A PILOT/VND/GRASP Hybrid for the Static Balancing of Public Bicycle Sharing Systems*

Petrina Papazek, Günther R. Raidl, Marian Rainer-Harbach, and Bin Hu**

Institute of Computer Graphics and Algorithms
Vienna University of Technology
Favoritenstraße 9–11/1861, 1040 Vienna, Austria
{papazek|raidl|rainer-harbach|hu}@ads.tuwien.ac.at

Abstract. Due to varying user demands in bicycle sharing systems, operators need to actively shift bikes between stations by a fleet of vehicles. We address the problem of finding efficient vehicle tours by an extended version of an iterated greedy construction heuristic following the concept of the PILOT method and GRASP and applying a variable neighborhood descend (VND) as local improvement. Computational results on benchmark instances derived from the real-world scenario in Vienna with up to 700 stations indicate that our PILOT/GRASP hybrid especially scales significantly better to very large instances than a previously proposed variable neighborhood search (VNS) approach. Applying only one iteration, the PILOT construction heuristic followed by the VND provides good solutions very quickly, which can be potentially useful for urgent requests.

1 Introduction

Public bicycle sharing systems (BSSs) emerge worldwide in various cities. Such systems augment public transport very well, reduce the amount of motorized traffic, congestions, parking problems, and last but not least are an incentive for sports, thereby contributing to public health [4]. Typically, modern BSSs offer automated rental stations distributed over the city, where users may rent or return bicycles anytime. Operators face one important issue: most stations show asymmetric usage patterns, e.g., people tend to rent bikes at topographically higher stations and return them at lower stations. Other frequent influences are commuting patterns across working days and the weather situation [9]. In order to avoid critical situations where stations run completely empty or full, operators actively move bikes between stations, usually by a fleet of cars with trailers. The Balancing Bicycle Sharing System (BBSS) Problem deals with optimizing these vehicle tours together with corresponding loading or unloading directions.

2 The Balancing Bicycle Sharing System Problem

The BBSS problem is defined on a complete directed graph $G_0 = (V_0, A_0)$, with node set $V_0 = V \cup \{0\}$ consisting of the nodes for rental stations V and the vehicles' depot 0.

* This work is supported by the Austrian Research Promotion Agency (FFG) under contract 831740.

** We thank the Austrian Institute of Technology (AIT) and Citybike Wien for the collaboration in this project.

Arcs $(u,v) \in A_0$ connecting all $v \in V$ are weighted with a time value $t_{u,v} > 0$ that consists of the time needed for driving from u to v and for servicing v. Let the subgraph induced by the bike stations V only be $G = (V,A)$, $A \subset A_0$. For each station $v \in V$ we are given the bike capacity $C_v \geq 0$, the number of present bikes when beginning the rebalancing process p_v, as well as a target number of bikes q_v, with $0 \leq p_v, q_v \leq C_v$. The BSS operator employs a fleet of vehicles $L = \{1, \ldots, |L|\}$ for moving bikes. Each vehicle $l \in L$ starts empty at the depot 0, has a capacity of Z_l bikes and may visit an arbitrary number of stations before returning empty to the depot again as long as the total tour length t_l does not exceed an available time budget $\hat{t}$.

Solutions to the BBSS problem consist of a route for each vehicle $l \in L$ specified by an ordered sequence of visited stations $r_l = (r_l^1, \ldots, r_l^{\rho_l})$ with $r_l^i \in V$, $i = 1, \ldots, \rho_l$, and ρ_l being the number of visited stations. Note that each station may be visited multiple times by several vehicles. Each visit has associated loading instructions $y_l^i \in \{-Z_l, \ldots, Z_l\}$ with $l \in L$, $v \in V$, and $i = 1, \ldots, \rho_l$, specifying how many bikes are to be picked up ($y_l^i > 0$) or delivered ($y_l^i < 0$) at that visit.

