# An Adaptive Ant Colony System for Public Bicycle Scheduling Problem

Di Liang[1], Zhi-Hui Zhan[1(✉)], and Jun Zhang[2]

[1] School of Computer Science and Engineering,
South China University of Technology,
Guangzhou 510006, People's Republic of China
zhanapollo@163.com
[2] Guangdong Provincial Key Laboratory of Computational Intelligence
and Cyberspace Information, South China University of Technology,
Guangzhou 510006, People's Republic of China

**Abstract.** Public bicycle scheduling problem (PBSP) is a kind of problem that how to design a reasonable transportation route in order to reduce cost or improve user satisfaction under certain constraints. PBSP can be regarded as a specific combinatorial optimization problem that ant colony system (ACS) can solve. However, the performance of conventional ACS is sensitive to its parameters. If the parameters are not properly set, ACS may have the disadvantage of being easy to fall into local optimum, resulting in poor accuracy and poor robustness. This paper proposes an adaptive ACS (AACS) to efficiently solve PBSP. Instead of fixed parameters in ACS, each ant is configured with own different parameters automatically to construct solutions in AACS. In each generation, AACS regards the parameters in the well-performed ants as good parameters and spreads these parameters among the ant colony via selection, crossover, and mutation operators like in genetic algorithm (GA). This way, the key parameters of ACS can be evolved into a more suitable set to solve PBSP. We applied AACS to solve PBSP and compared AACS with conventional ACS and greedy algorithm. The results show that AACS will improve the accuracy of the solution and achieve better robustness.

**Keywords:** Public Bicycle Scheduling Problem · Ant Colony System
Genetic Algorithm

## 1 Introduction

Inspired by the sharing economy, public bicycle sharing systems (PBSS) are bringing a revolutionary new environmental protection method to smog-ridden city traffic [1]. Despite their obvious success as a new form of public transportation, problems with borrowing bicycles and high bicycle scheduling cost still arise in the actual operation. The emergence of these problems is largely due to the irrationality of bicycle scheduling, which makes the public bicycle scheduling problem (PBSP) an important subject worth studying. As a variant of vehicle routing problem (VRP) [2], the PBSP belongs to combinatorial optimization problems (COPs), which can be solved by meta-heuristic methods [3].


L. Cheng et al. (Eds.): ICONIP 2018, LNCS 11302, pp. 417–429, 2018.
https://doi.org/10.1007/978-3-030-04179-3_37

As a kind of intelligent bionic optimization algorithm, ant colony optimization (ACO), especially its ant colony system (ACS) [4] variant has been successfully applied to many COPs [5], such as aircraft arrival sequencing and scheduling problem [6], virtual machine placement in cloud computing [7], and multiobjective cloud workflow scheduling [8]. Furthermore, ACO performs well on vehicle routing issues [2, 9]. A distributed ACS algorithm was proposed to solve school bus scheduling problem in [9]. Liu et al. [10] used ACO to optimize the scheduling path in PBSP, but did not consider the various demands of users in different periods of the day. Zhang et al. [11] considered the time factor, but simply used ACO to solve the problem, which was easy to converge to local optimal solution.

Given that the parameters such as $\beta$, $\varepsilon$, $\rho$, which are the heuristic factor and two parameters of pheromone updating rules in ACS respectively, have significant effect on the performance of ACS [12], many studies have been conducted to analyze parametric relation and find the best parameter values of ant-based algorithms [12–15]. Yu et al. [12] proposed to adjust the vales of $\rho$ and $\varepsilon$ dynamically in different states of the optimization process. In this paper, we expand parameters to be optimized (i.e., $\alpha$, $\beta$, $\rho$, $\varepsilon$, and $q_0$) and adopt the evolutionary mechanisms in GA to adaptively control these parameters. Two models are established to minimize transportation cost and maximize user satisfaction, and an adaptive ACS (AACS) is proposed to solve these models in this research. The results show that the AACS method can effectively improve the quality of solutions.

The rest of this paper is organized as follows. Section 2 describes PBSP. Section 3 introduces AACS. In Sect. 4 we compare AACS algorithm with other general algorithms: conventional ACS and greedy algorithm. Section 5 summarizes the study and points out the research direction in the future.

