Balancing Bicycle Sharing Systems: An Analysis of Path Relinking and Recombination within a GRASP Hybrid

Petrina Papazek, Christian Kloimüllner, Bin Hu, and Günther R. Raidl*

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

Abstract. In bike sharing systems, a vehicle fleet rebalances the system by continuously moving bikes among stations in order to avoid rental stations to run entirely empty or full. We address the static problem variant assuming initial fill levels for each station and seek vehicle tours with corresponding loading instructions to reach given target fill levels as far as possible. Our primary objective is to minimize the absolute deviation between target and final fill levels for all rental stations. Building upon a previously suggested GRASP hybrid, we investigate different approaches for hybridizing them with Path Relinking (PR) and simpler recombination operators. Computational tests on benchmark instances derived from a real world scenario in Vienna give insight on the impacts of the PR and recombination techniques and manifest that certain PR extension improve the results significantly. Ultimately, a hybrid exclusively searching a partial PR path in the neighborhood of the guiding solutions turns out to be most fruitful.

1 Introduction

Public Bicycle Sharing Systems (BSSs) are booming in many cities around the globe. A BSS comprises a set of automated rental stations, which allows users to rent and return bikes at any station of the system at any time. Establishing BSSs in cities is beneficial as they augment public transport very well, are "green" alternatives to motorized traffic, and contribute to public health [1]. However, BSSs face one major issue: without actively redistributing bikes among stations, i.e. balancing the stations, most would either run out of available bikes or free slots. This natural disbalance of a BSS originates from diverse circumstances, such as topography, commuting patterns, and weather conditions. To keep the system in balance, most BSS operators actively redistribute bikes among stations by a vehicle fleet.

* This work is supported by the Austrian Research Promotion Agency (FFG), contract 831740. The authors thank Citybike Wien, the Austrian Institute of Technology (AIT), and Energie und Umweltagentur Niederösterreich (eNu) for the collaboration in this project.

We consider the *Balancing Bicycle Sharing System* (BBSS) problem as introduced in [2]: Given a set of vehicles and a certain time budget, our goal is to find vehicle tours with corresponding bicycle loading instructions such that the stations' fill levels are brought to specified target values as far as possible.

Building upon our previous work, in particular a GRASP hybrid, we investigate different extensions by Path Relinking (PR) and simpler recombination operators and their impacts. While straight-forward applications of PR do not yield significant improvements in solution quality but rather increase the computational costs substantially, a variant, where only a part of the complete PR path in the vicinity of the guiding solution is searched, turns out to be fruitful. The GRASP hybrid in conjunction with this restricted form of PR is able to improve previously leading results on mid-size instances significantly.

2 Related Work

Although BBSS is a relatively new problem domain already several different algorithmic approaches have been suggested for diverse variants of it. Most of them apply integer linear programming techniques [3–6], whose applicability, however, is restricted to small instances or strongly simplified problem variations.

In our previous work we have developed a greedy construction heuristic which we improved by the PILOT method [7]. As metaheuristic approaches we established a Variable Neighborhood Search (VNS) [2] with an embedded Variable Neighborhood Descent (VND) and GRASP [7]. Our results about the static case have been refined together with comprehensive computational tests in the *Journal of Global Optimization* [8]. A more detailed description of these approaches can be found in Section 4. Comparing different work on BBSS is usually difficult because they consider various specific problem variants. Di Gaspero et al. [9, 10] proposed approaches based on constraint programming, ant colony optimization, and large neighborhood search and tested them on the same benchmark instances as ours, though they could not outperform our previous approaches mentioned above.

For the dynamic case, where the system is rebalanced while usage is simulated, only few work has been published so far [11]. In a recent work we considered this scenario and extended our techniques to handle expected dynamic user demands without relying on an expensive time-discretization [12].

Much more work exists on other aspects of BSSs, such as an optimal network design [13], system characteristics and usage patterns [14], but they are not in the scope of this paper.

3 Problem Definition

We consider the static or offline BBSS problem variant which neglects user interaction during rebalancing and model the BSS as a complete directed graph $G_0 = (V_0, A_0)$. The vertices V_0 comprise all rental stations V and the depot 0, which is the start and end point of the tours. The arcs A_0 connect all vertices

and are weighted with time $t_{u,v}$. This time represents the travel time between nodes $u, v \in V_0$ including an average service time for loading or unloading bikes at node v. Furthermore, we are given for each station $v \in V$ the capacity C_v, i.e., the number of bike slots, the initial fill level p_v, and a target fill level q_v. The BSS operator employs a fleet of vehicles L for distributing the bikes. Each vehicle $l \in L$ starts empty at the depot and may *visit* an arbitrary number of stations before it has to return *empty* to the depot again. Each vehicle $l \in L$ has associated a bike capacity Z_l.

