Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 719409, 14 pages
http://dx.doi.org/10.1155/2015/719409

Research Article

Improved Genetic Algorithm with Gene Recombination for Bus Crew-Scheduling Problem

Cuiying Song,[1] Wei Guan,[2] Jihui Ma,[1] and Tao Liu[3]

[1]*MOE Key Laboratory for Urban Transportation Complex Systems Theory and Technology, School of Traffic and Transportation, Beijing Jiaotong University, 3 Shang Yuan Cun, Haidian District, Beijing 100044, China*
[2]*Institute of Systems Engineering and Control, School of Traffic and Transportation, Beijing Jiaotong University, 3 Shang Yuan Cun, Haidian District, Beijing 100044, China*
[3]*Department of Civil and Environmental Engineering, Transportation Research Centre, The University of Auckland, 20 Symonds Street, Auckland 1142, New Zealand*

Correspondence should be addressed to Wei Guan; weig@bjtu.edu.cn and Jihui Ma; jhma@bjtu.edu.cn

Received 8 December 2014; Accepted 9 April 2015

Academic Editor: Michael Small

This paper presents an improved genetic algorithm (GA) with gene recombination for bus crew-scheduling problem in bus company. Unlike existing methods that rely on designing a fixed potential shift set by software, our new method does not need such a potential shift set information. In our method, satisfied shifts are generated through gene recombination in genetic algorithm. We conduct extensive studies based on real-life instances from Beijing Bus Group. Compared with results generated by the current manual method, ant colony algorithm, and CPLEX, computational results show that our algorithms demonstrated very good computational performances. In our tests, the number of the maximum reducing shifts can be beyond 30, especially when trip number is very large. The high relative percentage deviation demonstrated the effectiveness of the algorithm proposed.

1. Introduction

It has been known for more than 50 years [1] that the crew-scheduling problem for bus drivers (CSP-BD) is one of the most important operational-planning problems in a bus company, because, from the transit agencies' perspective, the largest cost of providing service is generated by drivers' wages and fringe benefits [2]. The CSP-BD is aimed at assigning vehicle trips to crews in such a way that each trip is covered by a shift, while guaranteeing that all other duty functions are feasible and the total cost of all duties is minimized [2–6].

CSP-BD has attracted the interest of many researchers since the 1960s, and research in this area has become very active since the 1990s. Most of the methods in the existing studies are based on mathematical programming or on hybrid approaches, which are combination of heuristics and Integer Linear Programming (ILP) [7–10]; the success and limitations of these methods have been discussed in Kwan et al. [11, 12] and Li and Kwan [13]. In the mathematical programming-based approach, CSP-BD is formulated as covering/partition problems that aimed at generating a subset of shifts to cover all pieces of trips, with the objective of minimizing total costs or total number of shifts [13]. In recent years, metaheuristics have been widely used for searching practical near-optimal solutions to NP-hard problems. Metaheuristics offer three main advantages: (a) they are usually very efficient in searching through very large solution space; (b) they can result in a feasible solution; (c) each class of metaheuristics has its own methodical and strategic structure. Until now, much effort has been made in exploring metaheuristics, such as tabu searches (TS) [9, 14, 15], ant system [16–18], and simulated annealing [19].

Genetic algorithms (GAs), one of the most important metaheuristics, form another major class for crew scheduling. In general, structure of GA for CSP-BD could be divided into two classes: one is the shift-based chromosome structure and the other is called the piece-based chromosome structure. In the shift-based structure, the chromosome length is not fixed. By changing the gene in chromosomes repeatedly through genetic operators, a feasible solution can be obtained [20, 21].

In the piece-based chromosome structure [22], a gene usually represents each piece. If a gene is chosen during the genetic operation, the corresponding shift covering the chosen gene will also be selected. Therefore, the simple crossover operation may be infeasible based on the units of shifts, let alone mutation. Furthermore, once a shift unit existed in a chromosome, no new shift units were produced except for exchanging the parent chromosomes. Thus, the efficiency of algorithm largely depends on initial parent chromosomes.

Most of researches mentioned above were based on a pregenerated set of potential fixed shifts. Through adding or removing shift units during genetic operations, an optimum solution with the least shifts is obtained. Different from the above generated method, this work introduces a new method without using the potential shift set; instead, satisfied shifts generated from process of gene recombination in genetic algorithm are employed. A different piece-based structure is taken into account and the optimal solution is generated by repeated recombination process.

The rest of paper is organized as follows. Section 2 introduces the crew-scheduling problem for bus drivers (CSP-BD) in general terms and elaborates the construction of the objective function and model. Section 3 describes the proposed genetic algorithm with gene recombination (GAGR) approach involving representation of chromosomes followed by gene recombination method. The genetic process of this section contains initialization of population, crossover, and mutation operators. In the last part of this section, an overall framework summarizes the whole process of GAGR. Section 4 provides the experiments with different parameters based on GAGR. Finally, Section 5 gives conclusion remarks and suggestions for future research.

2. Crew-Scheduling Problem for Bus Drivers

A vehicle block may be considered as a unit of work, which starts and ends at a relief opportunity (RO) that means a time and place at which a change of drivers is possible, for a vehicle in the course of one day. A driver's shift (or duty) is a set of pieces of work that can be assigned from the driver's signing on until his/her signing off at the depot. A piece of work denotes a shorter work between two consecutive ROs completed by the same vehicle. There are often some constraints on shifts, such as labor-agreement rules that limit work hours or the time of a break at the relief points. Several successive units (called spells) of a block form a portion of a shift. The concepts of block, RO, shift, spell, and piece of work are illustrated in Figure 1. In it, the process of generating two shifts on four blocks corresponding to four vehicles is illustrated. A shift typically starts from the crew signing on depot to a signing off depot. For shift 1, two spells combining a part of Block 1 and Block 2 are involved. For shift 2 three spells are used. There is a spell that contains only a piece of work in Block 3.