## 2 Description of PBSP

In PBSP, the location of each bicycle rental station is fixed and known. The scheduling vehicle departs from one of these stations, collects or delivers a certain number of bicycles according to the needs of each station passing by, and finally returns to the starting station to complete a dispatch task.

The problem can be modeled under the environments of flat-peak period or peak period [11]. In the flat-peak period, users need fewer bicycles and each rental station can basically meet the needs of users. The vehicle mainly considers how to plan the route to minimize scheduling cost. It is assumed that the cost is positively correlated with the distance of the scheduling path, that is, a longer path means a higher cost. Therefore, the objective can be transformed to minimize the path length. While in the peak period, users have greater demand, and their demand time is more concentrated. Therefore, time factor should be taken into account in the scheduling process. The primary consideration is to maximize user satisfaction. According to two different situations above, PBSP can be modeled as '*distance model*' in flat-peak period and as '*time window model*' in peak period, respectively. We will describe these two models in the following of this section.

### 2.1 Distance Model

Undirected graph $G = (V, E)$ denotes the scheduling domain, where $V = \{1, 2, \ldots, n\}$ denotes the set of rental stations, $E = \{(i,j)|i,j \in V, i \neq j\}$ denotes the edges between these stations. $d_i$ denotes the demand for bicycles in station $i(i \in V)$. Herein,$d_i > 0$ means that $i$ has extra bicycles that need to be taken away, $d_i > 0$ means $i$ needs to be placed bicycles, and $d_i = 0$ means that $i$ does not need to be served. Here the vehicle departs from the starting station with no bicycle. A complete scheduling route $R = \{r_1, r_2, \ldots, r_{n+1}\}, r_i \in V$ is formed when the vehicle returns to the starting station after visiting all the stations. Our goal is to minimize the length of $R$, thereby reducing the scheduling cost.

Three constraints need to be met when choosing the next station to serve. (i) The station has not been accessed before. (ii) If the station's demand is positive, the number of bicycles that need to be taken away should be less than or equal to the left capacity of the vehicle. (iii) If the station's demand is negative, the number of bicycles required should be less than or equal to the available number of bicycles on the vehicle this moment.

Above, distance model can be expressed as follows.

Firstly, the optimization objective is as

$$\min L = \sum_{r_x, r_{x+1} \in R} l(r_x, r_{x+1}) \quad x = 1, \ldots, n \tag{1}$$

Then, the constraints are as

$$0 \leq q_w = d_i + q_{w-1} \leq Q \quad w = 2, \ldots, n, i \in V \tag{2}$$

$$r_1 = r_{n+1} \tag{3}$$

$$r_x \neq r_y \quad x, y = 1, \ldots, n, x \neq y \tag{4}$$

$$\sum_{i \in V} d_i \geq 0 \tag{5}$$

where $l(r_x, r_{x+1})$ is the distance between stations $r_x$ and $r_{x+1}, q_w$ is the number of bicycles on the vehicle at time $w$, and $Q$ is the capacity of the vehicle. Equation (2) shows the condition that the demand of next station should be satisfied. Equation (3) shows that the vehicle finally returns to the starting station after finishing a scheduling task. Equation (4) indicates that each station has only one chance to be served in a scheduling process. Equation (5) indicates that the algebraic sum of all stations' demand should be greater than or equal to zero. Otherwise, this problem can not be solved.

### 2.2 Time Window Model

Time window model is based on distance model. In this model, the vehicle also completes a scheduling route $R$, and during the scheduling process, a corresponding

number of bicycles are collected or put into service according to the needs of each station. Unlike distance model, the objective is no longer to minimize the path length, but to maximize user satisfaction.

In time window model, each station maintains a time window that represents the period when users expect to be served at that station. The vehicle should try to reach the station within the specified time window to reduce the possibility that the user is rejected when borrowing or returning a bicycle. The calculation of user satisfaction at each station $i$ follows the function of visiting time $t_i$ shown in Fig. 1.