Let a_v be the final number of bikes at each station $v \in V$ after rebalancing and let $\delta_v = |a_v - q_v|, \forall v \in V$. Our objective function is given by

$$\min \quad \omega^{\text{bal}} \sum_{v \in V} \delta_v + \omega^{\text{load}} \sum_{l \in L} \sum_{i=1}^{\rho_l} |y_l^i| + \omega^{\text{work}} \sum_{l \in L} t_l. \tag{1}$$

Scaling factors $\omega^{\text{bal}}, \omega^{\text{load}}, \omega^{\text{work}} \geq 0$ control the relative importance of the respective terms. The primary objective is to minimize deviations δ_v and only secondarily the number of loading activities as well as the overall tour lengths. We use the setting $\omega^{\text{bal}} = 1$ and $\omega^{\text{load}} = \omega^{\text{work}} = 1/100\,000$ in all our tests.

We simplify the problem by restricting the fill levels of stations to *monotonicity*. Let $V_{\text{pic}} = \{v \in V \mid p_v > q_v\}$ denote *pickup stations* and $V_{\text{del}} = \{v \in V \mid p_v < q_v\}$ denote *delivery stations*. A vehicle must only load bikes at pickup stations and unload bikes at delivery stations. As shown in previous work, this restriction has only a minimal impact on the theoretically achievable best solution quality [8].

3 Related Work

The BBSS problem is related to variants of the vehicle routing problem (VRP). Significant differences, however, include allowing multiple visits of stations, even by different vehicles, and the possibility of loading or unloading an arbitrary number of bikes. BBSS can be regarded as a capacitated single commodity split pickup and delivery VRP.

Each approach deals with different application characteristics, making a direct comparison difficult. In particular, Chemla et al. [2] require achieving perfect balance as a hard constraint. Their approach is designed for a single vehicle and consists of a branch-and-cut algorithm on a relaxed MIP model in conjunction with a tabu search for the local improvement of solutions. Benchimol et al. [1] focus on approximation algorithms for selected special situations. Their approaches also assume balancing as a hard constraint and are limited to a single vehicle. Raviv et al. [9] propose MIP models for the multiple-vehicle case. They consider a convex penalty objective function minimizing

user dissatisfaction and tour lengths, but ignore the number of loading operations. The assets and drawbacks of the models are compared on instances with up to 104 stations, two vehicles and a time horizon of up to five hours. Contardo et al. [3] investigate the dynamic scenario where user activities during rebalancing are taken into account. They describe a hybrid MIP approach utilizing Dantzig-Wolfe and Benders decomposition. Upper and lower bounds can be derived relatively quickly for instances up to 100 stations, but significant gaps remain. Schuijbroek et al. [11] decompose the problem into separate single-vehicle routing problems by solving a clustering problem. The routing problems are handled by a clustered MIP heuristic or a constraint programming approach.

In [8], we propose a greedy construction heuristic followed by a variable neighborhood search/variable neighborhood descent (VNS/VND) metaheuristic for efficiently finding vehicle routes. Three alternative auxiliary algorithms calculate meaningful loading instructions for given tours. In [7] we develop a forth alternative for deriving loading instructions and describe an effective way for applying all of them in a hybrid fashion. The current work extends our methods by applying the PILOT method [12] in the construction heuristic and GRASP as an alternative to the VNS, as well as by performing experiments on larger instances of up to 700 stations.

4 Construction Heuristics

We employ two alternative construction heuristics for creating initial solutions: a greedy construction heuristic (GCH) and an extended version following the PILOT method.

4.1 Greedy Construction Heuristic

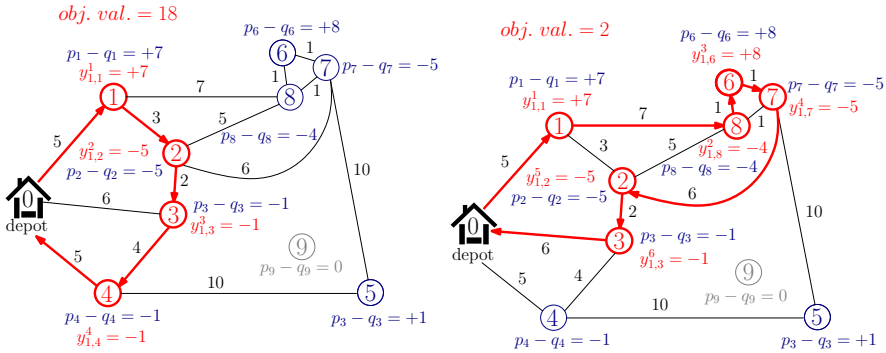
The greedy construction heuristic, which is in detail described in [8], sequentially constructs vehicle tours in a pure greedy manner following a local best successor strategy. First, we compute the maximum number of bicycles γ_v that can be picked up or delivered at any station v in the set of feasible, i.e., not yet balanced, successor stations F:

$$\gamma_v = \begin{cases} \min(a_v - q_v, Z_l - b_l) & \text{for } v \in F \cap V_{\text{pic}} \text{ and} \\ \min(q_v - a_v, b_l) & \text{for } v \in F \cap V_{\text{del}}, \end{cases} \tag{2}$$

where b_l expresses the final load of vehicle l so far and a_v the final number of bikes at station v in the partial tour. Next, we evaluate the ratio $\gamma_v / t_{u,v}$ for all $v \in F$. As vehicles need to return empty to the depot, we additionally apply a correction at pickup stations: We restrict the number of pickups at a station by potential deliveries after this stop. Eventually, we append the station offering the highest ratio to the tour r_l and derive loading instructions as follows:

$$y_l^{\rho_l} = \begin{cases} \gamma_v & \text{if } v \in V_{\text{pic}} \text{ and} \\ -\gamma_v & \text{if } v \in V_{\text{del}}. \end{cases} \tag{3}$$

After updating b_l and a_v, the procedure continues with the next extension.



(a) GCH. (b) PILOT-CH.

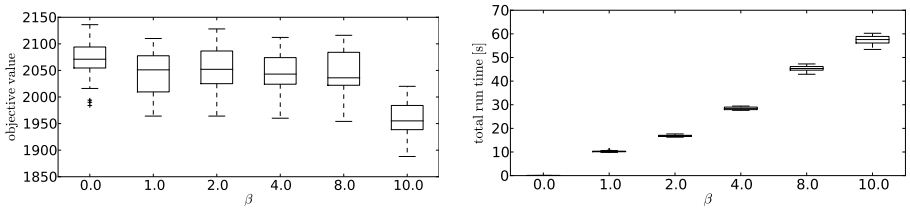
Fig. 1. Exemplary solutions of GCH vs. PILOT-CH with $|L| = 1$ and $\hat{t} = 30$min.

4.2 PILOT Construction Heuristic

The drawbacks of the fast GCH – always choosing the single locally best successor – are possible shortsighted results, e.g., we might never service a more distant cluster of stations offering a substantial balance gain. The PILOT (Preferred Iterative LOok ahead Technique) method addresses this issue by looking ahead in order to escape this greedy trap [12]. Consequently, the PILOT construction heuristic (PILOT-CH) extends GCH by evaluating each potential successor in a deeper way by constructing a complete temporary route. For this purpose, we utilize the objective function value as evaluation criterion and select the candidate station with the highest benefit. Figure 1 demonstrates an example where PILOT-CH surpasses GCH. For simplicity we merely visualize the most lucrative connections weighted with symmetric traveling times. Due to the recursive evaluation of candidates the time complexity of PILOT-CH is higher than GCH by a factor of $O(|V|)$. To speed up the computation we may limit the *PILOT depth* β, i.e., restrict the number of successor stations of the recursive look-ahead. In this case we adopt the evaluation criterion of GCH, i.e., the ratio of the balance gain and the time for the whole extension, as the objective function only makes sense for complete solutions. Figure 2 illustrates obtained objective values and computation times for varying β on benchmark instances, where $\beta = 0$ represents GCH and $\beta = \infty$ unrestricted depth. For details on the instances and hardware see Section 7. As $\beta = \infty$ runs fast compared to our other approaches while yielding significantly better results than all depth-restricted cases, we use it in all further experiments in this article.

5 Variable Neighborhood Descent

For locally improving candidate solutions, we employ a Variable Neighborhood Descent (VND) [5] with several classical neighborhood structures that were successfully applied in VRPs [6] as well as new neighborhood structures specific to BBSS. The neighborhoods are described in detail in our previous work [8]: Remove station (REM-VND), insert unbalanced station (INS-U), replace station (REPL), intra or-opt (OR-



(a) Objective values.

(b) CPU time.

Fig. 2. PILOT-CH: Finally best objective values and CPU times [s] for instances with $|V| = 700$, $|L| = 14$, $\hat{t} = 8h$ and different PILOT depths β.

