An improved GRASP for the bike-sharing rebalancing problem

Haitao Xu, Jing Ying

Zhejiang University, Hangzhou, Zhejiang 310007, China;
citycomputing@163.com

Abstract—**A bike-sharing system is a service in which bicycles are made available for shared use to individuals on a very short term basis. Due to varying user demands in bike-sharing systems, employees need to actively shift bicycles between stations by a fleet of vehicles. In this paper, an improved greedy randomized adaptive search procedure(GRASP) algorithm is utilized to find efficient vehicle tours. Firstly, a greedy construction heuristic is used to construct an initial solution, and then a local search algorithm is used to improve the solution. In addition, some improvements are made in some phases of the algorithm according to the feature of rebalancing problem. Practice examples and comparison with the typical algorithm in the fields are made. The results show that the proposed algorithm is efficient and it can produce good results. The research result has been implemented in Hangzhou.**

Keywords-Bike-sharing System; Bicycle Rebalancing; Improved GRASP

I. INTRODUCTION

Bike-sharing system has grown rapidly in the past decade, as an answer to an increasing need of green and versatile public transportation in cities. By 2016, the number of cities offering bike-sharing systems has increased from just a handful in the late 1990s to over 1200. On May 1, 2008, Hangzhou city government launched the first information technology–based bike-sharing system in mainland China. Now, Hangzhou city bike-sharing system has covered over 3000 service stations, a total of about 60000 bicycles. It has surpassed Vélib's bicycle system as the largest bike sharing program in the world.

Like many cities Hangzhou public bicycle is unattended, sometimes it is a common problem to find no bicycle to rent or no place to return at some stations. In order to avoid these situations, employees should actively move bicycles between stations, usually by a fleet of vehicles. The city is divided into some areas on which only one vehicle operates, then each vehicle has to work on about 50-150 stations. This paper focus on this part of this operating problem: how to deal with a part of the city assigned to a single vehicle. This problem is called the bike-sharing rebalancing problem by some academics. The primal goal of this research is to find out an efficient algorithm to deals with optimizing the vehicle tours together with corresponding loading or unloading directions.

The main contributions of the current work are summarized as follows.

- To the best of our knowledge, the algorithm we develop in this paper is the second attempt to solve the bike-sharing rebalancing problem using GRASP.
- We model the bike-sharing rebalancing problem and propose a new objective function to the problem, which meets the actual circs better.
- We build a rebalancing system by using a real-world dataset generated by Hangzhou bike-sharing system. Experiments show that our method is effective.

The rest of the paper is organized as follows: the second part describes related work of the problem. In the third part, the bike-sharing rebalancing problem is presented. In the fourth section, we first give a high-level description of a general GRASP with path-relinking heuristic. Then the improved GRASP for the bike-sharing rebalancing problem is described in the fifth section. Computational results obtained from experiments on real-world data of Hangzhou are reported in the sixth section. Concluding remarks are drawn in the last section.

II. RELATED WORK

There are some studies on the bike-sharing rebalancing problem. Because public bicycles are an emerging transportation tool, the amount of papers on it is not much so far. But the amount is growing rapidly. From these papers, we can find that the rebalancing problem is not uniquely defined. The problem is complicate enough, so various studies make different assumptions. Some authors viewed the problem is similar to the one-commodity pickup-and-delivery traveling salesman problem. Some studies consider the total distance travelled by the rebalancing vehicle as the objective function. Some studies consider the total amount of pickup-and- delivery as the objective function.

Chemla et al. propose a math-heuristic for the single vehicle version of the problem, assuming each station can be visited more than once in [1]. In that paper, a branch-and-cut algorithm is proposed for solving a relaxation of the problem. The objective function is to find a route that minimizes the total traveling cost. Erdogan et al. [2] think that there are some degrees of flexibility in the desired number of bicycles at each station at the end of the rebalancing. They present a branch-and-cut algorithm and a Bender decomposition that allow solving instances with up to 50 stations. The rebalancing problem with a service level constraint is studied by Schuijbroek et al. in [3]. Variable neighborhood search heuristics are proposed by Rainer-Harbach et al.[4] and Raidl et al. [5] for the rebalancing problem that take the loading and unloading times into account. Iris A.Petrina

978-1-5386-2813-3/17 $31.00 © 2017 IEEE
DOI 10.1109/ICSGEA.2017.117

Papazek et al. [6] proposed a GRASP hybrid for the problem. But their primary objective is to minimize the absolute deviation between target and final fill levels for all rental stations. This method suits for their test data in Vienna. But it may be not suitable for many congestion cities. Forma et al. [7] proposed a 3-step mathematical programming based heuristic.