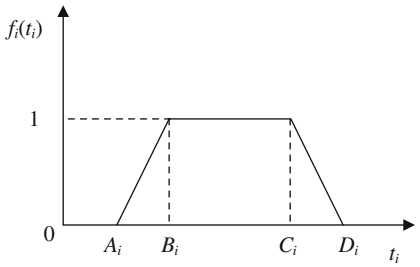


**Fig. 1.** User satisfaction function

In $(B_i,C_i)$, user satisfaction is 1, which is called the best visiting period. In $(A_i,B_i)$ and $(C_i,D_i)$, user satisfaction changes linearly with $t_i$ within (0,1). In $(A_i,D_i)$, user satisfaction is positive, which is called acceptable visiting time. Outside $(A_i,D_i)$, user satisfaction is 0, which is unacceptable visiting period. The user satisfaction function at station $i$ can be expressed as:

$$f_i(t_i) = \begin{cases} 1, & \text{if } t_i \in (B_i, C_i) \\ \frac{t_i - A_i}{B_i - A_i}, & \text{if } t_i \in (A_i, B_i) \\ \frac{D_i - t_i}{D_i - C_i}, & \text{if } t_i \in (C_i, D_i) \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Above, time window model is expressed as follows, with the objective as

$$\max S = \sum_{i \in V} f_i(t_i) \tag{7}$$

where $S$ is the sum of user satisfaction at all stations. The constraints are set the same as those in distance model.

## 3 AACS to Solve PBSP

In general, the parameters in ACS are set based experience or through trial and error, which may result in low precision and low efficiency respectively. Different from conventional ACS that each ant shares the same parameters to build solutions, AACS

assigns different parameters for each ant and combines the evolutionary mechanisms in GA (i.e., selection, crossover, and mutation) to adjust these parameters online. In AACS, each ant constructs solutions with independent parameters. According to this method, different ants may have different performance on this problem, mainly due to their different parameters. In order to deliver the good parameters in the well-performed ants to the poorly-performed ants, evolutionary mechanisms, such as selection, crossover, and mutation, are used to improve these parameters in ant colony. This way, the parameters in the ant are treated as a chromosome in GA.

When AACS is applied to solve PBSP, an ant can be considered as a vehicle with a constant capacity. First of all, a set of gene sequence values $\{\alpha_i, \beta_i, \rho_i, \varepsilon_i, q_{0i}\}$ is initialized randomly in certain ranges for each ant $i$. The ants construct solutions according to ACS rule under the control of their own parameters. After a certain number of iterations $I$, calculate the fitness of ants and update their parameters (or genes) by selecting some good parameters in the well-performed ants. Cross and mutate these parameters to generate better parameters for the colony.

### 3.1 Key Components in Designing AACS

**Pheromone Initialization.** The pheromone on the edge between each station is initialized to

$$\tau_0 = (N \cdot L_{nn})^{-1} \tag{8}$$

where $N$ is the number of stations, $L_{nn}$ is the tour length produced by the nearest neighbor heuristic search.

**Constraints Processing.** For constraints (2) and (4), a set of candidate stations is set up when choosing the next station, and only stations that satisfy both of these conditions can join the set.

**Solution Construction.** In one iteration, each ant randomly chooses a departure station and determines the next station with the guidance of a state transition rule. The transition probability that ant $k$ moves from station $i$ to station $j$ is calculated according to

$$p_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)]^{\alpha} \cdot [\eta(i,j)]^{\beta}}{\sum_{u \in allowed_k} [\tau(i,u)]^{\alpha} \cdot [\eta(i,u)]^{\beta}}, \text{if } j \in allowed_k \\ 0, \quad \text{otherwise} \end{cases} \tag{9}$$

where $allowed_k$ is the set of candidate stations that $k$ is allowed to access next, $\tau(i,j)$ is the amount of pheromone, and $\eta(i,j)$ is the heuristic factor. Herein, $\eta(i,j) = 1/l(i,j)$ in distance model and $\eta(i,j) = C \cdot f_j(t_i + T_0 + l(i,j)/v_0)$ in time window model, where $t_i$ is the time when the ant reaches $i$, $T_0$ is the time it takes to load or unload bicycles, $v_0$ is the speed of movement, and $C$ is a parameter.