A solution to the BBSS problem comprises a tour for each vehicle $l \in L$ and corresponding loading instructions for each stop. We define this ordered sequence of visited stations as $r_l = (r_l^1, \ldots, r_l^{\rho_l})$ with $r_l^i \in V$, $i = 1, \ldots, \rho_l$, and ρ_l being the number of visited stations. An important aspect is that stations may be visited multiple times by the same or different vehicles. Each visit has associated loading instructions $y_l^i \in \{-Z_l, \ldots, Z_l\}$ with $l \in L$ 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. A solution is feasible if vehicle and station capacities are never exceeded, the number of bikes available at stations is never below zero, a vehicle does not deliver more bikes than it has actually loaded, and if the working time t_l of a vehicle $l \in L$ does not exceed a given total time budget $\hat{t}$.

Let a_v be the final number of bikes at each station $v \in V$ after rebalancing. The primary objective is to minimize the deviation from the target values q_v, i.e., the disbalance $|a_v - q_v|$ at each station $v \in V$. As secondary objectives we want to minimize the total number of loading operations $|y_l^i|$ over all visits ρ_l and all vehicles L and the working time t_l of all drivers $l \in L$. This is expressed by the following objective function:

$$\min \quad \underbrace{\omega^{\text{bal}} \sum_{v \in V} |a_v - q_v|}_{\text{disbalance}} \quad + \quad \underbrace{\omega^{\text{load}} \sum_{l \in L} \sum_{i=1}^{\rho_l} |y_l^i|}_{\text{loading operations}} \quad + \quad \underbrace{\omega^{\text{work}} \sum_{l \in L} t_l}_{\text{working time}} \qquad (1)$$

The scaling factors $\omega^{\text{bal}}, \omega^{\text{load}}$, and ω^{work} control the relative importance of these terms. We assume that any improvement in balance is always preferred over decreasing the number of loading actions or reducing the working time and set the scaling factors accordingly ($\omega^{\text{bal}} = 1$ and $\omega^{\text{load}} = \omega^{\text{work}} = 1/100\,000$). The reason is that a balanced system is top priority for maximizing customer-satisfaction while from the operator's point of view, workers are paid for the whole shift length anyway, and therefore a reduction in the tour lengths is just a secondary aspect.

4 Metaheuristics Approaches

This section gives an overview on the metaheuristic approaches we proposed in [8]. All of these approaches utilize an incomplete solution representation by considering vehicle tours only; corresponding loading operations are computed via an auxiliary procedure whenever a solution is evaluated. From the different

variants for this procedure considered in [8], we use here the fastest greedy heuristic [2, 8], which is the most reasonable approach in practice since it scales best for large instances and nevertheless yields close to optimal loading operations.

To create an initial solution we employ two alternative construction heuristics: a greedy construction heuristic (GCH) and an extension of it based on the PILOT method [15]. GCH sequentially constructs vehicle tours following a local best successor strategy. To derive a tour for each vehicle, we compute the maximum number of bicycles γ_v that can be picked up or delivered at any not yet balanced station v without exceeding/deceeding its target value. For each vehicle, we construct a tour by starting at the depot and evaluating the next best successor by the ratio $\gamma_v/t_{u,v}$. Here, we only consider a station v if enough time remains to return to the depot afterwards. If there are no further feasible stations left, we proceed with the next vehicle. Afterwards, we derive loading instructions by an auxiliary heuristic. The PILOT construction heuristic (PCH) extends the greedy construction heuristic by overcoming possibly shortsighted successors. In particular, PCH evaluates each potential successor more accurately by constructing a complete temporary route utilizing the objective function value as evaluation criterion. This approach requires more time than GCH, but yields substantially better results in return.

For locally improving candidate solutions, we employ a Variable Neighborhood Descent (VND) [16] using seven neighborhood structures and applying a best improvement strategy [8]: remove station, insert unbalanced station, replace station, intra or-opt, 2-opt* inter-route exchange, and intra-route 3-opt.

