Adapting the *ng*-path Relaxation for Bike Balancing Problems

Christian Tilk*,[a]

[a] *Chair of Logistics Management, Johannes Gutenberg University Mainz, Jakob-Welder-Weg 9, D-55128 Mainz, Germany.*

Abstract

This paper deals with the multiple vehicle balancing problem, a static bike relocation problem that deploys a fleet of vehicles to redistribute shared bicycles. To solve the problem to optimality, we present a branch-price-and-cut algorithm for which we adapt several state-of-the-art components. Moreover, a new path relaxation for the pricing problem is introduced that relaxes the constraints on the maximum number of bikes to move at each station in a similar fashion as elementary can be relaxed in standard vehicle routing problems. Computational results show that our algorithm outperforms the former state-of-the-art.

Keywords: Branch-and-Price, Bike Balancing, Path Relaxation, Split-pick-up, Split-delivery

1. Introduction

Bike sharing systems have been rapidly developed and adopted world wide since the middle of the 2000s. According to the website `bikesharingworldmap.com`, more than 2000 bike sharing systems are operating worldwide. Bullock *et al.* (2017) have shown how bike sharing can supply significant economic returns for the urban economy. One central problem faced by shared mobility systems operators is to maintain an adequate number of vehicles in every station. Indeed, too large numbers can impede the return of vehicles whereas too small numbers may translate into lost demand (Laporte *et al.*, 2018).

The paper at hand deals with the *multiple vehicle balancing problem (MVBP)*. It is a static bike relocation problem that deploys a fleet of vehicles to redistribute shared bicycles during night time when customer demand is negligible (Shui and Szeto, 2020). Recently, Casazza *et al.* (2021) proposed a branch-price-and-cut (BPC) algorithm based on the decomposition of routes into simpler substructures called clusters. For an overview on problem-specific and related literature, the reader is referred to (Casazza *et al.*, 2021).

We present an exact BPC algorithm for the MVBP that is based on the column-generation part of the matheuristic presented in (Casazza *et al.*, 2018). In particular, we introduce a new path relaxation for the pricing problem that relaxes the constraints on the maximum number of bikes to move at each station in a similar fashion as the *ng*-path relaxation (Baldacci *et al.*, 2011) relaxes elementary. Additionally, several known state-of-the-art components that are frequently used in BPC algorithms for vehicle routing problems (VRPs) are adapted. In a computation study, we evaluate the impact of the new path relaxation and show that our BPC outperforms the former state-of-the-art algorithm.

2. Problem Description and Branch-Price-and-Cut

A fleet of F vehicles with a capacity of Q bikes is initially located at the depot vertex 0 and must end their journey at the end depot $n+1$. Let N be the set of all stations that needs to be balanced. Each station has a positive or negative demand $d_i \in \mathbb{Z}$ representing the number of bikes that needs to be delivered or picked-up. It is not allowed to temporarily store a bike at a station, i.e., each station $i \in N$ is either a

*Corresponding author.
Email address: `tilk@uni-mainz.de` (Christian Tilk)

pick-up station ($d_i > 0$) or a delivery station ($d_i < 0$). Hence, we define $N^+ := \{i \in N : d_i > 0\}$ to be the set of all pick-up stations and $N^- := \{i \in N : d_i < 0\}$ to be the set of all delivery stations. The pick-up or delivery of a single bike at a station i is referred to as performing a *task* at station i. Moreover, we assume w.l.o.g. that (i) the absolute demand at each station is smaller than the vehicle capacity, and (ii) the overall demand allow a perfect balancing, i.e., $\sum_{i \in N} d_i = 0$.

