{max}(x) < 0$ or $L^{<}_{min}(x) + q_{min}(x) > Q$, because the presence of event x would instantly result in a fail of the propagator.

2.1.2 Discovering new precedence relations

Let $\Pi^{<}_{min}(x) = -\sum_{y \in B(x) \cap ((P \setminus \overline{O}) \cup (C \cap O))} q_{max}(y)$ be the smallest amount of resources that has to be produced *before* event x commences. Intuitively, if $\Pi^{<}_{min}(x)$ yields a positive value, then, a number of production events must be scheduled before x, thereby producing at least $\Pi^{<}_{min}(x)$ resources. Let $P(x) = U(x) \cap P \setminus \overline{O}$ be the production events which can be scheduled either before or after event x. If there exists a $y \in P(x)$ such that:

$$\sum_{z \in P(x) \cap (B(y) \cup U(y))} q_{max}(z) < \Pi^{<}_{min}(x) \tag{3}$$

then a constraint can be posted, stating that y must precede x. Intuitively, Condition (3) reads: take a production event $y \in P(x)$, which can be scheduled before or after x, and iterate over all remaining production events in $P(x)$ that can potentially precede y. If these events cannot produce at least $\Pi^{<}_{min}(x)$ resources, then naturally y must precede x. Observe that when $x \in O \cap C$, the right hand side of Condition (3) can be strengthened to: $\Pi^{<}_{min}(x) + q_{max}(x)$. Furthermore, when $x \in \widetilde{O}$, an additional constraint can be posted, stating that presenceOf$(x) \implies$ presenceOf(y).
Following a similar line of reasoning, define $\Pi^{<}_{max}(x) = \sum_{y \in B(x) \cap ((C \setminus \overline{O}) \cup (P \cap O))} q_{min}(y) - Q$ as the least amount of resources that has to be consumed before x commences, and let $C(x) = U(x) \cap C \setminus \overline{O}$. If there exists a $y \in C(x)$ such that:

$$-\sum_{z \in C(x) \cap (B(y) \cup U(y))} q_{min}(z) < \Pi^{<}_{max}(x) \tag{4}$$

then a constraint can be posted, stating that y must precede x. When $x \in O \cap P$, the right hand side of Condition (4) can be strengthened to: $\Pi^{<}_{min}(x) + q_{min}(x)$. Furthermore, when $x \in \widetilde{O}$, an additional constraint can be posted, stating that presenceOf$(x) \implies$ presenceOf(y).

3 Mixed Integer Programming models

To compare the performance of our CP model in Section 4, we use a MIP model which essentially combines the models for a Pickup-and-Delivery problem with Time Windows (Ropke et al., 2007) and a model for a related Bicycle Rebalancing Problem (Dell'Amico et al., 2014). In contrast to this work, Dell'Amico et al. (2014) do not consider time windows, and they assume a homogeneous fleet. Consequently, their

[2] Observe that when x is a consumption event, i.e. $q(x) < 0$, then by definition, $q_{max}(x)$ corresponds to the largest (least negative) value in the domain of $q(x)$.

problem is closer related to the Traveling Salesman Problem with Pickup and Deliveries as presented by Hernndez-Prez and Salazar-Gonzlez (2004).
For each arc $(i,j) \in A$, let binary variable x_{ij}^k denote whether vehicle $k \in K$ travels from station i to j. For each station $i \in V$, let variable C_i denote the time at which servicing station i completes. Finally, for each arc $(i,j) \in A$, variable f_{ij} counts the number of bikes in the vehicle that traverses arc (i,j). Obviously, $f_{ij} = 0$ if $x_{ij}^k = 0$ for all $k \in K$.

MILP :

$$\min \sum_{k \in K} c_k \sum_{(i,j) \in A} t_{ij} x_{ij}^k \tag{5}$$

$$\text{s.t.} \sum_{j \in \delta^+(0)} x_{0j}^k \leq 1 \qquad \forall k \in K \tag{6}$$

$$\sum_{k \in K} \sum_{j \in \delta^+(i)} x_{ij}^k = 1 \qquad \forall i \in V \setminus \{0\} \tag{7}$$

$$\sum_{i \in \delta^-(j)} x_{ij}^k = \sum_{i \in \delta^+(j)} x_{ji}^k \qquad \forall j \in V \setminus \{0\}, k \in K \tag{8}$$