As the solution construction followed by VND is still quite fast and improvement potential remains, the approach is further extended into a Variable Neighborhood Search (VNS) [8] with the following shaking operators: move-sequence, exchange-sequence, destroy-&-recreate, and remove-stations.

In addition to the VNS, we investigated a hybrid Greedy Randomized Adaptive Search Procedure (GRASP) [17] by iteratively applying randomized versions of either GCH or PCH, locally improving each solution with the VND, and finally returning the overall best solution. Here, the PCH version with a random neighborhood in the VND turned out to perform best.

In the experimental evaluation in [8], VNS yields the best results on small and mid-size instances with up to 300 stations, while PCH-GRASP performed better on large instances with up to 700 stations.

5 Path Relinking and Recombination in GRASP

Glover et al. [18] define PR as the evolutionary technique of altering an initial solution I towards a guiding solution G – typically a solution from an elite set – by a series of simple moves. These moves describe a trajectory or path of (not necessarily feasible) solutions, and a best encountered solution is returned as result. According to numerous studies such as [19, 20], PR yields promising results on diverse related vehicle routing problems. Moreover, simpler recombination techniques as they are mainly used in evolutionary algorithms are able to yield

possibly promising candidate solutions from the joined properties of two input solutions in a fast manner. Therefore, we investigate extensions of the above hybrid GRASP by PR and simpler recombination techniques. This section describes several specific approaches in detail, which we found most meaningful in the context of BBSS, and explains how to embed it into GRASP.

5.1 PR and Recombination Variants

In order to iteratively transform solution I into solution G inside PR, we need to define basic moves. Again, we consider loading instructions to be always calculated by the auxiliary greedy heuristic for each intermediate candidate solution on the fly and thus, solely concentrate on the tours. One basic move edits a single vehicle tour r_l by removing, adding, or replacing exactly one station.

We adopt the common principle from PR for other vehicle routing problems to match each tour of I with a tour in G and individually relink or recombine each pair of corresponding tours. As we consider the case of having a not necessarily homogeneous vehicle fleet, we relate each vehicle tour of I with the tour of the same vehicle in G. Then, we transform the tours from I into corresponding ones from G, yielding a series of intermediate solutions. These intermediate solutions are not necessarily feasible as the relinked tours may exceed the allowed time budget $\hat{t}$. Consequently, we repair such infeasible tours by pruning them from the end before they are evaluated. In the following we denote by $r_l(G)$ the tour of vehicle l in solution G and by $r_l(I)$ the corresponding tour in solution I, $l \in L$; $\rho_l(G)$ and $\rho_l(I)$ denote their respective tour lengths. We consider the following operators for systematically generating intermediate solutions.

Sequential Replace PR (Seq-PR). Within this basic PR operator, we sequentially transform each tour $r_l(I)$, $l \in L$ step-by-step with basic moves into G's corresponding tour. This is achieved by iterating over the stops r_l^i with $i = 1, \ldots, \rho_l(G)$ and replacing each corresponding stop $r_l^i(I)$ by $r_l^i(G)$ or adding it if I's original tour was shorter.

One-Point-Recombination (OP-Rec). Starting from the parent solutions I and G, we randomly select a vehicle $l \in L$ and a crossover position $p \in \{1, \ldots, \min(\rho_l(I), \rho_l(G))\}$. In a first offspring, the tour r_l becomes

$$(r_l^1(I), \ldots, r_l^{p-1}(I), r_l^p(G), \ldots, r_l^{\rho_l(G)}(G))$$

and in a second

$$(r_l^1(G), \ldots, r_l^{p-1}(G), r_l^p(I), \ldots, r_l^{\rho_l(I)}(I)).$$

For each of the next vehicles $l \in L$ we randomly decide whether its tour is copied from $r_l(G)$ or $r_l(I)$ without any further change.

Single Tour Recombination (ST-Rec). This recombination variant basically works like OP-Rec by first selecting a vehicle and performing one-point crossover on the two corresponding tours, but then, all remaining tours are only adopted from I for the first offspring and only G for the second offspring, respectively. Thus, this operator changes a parent solution only in a part of one tour.