The crew-scheduling problem for bus drivers (CSP-BD) aims at finding a set of feasible shifts or duties that can cover all trips or vehicle blocks in a particular scheduling horizon. Each trip is a scheduled activity with specific starting and ending times and locations. The feasibility of a solution

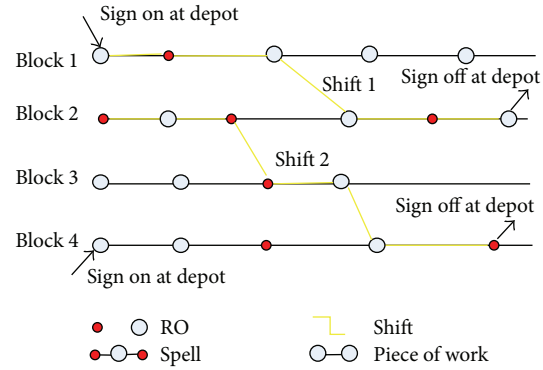

Figure 1: Procedure of generation shifts with spells.

mainly depends on many constraints, for example, if there is enough time to enable the connection of two successive trips. Generally speaking, CSP-BD involves both a vehicle-scheduling problem and a bus driver-scheduling problem.

3. Objective Function of CSP-BD

Let Ω be the set of m feasible shifts, represented by $\Omega = \{S_1, S_2, \ldots, S_m\}$ obtained from n pieces. Pieces set P can be formulated as $P = \{p_1, p_2, \ldots, p_m\}$; that is, $n = \sum_{i=1}^{m} p_i$. Let binary variable x_j equal 1 if shift j is selected and equal 0 otherwise. Let C_j^{Basic} be the basic minimum payment for a shift j, including the fixed salary, routine maintenance, and petrol cost. Note that bus purchase cost is not considered in this research. Let binary variable a_{ij} be 1 if shift j covers piece i and 0 otherwise. Let T_j denote the total driving time for shift j. Let $T_{\text{driving\_shift}}^{\max}$ be the maximum driving time stipulated by bus group. Let T_{ij} denote the driving time for one way. Let t_{idle}^{ij} be the idle time between pieces i and $i+1$ in shift j, ranging between the stipulated minimum idle time $t_{\text{idle}}^{\min}$ and maximum idle time $t_{\text{idle}}^{\max}$. Let $T_{\text{shift}}^{\text{limit}}$ denote the working time limit, say 8 h, according to the labor-agreement rules for a shift. It includes the total driving time, the total idle time, and another time ω for having a meal and signing on or signing off. C is a large constant to penalize the number of shifts in a final schedule. In addition, each bus must start and end its workday at the same depot. The crew-scheduling problem is usually formulated as the following ILP set partitioning model:

$$\min \quad \sum_{j=1}^{m} \left(C_j^{\text{Basic}} + C\right) x_j \tag{1}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} a_{ij} x_j = 1, \quad i = 1, 2, \ldots, n, \tag{2}$$

$$0 < T_j \le T_{\text{driving\_shift}}^{\max} \quad j = 1, 2, \ldots, m, \tag{3}$$

$$T_j = \sum_{i=1}^{n} a_{ij} T_{ij} \quad j = 1, 2, \ldots, m, \tag{4}$$

$$t_{\text{idle}}^{\min} \le t_{\text{idle}}^{ij} \le t_{\text{idle}}^{\max} \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, m, \tag{5}$$

$$T_j + \sum_{i=1}^{p_j - 1} t_{\text{idle}}^{ij} + \omega \le T_{\text{shift}}^{\text{limit}} \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, m, \tag{6}$$

$$x_j = \begin{cases} 1 & \text{if shift } j \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \quad j = 1, 2, \ldots, m, \tag{7}$$

$$a_{ij} = \begin{cases} 1 & \text{if shift } j \text{ covers piece } i \\ 0 & \text{otherwise} \end{cases} \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, m. \tag{8}$$

Objective (1) is to minimize the total number of shifts so as to reduce a bus company's total costs while maintaining almost the same service levels. All the constraints (2)–(8) strictly limit the range of solution. Constraint (2) ensures that each piece is served once; constraint (3) guarantees that the total driving time has to be satisfied in a proper range; constraint (5) imposes upper and lower bounds on the idle time between two successive pieces.

To depict CSP-BD more clearly, we construct it on a graph. ROs are expressed as nodes and the connecting edges are according to the constraints between two consecutive driving pieces. Table 1 gives the example of a part of the Beijing Bus Group's #26 bus timetable and also shows the feasible connecting-node set with constraints of $t_{\text{idle}}^{\min} = 3$, $t_{\text{idle}}^{\max} = 50$, and $T_{\text{shift}}^{\max} \le 400$.

4. GAGR Approach

Genetic algorithm (GA) is a global search algorithm based on the evolutionary ideas of natural selection and evolution. Except for the classical procedures crossover, selection, and mutation in GAs, the main design for GAGR is presented in this section. Piece-based chromosome representation is firstly defined. Secondly, recombination method with or without deadheads in each chromosome is illustrated in detail. Then the initial population is obtained by greedy algorithm. Next, the genetic operators of selection, crossover, and mutation are devised to generate new offspring chromosomes for the next iteration. In the end, the framework of GAGR method is presented.

4.1. Chromosome Representation. This section adopts the idea of Wren [22] that constructs a chromosome considering the pieces as genes. As displayed in Figure 2, $a1, a2, \ldots, d3, \ldots$ are genes (pieces for short in schedule) and shifts are identified as $s1, s2, s3, s4, \ldots$. The whole chromosome represents a solution and the quality of this solution is evaluated by the number of shifts or cost.

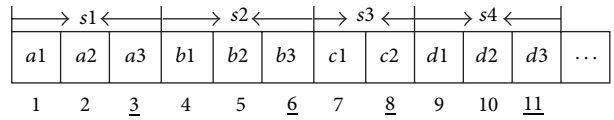

Figure 2: Representation of a chromosome with gene locus.

The number below represents each gene position (locus) and the number underlined is the last position for each shift. It may be noted that the genes are not in chronological order here as each shift is independent of the other. For example, shift 1 and shift 2 may exchange their positions and have no effect on the final solution in an iteration.

4.2. Gene Recombination. Gene recombination method starts using such generated chromosome mentioned above. In this process, several old shifts are merged into a new shift; thus, the number of shifts reduces effectively. Gene recombination could be decomposed into two main methods. In one method, no deadhead is permitted in the middle of the generated shifts; in the other method, one or two deadheads could be added to the new shift if regulations and recombination conditions meet. We set start or end points to be the joinable points. More details are depicted in Figure 3.