The MVBP can be formalized with the help of a directed graph $G = (V, A)$ with node set $V := \{0, n+1\} \cup N$ and arc set $A := \{(i, j) : i \neq j, i \neq n+1, \text{ and } j \neq 0\}$. Each arc $(i, j) \in A$ has a positive travel cost c_{ij} for which we assume that the triangle inequality holds. The goal is to find at most F feasible vehicle routes such that the demand of all stations is satisfied while minimizing travel cost. A vehicle route is a path $P = (0 = i_0, i_1, \ldots, i_m, i_{m+1} = n+1)$ in G together with a task pattern $T = (k_1, \ldots, k_m)$ where k_j specifies the amount of tasks performed at station i_j for all $j = 1, \ldots, m$. A route is feasible if it visits at most S stations and the vehicle capacity is never exceeded, i.e., if it holds

$$m \leq S \text{ and } 0 \leq \sum_{\substack{j \leq l \\ i_j \in N^+}} k_j - \sum_{\substack{j \leq l \\ i_j \in N^-}} k_j \leq Q \quad \text{for all } l = 1, \ldots, m.$$

Let Ω be the set of all feasible routes, for each route $r \in \Omega$, α_{ir} denotes the number of tasks performed at station i and c_r its travel cost. The path-based formulation of the MVBP introduced by Casazza *et al.* (2018) reads as follows:

$$\min \quad \sum_{r \in \Omega} c_r \lambda_r \tag{1a}$$

$$\text{subject to} \quad \sum_{r \in \Omega} \alpha_{ir} \lambda_r = |d_i| \qquad \forall i \in N \tag{1b}$$

$$\sum_{r \in \Omega} \lambda_r \leq F \tag{1c}$$

$$\lambda_r \in \{0, 1\} \qquad r \in \Omega \tag{1d}$$

The objective function (1a) minimizes the overall travel cost. Constraints (1b) enforces that the appropriate number of tasks is performed at each station. The size of the fleet is limited by constraint (1c). Finally, constraints (1d) restrict routing variables to be binary. Note that Casazza *et al.* (2018) have shown that this restriction is valid since it is assumed that $|d_i| \leq Q$ for all $i \in N$. In the following, we call the linear relaxation of Formulation 1 in which the set of all feasible routes is replaced by a subset $\bar{\Omega}$ the restricted master program (RMP). A column-generation algorithm (Desaulniers *et al.*, 2005) is employed to solve the RMP, it alternates between the optimization of the RMP and the solution of the pricing subproblem that either generates new negative reduced-cost routes to be added to $\bar{\Omega}$ or proves that none exist. In the latter case, the column-generation process terminates with a solution to the linear relaxation of the extensive formulation and branching may be required to obtain an integer solution to the MVBP.

2.1. Pricing Problem

Let π_i and μ be the dual prices of constraints (1b) and (1c). The pricing problem asks for a route r with negative reduced cost $\bar{c}_r = c_r - \mu - \sum_{i \in N} \alpha_{ir} \pi_i$. This can be modelled as a *shortest-path problem with resource constraints* (SPPRC) and solved with a labeling algorithm.

A partial forward path $P = (0, \ldots, i)$ starts at the depot 0 and ends at some vertex $i \in V$. The associated forward label $L_i = (i, T_i^{cost}, T_i^{load}, T_i^{stops}, (\delta_i^m)^{m \in N})$ comprises the following attributes:

- i: the last visited vertex
- T_i^{cost}: the reduced cost of the path
- T_i^{load}: the number of loaded bikes
- T_i^{stops}: the number of visited stations
- $(\delta_i^m)^{m \in N}$: a vector containing for each station $\ell \in N$ the number δ_i^ℓ of tasks performed by the route

L_i can be extended to all vertices $j \in N$ with the help of resource extension functions (REFs) f_{ij}^k for all $k = 1, \ldots, d_j$ or to vertex $n+1$ with $f_{i,n+1}^0$. The parameter k denotes the number of tasks performed at station j. The resulting label $L_j = (j, T_j^{cost}, T_j^{load}, T_j^{stops}, (\delta_j^m)^{m \in N})$ can be computed as:

$$T_j^{cost} := \begin{cases} T_i^{cost} + c_{ij} - k\pi_j & \text{if } j \in N \\ T_i^{cost} + c_{ij} - \mu & \text{otherwise} \end{cases}$$

$$T_j^{load} := \begin{cases} T_i^{load} + k & \text{if } j \in N^+ \\ T_i^{load} - k & \text{otherwise} \end{cases}$$

$$T_j^{stops} := \begin{cases} T_i^{stops} + 1 & \text{if } j \in N \\ T_i^{stops} & \text{otherwise} \end{cases}$$

$$\delta_j^\ell := \begin{cases} \delta_i^\ell + k & \text{if } j = l \\ \delta_i^\ell & \text{otherwise} \end{cases}$$

