Zheng Ning, Chen Tao, Lin Fei and Xu Haitao*

A Hybrid Heuristic Algorithm for the Intelligent Transportation Scheduling Problem of the BRT System

Abstract: This work proposes a hybrid heuristic algorithm to solve the bus rapid transit (BRT) intelligent scheduling problem, which is a combination of the genetic algorithm, simulated annealing algorithm, and fitness scaling method. The simulated annealing algorithm can increase the local search ability of the genetic algorithm, so as to accelerate its convergence speed. Fitness scaling can reduce the differences between individuals in the early stage of the algorithm, to prevent the genetic algorithm from falling into a local optimum through increasing the diversity of the population. It can also increase the selection probability of outstanding individuals, and speed up the convergence at the late stage of the algorithm, by increasing the differences between individuals. Using real operational data of BRT Line 1 in a city of Zhejiang province, the new scheduling scheme can be obtained through algorithm simulation. The passengers' total waiting time in a single way will be reduced by 40 h on average under the same operating cost compared with the original schedule scheme in a day.

Keywords: Hybrid heuristic algorithm, intelligent scheduling, BRT scheduling.

2010 Mathematics Subject Classification: 68T20 Problem solving.

DOI 10.1515/jisys-2014-0134
Received September 18, 2014; previously published online January 17, 2015.

1 Introduction

Bus scheduling is the key to daily operational activities, and it directly affects the operational cost and passenger satisfaction. The scheduling scheme, in accordance with customer traffic, can adjust the headway according to changes in the traffic. It will enhance the pertinence of bus services, and also reduces the passenger waiting time, improves the public transport service quality, and adds attractiveness to public transport. Meanwhile, bus scheduling is subject to various constraints, such as operating cost, service quality, and so on. Achieving intelligent bus dispatching to enable the schedule scheme to meet the traffic requirements and constraints in a reasonable time is an important issue.

There have been many studies on the bus scheduling problem; for example, Hadjar et al. [7] proposed a branch-and-bound algorithm for solving the multiple depot vehicle scheduling problem, which combines column generation, variable fixing, and cutting planes. Barkaoui and Gendreau [1] introduced an adaptive evolutionary approach for real-time vehicle routing and dispatching. Liu et al. [12] proposed a genetic algorithm (GA) incorporating Monte Carlo simulation to solve the optimization model of the bus stop-skipping problem. Guihaire and Hao [6] proposed a model and a solution based on tabu search and neighborhood specifically developed to improve the quality of service for passengers through the number and quality of

***Corresponding author: Xu Haitao,** Hangzhou Dianzi University, No. 215 Sixth Avenue, Xiasha Higher Education Zone, Hangzhou, 310018, China, e-mail: xuhaitao@hdu.edu.cn
Zheng Ning, Chen Tao and Lin Fei: Hangzhou Dianzi University, Hangzhou 310018, China

transfers. Hao et al. [8] established a model to minimize passengers' waiting cost and vehicles' operation cost, and to optimize the headway and bus scheduling combination, and solved it by using an improved GA. Petersen et al. [14] proposed a planning approach to obtain a favorable trade-off between the two contrasting objectives, passenger service and operation cost, by modifying the timetable. Jerald et al. [10] used GA to solve the synchronization problem of the arrival time between different bus lines. Bielli et al. [2] proposed a new method to compute fitness function values in GAs for bus network optimization. The goal is to design a heuristic that allows achieving the best bus network, satisfying both the demand and the offers of transport. Ceder proposed a visualized method [4] to obtain the departure frequency according to traffic law in a day, and then improved this method to approach even-load and even-headway timetables using different bus sizes [5], to meet the demand of regular passengers and randomly arriving passengers simultaneously. Tang and Yang [15] proposed an improved bus scheduling model using an ant colony algorithm based on the punishment strategy. Bunte and Kliewer [3] discussed the similarities and differences between a variety of bus scheduling models and introduced some of the latest scheduling models and their basic concepts.

The optimal solution or approximate optimal solution of the bus scheduling problem can be found within a reasonable time using GA, which is the reason why many researchers use it to solve the bus scheduling problem. The proposed algorithm combines the simulated annealing (SA) algorithm and GA, and joins the elite reserved strategy and fitness scaling method. The individual who has the highest fitness in the population will be copied directly to the next generation. Thus, the best individual can be protected from crossover or mutation operation. Fitness scaling can reduce the differences between individuals in the early stage of the algorithm, to prevent the GA from falling into a local optimum through increasing the diversity of the population. It can increase the selection probability of outstanding individuals, and speed up the convergence at the late stage of the algorithm, by increasing the differences between individuals. After the algorithm implementation and comparison with the actual data, the algorithm can obviously improve the disadvantages of GA of easily falling into local optima and slow convergence to the optimal solution.

