Australasian Transport Research Forum 2015 Proceedings
30 September - 2 October 2015, Sydney, Australia
Publication website: http://www.atrf.info/papers/index.aspx

An artificial bee colony algorithm for the public bike repositioning problem

C.S. Shui[1], W.Y. Szeto[1]

[1]The University of Hong Kong, Hong Kong

Email for correspondence: samshui@hku.hk

Abstract

Public bike repositioning is crucial in public bike sharing systems due to the imbalanced distribution of public bikes. This paper models the public bike repositioning problem (PBRP) involving two non-linear objectives, which are to minimize total service duration and the duration of the longest vehicle route. It includes practical constraints such as the tolerance of demand dissatisfaction and the limitation of duration on the longest route. These objective functions and constraints make the PBRP become NP-hard, so here introduces an artificial bee colony (ABC) algorithm to solve this PBRP. Three neighbourhood operators are introduced to improve the solution search. A modified ABC is proposed to further improve the solution quality. The performance of the modified heuristic was evaluated with the network of Vélib', and compared with the original heuristic and the Genetic Algorithm. These results may therefore prove that the modified heuristic can be an alternative to solve the PBRP. The numerical studies demonstrated that the two objective functions performed differently in which the increase in fleet size may not improve the objective value. This paper will therefore discuss on the practical implications of the trade-offs and provide suggestions about similar repositioning operations.

1. Introduction

The use of bicycles as a mode of transport has always been a significant issue to the transport system around the globe. Thanks to its economical and environmentally friendly nature, people have been placing more and more importance on it. In order to make the bicycle trips more convenient and efficient, the concept of bike-sharing was introduced to allow automatic rental use of bicycle in specified stations within a city and return in any other stations whenever a short-distance trip is required (Raviv, et al., 2013). In order to provide enough bikes at each station, repositioning of the bikes has become the largest concern of the operating companies. With a fixed number of bikes within the city, the operators need to transfer the bikes from bike surplus stations to bike deficient stations usually by truck. The problem therefore becomes determining the routes of the repositioning vehicles so that the bikes can be repositioned efficiently and effectively.

PBRP has attracted the interest of many researchers in recent years and can be modelled as either static or dynamic optimization problem (Raviv, et al., 2013). This study focuses on static repositioning activity at night, which has been studied by a number of publications (e.g. Benchimol, et al., 2011; Chemla, et al., 2013; Di Gaspero, et al., 2013; Nair, et al., 2013; Erdoğan, et al., 2014; Ho & Szeto, 2014; Rainer-Harbach, et al., 2014). More recently, Schuijbroek, et al. (2013) propose a new cluster-first route-second algorithm to decompose the problem into separate vehicle routing problems so as to solve the service level requirement and repositioning activity together. Dell'Amico, et al. (2014) presented four MILP models considering multiple vehicles with both capacitated and uncapacitated cases and developed a tailor-made branch-and-cut algorithm to solve them. Raviv, et al. (2013) presented two mixed integer linear programming (MILP) formulations that have included two vehicle cases and their focus were to minimize the demand dissatisfaction without considering the tour lengths. Exact and heuristic methods are proposed to solve the problem with shorter computation time.

In this paper we consider the PBRP about determining the routing sequences of multiple repositioning vehicles, so as to minimize firstly the demand dissatisfaction and subsequently the service duration of the operating vehicles. Rather than achieving perfect balance or minimum deviations, this paper introduces the concept of tolerance of demand dissatisfaction (TDD), in which the total deviation of target level of bikes from all routes should not exceed a predefined tolerable limit. Also, as the routing problem is a combinatorial problem, larger network involves more binary variables and creates large

number of combinations, which require high computation time to solve by commercial optimizers. To solve the routing problem effectively, past studies adopted different algorithms, including Simulated Annealing (Martinovic, et al., 2009), Genetic Algorithm (Zhao, et al., 2009) and Variable Neighbourhood Search (Hosny & Mumford, 2010). This study proposes Artificial Bee Colony algorithm (ABC) to solve the problem as the ABC is not bounded by the mathematical properties of the objectives and constraints so it can find the near optimal solutions with much shorter computational time (Szeto, et al., 2011). ABC is shown to be effective in solving various combinatorial problems including vehicle routing problem (Brajevic 2011; Szeto et al. 2011), minimum routing cost spanning tree problem (Singh and Sundar 2011) and job shop scheduling problems (Li et al. 2011).

The outline of the paper is listed below. Section 2 discusses the problem setting and formulation of the repositioning shared bikes problem. Section 3 describes the solution method based on ABC and the modifications on the operators. Section 4 analyses the computational results and discusses on the findings related to operating strategies. Finally, conclusions are given in Section 5.