III. THE BIKE-SHARING REBALANCING PROBLEM

Daniel Chemla et al. have done a good work on modelling the bike-sharing rebalancing problem in [1]. In this paper, we employ its problem definition mostly and make some modification in light of the conditions.

The bike-sharing rebalancing problem can be formalized as follows. Let G = (V,A) be a complete directed graph where V = {0, . . . , n} is the node set composed by n + 1 nodes, the nodes in {1, . . . , n} representing rental stations and the node 0 representing the garage, and where A is the set of arcs. Each station i ∈ V has associated a capacity Qi ⩾ 0, i.e., the number of available parking positions, the number of available bikes at the beginning of the rebalancing process p_i ⩾ 0, and a target number of available bikes after rebalancing q_i ⩾ 0. A station is in excess (resp. in default) if $p_i > q_i$ (resp. $p_i < q_i$). Some station can be initially balanced i.e. $p_i = q_i$. Moreover, throughout the paper the imbalance di $=p_i - q_i$ is used and the garage is always assumed to have no bike i.e. $Q_0 = p_0 = q_0 = 0$ and $d_0 = 0$. For each arc (i, j) ∈ A, we denote by C(i,j) the cost of the arc (i, j). The cost is assumed to satisfy the triangular inequality (i.e., C(i, j) + C(j, k) ⩾ C(i, k) for all i, j, k ∈ V). The public bicycle system operator employs a fleet of vehicles for moving bikes. Each vehicle services one area of city. It starts empty at the garage 0, has a capacity of z bikes and may visit an arbitrary number of stations before returning empty to the garage again as long as the total tour length does not exceed an available time budge $\hat{t}$.

A feasible solution for the bike-sharing rebalancing problem is called a route, which consists of two parts. The first part is a sequence of nodes, starting and finishing with the depot 0, together with bike displacements within the limits of capacity constraints, at the end of which the system is balanced: each station i has been brought from its initial state pi to its target state qi. The route is specified by an ordered sequence of visited stations $\mathrm{r} = (r_0, r_1, \dots, r_n, r_0)$ with r_i ∈ V, i = 1,...,n and n representing the number of stations. The second part consists of loading and unloading instructions $y_v^{+,x}, y_v^{-,x} \in \{0, \dots, Q\}$ with v ∈ V, and x = 1,..., N, specifying how many bikes are picked up or delivered, respectively, at station v at the x-th stop of the vehicle. It should be noted that $\forall \mathrm{v} \neq r_i: y_v^{+,x} = y_v^{-,x} = 0$.

The goal of the bike-sharing rebalancing problem is to find the minimum cost route. It is notable that the convergence for each node from pi to qi is not required to be monotonous: drops and multiple visits of the stations are allowed, i.e. bikes can be loaded from stations in default or unloaded at stations in excess. Figure1 shows an example of instance with 9 nodes. Node 0 is the garage and node 1-8 are stations. Among them, there is one initially balanced station. The pair of (p_i, q_i), which represent the number of available bikes at the beginning of the rebalancing process and a target number of available bikes after rebalancing, is displayed next each node. The capacity of the vehicle is set to 8. Figure1 also shows a feasible solution with station sequence (0,7,6,4,5,8,1,2,0).

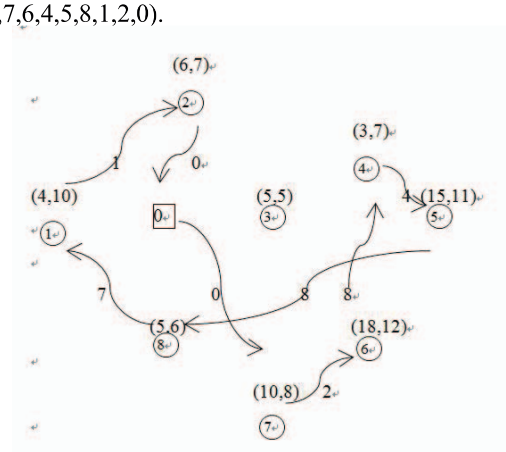

Figure 1. Example of a feasible solution

The objective of the rebalancing problem is to find a set of routes that minimize the total cost. The objective function can be written as:

$$\min \sum_{(i,i')\in A} \sum_{t=1}^{\beta_i} \sum_{t'=1}^{\beta_{i'}} c(i,i') z_{i,t,i',t'}$$

s. t.