The extension is feasible if (i) $0 \leq T_j^{load} \leq Q$, (ii) $T_j^{stops} \leq S$, (iii) $\delta_j^\ell \leq d_\ell$ for all $\ell \in N$, and (iv) $T_j^{load} = 0$ if $j = n+1$. The initial label is given by $L_0 := (0, 0, 0, 0, \mathbf{0})$. To avoid enumerating all paths, we apply the following dominance rule:

Rule 1. *Label* $L_1 = (i, T_1^{cost}, T_1^{load}, T_1^{stops}, (\delta_1^m)^{m \in N})$ *dominates* $L_2 = (i, T_2^{cost}, T_2^{load}, T_2^{stops}, (\delta_2^m)^{m \in N})$ *if* $T_1^{cost} \leq T_2^{cost}$, $T_1^{load} = T_2^{load}$, $T_1^{stops} \leq T_2^{stops}$, *and* $(\delta_1^i) \leq (\delta_2^i)$ *for all* $i \in N$.

The main difficulty of the resulting labeling algorithm relies in the handling of the resources $(\delta_i^m)^{m \in N}$. Instead of using an exact labeling algorithm, it is quite common to solve SPPRC pricing problems by relaxing the elementary requirement, e.g., using the *ng*-path relaxation (Baldacci *et al.*, 2011). Note that routes in the MVBP do not need to be elementary in terms of visited stations, we therefore need another kind of relaxation. Casazza *et al.* (2018) propose a relaxation to compute completion bounds in which the number of tasks performed at each station is ignored. As a result, routes can perform more tasks at a station than are available there. Inspired by this idea, we present an adapted version of the *ng*-path relaxation called *limited task memory (LTM)* relaxation for the MVBP. The idea of LTM is to allow routes to contain some cycles that exceed the number of available tasks at certain stations. In particular, a label will reset the information about the number of tasks performed at some other stations, thereby drastically strengthening the dominance.

A LTM relaxation is parametrized by neighborhoods N_i for each vertex $i \in V$ which contain i and its sz closest stations. If a label $L_i = (i, T_i^{cost}, T_i^{load}, T_i^{stops}, (\delta_i^m)^{m \in N})$ is extended with REF f_{ij}^k, we only alter the update of the resources $(\delta_i^m)^{m \in N}$ as follows:

$$\delta_j^\ell := \begin{cases} \delta_i^\ell + k & \text{if } j = \ell \\ 0 & \text{if } \ell \notin N_j \\ \delta_i^\ell & \text{otherwise} \end{cases}$$

Figure 1 shows an example of a feasible path in the LTM relaxation that is not feasible for the MVBP. The example depicts the path $P = (0, 1, 2, 3, 1, 4, 2, n+1)$, the demand and the neighborhood of each vertex are depicted above. On each arc (i, j) the value k of the number of tasks performed at j is depicted. The values of the resources T^{load}, δ^1, δ^2, δ^3, and δ^4 at each vertex are depicted below the path. The path contains two cycles, $1-2-3-1$ and $2-3-1-4-2$. While the former is feasible for the MVBP, the latter is only feasible in the LTM relaxation: The second visit of vertex 1 is feasible for the MVBP since the amount of tasks performed at vertex 1 is six (four on arc (0,1) and two on arc (3,1)) and hence smaller than $|d_1| = 7$. When vertex 2 is visited for the second time, the overall amount of tasks performed at station 2 would be six, four on arc (1,2) and two on arc (4,2), and thereby infeasible for the MVBP since it is greater than $|d_2| = 5$.