2. Mathematical formulation

To facilitate the presentation of the mathematical formulation, the basic settings of this repositioning problem are firstly described. For a network with n bike stations, a number of vehicles $|V|$ is employed for bike repositioning among these stations and each vehicle is assumed to be capacitated and have the same maximum capacity k. It is assumed that the exact and targeted number of bikes in every station is known in advance, so the number of bikes needed to be loaded or unloaded can be determined before the repositioning activity. As the repositioning activity is carried out during night time, the number of bikes in all stations is constant throughout the whole journey of the vehicle. Only one depot is used in this paper. Every vehicle therefore starts from the depot at the same time, travels to the bike stations assigned to it, and finally returns to the depot. Moreover, each station is assumed to be visited once by exactly one vehicle, regardless of the number of vehicles employed. The objective of this paper is to determine the optimal routes for all operation vehicles to travel to all stations, while effective and efficient routes can be achieved by minimizing (1) demand dissatisfaction and (2) service time of the operating vehicles. Demand dissatisfaction here is defined as the sum of outstanding bikes of all stations after the repositioning activity. This paper, therefore, proposes two mathematical models to determine the route sequences of the fleet of repositioning vehicles.

2.1 Model 1

The first objective is to minimize the total service times for the whole fleet of repositioning vehicles. For the service time, in addition to the travel time, this paper takes the loading and unloading time into consideration. The sets and the notations adopted in this study are stated below:

Set/ Indices

N_0 set of nodes;

N set of nodes excluding depot;

V set of vehicles;

i, j indices of nodes;

v index of vehicles.

Parameters

s_i number of excess bikes at node i before the repositioning operation starts;

d_i number of outstanding bikes at node i before the repositioning operation starts;

k capacity of the operating vehicle;

M Very large positive constant, 100,000;

t_{ij} traveling time from stations *i* to *j*;

L time needed to remove a bike from a bike station and load it onto a vehicle;

U time needed to unload a bike from a vehicle and hook it to a locker in a bike station;

TDD tolerance of demand dissatisfaction (or tolerance limit);

T_{total} total service time of all vehicles;

t_T maximum service duration for the repositioning activity.

Decision variables/ Functions

x_{ijv} 1 if vehicle *v* directly travels from node *i* to node *j*; 0 otherwise;

q_{ijv} number of bikes carried by vehicle *v* when it travels directly from nodes *i* to *j;*

$D_{dis,v}$ total demand dissatisfaction along the whole route of vehicle *v*;

$D_{sur,v}$ total demand surplus along the whole route of vehicle *v;*

T_v travel time of the whole route of vehicle *v;*

S_v service time of the whole route of vehicle *v;*

g_{iv} auxiliary variable associated with node i of vehicle v used for the sub-tour elimination constraint.

Based on the above notations, model 1 is formulated as follows:

$$\min Z = \max\left\{\sum_{v\in V} D_{dis,v} - TDD, 0\right\} + \sum_{v\in V}\left(T_v + S_v\right) \tag{1}$$

subject to

$$D_{dis,v} = \sum_{j\in N_o}\left\{\max\left(d_j - \sum_{i\in N_0} q_{ijv}, 0\right)\right\} \qquad \forall v \in V \tag{2}$$

$$D_{sur,v} = \sum_{j\in N_o}\left\{\max\left(\sum_{i\in N_0} q_{ijv} + s_j - k, 0\right)\right\} \qquad \forall v \in V \tag{3}$$

$$T_v = \sum_{i\in N_0}\sum_{j\in N_0, j\neq i} x_{ijv} t_{ij} \qquad \forall v \in V \tag{4}$$

$$S_v = \left(\sum_i\sum_j x_{ijv} s_j - D_{sur,v}\right) L + U \qquad \forall i, j \in N_0, \forall v \in V \tag{5}$$

$$q_{ijv} \le k x_{ijv} \qquad \forall i, j \in N_0, i \neq j, \forall v \in V \tag{6}$$

$$\sum_{j\in N_0, j\neq i} x_{ijv} = \sum_{j\in N_0, j\neq i} x_{jiv} \qquad \forall i \in N_0, \forall v \in V \tag{7}$$

$$\sum_{v\in V}\sum_{j\in N_0, j\neq i} x_{ijv} = 1 \qquad \forall i \in N \tag{8}$$