$$x_{i,t} \leq x_{i,t-1} \quad \forall i \in V, \forall t \in \{2, \dots, \beta_i\} \qquad (1)$$

$$y_{i,t,i',t'} < Q z_{i,t,i',t'} \quad \forall i, i' \in V, \forall t \in \{1, \dots, \beta_i\}, \forall t' \in \{1, \dots, \beta_{i'}\} \qquad (2)$$

$$0 \leq \sum_{i' \in V\{i\}} \sum_{\tau=1}^{t} \sum_{\tau'=1}^{\beta_{i'}} (y_{i',\tau',i,\tau} - y_{i,\tau,i',\tau'}) + p_i \leq C_i \quad \forall i \in V, \forall t \in \{1, \dots, \beta_i\} \qquad (3)$$

$$\sum_{i' \in V\{i\}} \sum_{\tau=1}^{\beta_i} \sum_{\tau'=1}^{\beta_{i'}} (y_{i',\tau',i,\tau} - y_{i,\tau,i',\tau'}) + p_i = q_i \quad \forall i \in V \qquad (4)$$

$$z_{i,t,i',t'}, u_{i,t,i',t'} \in \{0,1\} \quad \forall i, i' \in V, \forall t \in \{1, \dots, \beta_i\}, \forall t' \in \{1, \dots, \beta_{i'}\} \qquad (5)$$

$$x_{i,t} \in \{0,1\} \quad \forall i \in V, \forall t \in \{1, \dots, \beta_i\} \qquad (6)$$

In the above formula, β_i is an upper bound of the number of times vehicle has to visit the station i in any optimal solution. $x_{i,t}$=1 only if the station i is visited at least t times. $z_{i,t,i',t'} = 1$ only if the vehicle visits station i' for the t'th time just after having visited i for the t th time. $y_{i,t,i',t'}$ is the number of bikes that are carried by the vehicle during this move.

According to the definition of $x_{i,t}$, constraint (1) holds. Constraint (2) limit the number of bicycles transported between two stations to the capacity of the vehicle. Constraint (3) makes the number of bicycles parked at a station i not exceed its maximal capacity C_i. Constraints (5) and (6) make sure that all stations should reach their target state after the rebalancing.

IV. General GRASP

Greedy Randomized Adaptive Search Procedure (GRASP) is a multi-start or iterative metaheuristic. It is already applied successfully to many optimization problems. In GRASP, each iteration consists of two phases: construction and local search. The construction phase builds a feasible solution, whose neighborhood is investigated until a local minimum is found during the local search phase. The best overall solution is kept as the result. The pseudo-code of the main blocks of a basic GRASP procedure for minimization is illustrated as follows.

Input:
The number of iterations N.
Output:
The best solution x^*.
Steps:

```
x* ← ∅;
f* ← ∞;
For k=1,…,N do
Begin
   x ← ConstrunctionPhase();
   x ← LocalSearch(x);
   if f(x)<f(x*) then
      x* ← x;
End
Return x*;
```

The construction phase and the local search applied to the constructed solution are executed iteratively. Then the quality of the obtained solution is compared to the current best found and, the best solution is updated if necessary. The best solution is finally returned.

The construction and local search phases of GRASP should be customized for each problem. The general procedure of construction phase is illustrated as follows.

Input:
Null.
Output:
The solution x.
Steps:

```
x ← ∅;
while x is not a complete solution do
begin
   Build the RCL;
   e ← SelectRandom(RCL);
   x ← x ∪ {e};
end
return x;
```

The elements of the solutions are selected one by one from the restricted candidate list (RCL) and incorporated into the partial solution until it is completely built. RCL is built by the best elements, whose incorporation into the current solution results in the smallest incremental costs. The element to be added into the partial solution is randomly selected from those in the RCL. Once the selected element is added into the partial solution, RCL should be updated and the costs need be reevaluated.

The solution which obtained in the construction phase of GRASP usually is not locally optimal. The local search phase which is a hill-climbing process attempts to improve the solutions built in the construction phase. It uses the solution obtained in the construction phase as the starting point and then explores the neighborhood of the solution. The neighborhood of a solution is defined by a function that relates this solution with a set of other solutions. The local search is performed again if a better solution is found. It terminates when no better solution is found in the neighborhood. The pseudo-code of a local search algorithm is as follows.