$$C_j \geq C_i + t_{ij} - M_{ij}(1 - x_{ij}^k) \qquad \forall (i,j) \in A : j \neq 0, k \in K \tag{9}$$

$$\sum_{j \in \delta^+(i)} f_{ij} - \sum_{j \in \delta^-(i)} f_{ji} = d_i \qquad \forall i \in V \setminus \{0\} \tag{10}$$

$$\sum_{k \in K} \max\{0, d_i, -d_j\} x_{ij}^k \leq f_{ij} \leq \sum_{k \in K} \min\{q^k, q^k + d_i, q^k - d_j\} x_{ij}^k \qquad \forall (i,j) \in A : j \neq 0 \tag{11}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, k \in K \tag{12}$$

$$a_i \leq C_i \leq b_i \qquad \forall i \in V \setminus \{0\} \tag{13}$$

M_{ij} is a constant, defined as: $M_{ij} = \max\{0, b_i + t_{ij} - a_j\}$. The objective function (5) minimizes the total distance traveled by each vehicle, weighted by the vehicle's cost. Constraints (6) ensures that a vehicle either stays at the depot (and hence is not used), or that it leaves the depot to service one or more stations. Constraints (7)-(8) ensure that each station is serviced exactly once. Constraints (9), (13) ensure that each station is only serviced within its respected time window. In addition, Constraints (9) serve as subtour elimination constraints as they are a generalization of the Miller-Tucker-Zemlin subtour elimination constraints. Constraints (10)-(11) ensure that the desired amount of bikes are added or removed from each station, while simultaneously enforcing vehicle capacities. In particular, Constraints (11) ensure that:

- the number of bikes in a vehicle $k \in K$ is always between 0 and q^k.
- if a vehicle collects bikes at node i, i.e. $d_i > 0$ then $f_{ij} \geq d_i$ after leaving node i, for some $j \in V$.
- if a vehicle delivers bikes at node j, i.e. $d_j < 0$ then the vehicle must have sufficient bikes in its inventory before reaching station i, i.e. $f_{ij} \geq -d_j$.
- and vice versa for the other direction.

As shown by Desrochers and Laporte (1991), the bounds on the completion time variables C_i, $i \in V$ may be strengthened:

$$C_i \geq a_i + \sum_{k \in K} \sum_{j \in \delta^-(i)} \max\{0, a_j + t_{ji} - a_i\} x_{ji}^k \qquad \forall i \in V \setminus \{0\} \tag{14}$$

$$C_i \leq b_i - \sum_{k \in K} \sum_{j \in \delta^+(i) \setminus \{0\}} \max\{0, b_i - b_j - t_{ij}\} x_{ij}^k \qquad \forall i \in V \setminus \{0\} \tag{15}$$

Similarly, if for a given pair $i, j \in V$ both arcs (i,j) and (j,i) are contained in A then Constraint (9) may be replaced by a stronger equivalent:

$$C_j \geq C_i + t_{ij} - M_{ij}(1 - x_{ij}^k) + (M_{ij} - t_{ij} - \max\{t_{ji}, a_i - b_j\}) x_{ji} \qquad \forall (i,j) \in A : j \neq 0, k \in K \tag{16}$$

3.1 Valid inequalities

The family of clique inequalities described by Dell'Amico et al. (2014) can be modified to our problem. Let $S(i,j,\overline{q}) = \{h \in \delta^+(j), h \neq i : |q_i + q_j + q_h| > \overline{q}\}$ for a given pair of nodes $i,j \in V$, $j \neq 0$, $(i,j) \in A$ and a capacity $\overline{q} \geq 0$. Similarly, let $T(i,j,\overline{q}) = \{h \in \delta^-(i), h \neq j : |q_i + q_j + q_h| > \overline{q}\}$ for a given pair of nodes $i,j \in V$, $i \neq 0$, $(i,j) \in A$ and a capacity $\overline{q}$. Finally, let $\overline{Q} = \{q^k, k \in K\}$ be the set of different vehicle capacities. Then the following inequalities are valid for BRP-TW:

$$\sum_{\substack{k \in K: \\ q^k \leq \overline{q}}} \left(x_{ij}^k + \sum_{h \in S(i,j,\overline{q})} x_{jh}^k \right) \leq 1 \qquad \forall (i,j) \in A : j \neq 0, \overline{q} \in \overline{Q}, S(i,j,\overline{q}) \neq \emptyset \tag{17}$$

$$\sum_{\substack{k \in K: \\ q^k \leq \overline{q}}} \left(\sum_{h \in T(i,j,\overline{q})} x_{hi}^k + x_{ij}^k \right) \leq 1 \qquad \forall (i,j) \in A : i \neq 0, \overline{q} \in \overline{Q}, T(i,j,\overline{q}) \neq \emptyset \tag{18}$$