(1) No Deadhead in Recombination. In Figure 3(a), shift 1 and shift 3 may have no chance to merge into a new shift based on the total working time constraint. Moreover, other constraints like the maximum and minimum idle time limit must be considered. Then the remaining combination is shown as $C1$, $C2$, $C3$, and $C4$. The first or last node in shift 1 may join with shift 2 and the potential new shifts in the above picture after combination may be 1-2-3 or 3-1-2 as well as 3-4-5-6 or 4-5-6-3 as another two generated new shifts in a schedule.

(2) One Deadhead in Recombination. Obviously, in the process mentioned above, there are no repeated pieces in new solution. Sometimes, it may be proper to add some pieces to construct new shifts and thus reduce shifts efficiently when conditions are satisfied between two shifts after first recombination. In fact, because of the inexistence, those repeated pieces are replaced by deadheads. Figure 3(b) illustrates the recombination procedure. Four pieces m, n, m', and n' in original piece set prepare to bridge two shifts. If the connectivity between shift 1 and shift 2 is available, it means m, n, m', and n' must satisfy all the necessary constraints like the total driving time and depot. In addition, the idle time limitations also confine those corresponding pieces:

$$t_{\text{idle}}^{\min} \le t_{\text{idle}}^{2,m} \le t_{\text{idle}}^{\max}, \quad t_{\text{idle}}^{\min} \le t_{\text{idle}}^{m,3} \le t_{\text{idle}}^{\max}. \tag{9}$$

(3) Two Deadheads in Recombination. This procedure performs if two methods mentioned above are even not helpful to decrease the shift count. Two pieces from original piece set are selected to bridge two shifts. The feasible connection between

Table 1: Part of #26 bus timetable and feasible connecting nodes with constraints.

Trip number	Dep. depot	Dep. time	Dest. depot	Arr. time	Running time (minute)	Feasible connecting nodes
1	Erlizhuang	5:30	Xibianmen	6:48	78	8, 9
2	Erlizhuang	5:40	Xibianmen	6:58	78	8, 9
3	Erlizhuang	7:19	Xibianmen	8:49	90	10
4	Erlizhuang	9:15	Xibianmen	10:45	90	11, 12
5	Erlizhuang	9:29	Xibianmen	10:59	90	12
6	Erlizhuang	11:06	Xibianmen	12:32	86	—
7	Xibianmen	5:30	Erlizhuang	6:46	76	3
8	Xibianmen	7:21	Erlizhuang	8:49	88	4, 5
9	Xibianmen	7:26	Erlizhuang	8:54	88	4, 5
10	Xibianmen	9:27	Erlizhuang	10:55	88	6
11	Xibianmen	11:00	Erlizhuang	12:24	84	—
12	Xibianmen	11:14	Erlizhuang	12:38	84	—

shift 1 and shift 2 may construct two potential shifts. Take connection $C1$ as an example, shift 1 and shift 2 are connected by pieces m, n. The idle time constraints for those pieces are

$$\begin{aligned} t_{\text{idle}}^{\min} &\le t_{\text{idle}}^{1,m} \le t_{\text{idle}}^{\max}, \\ t_{\text{idle}}^{\min} &\le t_{\text{idle}}^{m,n} \le t_{\text{idle}}^{\max}, \\ t_{\text{idle}}^{\min} &\le t_{\text{idle}}^{n,2} \le t_{\text{idle}}^{\max}. \end{aligned} \tag{10}$$

4.3. Initialization. An initial population can be easily built by constructing individuals. Each individual is a solution generated by selecting pieces from its initial piece set. Length of an initial individual, m, is the same as the number of pieces (genes) in a schedule. And n denotes the population size. The process for initial population construction includes six stages.

(1) Randomly select pieces into start pieces set $S = \{s_1, s_2, \ldots, s_n\}$. Length of S is equal to n.

(2) Put the start pieces s_j into tabu list, a pieces set containing used pieces. Set $p_0 = s_j$, the first node in tabu list.

(3) Find a feasible piece in nontabu list, a point set with unused pieces to connect the last piece in tabu list.

(4) If existing, put the satisfied piece into tabu list and update nontabu list, and then go to step (3). Otherwise, one shift is generated and go to step (5).

(5) If nontabu list is not empty, randomly select one piece from nontabu list and put it into tabu list, and then go to step (3). Otherwise, go to step (6).

(6) Replicate all pieces in tabu list into chromosome Ch(j), and then empty tabu list.

(7) Repeat (2), (3), (4), (5), and (6) until $j = n$.

4.4. Crossover with Duplicate Gene Removal and Repairing Strategies. The standard roulette wheel method is adopted to select promising individuals according to the fitness function, an inverse relationship with cost function, from the current population $P(t)$, $t = 1, 2, \ldots, N$, where N represents generation iterations. And each chromosome may be corresponding to each individual.

A simple two-point crossover method is employed in GAGR. In view of convergence and integrity for a schedule, the duplicate genes should be removed and the lost genes in a chromosome should be readded to the crossover chromosome. Two chromosomes $\text{Ch}_i = \{a_1, a_2, \ldots, a_k, a_{k+1}, \ldots, a_m\}$ and $\text{Ch}_j = \{b_1, b_2, \ldots, b_k, b_{k+1}, \ldots, b_m\}$ are chosen from population and their cross point positions are e and f, where $1 < e < f < m$. Therefore, both chromosomes are divided into three parts: $\{a_1, \ldots, a_{e-1}\}$, $\{a_e, \ldots, a_f\}$, and $\{a_{f+1}, \ldots, a_m\}$ for Ch_i and $\{b_1, \ldots, b_{e-1}\}$, $\{b_e, \ldots, b_f\}$, and $\{b_{f+1}, \ldots, b_m\}$ for Ch_j. Let D_{a_e}, D_{a_f}, D_{b_e}, and D_{b_f} denote the relative shifts at e and f in the two exchanged chromosomes. Then, new chromosomes after crossover procedure turn into $\text{Ch}_i' = \{a_1, \ldots, a_{e-1}, b_e, \ldots, b_f, a_{f+1}, \ldots, a_m\}$ and $\text{Ch}_j' = \{b_1, \ldots, b_{e-1}, a_e, \ldots, a_f, b_{f+1}, \ldots, b_m\}$. However, simple crossover leads to multiple duplicate genes and losing genes in respective exchanged chromosome. Therefore, removal and repairing strategies are applied in GAGR.