In AACS, the state transition rule is as follows: ant $k$ positioned on station $i$ chooses the next station $j$ from the candidate stations $allowed_k$ by applying the rule given by

$$j = \begin{cases} \arg\max_{u \in allowed_k} \{[\tau(i,u)^{\alpha}] \cdot [\eta(i,u)^{\beta}]\}, & \text{if } q \le q_0 \\ J, \quad \text{otherwise} \end{cases} \tag{10}$$

where $q$ is a random number uniformly distributed in [0,1], $J$ is a random variable selected from $allowed_k$ according to the probability distribution given in (9), and $q_0$ ($0 \le q_0 \le 1$) is a parameter which is used to balance exploration of new edges and exploitation of accumulated knowledge about the problem.

**Pheromone Updating Rule.** Pheromone local updating and global updating are performed in the optimization process. The significance of local updating is to explore more potential solutions, while the significance of global updating lies in exploitation of the optimal solutions. When an ant has built a path, local updating rule (11) is applied on each edge of the path to decay the pheromone, which leads to a greater probability of exploring other edges. In contrast, only the globally best path so far is allowed to deposit pheromone according to global updating rule (12) after all the ants have completed their tours

$$\tau(i,j) = (1-\rho) \cdot \tau(i,j) + \rho \cdot \tau_0 \tag{11}$$

$$\tau(i,j) = (1-\varepsilon) \cdot \tau(i,j) + \varepsilon \cdot \Delta\tau(i,j), \forall (i,j) \in T^{gb} \tag{12}$$

where $\rho(0<\rho<1)$ and $\varepsilon(0<\varepsilon<1)$ are the decay parameters of local pheromone and global pheromone respectively. Herein, $\Delta\tau(i,j) = \left(L_{gb}\right)^{-1}$ in distance model, where $L_{gb}$ is the length of the globally best Tour $T_{gb}$; $\Delta(i,j) = C_1 \cdot S_{gb}$ in time window model, where $S_{gb}$ is the maximum user satisfaction value, and $C_1$ is a parameter.

**Fitness Function.** The fitness function is designed to evaluate the adaptability of ants. Since that ACS is a stochastic search method and each ant chooses the starting station randomly, which affects the evaluation of solution, especially in time window model, we allow an ant to construct solutions for number of iterations $I$ and take the value of the best solution produced by the ant as its fitness. The fitness function in distance model and time window model is implemented as (13) and (14) respectively.

$$F = (L_b)^{-1} \tag{13}$$

$$F = S_b \tag{14}$$

where $L_b$ is the length of the optimal scheduling route that the ant has passed and $S_b$ is the maximal user satisfaction value in $I$ iterations.

### 3.2 Complete AACS Algorithm

The complete AACS algorithm contains the following eight steps.

Step 1: Initialize parameters such as population size $M$, iteration times $I$, maximum generation $G$, crossover probability $p_c$ and mutation probability $p_m$, and initialize

the amount of pheromone $\tau_0$. Randomly Generate $\{\alpha_i, \beta_i, \rho_i, \varepsilon_i, q_{0i}\}$ for each ant $i$ and set its initial fitness value to 0.
Step 2: Set the iteration $t = 1$.
Step 3: Each ant randomly chooses the starting station and serves it. Determine the next station according to state transition rule. The ant goes on like this until it finishes serving all the stations and then goes back to the starting station. Perform local pheromone updating rule.
Step 4: For each ant, calculate $L^{-1}$ (or $S$) and compare it with the original fitness value. The better will be the new fitness value.
Step 5: Until all the ants have constructed their tours, perform global pheromone updating rule. Set $t = t+1$.
Step 6: Judge whether $t$ is equal to $I$. If so, turn to Step 7, otherwise turn to Step 3.
Step 7: Update parameters of these ants by selection, crossover, and mutation operations.
Step 8: Judge whether the number of evolutionary generations is equal to $G$. If so, output the optimal solution, otherwise turn to Step 2.

The flowchart of AACS is shown in Fig. 2.