$$T_v + S_v \le l_T \qquad \forall v \in V \tag{9}$$

$$g_{jv} \ge g_{iv} + 1 - M\left(1 - x_{ijv}\right) \qquad \forall v \in V, i \in N_0, j \in N, i \ne j \tag{10}$$

$$x_{ijv} = \{0,1\} \qquad \forall v \in V, i \in N_0, j \in N, i \ne j \tag{11}$$

$$D_{dis,v}, D_{sur,v} \in \mathrm{Z}^+ \tag{12}$$

$$q_{iv} \ge 0 \qquad \forall v \in V, i \in N_0 \tag{13}$$

Equation (1) is the objective function of model 1. Similar to the problem studied by Raviv et al., it aims firstly to minimize the demand dissatisfaction and then the total service time of all repositioning vehicles. The TDD term guarantees the routes that the demand dissatisfaction is lower than it are not penalized. Constraints (2) to (3) are related to the demand dissatisfaction. Constraint (2) defines the total demand dissatisfaction of the bike stations served by vehicle *v*. Constraint (3) defines the total bicycle surplus in the stations served by vehicle v. Constraint (4) defines the travel time of each vehicle, while constraint (5) defines the overall loading and unloading times along its repositioning route. Constraint (6) limits the quantity of bikes carried on each vehicle to its capacity. Constraint (7) ensures that the conservation of vehicle flow. Constraint (8) ensures each node is visited exactly once by one of the vehicle only. Constraint (9) ensures that all repositioning vehicles should finish its activity within the required service duration. Constraint (10) is the sub-tour elimination constraints. Finally constraint (11) defines x_{ijv} to be a binary variable. Constraint (12) restricts both bike deficiency and surplus for each vehicle route to be non-negative integers. Constraint (13) ensures the auxiliary variables to be non-negative.

2.2 Model 2

The second objective is to minimize the maximum route duration, which is the maximum travel time and loading and unloading time of an individual vehicle, among all vehicles. This maximum route duration is significant with the practical consideration of the service duration. Note that the shorter the maximum route duration is, the more is the flexibility of the service periods. Considering the importance of maximum service duration, model 2 is formulated as below:

$$\min Z = \max\left\{\sum_{v \in V} D_{dis,v} - TDD, 0\right\} + \arg\max_{v \in V}\left(T_v + S_v\right) \tag{14}$$

subject to equations (2) – (13)

Equation (14) is the objective function of model 2. Similar to model 1, model 2 is firstly to minimize the demand dissatisfaction, but it subsequently minimizes the maximum route duration within the whole fleet. Meanwhile, these models share the same set of constraints, in which both models are bounded by the same conditions.

3. Solution method

3.1 Artificial Bee Colony (ABC) Algorithm

The ABC algorithm is a swarm based meta-heuristic algorithm, which was introduced by Karaboga in 2005 to solve problems related to numerical optimization. The algorithm was developed based on the intelligent behaviour of the honey bees' foraging process (Karaboga, 2009). This process consists of three different types of bees:

(1) Employed Bees

Each employed bee is assigned to a food source. It is responsible for collecting nectar from that food source and fly back to its hive to share the information of the food source, including location, profitability of the nectar in that food source, etc., with other honey bees who are unemployed.

(2) Onlooker Bees

All onlooker bees are unemployed and waiting at the hive. The employed bees will carry out a process called "waggle dance" to share the information of its assigned food source with the onlooker bees. After that, each onlooker bee will choose a food source by probability. The more profitable the food source is, the higher chance the onlooker bees will choose that food source.

(3) Scout Bees

If a food source does not have profitable nectar any more, the employed bees will abandon that food source and become scout bees. All scout bees are unemployed and will choose a new source near their hive randomly.