OPT), 2-opt* inter-route exchange (2-OPT*), and intra-route 3-opt (3-OPT). All neighborhoods are searched using the best improvement strategy, and they are applied in random order, which turned out to work better in conjunction with PILOT-CH than the static order from [8].

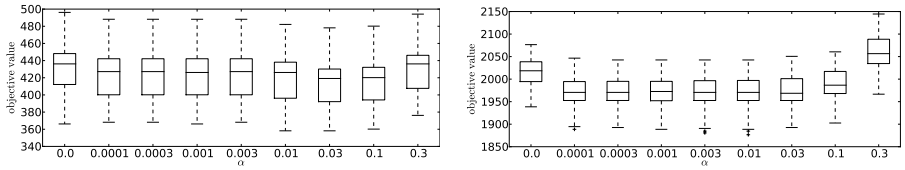
The VND only searches the space of vehicle routes, while loading instructions are calculated for each candidate solution by an auxiliary heuristic following the greedy strategy from GCH; see also [8], Chemla et al. [2].

6 Greedy Randomized Adaptive Search Procedure

For optimizing solutions further, we realized a *Greedy Randomized Adaptive Search Procedure* (GRASP) [10] by iteratively applying randomized versions of either the GCH or PILOT-CH, locally improving each solution with the VND, and finally returning the overall best solution. In the randomized construction heuristics we select a random successor station from a restricted candidate list $RCL \subseteq F$ instead of always picking the best candidate:

$$RCL = \{v \in F \mid g(v) \geq g_{\max} - \alpha\,(g_{\max} - g_{\min})\}, \tag{4}$$

where $g(v)$ is the greedy value of candidate station v, while $g_{\max} = \max\{g(v) \mid v \in F\}$ and $g_{\min} = \min\{g(v) \mid v \in F\}$ are the maximum and minimum evaluation values in F, respectively. Accordingly, $\alpha \in [0,1]$ controls the strength of the randomization, with $\alpha = 0$ representing a pure greedy and $\alpha = 1$ a completely random construction method. In this context, we may choose either a fixed α, i.e., remaining constant throughout all GRASP iterations, or a randomized $\alpha \in [0, \alpha_{\max}]$, changing in a random manner at each iteration. Evaluating both variants of α on the benchmarks instances disclose that the randomized version is more robust, and thus we employ it in all further tests. Moreover, the tests indicate that large instances w.r.t. $|V|$, $|L|$, and $\hat{t}$ require smaller values for α than small instances. Figure 3 shows the impact of different values for α exemplarily for GCH-based GRASP (GCH-GRASP) on large instances. Based on many preliminary tests, we finally decided to choose $\alpha = 0.11 \cdot e^{-\frac{|V|}{187}}$.



(a) Instances with $|V| = 180$, $|L| = 4$.

(b) Instances with $|V| = 700$, $|L| = 14$.

Fig. 3. GCH-GRASP: Final objective values in dependence of α; $\hat{t} = 8h$.

7 Computational Results

We performed computational tests on benchmark instances[1], which range from 10 to 700 stations and are derived from real-world data provided by Citybike Wien, which operates a BSS of 92 stations in Vienna, Austria. For larger instances the Austrian Institute of Technology (AIT) supplied us with another 664 artificial stations placed at reasonable locations. For the following tests we consider $|V| \in \{30, 60, 90, 180, 300, 400, 500, 600, 700\}$ and a shift length $\hat{t}$ for all vehicles ranging from 2 to 8 hours. We set $|L|$ between 1% and 12% of $|V|$ depending on $\hat{t}$. All instance sets include 30 instances and represent unique combinations of $|V|$, $|L|$, and $\hat{t}$. We implemented and ran each test on a single core of an Intel Xeon E5540 machine with 2.53 GHz. For a fair comparison we terminate our algorithms after a defined run time t_{max} depending on the instance size.

Note that the scaling factors $\omega^{bal} = 1$, $\omega^{load} = \omega^{work} = 1/100\,000$ induce that an improved balance always effects the objective values more than a decrease in tour lengths or number of loading operations. Since the values of ω^{load} and ω^{work} cause small differences of objective values, these small values are still crucial for evaluating the quality of solutions. In order to ease these comparisons we also list the number of runs for which the variant yields the best results of all variants in the #best column.