Restricted Multistart PR (MS-PR). This operator combines concepts from Seq-PR and ST-Rec. A main drawback of Seq-PR is its high computational cost. To speed up Seq-PR, we skip the evaluation of solutions in the middle part of the tours, since most potential solutions are in the close neighborhood of the optimized tours in I and G. As we found out in preliminary analysis, the best solutions on trajectories from I to G are almost always located relatively close to G and for this reason we focus on the final stations of tours. We therefore restrict ourself in MS-PR to *final parts* of paths from I to G. This is achieved by starting with a whole copy of G and replacing just one tour $r_l(G)$, $l \in L$, by a copy from $r_l(I)$. This tour $r_l(I)$ is then step-by-step relinked as in Seq-PR until G is reached again. In particular, it turned out most successful to solely evaluate the first and three final stations of the tour. If there is more than one vehicle, we perform the same procedure also for all the other vehicles in L; i.e., we perform multiple starts, one for each vehicle, which gives the operator its name.

Applying VND to the final best solution from the PR operator as suggested by [21] typically improves our results, because of frequent unfavorable ordering of stations by merely merging arbitrary tours. Note that our PR primarily uses replace operations without more sophisticated best improvement insert strategies. To integrate a reordering of stations into the PR or recombination operator, we can additionally employ VND to each intermediate/offspring solution.

However, performing the full VND at each step may soon become too expensive for large instances. In order to speed up the process, we use a limited variant of VND, which works as follows: For each intermediate PR solution, we test if applying a single move of each VND neighborhood leads to a solution that is better than I. Only if this is the case, a full VND is applied. This way lots of mediocre solutions on the path are skipped relatively quickly.

5.2 Embedding in GRASP

For applying PR and recombination operators within our PILOT-GRASP from [8], we add a memory to GRASP through an elite set of best solutions. Fiesta et al. [21] proposed different adaptive variants to join PR and GRASP, in particular Path Relinking GRASP (PR-GRASP), which we adopt here.

In PR-GRASP we employ the PR or recombination operator on each new VND improved GRASP solution and a randomly selected member of the elite set. The elite set is initialized by adding solutions from the GRASP procedure with the condition that they must be different from each other. If the initial elite set is sufficiently large (e.g., after five GRASP iterations), we employ the PR or recombination operator in each GRASP iteration. Before embarking the next

GRASP iteration, we consider to incorporate the PR or recombination enhanced candidate solution to the elite set in case it is different from all other already included members. In this context, a pure random update strategy turned out to perform best: We select a random element from the elite set and replace it by the new candidate solution if the objective value of the new one is better. In order to enhance diversity and quality in the elite set, we also tested the common approach of considering a minimal solution edit distance as proposed by [21]. However, preliminary tests revealed that this distance-based replacement strategy did not yield significantly better solutions but introduced a non-negligible runtime overhead due to the distance calculations.

6 Computational Analysis

In this section we show the results of our PR and recombination hybrids and compare them with the leading approaches from [8]. The instances of our benchmark sets[1] have been derived from the real-world scenario in Vienna provided by Citybike Wien and the AIT. The initial fill levels p_v of the stations are taken from a historical snapshot of the system in Vienna whereas the target fill levels q_v are calculated according to the estimated user demands at the particular stations. For further reference on the computation of target values, please, refer to our publication in the *Journal of Global Optimization* [8]. We consider an amount of stations $|V| \in \{60, 90, 180, 300, 400, 500\}$, a time budget $\hat{t}$ of 2, 4, 8 hours, and a number of vehicles $|L|$ between 1% and 2% of $|V|$ in order to test reasonable settings for practical scenarios. Large instances beyond 500 stations are neglected because the PR-GRASP variants are able to run only a few iterations, and PR extensions would make no sense in such situations. All benchmark sets include 30 instances and represent unique combinations of $|V|$ and $|L|$. We run each instance on a single core of an Intel Xeon E5540 machine with 2.53 GHz. For every algorithmic variant we use a common CPU time limit of 15 minutes for small instances with 30 stations, 30 minutes for medium instances with 60 to 90 stations, and 60 minutes for large instances with more than 90 stations.

Tables 1 and 2 include the results for the basic PR and recombination variants in conjunction with PILOT-GRASP as well as our previously developed metaheuristics from [8] and our most successful PR-GRASP variants: PILOT-GRASP with ST-Rec including full VND and MS-PR including limited VND. We list the mean objective values $\overline{obj}$, the counted best values #best (i.e., the number of instances where the approach has been superior to all other variants over both tables), and the number of major GRASP-iterations $\overline{g_{tot}}$ for each instance.