(1) Duplicate Gene Removal Strategy. This strategy is composed of two steps: (a) seek for the duplicate genes and the corresponding shifts in the whole chromosome and (b) split those shifts and remove the duplicate genes. The most difficult part occurs in dealing with shifts at the start and end points in each exchanged part. Thus, four cases are taken into account: (1) $D_{a_e} \ne D_{a_{e-1}}$ $D_{a_f} \ne D_{a_{f+1}}$, where it states that shifts in the two exchanged parts are all intact; thus no split exists; (2) $D_{a_e} = D_{a_{e-1}}$ $D_{a_f} \ne D_{a_{f+1}}$, where the shift at the start point has to be divided into two segments $D_{a_{e-2}}$ and D_{a_e}; (3) $D_{a_e} \ne D_{a_{e-1}}$ $D_{a_f} = D_{a_{f+1}}$, where the shift at the end point has to be divided into two segments $D_{a_{f-1}}$ and $D_{a_{f+1}}$; (4) $D_{a_e} = D_{a_{e-1}}$ $D_{a_f} = D_{a_{f+1}}$, where both shifts at start and end points should be split into two parts $D_{a_{e-2}}$ and D_{a_e} as well

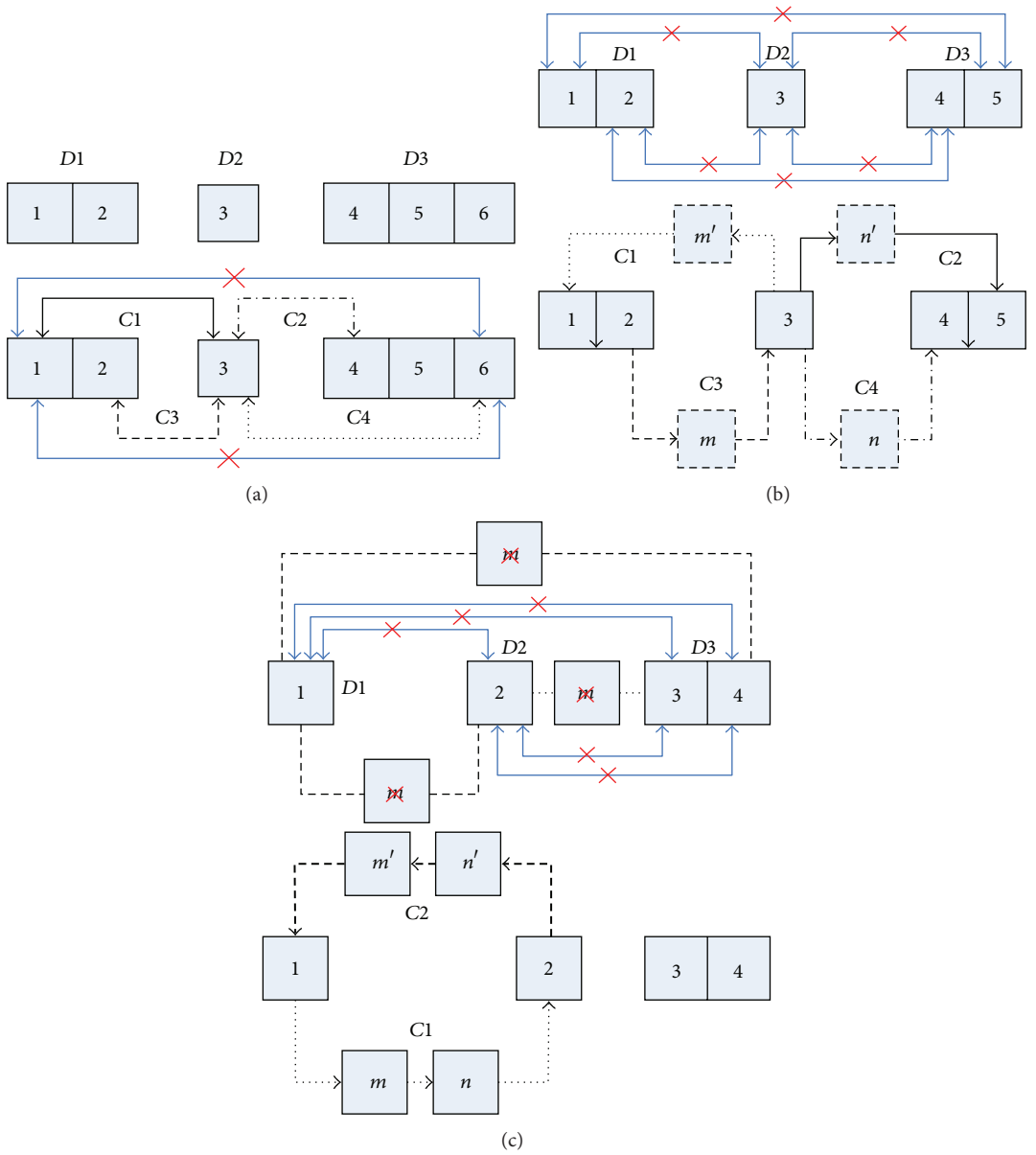

FIGURE 3: The process of recombination with different situations (a) with no deadhead, (b) with one deadhead, and (c) with two deadheads. There are three shifts $D1$, $D2$, and $D3$ and six pieces chosen from one schedule containing one depot; the driving time for each piece is 90 minutes and maximum total driving time is 400 minutes for hypothesis.

as $D_{a_{f-1}}$ and $D_{a_{f+1}}$. See more details in Figures 4(a), 4(b), and 4(c).

Seek for the duplicate genes in exchanged part and nonexchanged parts, and then pick the duplicate genes and the corresponding shifts out. Delete those genes in nonexchanged parts and move the remaining genes to the end of chromosome. If remaining genes are joinable, then they combine into a new shift. Otherwise, the single gene itself constructs a new shift. The process of gene removal strategy is illustrated in Figure 4(d).

(2) Losing Gene Repairing Strategy. Genes not existing in their exchanged part are the losing genes to replenish. Let B_i, B_j denote the difference set of two exchanged parts, $A_i = \{a_e, a_{e+1}, a_{e+2}, \ldots, a_{f-1}, a_f\}$ and $A_j = \{b_e, b_{e+1}, b_{e+2}, \ldots, b_{f-1}, b_f\}$:

$$B_i = A_i \setminus A_j,$$
$$B_j = A_j \setminus A_i. \tag{11}$$

Put the losing genes in B_i (or B_j) into the end of new chromosome Ch_i' (or Ch_j'). Each losing gene constructs a new shift.