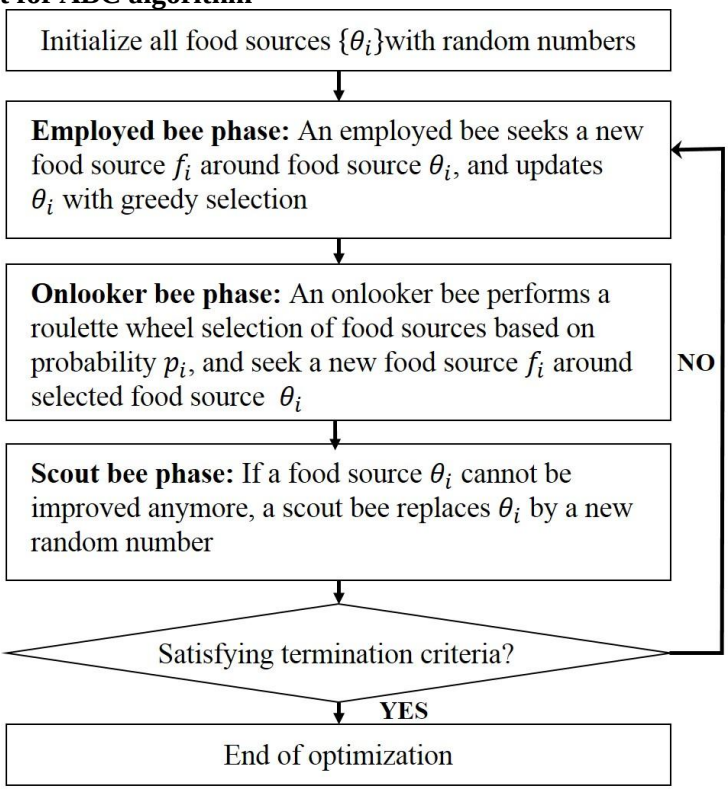
The ABC algorithm makes use of two characteristics of the foraging behaviour: recruitment of foragers to rich food sources giving positive feedback and abandonment of poor sources by foragers leading to negative feedback (Karaboga, 2009). Its critical part is to repeat this foraging process in order to keep searching better food sources so the ABC algorithm is regarded as an iterative algorithm, and therefore a stopping criterion is applied to terminate the foraging process. The outline of the algorithm is illustrated in Figure 1, and the details of the algorithm are discussed below:

First, a certain number of food sources θ_i are randomly generated. Each employed bee is assigned to a food source and the fitness of each food source is evaluated. After that, each employed bee searches a new food source f_i around a food source θ_i using a neighbourhood operator and the fitness of f_i is evaluated. If the new source is fitter than that of the old one, it replaces the old source and the employed bee changes to exploit the new food source.

When the employed bees go back to their hive, it shares the information with the onlooker bees. Each onlooker bee chooses a food source θ_i by a roulette wheel selection method. The higher the fitness value of the food source, the larger chance the food source is chosen. Then, each onlooker bee searches a new food source f_i near to its selected food source θ_i by a neighbourhood operator and the fitness of the new food source f_i is evaluated. If the fitness value of the new one is better than that of the old one, it replaces the old food source. For a food source that has more than one onlooker bee, the best new food source replaces the old one.

If a food source has applied a neighbourhood operator for a certain number of times, called "Limit", it is expected that the quality of the food source cannot be improved. The food source is abandoned and the employed bee assigned to that food source becomes a scout bee and searches for a new food source randomly. Again, each employed bee is assigned to a food source. The whole foraging process is repeated. The foraging process terminates when the number of predefined "Maximum Cycle" is reached.

Figure 1 Flow chart for ABC algorithm



3.2 Solution representation

Each ABC solution is represented by a sequence of the bike stations. Each station is given an index number, e.g. 1 to 10. And the depot is represented by 0. And the number of 0s used in a sequence determines the number of vehicles which are used in that solution. For example, if a solution has 10 stations which are travelled by 3 vehicles, the representation is:

$$0 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 0 \rightarrow 2 \rightarrow 6 \rightarrow 3 \rightarrow 0 \rightarrow 8 \rightarrow 10 \rightarrow 7 \rightarrow 9 \rightarrow 0$$

The first vehicle departs from the depot, passing through station 1, 4, and 5 subsequently; and then travels back to the depot. The second vehicle departs from the depot, passing through station 2, 6, and 3 subsequently; and then travels back to the depot. The third vehicle departs from the depot, passing through station 8, 10, 7, and 9 subsequently; and the travels back to the depot. All vehicles are assumed to depart from the depot simultaneously.

3.3 Neighborhood operators

To provide solution variety, three neighbourhood operators are proposed to have local search of the solution. They are used to alter the positions of different bike stations in a solution sequence so that a new solution can be obtained from the old solution. The *Swap* operator randomly selects two particular bike stations in a solution sequence and swaps their positions. The *Reverse* operator selects a sub-sequence with random length in a solution sequence and reverses the order of that sub-sequence. The *Swap Reverse* operator combines the characteristics of the two above operators. First, it selects two sub-sequences with random lengths in a solution sequence and swaps their positions. Then, each swapped sub-sequences has a 50% chance to carry out the Reverse operator.