Input:
The solution x.
Output:
The better solution x.
Steps:

```
while x is not local optimal do
begin
   Find y ∈ Neighbour(x) with f(y) < f(x);
   x ← y;
end
return x;
```

V. Improved GRASP for the Rebalancing Problem

GRASP has been successfully applied for different problems. In this paper, we applied GRASP to the rebalancing problem in public bicycle systems after making some improvement. The framework of the proposed algorithm is as follows:

Input:
The number of iterations N, RCL parameter L.
Output:
The best solution x^*.
Steps:

```
f* ← ∞;
for k=1,…,N do
begin
   x←Construction(L);
   x←LocalSearch(x);
end;
return x*;
```

A. Construction phase

The following code is the pseudo-code of the construction phase of the proposed algorithm. It is similar to the greedy randomized construction of GRASP, which seeks to produce a good-quality starting route. In the code, x is the partial solution, which is under construction in a given iteration. C is the candidate station set with all the remaining stations that can be added to x. $g(c)$ is a greedy function, which computes the value of each candidate station $c \in C$ to the partial solution x. The goal of the bike-sharing rebalancing problem is to find the minimal cost route. So the greedy function $g(c)$ is used for measuring the cost of adding candidate station to the route.

Restricted Candidate List (RCL) consists of candidate stations which can be added according to the value of $g(c)$ at each step. The station to be added to the partial solution x is picked randomly from this list. The parameter named α decides the length of the RCL. When $\alpha = 0$, only the best station with g_{min} will be added to the RCL. In that case the construction process becomes a pure Greedy algorithm. In another way, the construction phase will be completely random when $\alpha = 1$, because all possible stations may be present in RCL. The value of α is one of the important factors that affect the algorithm for global search performance. In the practical application, the value of α needs a number of adjustments based on the experimental results. After a lot of practical experiments, we found that the performance when α is not fixed is better than that when α is fixed. So a parameter set L is used in the algorithm. In every iteration, α is selected randomly from L. The result solution of the construction phase is then used in the local search phase.

Input:
RCL parameter L.
Output:
The solution x.
Steps:
x=$\emptyset$;
compute C with the candidate elements that can be added to x;
while C$\neq \emptyset$ do
begin
for all c$\in$C compute g(c). $g_{min} = min_{c\in C} g(c)$ and $g_{max} = max_{c\in C} g(c)$;
select α from L randomly;
define $RCL \leftarrow \{c \in C | g(c) \leq g_{min} + \alpha(g_{max} - g_{min})\}$ with $\alpha \in [0,1]$;
select c^* from RCL randomly;
$x = x \cup \{c^*\}$;
update C with the candidate elements that can be added to x;
end;
if x is infeasible then
apply a repair procedure to make x feasible;
return x;

B. Local search

2-opt is a simple local search algorithm first proposed by Croes for solving the traveling salesman problem. The main idea of it is to take a route that crosses over itself and the neighborhood. So it will provide a different chance to find a better solution and avoid the algorithm trapping into local minimum. This technique of 2-opt has been widely applied to the travelling salesman problem (TSP), the vehicle routing problem (VRP) as well as many related problems. In this paper, the 2-Opt algorithm is used in the local search phase of GRASP. The local search algorithm is as follows:

Input:
The solution x.
Output:
The new solution x.
Steps:
For i=0,1,…,n-2 do
For j=i+2,…,min{n,i+n-2} do
begin
r = $(r_0, \dots, r_i, r_{i+1}, \dots, r_j, r_{j+1}, \dots, r_n)$ is an ordered sequence of visited stations in solution x,
if $C(r_i, r_{i+1}) + C(r_j, r_{j+1}) > C(r_i, r_j) + C(r_{i+1}, r_{j+1})$ then
begin
r = $(r_0, \dots, r_i, r_j, r_{j-1} \dots, r_{i+1}, r_{j+1}, \dots, r_n)$;
Update the sequence of loading and unloading instructions in x according to r.
end;
return x;
end;

VI. PRACTICAL EXAMPLES

To practice the effectiveness of the improved GRASP, we performed extensive practical experiments. The experimental data are derived from real-world data of Hangzhou public bicycle system. As mentioned above, Hangzhou public bicycle system, which has covered over 3000 service stations, is the largest bike sharing program in the world. The city is divided into some areas on which only one vehicle operates. We performed tests on a PC Intel Core i5 clocked at 2.6 GHz, with 8 GB of RAM. The tests were applied on many areas, which range from 50 to 150 stations. Table1 shows the scale of the practical instances.

Table1. Instance scale

Practical Instance	Number of Stations
E1	50
E2	75
E3	101
E4	132
E5	150

We can know from the above introduction, the proposed algorithm includes construction, local search. There are some parameters in the algorithm. One feature of the algorithm is that its execution result is closely related to the selection of algorithm parameter. Better execution results can be obtained by selecting the appropriate algorithm parameters depending on the application environment.