As indicated by the mean objective values $\overline{obj}$ and #best in table 1 and confirmed by Wilcoxon signed-rank test with an error probability of 5%, the PILOT construction heuristic (PILOT-CH) clearly outperforms the simple greedy construction heuristic (GCH) on each instance set. Naturally, PILOT-CH consumes more median computation time as listed in the column $\widetilde{t_{tot}}$. Nevertheless, it is still faster than the more complex optimization methods and consequently a good compromise between solution quality and computation time. PILOT-CH might be a good option for practical applications requiring short computation times. Table 2 includes the final results of selected instance sets for VNS as proposed in our previous work [8], GCH-GRASP (GRASP with randomized GCH), and PILOT-GRASP (GRASP with randomized PILOT-CH). According to a Wilcoxon signed-rank test (with less than 5% error probability), PILOT-GRASP yields significantly better results than GCH-GRASP. When comparing our VNS with PILOT-GRASP, we observe that all approaches perform almost equally good on small instances with 30 stations. However, while VNS dominates the medium-sized instances with 60 to 180 stations, PILOT-GRASP is superior on large instances with 400 or more stations.

[1] Available at: https://www.ads.tuwien.ac.at/w/Research/Problem_Instances

Table 1. Computational results of GCH and PILOT-CH.

Instance set			GCH			PILOT-CH		
$\|V\|$	$\|L\|$	$\hat{t}$ $[h]$	#best	$\overline{\text{obj}}$	$\widetilde{\text{t}_{\text{tot}}}$ $[s]$	#best	$\overline{\text{obj}}$	$\widetilde{\text{t}_{\text{tot}}}$ $[s]$
30	1	2	4	141.401410	< 0.1	**29**	138.134850	< 0.1
30	1	4	1	99.203000	< 0.1	**30**	93.536490	< 0.1
60	1	4	3	279.136470	< 0.1	**29**	271.136660	< 0.1
60	2	2	2	302.936050	< 0.1	**29**	291.603090	< 0.1
90	2	4	1	390.739620	< 0.1	**29**	379.273230	< 0.1
90	2	8	1	236.545910	< 0.1	**29**	220.812930	0.2
180	4	4	1	760.812500	< 0.1	**29**	735.146480	0.3
180	4	8	0	448.825240	< 0.1	**30**	421.425890	1.2
300	6	4	0	1361.285300	< 0.1	**30**	1310.753180	1.1
300	6	8	0	865.571370	< 0.1	**30**	819.039150	5.0
400	8	4	0	1833.891260	< 0.1	**30**	1760.959600	2.3
400	8	8	0	1161.650440	< 0.1	**30**	1096.118750	11.3
500	10	4	0	2294.297610	< 0.1	**30**	2213.966180	4.5
500	10	8	0	1452.530120	< 0.1	**30**	1378.665150	22.2
600	12	4	0	2783.170230	< 0.1	**30**	2672.506080	6.9
600	12	8	0	1762.409230	< 0.1	**30**	1658.011680	35.9
700	14	4	0	3255.442790	0.1	**30**	3125.779220	10.3
700	14	8	0	2068.555330	0.1	**30**	1957.891110	57.6
		Total	13	21498.403880	0.2	534	20544.759720	158.8

Table 2. Computational results of VNS and two GRASP variants.