The OP-Rec operator is comparatively fast and enables many GRASP iterations. In particular, it requires only two VND evaluations per iteration and is thereby substantially faster than the Seq-PR operator. Both variants, Seq-PR as well as OP-Rec, are only competitive on smaller instances and are worse for

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

Table 1. Computational results of Seq-PR- and OP-Rec-GRASP.

Inst. set			Seq-PR-GRASP						OP-Rec-GRASP		
			full VND			limited VND					
$\|V\|$	$\|L\|$	$\hat{t}$	#best	$\overline{obj}$	$\overline{g_{tot}}$	#best	$\overline{obj}$	$\overline{g_{tot}}$	#best	$\overline{obj}$	$\overline{g_{tot}}$
30	1	2	26	147.33499	714751.5	26	147.33499	711656.0	26	147.33499	730164.5
30	1	4	**28**	95.40335	20523.0	22	95.80334	43938.5	27	95.47001	53617.5
30	1	8	27	29.20639	1111.5	22	29.27305	3584.5	**29**	29.20639	5457.5
60	1	4	25	270.00361	20477.5	18	270.87025	44621.5	26	269.93695	48482.0
60	1	8	**24**	170.20702	1168.0	13	171.00699	4190.5	23	170.47367	5580.5
60	2	2	**26**	293.80331	49955.5	24	293.93664	86270.0	**26**	293.80331	116731.5
90	2	4	**17**	346.00724	1720.5	6	347.60720	6583.5	16	345.94057	8753.5
90	2	8	1	175.61367	101.0	0	175.68032	673.0	1	176.48032	1007.5
90	4	4	2	197.21342	225.0	0	197.48005	1534.0	1	197.81338	2573.0
180	4	4	0	721.94756	265.0	1	721.41426	1465.5	2	721.41424	2434.0
180	4	8	0	385.36037	16.0	0	379.89380	150.0	0	384.29372	261.5
180	5	8	0	275.83278	10.5	1	270.49957	89.0	0	275.49946	164.0
300	6	4	0	1254.75482	37.5	0	1248.48823	312.0	0	1251.82153	528.5
300	6	8	0	724.50765	7.0	0	718.64109	38.0	0	724.77430	62.0
300	9	8	1	403.92498	6.0	2	399.12494	18.0	0	403.12491	28.5
400	8	4	0	1668.89514	16.0	0	1661.29525	128.5	0	1665.96185	240.5
400	8	8	1	956.38794	6.0	2	953.98794	18.0	0	958.32122	27.0
400	12	8	2	532.01093	6.0	0	530.01096	10.0	1	531.14422	14.0
500	10	4	0	2103.30225	9.0	1	2095.83567	60.5	0	2102.83560	108.0
500	10	8	0	1211.26824	6.0	3	1208.93495	11.0	1	1210.60163	15.5
500	15	8	0	668.03053	6.0	4	665.83060	7.0	1	667.03045	9.0
Total			180	12631.01619	810424.5	145	12582.95009	905359.0	180	12623.28272	976260.0

Table 2. Computational results of ST-Rec- and MS-PR-GRASP.