2 Establishing the Mathematical Model of BRT Scheduling

Bus scheduling is aimed at meeting the passengers' demand as far as possible to save operating costs. These two conflicting objectives determine that it is a multiobjective optimization problem. Bus schedule can be divided into two parts: static scheduling and dynamic scheduling. Static scheduling mainly refers to formulating the timetable of each line, whereas the task of dynamic scheduling is to adjust the timetable when an emergency occurs, such as an accident, traffic jam, and so on. In reality, static scheduling is given priority over dynamic scheduling. In this article, we will mainly discuss about how to use the improved hybrid heuristic algorithm to solve the static scheduling problem of BRT. Owing to the influence of many factors, solving the bus scheduling problem is a very complex process. To study algorithms for bus scheduling, we make some assumptions as follows:

1. After BRT vehicles arrive at the station, all passengers waiting for this line get on the bus. No one is left waiting for the next bus.
2. All the passengers should buy ticket before get in the bus, so we did not take into consideration the time for dropping-coin or swipe ID card.
3. There is only one type of vehicle in the line; that is, all vehicles have the same number of seats and the maximum load number.
4. There are enough vehicles for scheduling.
5. BRT vehicles depart from stations according to a timetable.
6. The sequence of vehicles on the road tallies with the sequence of departure time. No overhead occurs on the road.
7. Minute is the smallest unit in the scheduling.

The variables used in the model and their meaning are shown below:

- m denotes the departure frequency in the whole scheduling cycle.
- n denotes the total count of bus stop in one direction of the line.
- t_i denotes the i-th vehicle departure time in the scheduling cycle; its unit is minute, $i = 1, 2, \ldots, m$.
- r_j denotes the changes in arrival rate at station j over time.
- T denotes the passengers' total waiting time in the scheduling cycle.

The number of passengers that arrive between t_{i-1} and t_i is equal to $r_j \times (t_i - t_{i-1})$. Assuming the waiting time of each person is equal to $(t_i - t_{i-1})/2$, then the passengers' total waiting time T can be calculated as

$$T = \sum_{i=1}^{m} \sum_{j=1}^{n} r_j \times \frac{(t_i - t_{i-1})^2}{2}. \tag{1}$$

Assuming the cost of one passenger per minute waiting for the bus is ξ, then the passengers' total waiting cost T' equals to

$$T' = \xi \times \sum_{i=1}^{m} \sum_{j=1}^{n} r_j \times \frac{(t_i - t_{i-1})^2}{2}. \tag{2}$$

The cost of bus operation includes fixed costs and variable costs. However, there is little relation between fixed costs and bus scheduling. Therefore, we just considered the variable costs here. Assume that R denotes the earnings of the bus company in a scheduling cycle, P denotes the ride cost of one passenger on average, L denotes the length of the bus line, and C denotes the variable cost for one vehicle per kilometer. The sum of the passengers' ride fee minus the total variable costs is the revenue.

$$R = P \times \sum_{i=1}^{m} \sum_{j=1}^{n} r_j \times (t_i - t_{i-1}) - C \times L \times m. \tag{3}$$

Assuming μ indicates the weight coefficient of the passengers' waiting cost, ν denotes the weight coefficient of the agency's benefit. According to the quadratic penalty method, we can obtain the object function of the bus scheduling model as follows:

$$\min z = \mu \times T' - \nu \times R. \tag{4}$$

To make full use of the public transport resources, bus companies require that the vehicle load rate be higher than the expected load factor. Let $N_{\max}$ denote the maximum capacity of the bus and ρ the agency's excepted load rate. The constraint of load rate is shown below:

$$\sum_{j=1}^{n} r_j \times (t_i - t_{i-1}) \geq \rho \times N_{\max} \qquad i = 2,\ 3, \ldots,\ m. \tag{5}$$

The headway must be in a reasonable range to ensure that both the regular and random passenger can get on in a shorter time. Assuming $H_{\min}$ indicates the minimum specified headway, whereas $H_{\max}$ indicates the maximum specified headway, the real headway should satisfy the following constraint:

$$H_{\min} \leq t_i - t_{i-1} \leq H_{\max} \qquad i = 2,\ 3, \ldots,\ m. \tag{6}$$

Meanwhile, the difference between two adjacent headways of the BRT vehicles should not be too large, to avoid the discontinuity phenomenon from occurring. Assuming τ indicates the maximum difference specified by the agency,

$$|(t_{i+1} - t_i) - (t_i - t_{i-1})| \leq \tau \qquad i = 2,\ 3, \ldots,\ m-1, \tag{7}$$

use the penalty functions to deal with the constrained conditions. On the basis of the above description, the form of the objective function is as follows:

$$\begin{aligned}\min f(X) = \min z \\ &+\omega_1 \sum_{i=2}^{m} \{\max\{0, \rho \times N_{\max} - \sum_{j=1}^{n} r_j \times (t_i - t_{i-1})\}\} \\ &+\omega_2 \sum_{i=2}^{m} \{\max\{0, H_{\min} - (t_i - t_{i-1})\}\} \\ &+\omega_3 \sum_{i=2}^{m} \{\max\{0, (t_i - t_{i-1}) - H_{\max}\}\} \\ &+\omega_4 \sum_{i=2}^{m-1} \{\max\{0, |(t_{i+1} - t_i) - (t_i - t_{i-1})| - \tau\}\},\end{aligned} \tag{8}$$

where $\min f(x)$ is the value of the objective function after adding the penalty functions, and ω_1, ω_2, ω_3, and ω_4 denote the each penalty coefficient of the corresponding constrained conditions. The solution of the problem is a vector X of length m, with each of its component x_i indicating the difference of the i-th vehicle's departure time from that of the first vehicle, in minutes.