3.4 Selection of Food Sources

As mentioned in section 3.1, during each Onlooker Bee Phase, a food source s_i is selected by a roulette selection method for randomly selecting a food source. The probability of choosing that food source $P(s_i)$ is calculated as:

$$P(s_i) = \frac{f(s_i)}{\sum_{i=1}^{N} f(s_i)}, i = 1, \ldots, n \text{ where } f(s_i) = \frac{1}{Z(s_i)}.$$

3.5 Modified ABC algorithm (ABC)

It is shown that the basic ABC algorithm can solve certain problems with great success. However, according to our computational experiments in section 4, this is not the case for the repositioning problem. Therefore this section provides a modified ABC algorithm to improve the performance of the basic ABC algorithm through modifying the steps in the Onlooker Bee Phase and Scout Bee Phase.

Onlooker Bee Phase

After applying the neighbourhood operator and compare the two solutions, the new solution replace an old solution among all the food sources by fulfilling two criteria: (1) the value of Limit of the food source is the largest among all the existing food sources, which implies that the food source has been improved for the largest number of times and (2) the new solution is better than the old solution. This modified approach allows a larger chance for potential food sources to produce better neighbour solutions as well as excluding non-potential food sources which have been improved for a relatively large number of times and have been worse than the new solutions (Szeto, et al., 2011).

Scout Bee Phase

An old solution is replaced by a randomly generated new solution if the old one has reached the value of Limit. This procedure is modified as: if an old solution has reached the value of Limit, a neighbourhood operator is applied to it to generate a new solution. The new solution can be better or worse than the original one. This modified approach may be able to limit the search in bad regions of the solution space with no control of the quality of the food sources (Szeto, et al., 2011).

The steps of the MABC are summarized as follows:

1. A certain number of food sources s_i are randomly generated, where $i = 1, \ldots, N$.
2. Each employed bee is assigned to a food source. Each food source's fitness value $f(s_i)$ is evaluated.
3. Initialize $M = 0$ and $L_1 = L_2 = \ldots = L_N = 0$,
   where M = No. of times of repeating a whole foraging process;
   L_i = No. of times of applying a neighbour operator to food source i,
   $i = 1, \ldots, N$.
4. The foraging process is repeated:
   a. Employed Bee Phase
      i. Each food source is applied by a neighbourhood operator: $s_i \rightarrow \tilde{s}$
      ii. If $f(\tilde{s}) > f(s_i)$, s_i is replaced by $\tilde{s}$ and $L_i = 0$. Otherwise $L_i = L_i + 1$
   b. Onlooker Bee Phase
      i. Each onlooker bee selects a food source using the roulette wheel selection method based on their fitness values
      ii. If $f(\tilde{s}) > f(s_i)$, select s_j where L_j is the maximum among all food sources
      iii. If $f(\tilde{s}) > f(s_j)$, s_j is replaced by $\tilde{s}$ and $L_j = 0$. Otherwise $L_i = L_i + 1$.
   c. Scout Bee Phase
      i. For each food source, if L_i = Limit, it will be applied by a neighbourhood operator: $s_i \rightarrow \tilde{s}$ and s_i is replaced by $\tilde{s}$.
      ii. $M = M + 1$.
5. The foraging process is stopped when M = Maximum Cycle.

4. Computational Results

4.1 Comparison of neighbourhood operators

Experiments here were conducted to determine the most suitable neighbourhood operator for the MABC algorithm. Three different neighbourhood operators are considered with the MABC algorithm on instance 1v60n (i.e. 1 vehicle passing 60 bike stations) with TDD = 0 to compare their effectiveness in solution search, including (1) random swaps (Swap); (2) reversing a subsequence (Reverse); and (3) random swaps of reversed sub-sequences (Swap_Reverse). The value of Limit was set to be a fixed value, which equals to 500n/3 (i.e. 10,000 for n = 60), and a total number of 180,000 iterations was adopted. Table 1 shows the results found by each tested operator. It can be seen that the MABC algorithm with the neighborhood operator Reverse achieves the best average objective value, while Swap and Swap_Reverse did not yield promising results. The search using only the operator Swap may be too diversified, and may not lead to promising regions of the solution space. Therefore, a combination of the three operators, 'Combined', has been proposed and experimented, in which all 3 operators are selected with an equal probability. Using this combined operator, the MABC algorithm was run and obtained the best minimum and average objective value, as shown in Table 1. In addition, its computation time is shorter than the MABC algorithm combined with Swap_Reverse and comparable to the one combined with Reverse.