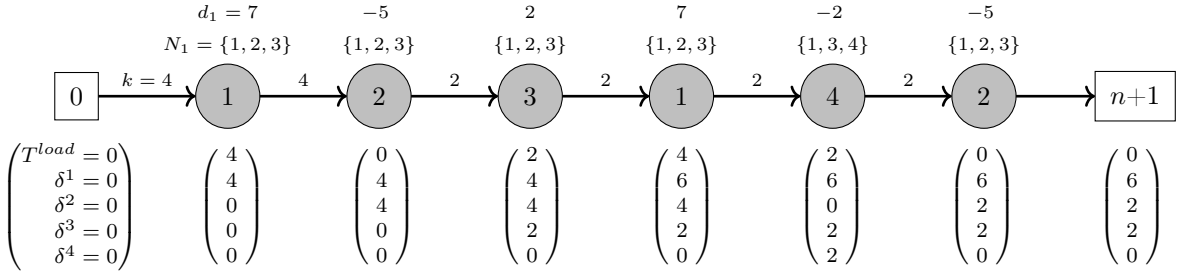

Figure 1: Example of a forward path that is feasible in the LTM relaxation but infeasible for the MVBP.

However, in the LTM relaxation the value δ^2 was reset to zero at station 4 because station 2 is not included in the neighborhood N_4, and hence, the path is feasible. Note that the corresponding route-variable of such a LTM-path can only have a fractional value in the RMP and will therefore be eliminated by cutting and branching (see Section 2.2). Next, we summarize acceleration techniques that we use for solving the pricing problem.

Bidirectional labeling. A partial backward path $P = (j, \dots, n+1)$ starts at some vertex $j \in V$ and ends at the depot $n+1$. The associated backward label $L_j^{bw} = (j, T_j^{cost}, T_j^{load}, T_j^{stops}, (\delta_j^m)^{m \in N})$ comprises the same attributes as the forward label. It can be extended backwards to all vertices $j \in N$ with the help of REFs g_{ij}^k for all $k = 1, \dots, d_i$ or to vertex 0 with $f_{0,j}^0$. The parameter k gives the number of tasks performed at station i. The resulting label $L_i = (i, T_i^{cost}, T_i^{load}, T_i^{stops}, (\delta_i^m)^{m \in N})$ can be computed as:

$$T_i^{cost} := \begin{cases} T_j^{cost} + c_{ij} - k\pi_i & \text{if } i \in N \\ T_i^{cost} + c_{ij} - \mu & \text{otherwise} \end{cases} \tag{2a}$$

$$T_i^{load} := \begin{cases} T_j^{load} + k & \text{if } i \in N^- \\ T_j^{load} - k & \text{otherwise} \end{cases} \tag{2b}$$

$$T_i^{stops} := \begin{cases} T_j^{stops} + 1 & \text{if } i \in N \\ T_j^{stops} & \text{otherwise} \end{cases} \tag{2c}$$

$$\delta_i^\ell := \begin{cases} \delta_j^\ell + k & \text{if } i = \ell \\ 0 & \text{if } \ell \notin N_i \\ \delta_i^\ell & \text{otherwise} \end{cases} \tag{2d}$$

The extension is feasible if (i) $0 \le T_i^{load} \le Q$, (ii) $T_i^{stops} \le S$, (iii) $\delta_i^\ell \le d_\ell$ for all $\ell \in N$, and (iv) $T_i^{load} = 0$ if $i = 0$. The initial label is given by $L_0^{bw} := (0, 0, 0, 0, \mathbf{0})$ and the same dominance rule as in the forward labeling can be applied.

A forward label $L_i^{fw} = (i, T_i^{cost}, T_i^{load}, T_i^{stops}, (\delta_i^m)^{m \in N})$ and a backward label $L_j^{bw} = (j, T_j^{cost}, T_j^{load}, T_j^{stops}$, $(\delta_j^m)^{m \in N})$ can be merged to obtain a feasible route if $T_i^{load} = T_j^{load}$, $T_i^{stops} + T_j^{stops} \le S$, and $\delta_i^\ell + \delta_j^\ell \le d_l$ for all $\in N$. The cost of the resulting path is given by $T_i^{cost} + T_j^{cost} + c_{ij}$.