The hybrid algorithm has the following key parameters: the number of iterations N, RCL parameter L. Table2 shows the parameters value in our practical examples.

Table2. Parameter setting of improved algorithm

Parameter	Value

Number of iterations N	1000
RCL parameter L	{0.5,0.6,0.7,0.8}

To test the validity of the proposed algorithm, we compare the quality of the obtained solutions of simulated annealing (SA) algorithm and the proposed algorithm. SA and the proposed hybrid algorithm were run 10 times. Table3 shows the comparison results. The first column presents the id of the instance. The second and third columns present the best cost values and the average cost values obtained by SA. The fourth and fifth columns present the best cost values and the average cost values obtained by the proposed algorithm.

These results show that the proposed hybrid algorithm was able to improve the quality of the solutions.

Table3. Quality Result Comparison

Practical Instance	SA		GRASP	
	Best	Average	Best	Average
E1	315	323	303	309
E2	458	468	459	465
E3	639	650	624	631
E4	801	813	789	797
E5	957	974	934	947

To test the efficiency of the proposed algorithm, we compare the time of finding the current best solutions. In order to do these tests, the termination conditions were changed. When the cost of current solution is equal to value of the best solution in table3, the algorithm will terminate. Table4 shows the comparison results of these algorithms. In this table, the first column presents the id of the instance. The second, third and fourth columns show the average execution time (in seconds) of SA and the proposed hybrid algorithm, which is obtained for 10 runs.

Table4. Time result comparison

Practical Instance	SA	GRASP
E1	19	18
E2	39	35
E3	58	50
E4	92	89
E5	147	131

We can know from above the introduction that value of α is one of the important factors that affect the algorithm for global search performance. So we also performed experiments to test the performance of the proposed algorithm when α had different values. Table5 shows quality comparison results between α=0 and α is selected randomly from L.

Table5. Quality Result Comparison

Practical Instance	Hybrid Algorithm when α=0		Hybrid Algorithm when $\alpha \in L$	
	Best	Average	Best	Average
E1	305	310	303	309
E2	464	472	459	465
E3	631	643	624	631
E4	796	809	789	797
E5	952	963	934	947

We can see from Table5 that the performance when α is selected randomly from L is better than that when $\alpha = 0$. The reason is that only the best station with g_{min} will be added to the RCL when $\alpha = 0$. This may lose the chance to find the better solution. But we also found that the algorithm is more quickly when $\alpha = 0$.

VII. CONCLUSIONS

GRASP has been successfully applied for different problems. In this paper, an improved GRASP is proposed and some improvements are made. Firstly, a greedy construction heuristic is used to construct an initial solution, and then a local search algorithm is used to improve the solution. In addition, some improvements are made in some phases of the algorithm according to the feature of the bicycle rebalancing problem. Practical experiments show that the proposed GRASP algorithm is effective. Nowadays public bicycle systems are flourishing all over the world. So the work has great practical meaning. The research result has been applied in Hangzhou.

REFERENCES

[1] Chemla D, Meunier F, Calvo R W. Bike sharing systems: Solving the static rebalancing problem[J]. Discrete Optimization, 2013, 10(2):120-146.

[2] Erdoğan G, Laporte G, Calvo R W. The One Commodity Pickup and Delivery Traveling Salesman Problem with Demand Intervals[J]. 2013, 17(1): 74-79.

[3] Schuijbroek J, Hampshire R C, Hoeve W J V. Inventory rebalancing and vehicle routing in bike sharing systems[J]. European Journal of Operational Research, 2017, 257(3):992-1004.

[4] Rainer-Harbach M, Papazek P, Hu B, et al. Balancing Bicycle Sharing Systems: A Variable Neighborhood Search Approach[M]// Evolutionary Computation in Combinatorial Optimization. Springer Berlin Heidelberg, 2013:121-132.

[5] Raidl G R, Hu B, Rainer-Harbach M, et al. Balancing Bicycle Sharing Systems: Improving a VNS by Efficiently Determining Optimal Loading Operations[M]// Hybrid Metaheuristics. Springer Berlin Heidelberg, 2013:130-143.

[6] Papazek P, Kloimüllner C, Hu B, et al. Balancing Bicycle Sharing Systems: An Analysis of Path Relinking and Recombination within a GRASP Hybrid[C]// Parallel Problem Solving from Nature – PPSN XIII. 2014:792-801.

[7] Forma I A, Raviv T, Tzur M. A 3-step math heuristic for the static repositioning problem in bike-sharing systems[J]. Transportation Research Part B Methodological, 2015, 71:230-247.