Inst. set			VNS			PILOT-GRASP			ST-Rec-GRASP			MS-PR-GRASP		
$\|V\|$	$\|L\|$	$\hat{t}$	#best	$\overline{obj}$	$\overline{g_{tot}}$	#best	$\overline{obj}$	$\overline{g_{tot}}$	#best	$\overline{obj}$	$\overline{g_{tot}}$	#best	$\overline{obj}$	$\overline{g_{tot}}$
30	1	2	28	147.20166	1215760.5	**29**	147.13500	618965.5	26	147.33499	728816.5	26	147.33499	729751.0
30	1	4	27	95.20336	269610.5	23	95.73667	100840.0	**28**	95.40335	52691.0	22	95.80334	58542.0
30	1	8	26	29.20639	37254.0	26	29.27305	10879.0	**29**	29.20639	5386.5	23	29.27305	6703.5
60	1	4	**29**	269.60362	310747.5	17	271.20358	77673.0	26	269.93695	48437.5	18	270.93692	54233.0
60	1	8	19	170.47367	44431.5	13	171.00699	10889.5	22	170.40701	5673.5	15	170.87366	7793.0
60	2	2	23	293.80331	505746.0	23	294.06999	190196.5	**26**	293.80331	117828.0	23	294.00330	81912.0
90	2	4	11	346.27390	50771.5	3	348.74052	17740.5	14	346.47388	8526.5	5	348.00719	9290.5
90	2	8	**17**	173.14705	7677.0	3	175.34702	2325.5	1	175.68034	1027.0	8	174.34701	1284.5
90	4	4	**20**	193.74679	15854.5	0	197.88006	6829.5	6	196.08009	2678.5	2	196.81337	2145.5
180	4	4	**15**	718.08096	14634.5	0	722.61423	5184.0	2	719.88094	2710.5	10	719.14761	2098.0
180	4	8	11	374.22726	1687.5	0	379.96048	739.0	0	382.49372	279.0	**19**	372.82722	318.5
180	5	8	10	264.03307	996.5	1	269.83290	510.0	0	272.09952	184.0	**18**	263.56637	194.5
300	6	4	**19**	1241.28831	3209.0	2	1248.82158	1355.5	0	1249.35488	686.5	9	1242.62164	454.5
300	6	8	6	715.10780	322.5	1	715.64113	201.0	0	721.57436	76.0	**23**	707.97455	78.5
300	9	8	1	403.25824	127.5	7	394.39177	114.5	0	402.19153	39.5	**19**	392.32501	35.0
400	8	4	14	1654.16199	1179.5	1	1658.49534	612.0	0	1660.69526	343.0	**15**	1654.09532	196.0
400	8	8	0	958.58794	120.0	8	949.05467	91.5	2	954.65456	38.0	**17**	947.45474	37.5
400	12	8	0	543.47730	53.0	**14**	524.34443	52.0	1	530.01100	20.0	12	525.14430	18.0
500	10	4	6	2092.16902	517.0	4	2091.90240	322.0	2	2095.30235	177.0	**17**	2086.90239	95.5
500	10	8	0	1225.46808	55.5	9	1202.93505	49.5	3	1209.33487	22.0	**14**	1202.86835	21.0
500	15	8	0	690.22984	26.5	**16**	660.43057	28.0	3	667.03054	13.0	6	662.76393	11.0
Total			282	12598.74956	2480782.0	200	12548.81743	1045598.0	191	12588.94984	975653.5	**321**	12505.08426	955213.0

mid-size and large instances. Thus, it is most interesting to concentrate on instances of smaller size for these PR variants. We observe that for these smaller instances it does not make sense to apply the limited VND since only a few solutions are actually improved. Moreover, ST-Rec yields significantly better results than the original OP-Rec operator because we destroy less well working parts of the tours.

Comparing with state-of-the-art approaches (i.e., VNS and PILOT-GRASP), the PR extensions are able to boost the performance of GRASP on medium instances which have so far been dominated by VNS. Among the PR-GRASP variants, we observe that MS-PR-GRASP is superior on medium and large instances with 180 to 500 stations whereas ST-Rec-GRASP is able to enhance GRASP for small instances with 60 to 90 stations. While VNS still performs best on medium instances with 90 stations in overall, the results indicate that our new GRASP variants are able to catch up on instances with smaller time budgets of 2 and 4 hours. These observations are confirmed by the Wilcoxon Rank Sum test on an error level of 5%.

7 Conclusions and Future Work

To improve results of a state-of-the-art PILOT-GRASP hybrid for the static BBSS variant, we analyzed different variants of PR and recombination operators to obtain new promising candidate solutions by joining parts of two parental solutions. Seq-PR follows a very traditional way of performing PR and introduces a quite high computational overhead. In contrast, OP-Rec is an implementation of a fast, rather classical one-point crossover in the context of our solution representation. It turned out that intermediate solutions that are constructed from large parts of both parent solutions are in general less promising than when most properties are inherited from one parent and only a smaller portion is adopted from the second parent. Consequently, we came up with ST-Rec, which performs the crossover only on a single tour and adopts all other tours from a single parent. It turned out that this generally less disruptive recombination yields significantly better results. We then also modified Seq-PR into MS-PR, where we investigate only parts of whole paths from an initial solution to the guiding solution that are close to the guiding solution and skip solutions in the middle of tours. This modification was most fruitful and yields the best results. In this way we could obtain new leading solutions and significantly improved average results especially on larger instances with up to 500 nodes.

More generally, we strongly believe that also in other problems, the parts of path relinking trajectories close to either the initial or the guiding solutions may be more promising than the middle ones, and consequently, focusing the search on those ends may yield significant improvements or speedups. With respect to BBSS, we intend to integrate PR also in our GRASP-variant for the dynamic BBSS [12], where user interactions during rebalancing are also considered.