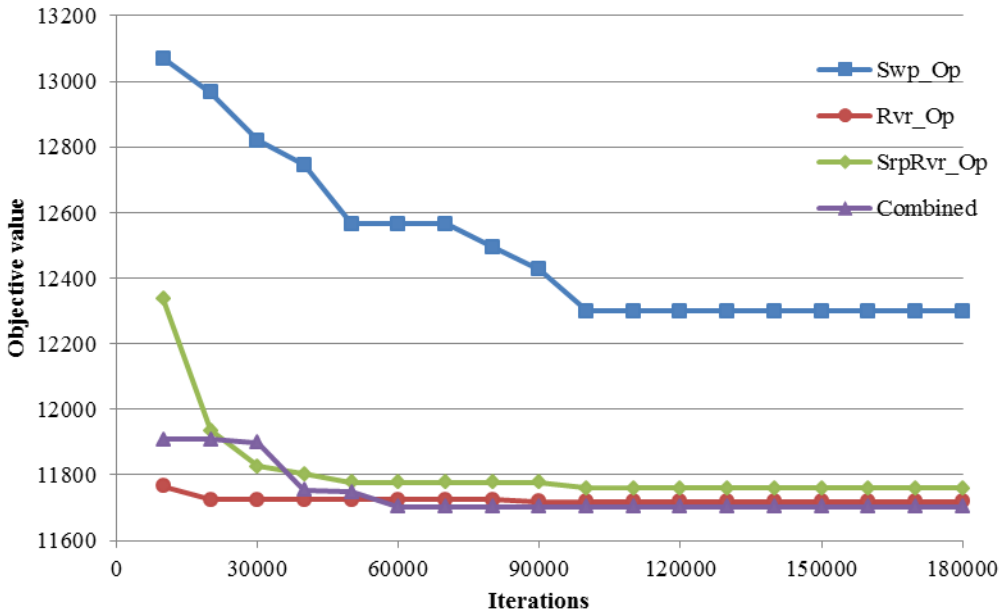
Figure 2 shows the plot of the best objective solution values found by the MABC algorithm using the operators Reverse, Swap_Reverse and Combine during one run on instance 1v60n with objective 1. It can be seen that the heuristic combined with Reverse converges faster than the heuristics combined other two operators. The MABC algorithm with the Swap_Reverse operator converges at a later stage and its solution is the worst among the three. Moreover, their solution qualities are not good compared with those obtained from the heuristc with the Combined operator. Although the heuristic combined with Combine has the slowest convergence rate, it leads to the best solution quality than the other two variants. This conclusion could also be drawn when these heuristics were experimented in other test instances. Therefore, for this reason, the setting of Combined was used in the remaining parts of this section.

Table 1 Comparison of performance of different neighborhood operators

Operator	Swap	Reverse	Swap_Reverse	Combined
Minimum [a]	12300.0	11719.5	11760.5	11702.5
Maximum [b]	12725.5	11776.5	11965.5	11789.5
Average [c]	12582.5	11749	11830.9	11740.4
Standard Deviation [d]	133.1	17.7	56.6	28.7
Run time [a]	0.28	0.34	0.39	0.35

[a] Minimum objective value obtained in 20 runs
[b] Maximum objective value obtained in 20 runs
[c] Average objective value obtained in 20 runs
[d] Standard deviations of objective values of 20 runs
[e] Average CPU runtime per run (in minutes)

Figure 2 Converging processes for Swap, Reverse, Swap_Reverse and Combined



4.2 Comparison between MABC with original ABC

To assess the performance of the modified ABC algorithm (MABC), this modified version was compared with the original version of ABC algorithm. Both versions were implemented by adopting the Combine setting of neighbourhood operators and the same number of limit and maximum cycle, and assessed by the set of 8 instances. In all test instances, they are tested by using model 1. The average and best results of MABC and OABC are summarized and compared in Table 2.

Table 1 show that MABC outperforms original ABC with the increasing network sizes and number of vehicles. It does not significantly improve the solutions in the instances with 30-nodes network compared with original ABC as all the p-values are greater than 0.01. Yet, according to the table, the modified version obtained solutions better than, or at least equivalent to those from the original version in all test instances. The mean percentage improvement of the average results of all test instances is 0.97% while the largest percentage improvement of the average result is 2.55%. Meanwhile, the improvements become statistically significant with the increasing network sizes (i.e. all p-values below 0.01). Also, the p-values for the instances with 1 vehicle are the largest compared with the instances with more vehicles. It shows that MABC is more capable to handle instances with multiple vehicles compared with basic version.