Bounding. Similar to (Casazza *et al.*, 2018), we compute completion bounds and use them to discard unpromising partial paths in the labeling algorithm. Using a LTM relaxation with neighborhoods $(N_i^1)_{i \in V}$, valid completion bounds for forward label can be computed by solving a backward labeling with another LTM relaxation with neighborhoods $(N_i^2)_{i \in V}$ that fulfill $N_i^2 \subseteq N_i^1$ for all $i \in V$. Let $\mathcal{L}^{bw}$ be the set of all backward label, according to the merge criterion, we can merge the forward label $L_i = (i, T_i^{cost}, T_i^{load}, T_i^{stops}, (\delta_i^m)^{m \in N})$ with all backward labels from the set $\mathcal{L}^{bw}(L_i) := \{L_j \in L^{bw} : T_i^{load} = T_j^{load},\ T_i^{stops} + T_j^{stops} \le S,\ \text{and } \delta_i^\ell + \delta_j^\ell \le d_l \text{ for all } l \in N\}$. Hence, a valid completion bound for L_i can be

computed as $lb(L_i) := \min_{L_j \in \mathcal{L}^{bw}(L_i)} T_j^{cost} + c_{ij}$ and all forward label with $T_i^{cost} + lb(L_i) > 0$ can be discarded. The computation of completion bounds for backward label works analogous. In our computational analysis, we compute the completion bound with a neighborhood size $sz = 0$.

Partial pricing. Instead of solving the pricing problem exactly, we use the following three pricing heuristics hierarchically: In the first heuristic, we strongly reduce the size of the pricing network by considering for each station only the two cheapest ingoing and outgoing arcs to other stations. Additionally, all arcs from and to the depot are kept. The second heuristic uses a heuristic dominance rule ignoring the resources $(\delta_i^m)^{m \in N}$ to drastically increase the number of discarded labels. In the third heuristic, we adapt the *limited discrepancy search* (Feillet *et al.*, 2007) to the MVBP. Therein, REFs f_{ij}^k (g_{ij}^k) are labelled good if $|d_j| = k$ ($|d_i| = k$) and all other REFs are labelled bad. Then the total number of bad arcs that can be traversed in a route is limited by two. Only if all heuristics fail, the pricing problem is solved exactly.

2.2. Cutting and Branching

We strengthen the linear relaxation of Formulation 1 by two classes of valid inequalities. First, we use rounded capacity inequalities (RCIs). Let b_{ijr} be the number of times route r uses arc (i, j), the RCIs for a set $S \subset N$ are given by:

$$\sum_{i \in S} \sum_{j \in N \setminus S} \sum_{r \in \Omega} b_{ijr} \lambda_r \geq \left\lceil \frac{\left| \sum\limits_{i \in S} q_i \right|}{Q} \right\rceil$$

For the separation, we adapt the model presented in (Fukasawa *et al.*, 2005) for the capacitated vehicle routing problem. Second, we use the robust profitable cuts introduced by Casazza *et al.* (2018) for pairs of either pick-up or delivery stations:

$$\sum_{r \in \Omega} (b_{ijr} + b_{jir}) \lambda_r \leq 1 \qquad \text{for all } i, j \in N^+ \text{ or } i, j \in N^-$$

This class of inequalities can be separated by inspection. Note that both classes of inequalities are robust, i.e., they only introduce an additional dual price that can be incorporated in the pricing problem by changing the arc cost c_{ij}. Both classes of valid inequalities are separated at each branch-and-bound node.

To finally obtain integer solution, we use a three-stage branching scheme. Let λ_r^* be the current solution of the RMP. First, we branch on the number of vehicles $F^* := \sum_{r \in \Omega} \lambda_r^*$, creating two branches with $\sum_{r \in \Omega} \lambda_r \geq \lceil F^* \rceil$ and $\sum_{r \in \Omega} \lambda_r \leq \lfloor F^* \rfloor$. Second, we branch on the number of visits to a station, let a_{ir} be the number of visits of route r to station i. If $a_i^* := \sum_{r \in \Omega} a_{ir} \lambda_r^*$ is fractional for some $i \in N$, we choose the station i with value $a_i^* - \lfloor a_i^* \rfloor$ closest to 0.5 to branch on and the two branches $\sum_{r \in \Omega} a_{ir} \lambda_r \geq \lceil a_i^* \rceil$ and $\sum_{r \in \Omega} a_{ir} \lambda_r \leq \lfloor a_i^* \rfloor$ are created. Third, we branch on the flow on arcs if $b_{ij}^* := \sum_{r \in \Omega} b_{ijr} \lambda_r^*$ is fractional for some $(i, j) \in A$. We choose the arc (i, j) with value $b_{ij}^* - \lfloor b_{ij}^* \rfloor$ closest to 0.5 to branch on and the two branches $\sum_{r \in \Omega} b_{ijr} \lambda_r \geq \lceil b_{ij}^* \rceil$ and $\sum_{r \in \Omega} b_{ijr} \lambda_r \leq \lfloor b_{ij}^* \rfloor$ are created. The first, second, and third stage all induce new dual prices that that can be incorporated in the pricing problem by changing the arc cost c_{ij}. Note that integer flows on arcs may not guarantee that the route variables are integer for the MVBP. However, fractional route variables could always be convex combined into feasible integer solutions (see, Desaulniers *et al.* (1998) and Jans (2010)).