References

1. DeMaio, P.: Bike-sharing: History, impacts, models of provision, and future. Public Transportation **12**(4) (2009) 41–56
2. Rainer-Harbach, M., Papazek, P., Hu, B., Raidl, G.R.: Balancing bicycle sharing systems: A variable neighborhood search approach. In Middendorf, M., Blum, C., eds.: Evolutionary Computation in Combinatorial Optimization. Volume 7832 of LNCS., Springer (2013) 121–132

3. Chemla, D., Meunier, F., Calvo, R.W.: Bike sharing systems: Solving the static rebalancing problem. Discrete Optimization **10**(2) (2013) 120–146
4. Raviv, T., Tzur, M., Forma, I.A.: Static repositioning in a bike-sharing system: models and solution approaches. EURO Journal on Transp. and Log. (2013) 1–43
5. 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) (2011) 37–61
6. Schuijbroek, J., Hampshire, R., van Hoeve, W.J.: Inventory Rebalancing and Vehicle Routing in Bike Sharing Systems. Technical Report 2013-E1, Tepper School of Business, Carnegie Mellon University (2013)
7. Papazek, P., Raidl, G.R., Rainer-Harbach, M., Hu, B.: A PILOT/VND/GRASP hybrid for the static balancing of public bicycle sharing systems. In: Proceedings of the 14th Int. Conf. on Computer Aided Systems Theory. Volume 8111 of LNCS., Springer (2013) 372–379
8. Rainer-Harbach, M., Papazek, P., Hu, B., Raidl, G.R.: PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems. Journal of Global Optimization (2013) DOI:10.1007/s10898-014-0147-5.
9. Di Gaspero, L., Rendl, A., Urli, T.: A hybrid ACO+CP for balancing bicycle sharing systems. In Blesa, M., Blum, C., Festa, P., Roli, A., Sampels, M., eds.: Hybrid Metaheuristics. Volume 7919 of LNCS. Springer (2013) 198–212
10. Di Gaspero, L., Rendl, A., Urli, T.: Constraint-based approaches for balancing bike sharing systems. In Schulte, C., ed.: Principles and Practice of Constraint Programming. Volume 8124 of LNCS. Springer (2013) 758–773
11. Contardo, C., Morency, C., Rousseau, L.M.: Balancing a dynamic public bike-sharing system. Technical Report CIRRELT-2012-09, Montreal, Canada (2012)
12. Kloimüllner, C., Papazek, P., Hu, B., Raidl, G.R.: Balancing bicycle sharing systems: An approach for the dynamic case. In: Evolutionary Computation in Combinatorial Optimization. (2014) 12 pages, to appear.
13. Lin, J.R., Yang, T.H., Chang, Y.C.: A hub location inventory model for bicycle sharing system design: Formulation and solution. Computers & Industrial Engineering **65**(1) (2013) 77–86
14. Nair, R., Miller-Hooks, E., Hampshire, R.C., Bušić, A.: Large-scale vehicle sharing systems: Analysis of Vélib'. Int. Journal of Sustain. Transp. **7**(1) (2013) 85–106
15. Voß, S., Fink, A., Duin, C.: Looking ahead with the PILOT method. Annals of Operations Research **136** (2005) 285–302
16. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers and Operations Research **24**(11) (1997) 1097–1100
17. Resende, M., Ribeiro, C.: Greedy randomized adaptive search procedures. In Glover, F., Kochenberger, G., eds.: Handbook of Metaheuristics. Kluwer Academic Publishers (2003) 219–249
18. Glover, F., Laguna, M., Marti, R.: Fundamentals of scatter search and path relinking. Control and Cybernetics **29**(3) (2000) 653–684
19. Ho, S.C., Grendreau, M.: Path relinking for the vehicle routing problem. Heuristics **12**(1-2) (2006) 55–72
20. Rahimi-Vahed, A., Crainic, T., Gendreau, M., Rei, W.: A path relinking algorithm for a multi-depot periodic vehicle routing problem. Heuristics **19**(3) (2013) 497–524
21. Festa, P., Resende, M.: Hybridizations of GRASP with path-relinking. In Talbi, E.G., ed.: Hybrid Metaheuristics. Volume 434 of Studies in Computational Intelligence. Springer (2013) 135–155