Table 2 Comparison of Experimental Results between MABC and original ABC

Instance	Modified ABC		Original ABC		Imp % [c]	P-Value [d]
	Average [a]	Minimum [b]	Average [a]	Minimum [b]		
1v30n	6090	6090	6090.3	6090	0	0.330
1v99n	23336.5	23096	23518.6	23376.5	0.78	1.27E-04
2v30n	6090	6090	6090	6090	0	N/A
2v99n	23356.9	22987	23806.3	23528	2.55	1.05E-08
3v30n	6090	6090	6090.8	6090	0.01	0.330
3v99n	23363.3	23069	23838.1	23559	2.03	1.36E-09
4v30n	6090	6090	6092.6	6090	0.04	0.129

4v99n	23360.58	22963	23851.9	23483	2.1	2.37E-07

[a] Average objective value obtained in 20 runs
[b] Minimum objective value obtained in 20 runs
[c] Calculated based on the mean objective values
[d] P-value refers to the t-test's p-value of 20 runs

4.3 The effect of fleet size towards repositioning activities

Table 3 Effect of demand towards service time and maximum route duration

		Objective: Total service time					Objective: Maximum route duration				
Fleet Size	TDD	Veh [a]	DD [b]	Min (mean) TST	Min (mean) MRD	CPU Time [c]	Veh [a]	DD [b]	Min (mean) MRD	Min (mean) TST	CPU Time [c]
2	29	2	29	6597.5 (6597.5)	4303 (4303)	0.130	2	29	3387.5 (3394.0)	6750.5 (6769.4)	0.131
2	9	2	29	Infeasible [d]			2	29	Infeasible [d]		
2	0	2	29	Infeasible [d]			2	29	Infeasible [d]		
3	29	2	29	6597.5 (6597.5)	4303 (4303)	0.127	3	29	2671.5 (2699.3)	7983.5 (7945.7)	0.127
3	9	3	9	8484 (8492.5)	3943.5 (4154.8)	0.127	3	9	3101 (3111.1)	9242 (9300.8)	0.126
3	0	3	9	Infeasible [d]			3	9	Infeasible [d]		
4	29	2	29	6597.5 (6597.5)	4303 (4303)	0.128	4	29	2620.5 (2629.6)	10375.6 (10158.5)	0.127
4	9	3	9	8484 (8504.6)	3943.5 (4303.6)	0.127	4	9	2776 (2811.8)	11033 (11152.6)	0.126
4	0	4	0	9846.5 (9901.3)	3811.5 (4233.2)	0.130	4	0	2939.5 (3003.0)	11597.5 (11821.8)	0.128
5	29	2	29	6597.5 (6597.5)	4303 (4303)	0.127	5	29	2620.5 (2620.5)	12433.5 (12760.5)	0.125
5	9	3	9	8484 (8517.8)	3943.5 (4287.1)	0.126	5	9	2694.5 (2744.8)	13397.5 (13409.7)	0.124
5	0	4	0	9846.5 (9897.2)	3811.5 (4320.9)	0.126	5	0	2866.5 (2883.0)	14153.9 (14173.9)	0.124

[a] Number of vehicles in the fleet that are assigned for the repositioning activity
[b] The mean demand dissatisfaction for 20 runs
[c] Average CPU runtime in minutes for each run

Table 3 shows the results of using 2 to 5 vehicles to reposition the bikes in 30-node network with higher level of uneven distribution of bikes. The CPU times for all instances are similar and apparently independent from the number of employed vehicles. These results show that (1) increase of number of vehicles does not change the minimum total service time (TST) for a specific TDD level; and (2) higher tolerance of demand dissatisfaction can reduce the operation time. Table 3 provides an illustration on the trade-offs among the number of vehicles, tolerance of demand dissatisfaction and the service durations. Note that the stricter the TDD level, the more the number of operating vehicles required to have feasible solutions. It is in accordance with the intuition, since for instances with strict TDD level, the overall required capacity for bike repositioning increases. Instances with 2 or 3 vehicles at TDD = 0 are examples for such a phenomenon. Meanwhile, it is clear that the more operating vehicles can reduce the maximum route duration of the whole operation given the same level of TDD. Once the minimum maximum route duration (MRD) is reached, further increase of operating vehicles will result in the increase of total service time of the repositioning activities, which

performs differently compared with the TST objective instances. For objective 2 with TDD = 29, the MRDs of 4-vehicles and 5-vehicles are the same, but the total service time for 5-vehicles instance is 19.8% more than the 4-vehicles one. When comparing with 4-vehicles instance for TDD = 0, the 5-vehicles instance reduces the MRD by 72.5 seconds (1.2 minutes) while it increases the total service time by 2556.4 seconds (42.6 minutes). In some repositioning activities, perfect balance may be costly compared with slight imbalance. From the solutions for model 1, it is found that the minimum demand dissatisfaction equals 9 for 3-vehicles instances and equals 0 for 4-vehicles instances. Meanwhile, the total service time found in the 3-vehicle case (8,484.0) is comparably lower than that in the 4-vehicle case (9846.5). This is equivalent to the average time reduction of 151.4 seconds (2.5 minutes) for each unsatisfied demand.