The penalty strategy can guarantee that the result is applicable in real life. The penalty of the passengers' number ensures that the bus will not be too crowded, namely guaranteeing the quality of service. The penalty of headway ensures that the headway remains in a reasonable range, which guarantees that the resources will not be wasted and part of the passengers will not be waiting too long. In the optimization process, once the solution violates these constraints, its fitness will decrease; thus, it will be at a disadvantage in the competition. Finally, it will be eliminated in optimization.

3 Improved Hybrid Heuristic Algorithm Design

3.1 Classical GA

GA is a heuristic search that imitates the process of natural evolution due to the processes of inheritance, mutation, selection, and crossover [9]. The main steps for solving the bus scheduling problem using GAs are described below. First, generate the initial population consisting of N individuals randomly and calculate the fitness of each individual according to the fitness function. Then, choose two individuals from the population and perform a crossover operation between the selected individuals to generate two new individuals. We use the roulette wheel selection scheme to implement a selection process in which the probability of the individual be selected is proportional to its fitness. Hence, an individual who has a higher fitness will have a higher probability of contributing one or more offspring to the next generation (see, e.g., Reference [2]). The slot size of each individual can be calculated as $F(X_i)/\sum_{i=1}^{N} F(X_i)$. Generate a random number between 0 and 1, rotate the wheel to this position, then perform other operations on the selected one. Finally, perform a mutation operation on every gene locus of the two individuals, then add these two individuals to the new population. Repeat these three steps, until the individual count of the new population reached N. This entire process is a complete one iteration of the GA. The iteration will not stop until it reaches a specified number of iterations or an individual with higher fitness cannot be generated.

The aim of the objective function is to obtain the minimum. To ensure that the fitness of each individual is greater than zero, and also to facilitate the use of the roulette selection scheme, we should transform the objective function into another form; the final form of the fitness function is

$$F(X) = \begin{cases} C_{\max} - f(X) & \text{if } f(X) < C_{\max} \\ 0 & \text{otherwise.} \end{cases}, \tag{9}$$

3.2 Combination of the SA Algorithm and GA

The SA algorithm is derived from the principle of solid annealing (see, e.g., Reference [11]). SA is based on an analogy with a physical annealing process and optimization problems. Annealing is a physical process; when the solid is heated to a sufficiently high temperature, the particles in its interior move freely in a state space. As the temperature decreases, the internal particles gradually stay on the different states. When the temperature decreases to the minimum, the particles inside reach a steady state. This process is similar to the problem of finding the optimal solution in the optimization process. The basic idea of the SA algorithm was first proposed by Metropolis et al. in 1953 [13]. In 1983, Kirkpatrick and Vecchi [11] successfully introduced it into the field of combinatorial optimization. The main steps of the SA algorithm are as follows: generate a new solution at first, then calculate the change in the objective function value. If the function value is higher than the old solution, then replace the previous solution with the new one. In other cases, accept the newly created solution with probability P. At last, perform the SA operation. Then, generate the next new solution. Repeat all these steps until the optimal solution is found.

The traditional GA is still characterized by its slow evolution and convergence in advanced issues in practical applications. The simulated annealing genetic algorithm (SAGA) is a heuristic search algorithm which combined the SA with GA, and it can receive some bad solutions randomly in the search process. Therefore, it attracts widespread attention, through which it can escape from the local optimal solution, and has a strong global search capability. In theory, the algorithm can overcome the shortcomings of the poor local search capability of the traditional GA, and can well prevent the SA algorithm from not allowing the search process to enter the most promising optimization area. Therefore, the improved algorithm is conducive to expanding the scope of the global search and local search ability, speeding up the convergence rate and improving the operational efficiency of optimization. After the crossover and mutation operations in GA, compare the fitness of the parent and the child individual. If the fitness becomes higher, put the child individual into the new population. In other cases, put the child individual into the new population with probability P. The SA algorithm needs to set the initial temperature T_0; the current temperature T_c is calculated as follows:

$$T_c = T_0 \times \sigma^{g-1}, \tag{10}$$

where σ is a constant between 0 and 1 (not containing the endpoints), which represents the rate of temperature decrease. The greater its value, the slower the temperature decrease. Conversely, the smaller its value, the faster the temperature decrease. The variable g is the number of the current iterations of the algorithm. When the newly generated individual's fitness decreases, the probability of accepting it is

$$P = \exp\left(\frac{F(X_{\text{new}}) - F(X_{\text{old}})}{T_c}\right), \tag{11}$$

where $F(X_{\text{new}})$ indicates the fitness of the new individual and $F(X_{\text{old}})$ indicates the fitness of the original individual.

From Figure 1, it can be concluded that when the temperature is high, the probability of accepting an inferior solution is high, which can ensure the diversity of the population. When the temperature is low, the probability of accepting a poor solution becomes smaller. Then, the algorithm often has entered into a late stage, which guarantees that the optimal solution will not be destroyed.