4.5. Mutation with Removal and Repairing Strategies. The mutation procedure is devised as follows: (1) Define some

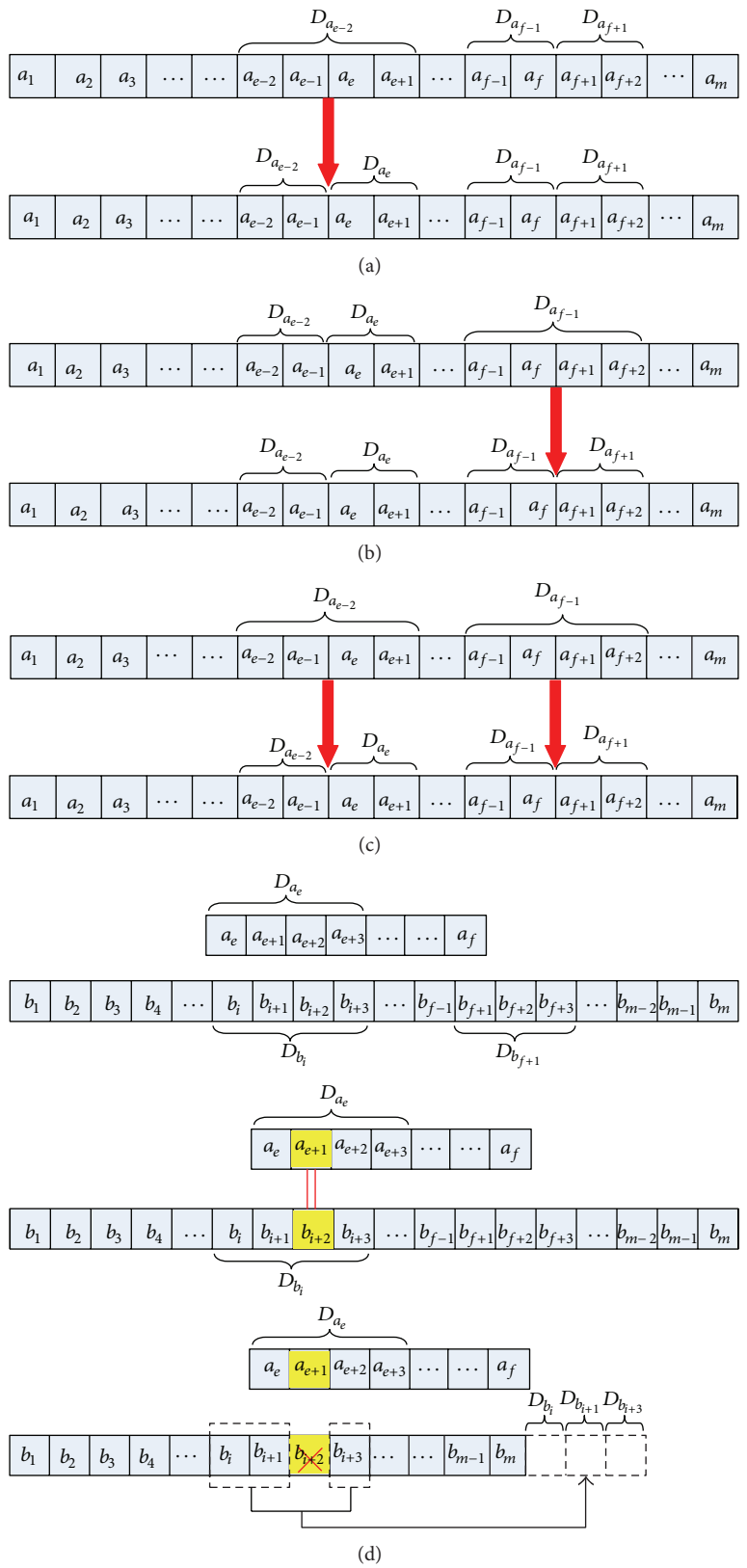

FIGURE 4: Illustration of a shift at the head of cross point divided into two segments for different situations: (a) $D_{a_e} = D_{a_{e-1}}$ and $D_{a_f} \neq D_{a_{f+1}}$, (b) $D_{a_e} \neq D_{a_{e-1}}$ and $D_{a_f} = D_{a_{f+1}}$, (c) $D_{a_e} = D_{a_{e-1}}$ and $D_{a_f} = D_{a_{f+1}}$, and (d) gene removal strategy. Exchanged part of Ch_i is above the nonexchanged part Ch_j and the value of gene a_{e+1} is equal to b_{i+2}.

parameters such as mutation rate p_m ($0 < p_m < 1$) and a small constant k (say 2 or 3) denoting the number of positions to be mutated in a chromosome; (2) randomly choose a number $p \in [0, 1]$ for each individual; (3) compare mutation rate p_m and the random number p. If $p < p_m$, randomly choose k positions. Then the shifts on these positions are selected and split into several segments according to the specific positions. (4) These segments are constructed into new shifts, then previous selected shifts are deleted at the meanwhile.

The final result in each chromosome may contain many piece shifts, that is, only one piece in a shift. Therefore, the recombination method mentioned above plays a significant role in decreasing objection.

4.6. Framework of GAGR. The framework of GAGR approach is outlined as follows:

(1) Initialize parameters such as the global target shifts $L = M$ (M is a large constant), maximum generations T, crossover rate p_c ($0 < p_c < 1$), mutation rate p_m ($0 < p_m < 1$), and generation gap g_{gap} ($0 < g_{\text{gap}} < 1$).

(2) Set $t = 1$, and construct an initial population $P(t)$ containing K individuals. Then transform $P(t)$ into $P'(t)$ by gene recombination methods with no deadhead. Update $L = L_g$ if a less global target L_g appears.

(3) For each individual $\text{Ch}_i(t) = \{c_1, c_2, c_3, \ldots, c_m\}$ in $P(t)$, calculate its fitness value according to the objective function presented in formula (1).

(4) By using roulette wheel, select a set of promising individuals $P'(t)$ according to the generation gap. And the size K' in $P'(t)$ is denoted as $K' = K * g_{\text{gap}}$.