To accelerate the solution process, we apply strong branching on stages 2 and 3 of the branching hierarchy. In both stages, we choose the eight most fractional candidates in the current solution and perform a rough evaluation of each candidate by solving the current RMP twice, adding the constraint corresponding to each child node and using only the first two pricing heuristics to generate additional columns. The resulting improvements in the lower bounds are usually overestimated. However, this evaluation strategy is fast and beneficial compared to just choosing the most fractional candidates to branch on. The candidate to branch on is then chosen according to the product rule (Achterberg, 2007). As branch-and-bound node-selection rule, we apply a best-bound-first strategy.

3. Computational Results

In this section, we evaluate the impact of the LTM relaxation and compare our BPC with the state-of-the-art in the literature. We, therefore, use the benchmark set from Casazza *et al.* (2018). It consists of 40 instances with up to 30 stations with demands in $[-10, 10]$, the vehicle capacity is set to 10 and the maximum number of stops to 9. Travel costs are based on Euclidean distances and fulfill the triangle inequality.

The BPC algorithm was implemented in C++ and compiled with MS Visual Studio 2019 into 64-bit single-thread code. The callable library of CPLEX 12.9.0 was used for (re)optimizing the RMPs. All results were obtained using a standard PC with an Intel® Core™ i7-5930K processor clocked at 3.5 GHz and 64 GB RAM running Microsoft Windows 10 Education. The time limit is set to three hours.

Table 1 compares our BPC with the column-generation algorithm of (Casazza *et al.*, 2018) and the BPC of (Casazza *et al.*, 2021). We evaluate our BPC with a labeling algorithm using LTM neighborhood sizes 0 and 6 as well as using the non-relaxed version. For each instance class, it contains the number of stations ($|N|$), the number of instances in the class (#) as well as the number of solved instances and the average solution times for all algorithms. The table shows that the LTM-relaxation with $\beta = 6$ clearly outperforms all other algorithms. It solves all but one instance to optimality, and the average computation time decreases by more than factor three compared to the other algorithms. All in all, 14 new proven optimal solutions are computed. Note that all variants of our BPC are superior to the algorithms presented in the literature.

instance	$\|N\|$	#	Solved instances					Time [sec]				
class			LTM 0	LTM 6	exact	cas18	cas21	LTM 0	LTM 6	exact	cas18	cas21
velib 10	10	20	20	20	20	20	20	5.0	1.5	18.1	605.9	9.9
velib 20	20	10	10	10	9	5	3	297.3	130.2	1415.9	6690.5	7851.3
velib 30	30	10	5	9	3	0	0	6516.7	2071.2	8097.6	10800.0	10800.0
all	-	40	35	39	32	25	23	1706.0	551.1	2387.4	4675.6	4667.8

Table 1: Comparison of the exact labeling algorithm, labeling with different LTM sizes, and (Casazza *et al.*, 2018, 2021).

Next, we evaluate the impact of forbidding split pick-ups and split deliveries. For that purpose, we solve the same set of instance but deleting all REFs f_{ij}^k with $k < |d_j|$ (and g_{ij}^k with $k < |d_i|$) in the labeling algorithm. Note that the resulting problem is much easier and all instances are solved to optimality in a fraction of the time needed to solve the corresponding instance of the MVBP. For all instances that were solved to optimality for both problems, Figure 2 depicts the increase in the optimal objective value when splitting is forbidden. In 32 out of 39 instances, the optimal objective value increases. The average increase is more than five percent with a maximum at around 25 percent.