Meanwhile, adding an elitist strategy into the SAGA ensures that the most outstanding individual of each population can successfully enter into the next generation to generate new individuals. After generating a new population, compare the fitness value of the best individual in the previous generation and the current generation. If the fitness of the best individual in the current generation is less than the one in the previous generation, then replace the worst individual in the current generation with the best individual in the previous generation. Otherwise, directly move into the next iteration.

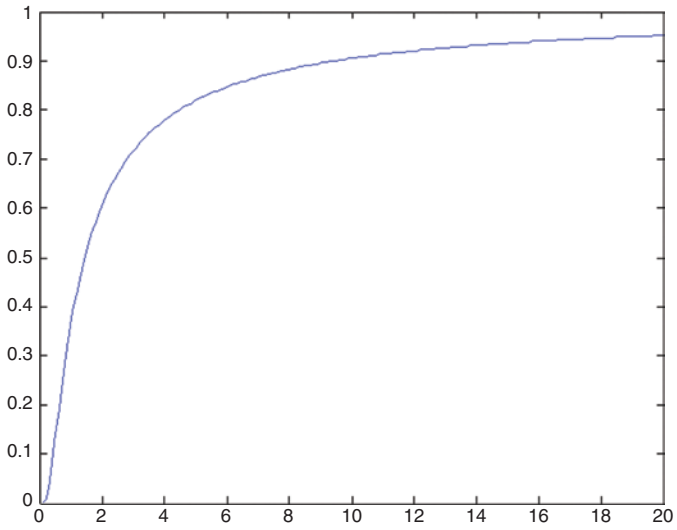

Figure 1. $y = \exp(-1/x)$.

3.3 Improved Hybrid Heuristic Algorithm

Prematurity is one of the most serious shortcomings of the GA; many researchers have used various algorithms to try to improve the GA in the last 10 years. The main reason for the prematurity of GA is selecting individuals according to the proportion to the total fitness. This method will lead the individual having a higher fitness to occupy the entire population quickly, thus weakening the individual's ability to jump out of the local optimal solution. Then, make the GA converge to a local optimal solution. To solve this problem, we must ensure the diversity of the population in the early stage of the algorithm (the fitness varies from low to high); however, in the late stage of the algorithm, it must be ensured that individuals with higher fitness than others have an advantage in the selection.

The fitness scaling method can be used to solve this problem. The so-called stretch refers to narrowing of the gap between the select probability of the individuals in the early stage of the algorithm, so that those individuals that have lower fitness still have a chance to enter the next generation. The diversity of the population will increase through this way. In the late stage of the algorithm, as the average fitness of the population reaches a higher level, those individuals with a higher fitness do not have an obvious advantage in the selection. After stretching, amplify the differences between individuals, increasing the probability of individuals with higher fitness to be selected, so as to accelerate the convergence rate.

The form of the fitness scaling function is as follows:

$$F(X_i)' = \frac{e^{\lambda F(X_i)/T_c}}{\sum_{i=1}^{N} e^{\lambda F(X_i)/T_c}}, \tag{12}$$

where $F(X_i)$ indicates the fitness of the individual X_i, $F(X_i)'$ is the fitness after scaling, T_c refers to the current temperature in SA algorithm, N denotes the size of population, and λ indicates the tensile coefficient. The following example displays the function of the fitness scaling method in SAGA. When T_c is equal to 5000, λ is equal to 200, assuming that there are 10 individuals. Their fitness values are 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0. The comparison of each individual's selection probability before and after scaling is shown in Table 1. When T_c is equal to 50, λ is equal to 200, assuming that there are 10 individuals. Their fitness values are 0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, and 0.80. The comparison of each individual's selection probability before and after scaling is shown in Table 2. P_1 is the probability of selection before scaling, and

Table 1. Comparison of Selected Probabilities When T_c is Equal to 5000.

No.	1	2	3	4	5	6	7	8	9	10
Fitness	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
P_1	0.0182	0.0364	0.0545	0.0727	0.0909	0.1091	0.1273	0.1455	0.1636	0.1818
P_2	0.0982	0.0986	0.0990	0.0994	0.0998	0.1002	0.1006	0.1010	0.1014	0.1018

Table 2. Comparison of Selected Probability When T_c is Equal to 50.

No.	1	2	3	4	5	6	7	8	9	10
Fitness	0.71	0.72	0.73	0.74	0.75	0.76	0.77	0.78	0.79	0.80
P_1	0.0940	0.0954	0.0967	0.0980	0.0993	0.1007	0.1020	0.1033	0.1046	0.1060
P_2	0.0830	0.0864	0.0899	0.0936	0.0974	0.1013	0.1055	0.1098	0.1143	0.1189

P_2 is the probability of selection after scaling. From the comparison, the following conclusions can be made: when the temperature is high, the difference between individuals is reduced after stretching; when the temperature is low, the difference between individuals becomes larger after stretching.