Instance set				VNS		GCH-GRASP		PILOT-GRASP	
$\|V\|$	$\|L\|$	$\hat{t}$ [h]	t_{max} [s]	#best	$\overline{\text{obj}}$	#best	$\overline{\text{obj}}$	#best	$\overline{\text{obj}}$
30	1	2	900	28	137.13486	27	137.13617	**29**	136.93617
30	1	4	900	29	89.26988	28	89.34470	**30**	89.27799
60	1	4	1800	**30**	267.00340	19	268.01137	25	267.27850
60	2	2	1800	**24**	287.40315	12	288.67231	**24**	287.47238
90	2	4	1800	**28**	368.00672	0	370.35554	4	369.42263
90	2	8	1800	**27**	205.21311	2	210.95327	2	210.75318
180	4	4	3600	**27**	714.21342	0	723.57714	3	719.04451
180	4	8	3600	**28**	396.82615	0	412.56891	2	408.03814
300	6	4	3600	**27**	1287.42007	0	1304.53134	4	1294.80010
300	6	8	3600	**17**	798.57261	3	813.78571	10	803.72637
400	8	4	3600	**19**	1737.75980	0	1762.41566	11	1743.21898
400	8	8	3600	1	1087.31864	0	1094.79654	**29**	1077.87467
500	10	4	3600	12	2193.83297	0	2217.90432	**18**	2192.24287
500	10	8	3600	0	1383.19814	1	1381.35180	**29**	1359.56638
600	12	4	3600	2	2664.17266	0	2691.98787	**28**	2651.39459
600	12	8	3600	0	1675.21100	0	1675.89018	**30**	1641.57620
700	14	4	3600	0	3128.84563	0	3150.20618	**30**	3102.14626
700	14	8	3600	0	1979.89031	0	1974.83820	**30**	1938.92603
		Total	52200	299	20401.29252	92	20568.32721	338	20293.69595

8 Conclusions and future Work

We presented and tested two GRASP approaches iterativly employing randomized versions of the construction heuristics and compared them to our previously proposed VNS approach. Computatinal results indicate that the PILOT-GRASP variant surpasses the GCH-GRASP and the VNS on large instances up to 700 stations. However, the VNS yields the best results on medium instances. Hence, we conclude that PILOT-GRASP scales better with respect to instance size and complexity.

In future work we will adapt our methods to the dynamic case of BBSS considering prognosis of demands involving stochastic aspects and investigate the hybridization of VNS with MIP approaches for computing lower bounds.

References

1. Benchimol, M., Benchimol, P., Chappert, B., De la Taille, A., Laroche, F., Meunier, F., Robinet, L.: Balancing the stations of a self service bike hire system. RAIRO – Operations Research **45**(1), 37–61 (2011)
2. Chemla, D., Meunier, F., Calvo, R.W.: Bike sharing systems: Solving the static rebalancing problem. Discrete Optimization **10**(2), 120–146 (2013)
3. Contardo, C., Morency, C., Rousseau, L.M.: Balancing a Dynamic Public Bike-Sharing System. Tech. Rep. CIRRELT-2012-09, Montreal, Canada (2012)
4. DeMaio, P.: Bike-sharing: History, impacts, models of provision, and future. Journal of Public Transportation **12**(4), 41–56 (2009)
5. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers and Operations Research **24**(11), 1097–1100 (1997)
6. Pirkwieser, S., Raidl, G.R.: A variable neighborhood search for the periodic vehicle routing problem with time windows. In: Proceedings of the 9th EU/MEeting on Metaheuristics for Logistics and Vehicle Routing. Troyes, France (2008)
7. Raidl, G.R., Hu, B., Rainer-Harbach, M., Papazek, P.: Balancing bicycle sharing systems: Improving a VNS by efficiently determining optimal loading operations. In: M.J. Blesa, et al. (eds.) Hybrid Metaheuristics, 8th Int. Workshop, HM 2013, *LNCS*, vol. 7919, pp. 130–143. Springer (2013)
8. Rainer-Harbach, M., Papazek, P., Hu, B., Raidl, G.R.: Balancing bicycle sharing systems: A variable neighborhood search approach. In: M. Middendorf, C. Blum (eds.) Evolutionary Computation in Combinatorial Optimization, *Lecture Notes in Computer Science*, vol. 7832, pp. 121–132. Springer Berlin Heidelberg (2013)
9. Raviv, T., Tzur, M., Forma, I.A.: Static repositioning in a bike-sharing system: models and solution approaches. EURO Journal on Transportation and Logistics pp. 1–43 (2013)
10. Resende, M., Ribeiro, C.: Greedy randomized adaptive search procedures. In: F. Glover, G. Kochenberger (eds.) Handbook of Metaheuristics, pp. 219–249. Kluwer Academic Publishers (2003)
11. Schuijbroek, J., Hampshire, R., van Hoeve, W.J.: Inventory Rebalancing and Vehicle Routing in Bike Sharing Systems. Tech. Rep. 2013-E1, Tepper School of Business, Carnegie Mellon University (2013)
12. Voß, S., Fink, A., Duin, C.: Looking ahead with the Pilot method. Annals of Operations Research **136**, 285–302 (2005)