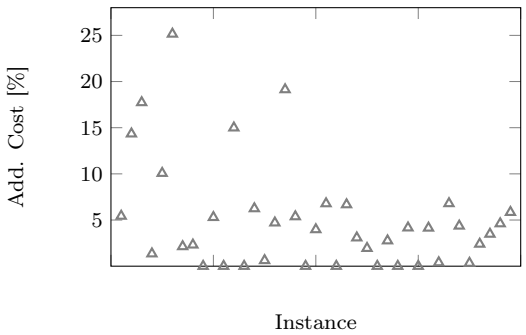

Figure 2: Cost increase in the optimal objective when splitting demands is forbidden.

To access the limitations of our BPC, we tested LTM 6 on the same instances with doubled vehicle capacity $Q = 20$ and different number of stops $S \in \{12, 15, 18\}$. The average computation time for the resulting instance sets is depicted in Figure 3. We can see that a larger vehicle capacity surprisingly just

slightly increases the average computation time to solve an instance. The average computation time increases linear with the number of stops with one exception: Increasing the number of stops further than 15, does slightly decrease the average computation time for the $Q = 20$ instances.

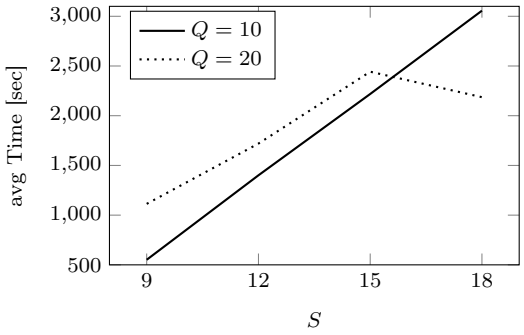

Figure 3: Average computation time when instance parameter S and Q are varied.

4. Conclusion

In this paper, we have presented a branch-price-and-cut algorithm for the multiple vehicle balancing problems. The algorithm adapts several known techniques that were successfully used in BPC algorithms for other vehicle routing problems. Moreover, we have introduced a new path relaxation for solving the pricing problem. Computational results clearly show the benefit of using the new relaxation. On the benchmark set from the literature, the new algorithm can compute all known optimal solutions in a fraction of the time of the former state-of-the-art algorithm. Moreover, 14 instances are solved to proven optimality for the first time. Further computational experiments indicate that the solution time scales linear with the instance parameter number of stops, while the vehicle capacity seems to have less impact on the solution time.

Acknowledgement

This research was funded by the Deutsche Forschungsgemeinschaft (DFG) under grant no. IR 122/9-2.

References

Achterberg, T. (2007). *Constraint Integer Programming*. Ph.D. thesis, Technische Universität Berlin, Fakultät II – Mathematik und Naturwissenschaften, Berlin, Germany.

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

Bullock, C., Brereton, F., and Bailey, S. (2017). The economic contribution of public bike-share to the sustainability and efficient functioning of cities. *Sustainable Cities and Society*, **28**, 76–87.

Casazza, M., Ceselli, A., Chemla, D., Meunier, F., and Wolfler Calvo, R. (2018). The multiple vehicle balancing problem. *Networks*, **72**(3), 337–357.

Casazza, M., Ceselli, A., and Wolfler Calvo, R. (2021). A route decomposition approach for the single commodity split pickup and split delivery vehicle routing problem. *European Journal of Operational Research*, **289**(3), 897–911.

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

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

Feillet, D., Gendreau, M., and Rousseau, L.-M. (2007). New refinements for the solution of vehicle routing problems with branch and price. *INFOR*, **45**(4), 239–256.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E., and Werneck, R. F. (2005). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. *Mathematical Programming*, **106**(3), 491–511.

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

Laporte, G., Meunier, F., and Calvo, R. W. (2018). Shared mobility systems: an updated survey. *Annals of Operations Research*, **271**(1), 105–126.

Shui, C. and Szeto, W. (2020). A review of bicycle-sharing service planning problems. *Transportation Research Part C: Emerging Technologies*, **117**, 102648.