As the range of the basic data types of the programming language that can be provided is limited, we should normalize the fitness of all individuals in the population before performing fitness scaling:

$$fitness' = \frac{fitness}{\max fitness}, \tag{13}$$

where *fitness* refers to the fitness of the individual, *fitness′* refers to the normalized fitness, and max *fitness* refers to the fitness of the best individual in the current population.

The improved hybrid heuristic algorithm mainly consists of selection, crossover and mutation, SA, elitist strategy, standardization, and fitness scaling operation. Perform these operations on the individuals in the population P_g, and then produce a new generation of population P_{g+1}. The basic algorithm is as follows:

1. Set the value of the parameters, such as population size N, chromosome length L_c, crossover probability P_c, mutation probability P_m, maximum generation $G_{\max}$, initial temperature T_0, annealing speed σ, and stretching factor λ.
2. Initialize the population P_0, i.e., generate N feasible solutions randomly. Evaluate the fitness of each individual, and then standardize the fitness, and finally perform the fitness scaling operation. Set the current generation g to be equal to 0.
3. Select two individuals from the current population P_g using the roulette wheel selection scheme.
4. Perform the crossover operation and SA operation. Perform the crossover operation on the two selected individuals, p_1 and p_2, using single point crossover strategy according to the probability P_c; then, produce two new individuals, c_1 and c_2. If the fitness of c_i is higher than that of p_i, then accept c_i; otherwise, accept it with the probability $\exp((F(c_i) - F(p_i))/T_c)$.
5. Perform the mutation operation and SA operation. Perform mutation on every offspring. If the fitness of the individual after the mutation is higher than that of the original one, then accept it; otherwise, accept it with the probability $\exp((F(c_i) - F(p_i))/T_c)$.
6. Put the new individuals to the new population P_{g+1}. If the count of the individuals in P_{g+1} is less than N, then go to step 3; otherwise, go to the next step.
7. Calculate the new fitness of each individual in the population, and standardize the fitness.
8. Perform fitness scaling on the individuals in the new population.
9. Implement the elitist strategy, and then replace the original population with the new population.
10. Decrease the temperature by the annealing rate.
11. Update the value of $g : g = g + 1$. Judge the algorithm termination condition. If the termination condition is satisfied, then output the optimal solution; otherwise, go to step 3.

4 Case Analysis and Simulation

Select the BRT Line 1 in a city of Zhejiang province as the research object, and only consider its up-run. The 10 bus stops on the up-run are Stop1, Stop2, Stop3, Stop4, Stop5, Stop6, Stop7, Stop8, Stop9, and Stop10, in order. The total length of this line is about 18 km. It takes nearly 40 min for the bus to run from the first bus stop to the terminal bus stop on average. The first bus departs at 6:00 am; the last bus departs at 6:30 pm. Thus, the total operating time in a day is 750 min. The maximum headway is 17 min, and the minimum headway is 4 min.

Using the real value encoding method, each individual consists of m real numbers in order. Each number in the individual represents the departure time from the first bus in minutes, and m represents the total number of the departure frequency. Its value is equal to 92 here. The earliest departure time x_1 is equal to 0, and the last departure time x_{92} is equal to 750. To determine the reasonable value of the parameters, such as crossover probability, mutation probability, termination condition, initial temperature, and so on, many experiments have been carried out. Table 3 shows the best fitness values of different combinations of crossover and mutation probability. According to the data in the table, the highest fitness was obtained when the crossover probability equals to 0.8 and the mutation probability equals to 0.1. It was also found that the running time changes in proportion to these two parameters, and the running time increases faster when we added the mutation probability (see Figure 2). Therefore, we should also take this factor into account when setting the values of these two parameters. Finally, the set population size N is equal to 200; crossover probability P_c is equal to 0.8; mutation probability P_m is equal to 0.1; maximum generation $G_{\max}$ is equal to 500; initial temperature T_0 is equal to 5000; annealing rate σ is equal to 0.99; stretch factor λ is equal to 900; and penalty function coefficient ω_1 is equal to 50, ω_2 is equal to 84, ω_3 is equal to 132, and ω_4 is equal to 120.

Table 3. Best Fitness of Different Combinations of Crossover and Mutation Probability.

P_m	fitness ($P_c = 0.8$)	fitness ($P_c = 0.9$)	fitness ($P_c = 1$)
0	63,240	61,140	64,560
0.01	107,460	107,580	107,460
0.05	111,180	114,120	109,560
0.1	119,100	118,140	116,700
0.2	108,780	116,100	113,400

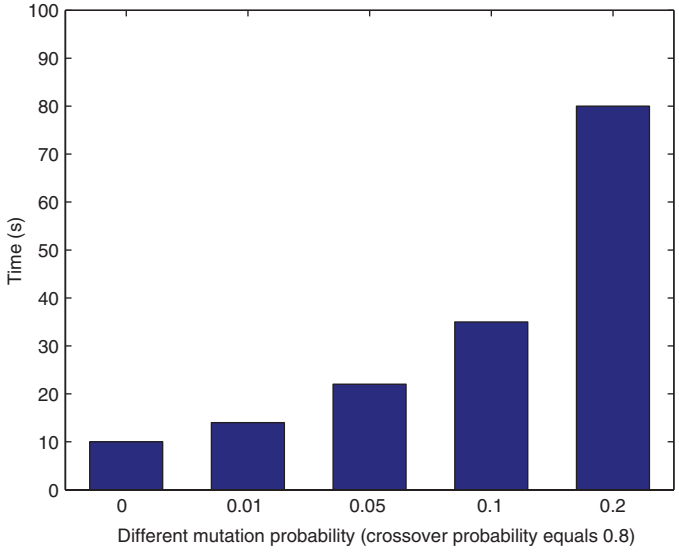

Figure 2. Running Time for Different Mutation Probabilities.