The validity of inequalities (17) follows from the fact that (1) each station must be visited by exactly one vehicle and (2) a vehicle $k \in K : q^k \leq \overline{q}$ does not have sufficient capacity to serve all three stations i, j, $h \in S(i,j,\overline{q})$. Similar for inequalities (18). Some of the inequalities (17), (18) are dominated by other inequalities in the same family, and can therefore be removed.
Separation of the clique inequalities (17), (18) is performed through complete enumeration. For every inequality in (17), (18) we evaluate the left hand side for a given solution $\overline{x}$. If the left hand side is strictly larger than one, the corresponding inequality is violated and is added to the problem.
Experiments were conducted with additional families of valid inequalities, such as the Fractional and Rounded capacity inequalities (Hernndez-Prez and Salazar-Gonzlez, 2004), but they did not have a positive impact on the performance of the MIP model. Hence they are omitted in this discussion.

4 Computational Results and Discussion

The CP and MIP models are implemented in resp. ILOG CPLEX and CP Optimizer (version 12.6.2), and executed with the default search parameters on an Intel i7-4790 with 8 cores. During the competition, 12 problem instances were provided, out of which we could solve 4 to optimality. For two instances, no solution could be found; the results for the remaining 10 instances are reported in Table 2. Table 2 shows for each instance the number of stations, and the number of vehicles. Furthermore, for the MIP approach, Table 2 provides the best upper (UB) and lower bound (LB) obtained after 30 minutes of computation time, the gap between these bounds and the actual computation time. Finally, for the CP approach, we show the upper bound, the gap between this upper bound and the MIP lower bound, and the actual computation time. Whenever no feasible solution could be found within the allotted time, the optimality gap is assumed to be 100%. As can be observed from Table 2, for the smaller instances, MIP outperforms CP: it solves these instances to optimality in a fraction of the computation time required by CP. However, for the larger instances, MIP is unable to find any feasible solutions. A similar trend was observed for alternative MIP based solution approaches used by competitors during the competition.
Part of the decision problem is determining how many bikes each vehicle carries when it leaves the depot. This aspect makes the problem significantly harder to solve for a CP based approach. In particular, when we fixed the number of bikes in the initial inventory for each vehicle, the CP method was able to obtain significantly better results.
Figure 1 analyses the impact of the Reservoir Balancing constraint presented in Section 2.1. We compare three situations: (1) the CP model without the reservoir constraint, (2) the CP model with the reservoir constraint but without generating precedence relations (blue bars), (3) the CP model with the complete reservoir constraint (orange bars). Figure 1 plots the difference between the the optimality gap for cases (2) and (3) and the optimality gap obtained using the model without the Reservoir Constraint (case (1)). For example, for instance 5, we obtain optimality gaps of resp. 40.68%, 39.77%, 37.10%, resulting in an absolute improvement of the optimality gap by resp. 39.77-40.58=-0.91% (blue) and 37.10-40.68=-3.58% (orange). As can be observed from Figure 1, for each instance, the Reservoir Balancing constraint improves the objective (orange bars); it never has a negative impact on the solution. Whenever no precedence relations are deduced, the effectiveness of the Reservoir Balancing constraint is significantly reduced, and, due to its computational overhead, may have a negative impact on the solution quality. The latter may be observed from instances 6, 7, and 8.
In future work, the Reservoir Rebalancing Constraint may be extended by adding conditions on the earliest start and latest end times of an interval, based on the availability of a resource (see Laborie