To conclude, the above observations show the trade-off of TDD level and fleet size towards the design objectives. In general, both total service time and maximum service duration can be reduced by increasing TDD level and/or fleet size, but these increases degrade the service level of the repositioning activity and increase the operating costs respectively.

5. Conclusion

A public bike repositioning problem (PBRP), which considers the demand dissatisfaction and service time, is investigated. The problem aims to determine the routes of the fleet of repositioning vehicles that the service time is minimized while the demand dissatisfaction is kept below an overall tolerable limit. As a NP-hard problem, an efficient solution method that employs the artificial bee colony (ABC) algorithm to determine the sequence is presented. A solution representation scheme is proposed to search the all of the possible route sequences efficiently. To improve the effectiveness of the solution process, three neighbourhood operators were proposed. A modified version of ABC was proposed and outperformed the original heuristic through computational experiments. Through numerical studies, the increase in fleet size may reduce the longest route duration but not the total service time of all vehicles and the practical implications of the trade-offs between the fleet size and travel time are discussed subsequently.

References

Benchimol, M et al. (2011) Balancing the stations of a self-service bike hire system *RAIRO-Operations Research* 45(1), 37-61

Brajevic I (2011) Artificial bee colony algorithm for the capacitated vehicle routing problem. In: *Proceedings of the European computing conference (ECC'11)*, 239–244

Chemla, D, Meunier, F and Wolfler Calvo, R (2013) Bike sharing systems: Solving the static rebalancing problem *Discrete Optimization* 10(2), 120-146

Dell'Amico, M, Hadjicostantinou, E, Iori, M and Novellani, S, (2014) The bike sharing rebalancing problem: Mathematical formulations and benchmark instances *Omega*, 7-19

Di Gaspero, L, Rendl, A and Urli, T (2013) A hybrid ACO+ CP for balancing bicycle sharing systems *Hybrid Metaheuristics*, 198-212

Erdoğan, G, Laporte, G and Wolfler Calvo, R (2014) The static bicycle relocation problem with demand intervals *European Journal of Operational Research* 238(2), 451-457

Hosny, M I and Mumford, C L (2010) Solving the one-commodity pickup and delivery problem using an adaptive hybrid VNS/SA approach Parallel Problem *Solving from Nature* 6, 189-198

Ho, S and Szeto, WY (2014) Solving a static repositioning problem in bike-sharing systems using iterated tabu search *Transportation Research Part E: Logistics and Transportation Review* 69, 180-198

Karaboga, D and Basturk, B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm *Journal of global optimization* 39(3), 459-471

Karaboga, N (2009) A new design method based on artificial bee colony algorithm for digital IIR filters *Journal of the Franklin Institute* 346, 328-348

Li J, Pan Q, Xie S, Wang S (2011) A hybrid artificial bee colony algorithm for flexible job shop scheduling problems. *International Journal of Computers Communication and Control* 6(2), 286–296

Martinovic, G, Aleksi, I and Baumgartner, A (2008) Single-commodity vehicle routing problem with pickup and delivery service *Mathematical Problems in Engineering* 2008, 1-17

Nair, R, Miller-Hooks, E, Hampshire, R C and Bušić, A (2013) Large-Scale Vehicle Sharing Systems: Analysis of Vélib' *International Journal of Sustainable Transportation* 7(1), 85-106

Rainer-Harbach, M et al. (2014) PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems *Journal of Global Optimization*, 1-33

Raviv, T, Tzur, M and Forma, I A (2013) Static repositioning in a bike-sharing system: models and solution approaches *EURO Journal on Transportation and Logistics* 2(3), 187-229

Schuijbroek, J, Hampshire, R and van Hoeve, W J (2013) Inventory rebalancing and vehicle routing in bike sharing systems

Singh, A, & Sundar, S (2011) An artificial bee colony algorithm for the minimum routing cost spanning tree problem *Soft Computing* 15(12), 2489-2499

Szeto, W Y, Wu, Y and Ho, S C (2011) An artificial bee colony algorithm for the capacitated vehicle routing problem *European Journal of Operational Research* 215(1), 126-135

Zhao, F, Li, S, Sun, J and Mei, D (2009) Genetic algorithm for the one-commodity pickup-and-delivery traveling salesman problem *Computers and Industrial Engineering* 56(4), 1642-1648