Finally, implement the algorithm in the C programming language; the algorithm will converge to a stable state eventually. The variation of the historical highest fitness during the optimization is shown in Figure 3. Compared with the average fitness of the population (Figure 4), we found that the increase speed of the average fitness is less than the increase speed of the highest fitness obviously at the early stage of the algorithm. This is due to the function of fitness scaling of reducing the differences between individuals.

The best individual obtained finally is X = 0, 10, 17, 23, 27, 31, 36, 41, 46, 51, 59, 67, 72, 79, 87, 92, 97, 103, 111, 117, 123, 131, 138, 145, 151, 157, 168, 178, 185, 196, 202, 214, 220, 232, 243, 251, 259, 266, 273, 287, 294, 303, 319, 331, 343, 352, 361, 370, 380, 390, 402, 409, 420, 429, 441, 450, 463, 468, 478, 485, 497, 506, 518, 523, 528, 534, 547, 554, 564, 571, 575, 584, 592, 601, 606, 615, 625, 629, 634, 639, 651, 657, 666, 671, 677, 690, 699, 707, 717, 726, 743, 750.

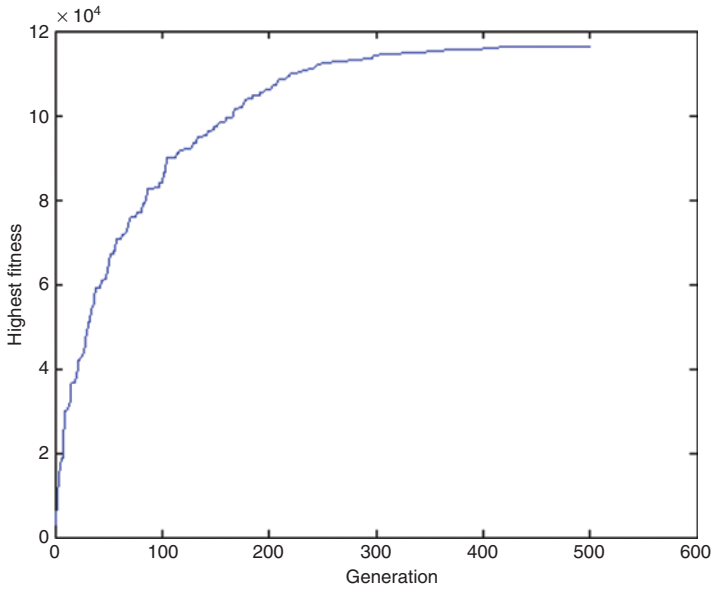

Figure 3. Fitness of the Historically Best Individual.

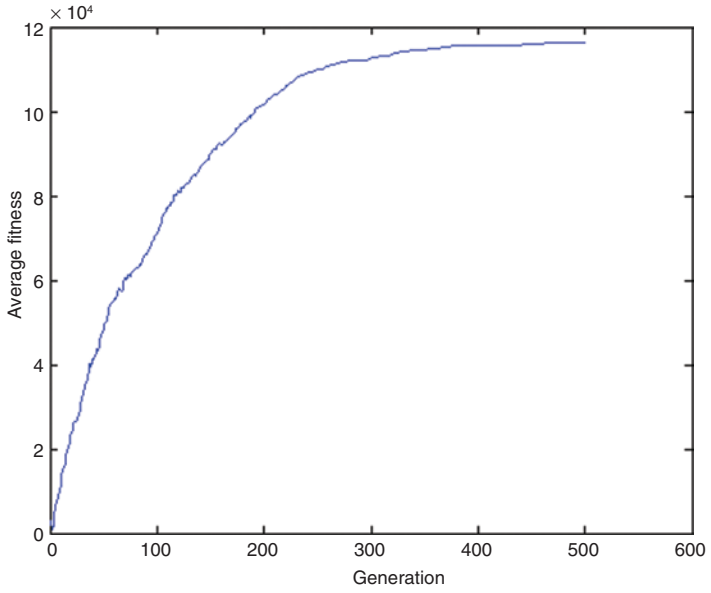

Figure 4. Average Fitness Curve.