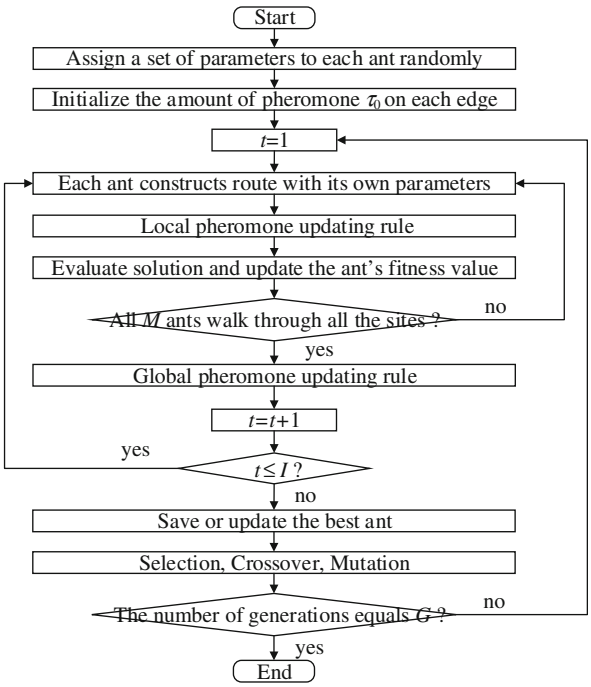


**Fig. 2.** The flowchart of AACS

## 4 Experiments and Discussions

In this section, we compare AACS with other general algorithms. Considering the existence of multiple constraints in PBSP, it is easy to generate an illegal solution in calculating process if using GA to solve. Therefore, we take conventional ACS and greedy algorithm as competitors in the experiments. We report three sets of experiments. Firstly, AACS is compared with the conventional ACS and GA on some symmetric TSP instances, include fri26, eil51, eil76, and eil101 in TSPLIB (https://wwwproxy.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/) to test the performance of each algorithm without bicycle restrictions. The second set and the third set compare their performance in distance model and time window model. The AACS performs well in all these cases. In addition, the time consumption of AACS is basically the same as that of ACS, which is verified through experiments.

### 4.1 Comparison Without Bicycle Constraints

The parameter settings are listed in Table 1. The parameters of ACS are set as originally suggested in [4, 16].

**Table 1.** Parameter settings

| Parameter | AACS | ACS |
|---|---|---|
| $M$ | 50 | 50 |
| $I$ | 500 | 50000 |
| $G$ | 100 | N/A |
| $\alpha$ | [0.01,4] | 1 |
| $\beta$ | [1, 7] | 5 |
| $\rho$ | [0.01,0.5] | 0.1 |
| $\varepsilon$ | [0.01,0.5] | 0.1 |
| $q_0$ | [0.5,1] | 0.9 |

Maximum generation $G$ is set to 100. Ant iterations $I$ is set to 500 in AACS and 50000 in ACS to be fair to compare. That is, $I \times G$=50000. Both AACS and ACS run 20 times independently on each instance, and the comparison is performed with respect to the best result, average result, and standard deviation (denoted as 'Std.dev.'). As the greedy algorithm is a deterministic algorithm, it runs only one time, with the result comparing. The results given in Table 2 show that AACS performs better than ACS and greedy. For clarity, the best results obtained are highlighted in **boldface**.

**Table 2.** Experimental results without bicycle constraints

| Instance | AACS | | | ACS | | | Greedy | Optimum |
|---|---|---|---|---|---|---|---|---|
| | Best | Average | Std.dev. | Best | Average | Std.dev. | | |
| fri26 | **937** | **937.00** | 0.00 | **937** | **937.00** | 0.00 | 965 | 937 |
| eil51 | **426** | **427.15** | 0.49 | **426** | 428.10 | 2.90 | 482 | 426 |
| eil76 | **538** | **543.70** | 4.38 | **538** | 546.45 | 5.16 | 608 | 538 |
| eil101 | **629** | **639.65** | 6.48 | 630 | 640.10 | 5.39 | 746 | 629 |

### 4.2 Experimental Results in Distance Model

In this experiment, we conduct two PBSP instances based on eil51 and eil76. The locations of the stations are the same as the corresponding ones in TSPLIB. The difference is that the PBSP instance contains more information that each station has a number of bicycle demand. Tables 3 and 4 give the demand for bicycles at each station in eil51 and eil76 respectively, where $i$ denotes the index of the station and $d_i$ is the number of bicycles needed in station $i$. The parameters of ACS and AACS are also set as Table 1, and the results are obtained by 20 independent runs on each instance. The results given in Table 5 show that AACS has general better performance than both ACS and greedy algorithm.