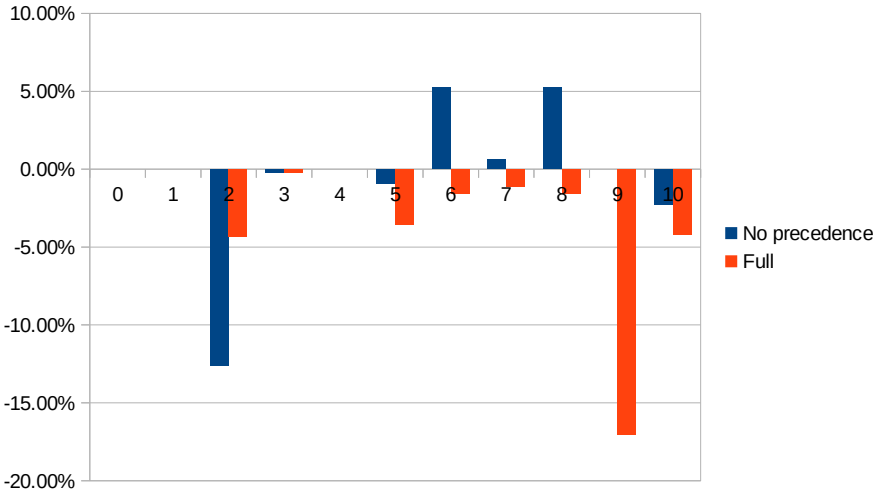

Figure 1: Impact of reservoir constraint. Smaller percentage represents a bigger reduction in the optimality gap.

(2003) for details). In addition, specific search procedures for the BRP-TW may be developed, based on for example the station's demands. We conducted some preliminary tests with a search procedure which fixed the presence status of the v_i^k variables one by one, starting with stations with a large demand $|d_i|$, but this approach did not yield a notable improvement in the solution procedure. Finally, one may wish to incorporate constraints to improve the propagation of the sequence constraints (see Bergman et al. (2015) for details).

Table 2: MIP vs CP.

			MIP				CP		
inst	$\|V\|$	$\|K\|$	LB	UB	t(s)	gap	UB	t(s)	gap
0	5	1	137992	137992	0	0.00%	137992	0	0.00%
1	7	2	432270	432270	0	0.00%	432270	1	0.00%
2	28	6	424629	424629	193	0.00%	518620	1800	18.12%
3	16	4	770988	770988	1	0.00%	770988	1800	0.00%
4	14	4	317053	317053	1	0.00%	317053	134	0.00%
5	27	12	278733.93	453197	1800	38.50%	443172	1800	37.10%
6	42	10	353770.92	-	1800	100%	895957	1800	60.51%
7	81	10	488855.4	-	1800	100%	2204450	1800	77.82%
8	42	10	353770.92	-	1800	100%	895957	1800	60.51%
9	151	20	570806.23	-	1800	100%	3348640	1800	82.95%
10	50	6	107534.84	-	1800	100%	255900	1800	57.98%

Acknowledgement

I would like to thank Philippe Laborie (IBM) for his many helpful suggestions and comments regarding the implementation of the Reservoir Rebalancing constraint.

References

D. Bergman, A. Cire, and W.-J. van Hoeve, "Lagrangian bounds from decision diagrams," *Constraints*, vol. 20, no. 3, pp. 346–361, 2015.

M. Dell'Amico, E. Hadjicostantinou, M. Iori, and S. Novellani, "The bike sharing rebalancing problem: Mathematical formulations and benchmark instances," *Omega*, vol. 45, pp. 7 – 19, 2014.

M. Desrochers and G. Laporte, "Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints," *Operations Research Letters*, vol. 10, no. 1, pp. 27 – 36, 1991.

H. Hernndez-Prez and J.-J. Salazar-Gonzlez, "A branch-and-cut algorithm for a traveling salesman problem with pickup and delivery," *Discrete Applied Mathematics*, vol. 145, no. 1, pp. 126 – 139, 2004.

P. Laborie, "Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results," *Artificial Intelligence*, vol. 143, no. 2, pp. 151 – 188, 2003.

P. Laborie and J. Rogerie, "Reasoning with conditional time-intervals," in *FLAIRS Conference*, 2008, pp. 555–560.

P. Laborie, J. Rogerie, P. Shaw, and P. Vilím, "Reasoning with conditional time-intervals. part ii: An algebraical model for resources," in *FLAIRS Conference*, 2009.

S. Ropke, J.-F. Cordeau, and G. Laporte, "Models and branch-and-cut algorithms for pickup and delivery problems with time windows," *Networks*, vol. 49, no. 4, pp. 258–272, Jul. 2007.