(5) If the random number p_k ($0 < p_k < 1$) satisfies $p_k < p_c$, pair individuals randomly in $P'(t)$. Then through crossover, the initial pair turns into a children pair. And the target shift number L may also decrease adaptively. The new offspring population is denoted by $PO'(t)$.

(6) Mutate each individual in $PO'(t)$ according to the operator p_m.

(7) Construct a new population $P(t + 1)$. Its individuals consist of all individuals from $PO'(t)$ and the best individuals from $P(t)$. Update $L = L_g$ if a better global target L_g appears.

(8) Set $t = t + 1$. If the termination is not met, that is, perform procedures from (2) to (7) mentioned above.

5. Computational Results

5.1. Comparison with CPLEX. To testify the accuracy for the results of GAGR, two simple examples with 36 and 72 pieces of work on single depot are tested on CPLEX, a useful tool on mathematical programming problems. The maximum iteration was set for 100 and the corresponding parameters were set for $p_m = 0.05$, $p_c = 0.2$, and $g_{\text{gap}} = 0.8$, respectively. The results are listed in Table 2 and the minimum results in each iteration are also depicted in Figure 5. From Table 2, little difference of the minimum results in both methods is shown, especially when problem size is small. Both of the running times are maintained within an acceptable range. The minimum shift for each iteration is displayed in Figure 5. Though the best result in GAGR for this example fails to reach the correct answer, the whole descending process presents the convergence of GAGR. Besides, the first result close to the correct answer shows the effectiveness for reducing shifts at the beginning of iteration.

Table 2: The minimum results compared with CPLEX.

Number of pieces	GAGR		CPLEX	
	Shifts	Running time (s)	Shifts	Running time (s)
36	6	4.02	6	2.58
72	13	6.21	12	3.13

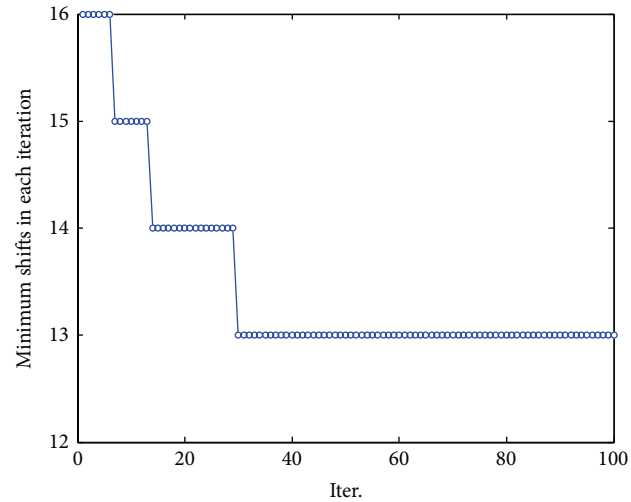

Figure 5: The minimum shifts in each iteration when the number of pieces is 72.

In order to test the performance of GAGR, experiments on 10 lines in Beijing have been done. The lines information is showed in Table 3. Manual results are also listed in the table, regarded as the best results. The computational experiments were carried out by programming in MATLAB. We set the population size m to be 100 and iterations to be 20. To simplify the computation difficulty, other experimental parameters were as follows:

$$T_{\text{driving}}^{\max} = 400;$$
$$t_{\text{idle}}^{\min} = 3; \quad (12)$$
$$t_{\text{idle}}^{\max} = 30.$$

Experiments were firstly carried out on the initial chromosomes by the gene recombination. We tried to apply different parameters in order to achieve the best result.

Table 3: Size of test instances and manual results.

Line	Number of blocks	Number of pieces of work	Manual results (MR) (shifts)
#306	18	288	33
#695	36	448	64
#348	16	482	31
#26	44	598	80
#43	25	582	49
#467	18	620	36
#28	27	756	52
#34	24	724	48
#345	72	1236	141
#322	74	2216	139

Finally, the gene recombination method with one or two deadheads is also applied to the last experiment.

5.2. Experiments on the Initial Population with Gene Recombination (GR) Method. From Table 4, we can see a large gap between results with GR and the ones without. The maximum reducing shifts are more than one hundred shifts on Line #322. An interesting discovery shows that more pieces of work on a line may lead to larger reducing shifts, except for #348 and #34. The reason may be that more pieces of work lead to more chances to choose randomly, thus generating large shifts. This paper also applied the relative percentage deviation (RPD) according to Li and Kwan [23] over the best known schedule to measure the quality of a heuristic schedule shown below. In Table 3, we noted that all the RPD results are positive; thus, it may not achieve ideal results if just using GR method.

5.3. Experiments on the GAGR with Different Parameters. Although GAGR approach presents its advantage on exploring new shifts and thus decreases the number of shifts, different parameters may influence its performance. These parameters consist of two types: one is called the computational parameters; parameter combinations considered in GAGR include (i) crossover rate p_c ($0 < p_c < 1$), (ii) the mutation rate p_m ($0 < p_m < 1$), and (iii) generation gap ($0 < g_{\text{gap}} < 1$); the other one is the objective parameters shown in the objective function, while in this paper, the objective parameter is C, a large constant in formula (1). Therefore, we set p_c from 0.1 to 1.0 stepped by 0.1, p_m from 0.05 to 0.3 stepped by 0.05, and gap from 0.5 to 0.9 stepped by 0.1 according to the classical GA method. The large constant C is set to be 10, 100, and 1000. Each line with each parameter combination runs for 10 times and each run iterates for 20 times.

(1) Experiments on GAGR with Different p_c. We set $p_m = 0.05$, $g_{\text{gap}} = 0.9$, and $C = 100$ as default values when $p_c \in [0.1, 1.0]$. Table 5(a) presents the minimum and average results of experiments for each bus line, and the last row displays their average relative percentage deviation (RPD).

We can see that most of the minimum results are better than the manual results (MR), especially #322, for all the results are much lower. It is testified to improve quality of solutions considering the negative values of average RPD. And tables also show something different from the classical GA; that is, a higher value of $p_c \in [0.8, 1.0]$ may not achieve the anticipated results. The reason may be that it is likely to produce more piece shifts after crossover operation if no more short shifts can be linked with others to generate new shifts. p_c ranging from 0.2 to 0.4 is more favorable to obtain optimum results especially when $p_c = 0.2$ for there are 5 optimum results and the lowest average RPD shown in tables. So we may set $p_c = 0.2$ in the following tests.