**Table 3.** Demand of each station in eil51

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $d_i$ | –2 | 2 | –10 | –9 | –2 | –15 | –13 | –15 | -5 | 17 | 5 | 20 | 4 |
| $i$ | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| $d_i$ | 20 | 10 | 3 | –8 | 11 | 7 | –13 | 8 | –10 | –11 | –1 | 1 | –9 |
| $i$ | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| $d_i$ | 15 | 5 | –4 | –3 | 6 | 15 | –9 | –7 | –12 | 14 | 18 | 6 | –7 |
| $i$ | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | |
| $d_i$ | 5 | –2 | –18 | –20 | 9 | 13 | 16 | –12 | –8 | 19 | –12 | –6 | |

### 4.3 Experimental Results in Time Window Model

To simulate real-road conditions more accurately, we adopt floating-point to record the distance between various stations and set $T_0 = 2$ min and $v_0 = 20$ km/h to correspondingly expand the distance to 20 times the original. The best visiting period and the acceptable visiting period of each station are set 5 min and 9 min respectively (i.e., $[B_i,C_i] = 5$ min, $[A_i,D_i] = 9$ min). For the choice of next station in time window model has a significant relationship with current time, the accumulated knowledge is not particularly valuable for ants. Therefore, the range of $\beta$ is adjusted to [3, 7] to increase the relative importance of desirability heuristic information. Tables 6 and 7 give the start time of the best visiting period (i.e., $B_i$) in eil51 and eil76, which use the bicycle constraints data in Tables 3 and 4 respectively. We set $I = 1000$ and $G = 100$ because of the complexity of the problem. Correspondingly, $I = 100000$ in ACS to be fair to compare. The results are based on 20 independent runs and are given in Table 8.

**Table 4.** Demand of each station in eil76

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $d_i$ | 4 | –5 | –10 | –9 | 20 | –15 | –13 | 8 | –7 | 17 | –2 | 20 | 4 |
| $i$ | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| $d_i$ | 3 | 10 | –12 | –8 | 22 | 7 | –11 | 9 | –10 | –20 | 1 | 25 | –1 |
| $i$ | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| $d_i$ | 3 | 15 | 5 | –22 | 15 | 5 | –14 | –7 | 12 | 14 | 6 | 18 | –16 |
| $i$ | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 |
| $d_i$ | 5 | –2 | –18 | 16 | 9 | –25 | 13 | –20 | –5 | –8 | 19 | –17 | –4 |
| $i$ | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 |
| $di$ | –20 | 6 | 19 | –21 | –10 | 11 | 8 | 5 | –24 | –5 | 15 | 13 | 6 |
| $i$ | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | | |
| $di$ | –8 | 2 | –25 | –3 | 16 | 6 | 9 | 25 | –15 | –3 | 10 | | |

**Table 5.** Experimental results in distance model

| Instance | AACS | | | ACS | | | Greedy |
|---|---|---|---|---|---|---|---|
| | Best | Average | Std.dev. | Best | Average | Std.dev. | |
| eil51 | **426** | **431.45** | 2.52 | **426** | 431.6 | 3.25 | 499 |
| eil76 | 558 | **566.20** | 4.00 | **555** | 567.20 | 7.51 | 648 |

**Table 6.** Start time of the best visiting period in eil51

| $i$ | Time | $i$ | Time | $i$ | Time | $i$ | Time | $i$ | Time | $i$ | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 07:04 | 10 | 07:29 | 19 | 07:35 | 28 | 08:25 | 37 | 08:35 | 46 | 08:08 |
| 2 | 07:08 | 11 | 07:17 | 20 | 07:36 | 29 | 08:22 | 38 | 08:42 | 47 | 08:48 |
| 3 | 07:40 | 12 | 07:27 | 21 | 07:39 | 30 | 08:15 | 39 | 08:50 | 48 | 08:53 |
| 4 | 08:20 | 13 | 07:30 | 22 | 07:43 | 31 | 08:28 | 40 | 08:45 | 49 | 08:02 |
| 5 | 07:13 | 14 | 07:55 | 23 | 07:45 | 32 | 08:06 | 41 | 07:58 | 50 | 09:00 |
| 6 | 07:05 | 15 | 08:30 | 24 | 07:48 | 33 | 08:05 | 42 | 08:00 | 51 | 08:33 |
| 7 | 07:20 | 16 | 07:00 | 25 | 07:52 | 34 | 08:10 | 43 | 08:18 | | |
| 8 | 07:10 | 17 | 07:30 | 26 | 08:30 | 35 | 08:27 | 44 | 07:50 | | |
| 9 | 07:23 | 18 | 07:32 | 27 | 08:37 | 36 | 08:23 | 45 | 07:56 | | |