The corresponding optimal timetable is as follows: 6:00 – 6:10 – 6:17 – 6:23 – 6:27 – 6:31 – 6:36 – 6:41 – 6:46 – 6:51 – 6:59 – 7:07 – 7:12 – 7:19 – 7:27 – 7:32 – 7:37 – 7:43 – 7:51 – 7:57 – 8:03 – 8:11 – 8:18 – 8:25 – 8:31 – 8:37 – 8:48 – 8:58 – 9:05 – 9:16 – 9:22 – 9:34 – 9:40 – 9:52 – 10:03 – 10:11 – 10:19 – 10:26 – 10:33 – 10:47 – 10:54 – 11:03 – 11:19 – 11:31 – 11:43 – 11:52 – 12:01 – 12:10 – 12:20 – 12:30 – 12:42 – 12:49 – 13:00 – 13:09 – 13:21 – 13:30 – 13:43 – 13:48 – 13:58 – 14:05 – 14:17 – 14:26 – 14:38 – 14:43 – 14:48 – 14:54 – 15:07 – 15:14 – 15:24 – 15:31 – 15:35 – 15:44 – 15:52 – 16:01 – 16:06 – 16:15 – 16:25 – 16:29 – 16:34 – 16:39 – 16:51 – 16:57 – 17:06 – 17:11 – 17:17 – 17:30 – 17:39 – 17:47 – 17:57 – 18:06 – 18:23 – 18:30.

Compared with the classical GA, the improved algorithm has a faster convergence speed. At the same time, the best individual's fitness is higher, and the ability to jump out of the local optima becomes stronger. We used two data samples to test the GA, SAGA, and the improved method, and the results show that the optimization capacity of the improved algorithm increased on a large scale compared with the other two methods (see Figures 5 and 6). In Figures 5 and 6, the horizontal axis represents the generation, and the vertical axis

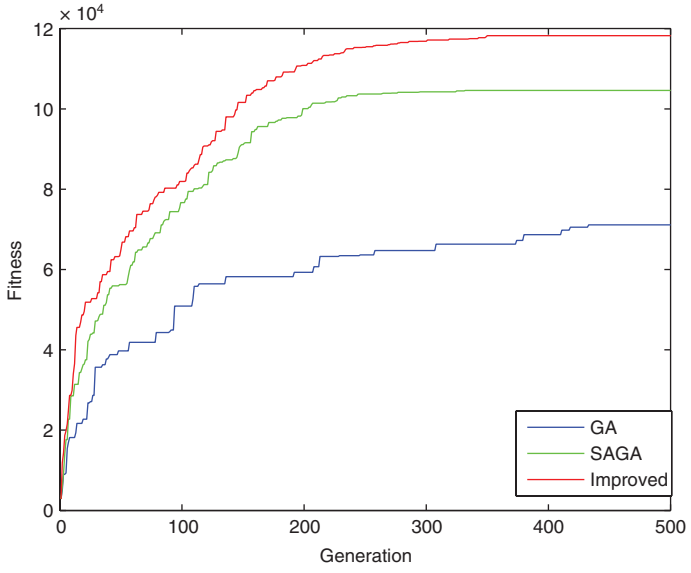

Figure 5. Data Sample 1.

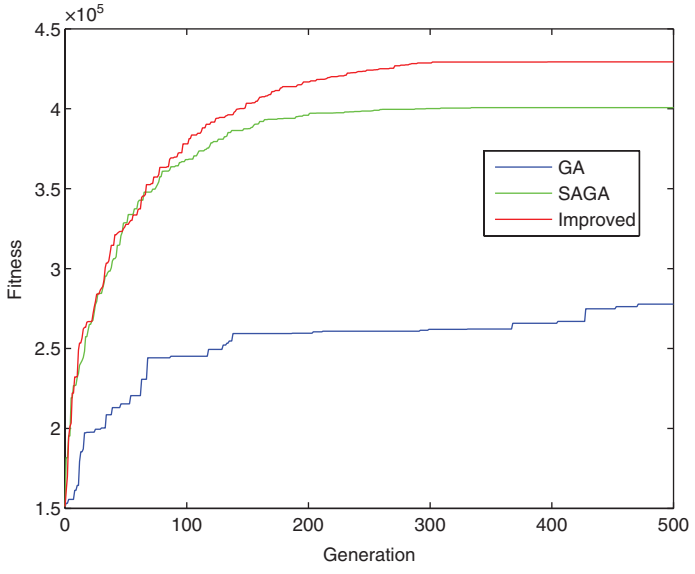

Figure 6. Data Sample 2.

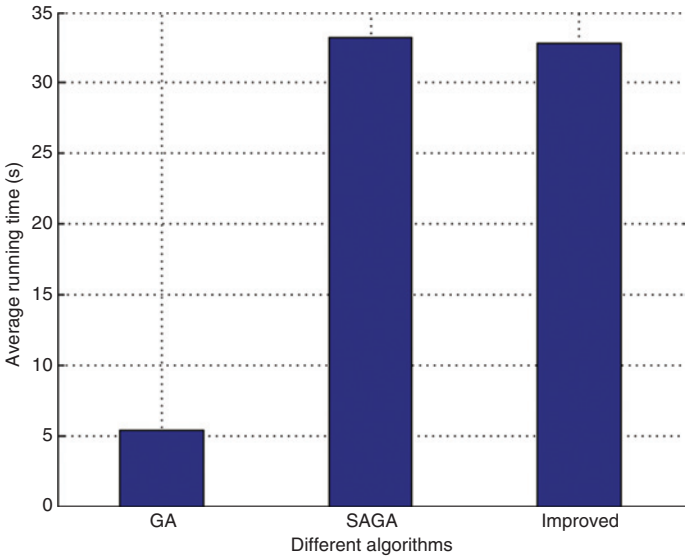

Figure 7. Running Time of Different Methods.