(2) Experiments on GAGR with Different p_m. From Table 5(b), we obtain that all the average RPD is negative and the deviation in the same line is small. It demonstrates that different p_m may not change much. The reason is likely to be that the mutation operator can raise the possibility to generate a better solution, but not necessary. When $p_m = 0.05$, optimal results for this value take up 70% and the average RPD is obviously lower than other values. Therefore, we determine to set $p_m = 0.05$.

(3) Experiments on GAGR with Different g_{gap}. As described in Table 5(c), average RPD for different g_{gap} values are also close. The minimum results equally distribute in various g_{gap} values, while, as seen from the average RPD of minimum and average results in Table 5(c), the suitable generation gap is set to 0.8 ultimately.

(4) Experiments on GAGR with Different C. In this section, we run all the programs again and compare the results in different C. From Table 5(d), we can see that almost all the results are better than the manual ones. However, all the best results may not appear in the largest C when shifts are in a large number, for example, #345 and #322. In other words, the large constant constraint may have little effect on large-scale problems using GAGR. In addition, we discover that the results on a large C approximate the minimum results. The reason may be the randomness in GAGR and fewer tests in this paper. To test the validity, more experiments will be made in our future work. To keep accuracy, we set $C = 1000$ in our following experiments.

In the process of seeking for the suitable parameter combinations, we may discover that some optimal values appear in different parameter combinations. For example, in #306, the optimal value 26 is presented in different parameter combinations. In addition, even if the selected excellent parameters are set to experiments, sometimes the optimum results cannot be achieved. We may guess that (a) the composition of optimum shifts may be different from each other; (b) the randomness of GRGA may lead to the same result in different parameter combinations; and (c) as most of the metaheuristic algorithm the chosen parameter combinations only raise the possibility to generate a better solution but may not guarantee the certainty.

Table 4: Results of initial population with GR with no deadhead.

Line	Before GR (shifts)			After GR (shifts)			Number of reducing shifts			RPD with minimum shifts (%)
	Max	Min	Avg.	Max	Min	Avg.	Max	Min	Avg.	
#306	59	46	52.29	45	37	40.73	18	5	11.56	12.12
#695	110	92	100.95	86	79	82.81	25	10	18.14	23.43
#348	82	56	68.87	51	40	45.36	37	14	23.55	29.03
#26	122	105	114.02	98	87	91.47	32	13	22.55	8.75
#43	115	95	104.42	86	74	79.41	35	15	25.01	51.02
#467	95	66	83.77	61	50	55.71	39	13	28.08	38.89
#28	107	80	88.89	77	61	67.68	40	13	21.21	17.31
#34	109	89	99.52	76	64	69.62	39	21	29.9	33.33
#345	237	206	223.58	182	166	173.58	65	37	50	17.73
#322	295	260	274.89	199	181	193.58	105	66	83.36	30.22

5.4. Experiments on the Final Population with GRD Method. Based on the previous experiments on the initial population, this method tends to obtain a better solution based on the final population considering deadhead; that is, after GA the whole population is recombined with one deadhead. 20 tests are performed with the selected parameter combinations and the minimum and average results before and after this method are also presented to testify the quality of this method.

As Table 6 shows, the results are improved after the combination method with one deadhead. It is interesting to find that the maximum reducing shifts reach 13; that is, 13 shifts are saved by adding one deadhead. In our tests, results in most lines are much better than the obtained optimal results except for some lines, for example, #348 and #345. Thus, the combination method with one deadhead is quite effective in reducing shifts.

5.5. Comparison with Ant Colony Algorithm (ACA). To further test the performance of GAGR, it is necessary to compare it with other metaheuristics methods. Though different GA methods [21] with shift-based chromosome structure for CSP-BD have been provided, it is difficult to completely restore these methods because of lacking a set of potential shifts in such a huge problem. Therefore, a similar selecting piece method, ant colony method, is proposed to compare the effectiveness and efficiency of the presented GAGR method. The computational testing of GAGR was carried out by applying the code and comparing the results to the ant colony algorithm (ACA) running under the identical experimental conditions. The parameters of the proposed GAGR are set as follows: $p_m = 0.05$, $p_c = 0.2$, and $g_{\text{gap}} = 0.8$, $C = 1000$. Meanwhile, the parameters for ant colony algorithm are as follows: (1) exploration threshold $q_0 = 0.95$; (2) the relative importance of pheromone trails versus heuristic function $\alpha : \beta = 1 : 2$; (3) trace persistence coefficient $1 - \rho = 0.9$. The process of ACA is illustrated in Figure 6 and its steps are outlined below.

Step 1. Initialize the necessary variables and parameters and define the heuristic function and initial pheromone density.

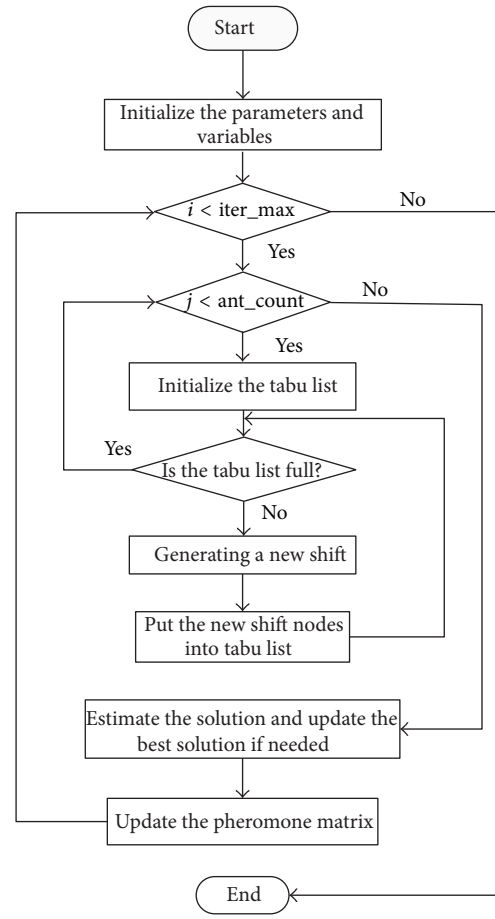

Figure 6: The process of ant colony algorithm.

Step 2. If the iteration number is less than the original maximum number, then judge the ant number, or else go to Step 6.