It can be seen from the above results that AACS outperforms conventional ACS generally. Table 2 shows that AACS has a stronger ability to find better solutions than according to 'Best' column and 'Average' column. Greedy algorithm performs poorly compared to AACS and ACS especially in large-scale problem. AACS generates smaller average tour lengths in Table 5. In Table 8 we can see that AACS performs better on satisfying the demand of users in time window model, in terms of both solution quality and stability. This is because the conventional parameter settings suggested for TSP are not suitable enough to solve PBSP due to its complexity,

**Table 7.** Start time of the best visiting period in eil76

| *i* | Time | *i* | Time | *i* | Time | *i* | Time | *i* | Time | *i* | Time |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 07:04 | 14 | 07:30 | 27 | 08:30 | 40 | 08:42 | 53 | 08:27 | 66 | 08:47 |
| 2 | 07:08 | 15 | 08:30 | 28 | 08:37 | 41 | 08:50 | 54 | 07:28 | 67 | 08:08 |
| 3 | 08:20 | 16 | 07:00 | 29 | 08:06 | 42 | 07:58 | 55 | 08:16 | 68 | 08:58 |
| 4 | 07:40 | 17 | 07:36 | 30 | 08:22 | 43 | 08:00 | 56 | 07:35 | 69 | 08:50 |
| 5 | 07:05 | 18 | 07:32 | 31 | 08:25 | 44 | 07:50 | 57 | 07:45 | 70 | 08:52 |
| 6 | 07:56 | 19 | 07:35 | 32 | 08:15 | 45 | 08:08 | 58 | 08:30 | 71 | 08:55 |
| 7 | 07:20 | 20 | 07:55 | 33 | 08:28 | 46 | 08:48 | 59 | 08:34 | 72 | 08:21 |
| 8 | 07:10 | 21 | 07:45 | 34 | 08:05 | 47 | 08:18 | 60 | 08:26 | 73 | 08:30 |
| 9 | 07:13 | 22 | 07:39 | 35 | 08:10 | 48 | 08:02 | 61 | 08:25 | 74 | 08:45 |
| 10 | 07:23 | 23 | 07:30 | 36 | 08:23 | 49 | 08:33 | 62 | 08:36 | 75 | 07:05 |
| 11 | 07:29 | 24 | 07:43 | 37 | 08:45 | 50 | 09:00 | 63 | 08:40 | 76 | 08:09 |
| 12 | 07:17 | 25 | 07:48 | 38 | 08:27 | 51 | 08:53 | 64 | 08:43 | | |
| 13 | 07:27 | 26 | 07:52 | 39 | 08:35 | 52 | 08:00 | 65 | 08:10 | | |

**Table 8.** Experimental results in time window model

| Instance | AACS | | | ACS | | | Greedy |
|---|---|---|---|---|---|---|---|
| | Best | Average | Std.dev. | Best | Average | Std.dev. | |
| eil51 | **35.19** | **34.46** | 0.19 | 34.56 | 34.39 | 0.24 | 30.11 |
| eil76 | **39** | **38.68** | 0.36 | **39** | 38.35 | 0.64 | 32.14 |

especially in the time window model. In contrary, the parameters in AACS can be dynamically adjusted with the problem in the optimization process, which means that the best parameter values are problem-independent. Therefore, the proposed AACS has better adaptability to different COPs than ACS.

## 5 Conclusion

This paper presents an adaptive ACS to solve PBSP, which is based on evolutionary mechanisms to optimize the key parameters of ACS instead of the direct assignment based on past experience. The adaptive convergence of parameters enables AACS better to find out the global optimal solution. The experimental results show that the proposed method can effectively overcome the drawbacks of ACS in solution quality and stability. Furthermore, for the parameters can converge adaptively with the problem to be solved, this method has good generality which can also be used to solve other COPs like PBSP.

In the future, we will combine parallel computing to reduce runtime, especially when dealing with problems with large scale. Real-time updating of bicycle demand to realize dynamic scheduling is also an aspect to research.