Table 4. Optimal Result Comparison of Improved Hybrid Heuristic Algorithm, GA, and SAGA.

Name	Original	GA	SAGA	Improved
Wait time (h)	185	166	157	153
Reduced (%)	–	10.23	15.25	17.29

represents the maximum fitness of individuals in the current population. This stems from the fact that the fitness scaling function amplifies the differences between individuals in the late stage. The possibility of individuals having a higher fitness value to be selected will increase when the difference of the individuals' fitness is small. The fitness scaling method will also help the algorithm find better solutions. In addition to the fact that the improved method can obtain better solutions, its running time almost equals that of SAGA's (see Figure 7). To carry out the SA operation, it is necessary to recalculate the fitness of the individual after the crossover and mutation operations; thus, SAGA is much more time consuming than GA. The scheduling scheme obtained by the improved hybrid heuristic algorithm can reduce the passengers' total waiting time by 7.86% compared with the classical GA, and by 2.4% compared with SAGA. The total waiting time of the scheduling scheme obtained by classical GA is approximately 166 h. The total waiting time of the scheduling scheme obtained by SAGA is roughly 157 h. However, the total waiting time of the scheduling scheme obtained by the improved hybrid heuristic algorithm is about 153 h (Table 4).

5 Conclusion

In summary, the integration of SAGA and the fitness scaling method has a good effect on the use of SAGA for solving the intelligent scheduling problems of BRT. It can find the optimal or approximately optimal solution to bus scheduling problems in a large solution space and at a faster rate. This also proves that the fitness scaling method can indeed improve the SAGA. However, in the process of designing the algorithm, we simplified the problem and made some assumptions. Furthermore, we also did not consider how to react to sudden changes of the passenger flow. Meanwhile, the passenger flow forecasting problem associated with bus scheduling, namely considering the weather, season, holidays, and other factors to predict the flow

accurately, also needs to be addressed. Theories and methods about these areas need further analysis and research in the future.

Acknowledgments: This article was financially supported by Public Projects of Zhejiang Province, China (no. 2013C33082).

Bibliography

[1] M. Barkaoui and M. Gendreau, An adaptive evolutionary approach for real-time vehicle routing and dispatching, *Comput. Oper. Res.* **40** (2013), 1766–1776.
[2] M. Bielli, M. Caramia and P. Carotenuto, Genetic algorithms in bus network optimization, *Transport. Res. Part C Emerg. Technol.* **10** (2002), 19–34.
[3] S. Bunte and N. Kliewer, An overview on vehicle scheduling models, *Pub. Trans.* **1** (2009), 299–317.
[4] A. Ceder, Methods for creating bus timetables, *Transport. Res. Part A Gener.* **21** (1987), 59–83.
[5] A. Ceder, S. Hassold and B. Dano, Approaching even-load and even-headway transit timetables using different bus sizes, *Pub. Trans.* **5** (2013), 193–217.
[6] V. Guihaire and J. K. Hao, Improving timetable quality in scheduled transit networks, in: *Trends in Applied Intelligent Systems*, pp. 21–30, Springer, Berlin, 2010.
[7] A. Hadjar, O. Marcotte and F. Soumis, A branch-and-cut algorithm for the multiple depot vehicle scheduling problem, *Oper. Res.* **54** (2006), 130–149.
[8] X. Hao, W. Jin and Y. Yang, Scheduling combination optimization research for bus lane line, *TELKOMNIKA Ind. J. Electr. Eng.* **12** (2014), 809–817.
[9] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*, University of Michigan Press, Ann Arbor, MI, 1975.
[10] J. Jerald, P. Asokan, R. Saravanan and A. Delphin Carolina Rani, Simultaneous scheduling of parts and automated guided vehicles in an FMS environment using adaptive genetic algorithm, *Int. J. Adv. Manuf. Technol.* **29** (2006), 584–589.
[11] S. Kirkpatrick and M. P. Vecchi, Optimization by simulated annealing, *Science* **220** (1983), 671–680.
[12] Z. Liu, Y. Yan, X. Qu and Y. Zhang, Bus stop-skipping scheme with random travel time, *Transport. Res. Part C Emerg. Technol.* **35** (2013), 46–56.
[13] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, Equation of state calculations by fast computing machines, *J. Chem. Phys.* **21** (1953), 1087–1092.
[14] H. L. Petersen, A. Larsen, O. B. G. Madsen, B. Petersen and S. Ropke, The simultaneous vehicle scheduling and passenger service problem, *Transport. Sci.* **47** (2012), 603–616.
[15] X. L. Tang and S. H. Yang, An improved bus timetable scheduling model using quantum genetic algorithm based on penalty strategy, *Appl. Mech. Mater.* **253** (2013), 1406–1409.