**Acknowledgments.** This work was partially supported by the Outstanding Youth Science Foundation with No. 61822602, the National Natural Science Foundations of China (NSFC) with No. 61772207 and 61332002, the Natural Science Foundations of Guangdong Province for Distinguished Young Scholars with No. 2014A030306038, the Project for Pearl River New Star in Science and Technology with No. 201506010047, the GDUPS (2016), and the Fundamental Research Funds for the Central Universities.

## References

1. Labadi, K., Benarbia, T., Barbot, J.P., Hamaci, S., Omari, A.: Stochastic petri net modeling, simulation and analysis of public bicycle sharing systems. IEEE Trans. Autom. Sci. Eng. **12**(4), 1380–1395 (2015)
2. Wang, X., Choi, T.M., Liu, H., Yue, X.: Novel ant colony optimization methods for simplifying solution construction in vehicle routing problems. IEEE Trans. Intell. Transp. Syst. **17**(11), 3132–3141 (2016)
3. Rabbani, M., Tahaei, Z., Farrokhi-Asl, H., Saravi, N.A.: Using meta-heuristic algorithms and hybrid of them to solve multi compartment vehicle routing problem. In: 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp. 1022–1026. IEEE, Singapore (2017)
4. Dorigo, M., Gambardella, L.M.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
5. Zhang, Z., Zou, K.: Simple ant colony algorithm for combinatorial optimization problems. In: 36th Chinese Control Conference (CCC), pp. 9835–9840. IEEE, Dalian (2017)
6. Zhan, Z.H., Zhang, J., Li, Y., et al.: An efficient ant colony system based on receding horizon control for the aircraft arrival sequencing and scheduling problem. IEEE Trans. Intell. Transp. Syst. **11**(2), 399–412 (2010)
7. Liu, X.F., Zhan, Z.H., Deng, J.D., Li, Y., Gu, T.L., Zhang, J.: An energy efficient ant colony system for virtual machine placement in cloud computing. IEEE Trans. Evol. Comput. **22**(1), 113–128 (2018)
8. Chen, Z.G., Zhan, Z.H., Gong, Y.J., et al.: Multiobjective cloud workflow scheduling: a multiple populations ant colony system approach. IEEE Trans. Cybern. (2018). https://doi.org/10.1109/tcyb.2018.2832640
9. Mouhcine, E., Khalifa, M., Mohamed, Y.: Route optimization for school bus scheduling problem based on a distributed ant colony system algorithm. In: Intelligent Systems and Computer Vision (ISCV), pp. 1–8. IEEE, Fez (2017)
10. Liu, Z.P., Li, K.P., Zhu, X.H.: Optimal dispatch between stations for public bicycle based on ant colony algorithm. J. Transp. Inf. Saf. **30**(4), 71–74 (2012)
11. Zhang, J.G., Wu, T., Jiang, Y.S.: Study on scheduling algorithm for public bicycle system based on ant colony algorithm. J. Xihua Univ. Nat. Sci. **33**(3), 70–76 (2014)
12. Yu, W.J., Hu, X.M., Zhang, J., Huang, R.Z.: Self-adaptive ant colony system for the traveling salesman problem. In: 2009 IEEE International Conference on Systems, Man and Cybernetics, pp. 1399–1404. IEEE, San Antonio (2009)
13. Zecchin, A.C., Simpson, A.R., Maier, H.R., Nixon, J.B.: Parametric study for an ant algorithm applied to water distribution system optimization. IEEE Trans. Evol. Comput. **9**(2), 175–191 (2005)

14. Jangra, R., Kait, R.: Analysis and comparison among ant system; ant colony system and max-min ant system with different parameters setting. In: 2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT), pp. 1–4. IEEE, Ghaziabad (2017)
15. Gong, Y.J., Xu, R.T., Zhang, J., Liu, O.: A clustering-based adaptive parameter control method for continuous ant colony optimization. In: 2009 IEEE International Conference on Systems, Man and Cybernetics, pp. 1827–1832. IEEE, San Antonio (2009)
16. Dorigo, M., Maniezzo, V., Colorni, A.: Ant system optimization by a colony of cooperating agents. IEEE Trans. Syst. Man Cybern. Part B (Cybern.) **26**(1), 29–41 (1996)