Step 3. Initialize the tabu list that contains the visited nodes and generate a new shift if the tabu list is not full. Before generating a new shift, feasible connecting-node set with

Table 5: Comparative minimum results and average results of 10 runs for six lines (a) with different p_c for $p_m = 0.05$, $g_{gap} = 0.9$, (b) with different p_m for $p_c = 0.2$, $g_{gap} = 0.9$, (c) with different g_{gap} for $p_m = 0.05$, $p_c = 0.2$, and (d) with different C for $p_c = 0.2$, $p_m = 0.05$, $g_{gap} = 0.8$. (The bold number is the minimum shift number for each bus line.)

(a)

p_c	Shifts										
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	MR
#306	30	**26**	29	29	29	30	28	31	30	32	33
#695	65	**63**	65	**63**	64	64	65	64	64	67	64
#348	22	**21**	22	22	22	23	24	25	24	27	31
#26	81	80	79	**78**	81	79	81	79	82	82	80
#43	**49**	50	50	51	54	53	52	56	52	59	49
#467	36	37	**35**	36	37	37	38	37	40	42	36
#28	45	45	**44**	47	46	47	48	49	51	51	52
#34	46	**45**	**45**	**45**	46	47	46	47	49	50	48
#345	128	**118**	121	124	126	128	132	132	140	139	141
#322	109	109	105	**103**	104	110	107	110	116	122	139
Avg. RPD (%)	−8.37	−10.78	−10.19	−9.5	−7.84	−6.7	−6.45	−4.47	−2.9	1.69	
#306	31.2	**30.1**	30.7	30.7	30.9	31.4	31.6	32.5	32.2	33.2	33
#695	66.3	**66**	67	66.4	65.6	66.1	67.1	66.9	68.3	68.3	64
#348	24	**23.2**	24.1	23.6	23.9	24.8	25	26.1	27.1	29.5	31
#26	81.9	82.5	81.7	**81.3**	82	82.1	82.7	82.4	83.2	83.4	80
#43	**53.6**	56.2	55.6	57.3	57	58.1	57.5	59.7	58.8	61.9	49
#467	**37.8**	38.6	38	38.1	39.1	39.1	40.2	40.3	41.9	44	36
#28	**47.9**	48.2	48	48.9	50	50.2	50.9	51.8	53.2	55.2	52
#34	47.5	**46.9**	47.4	47.1	47.8	49	49	49.2	51.5	52.6	48
#345	133.3	**127.7**	133.1	131.4	133.5	138.1	140.9	141.4	144.6	146.9	141
#322	112	112	**110.8**	111.9	111.6	113.3	114.4	117.8	120	112	139
Avg. RPD (%)	−4.15	−4.42	−3.82	−3.68	−2.86	−1.37	−0.42	1.11	3.06	5.58	

(b)

p_m	Shifts						
	0.05	0.1	0.15	0.2	0.25	0.3	MR
#306	**26**	30	29	**26**	29	29	33
#695	**63**	65	63	**61**	64	64	64
#348	**21**	22	22	23	24	23	31
#26	80	**78**	80	80	79	79	80
#43	**50**	52	53	53	51	53	49
#467	37	37	**36**	**36**	37	**36**	36
#28	**45**	47	47	47	47	48	52
#34	**45**	44	45	47	47	46	48
#345	**118**	122	125	126	127	124	141
#322	109	**104**	108	107	108	105	139
Avg. RPD (%)	−10.78	−8.68	−8.41	−8.89	−7.3	−7.94	
#306	**30.1**	31	31	30.6	30.7	30.9	33
#695	66	66.7	65.7	**65.6**	65.8	66.4	64
#348	**23.2**	23.7	24	25	24.9	24.7	31
#26	82.5	**81.2**	82.3	81.4	82	81.9	80
#43	56.2	56.1	56.6	**56**	**56**	56.8	49
#467	38.6	38.3	**37.4**	38.4	38	38.3	36

(b) Continued.

p_m	Shifts						
	0.05	0.1	0.15	0.2	0.25	0.3	MR
#28	**48.2**	48.4	49.6	49.1	49.6	49.2	52
#34	**46.9**	47.3	47.4	48.9	47.9	48.3	48
#345	**127.7**	131.7	133.6	132.4	132.9	133.2	141
#322	112	110.4	112.4	110.9	110.7	**109.5**	139
Avg. RPD (%)	−4.42	−3.86	−3.4	−3.14	−3.24	−2.98	

(c)

g_{gap}	Shifts					
	0.5	0.6	0.7	0.8	0.9	MR
#306	29	29	29	**26**	**26**	33
#695	62	62	63	**60**	63	64
#348	23	**21**	23	22	**21**	31
#26	**77**	79	79	80	80	80
#43	51	51	52	53	**50**	49
#467	37	**36**	**36**	**36**	37	36
#28	48	45	**44**	46	45	52
#34	**45**	47	46	**45**	**45**	48
#345	125	126	126	119	**118**	141
#322	**103**	108	104	104	109	139
Avg. RPD (%)	−8.91	−9.32	−9	−10.69	−10.78	
#306	30.8	31.2	30.3	28.4	**30.1**	33
#695	67.4	66.5	67	**65.7**	66	64
#348	24.6	23.8	24.1	23.5	**23.2**	31
#26	82.1	**81.8**	81.9	82	82.5	80
#43	56.5	56.5	55.5	**55.4**	56.2	49
#467	38.5	**37.4**	**37.4**	37.8	38.6	36
#28	50	49	**47**	48.1	**48.2**	52
#34	47.5	49	47.5	**46.9**	**46.9**	48
#345	132	134.3	133	130.5	**127.7**	141
#322	111.5	113.5	111.5	**110**	112	139
Avg. RPD (%)	−2.82	−3.01	−4.23	−5.3	−4.42	

(d)

C	Shifts				
	1	10	100	1000	MR
#306	28	29	29	28	33
#695	62	63	65	62	64
#348	21	21	21	22	31
#26	79	80	79	76	80
#43	49	52	49	48	49
#467	36	36	36	35	36
#28	45	46	47	45	52
#34	45	45	46	45	48
#345	121	120	122	122	141
#322	108	105	103	109	139
Avg. RPD (%)	−10.8	−9.7	−9.72	−10.89	−10.8
#306	30.4	30.9	30.6	**30**	33
#695	66.2	66.3	67	**65.3**	64
#348	**22.8**	23.5	24.3	23.8	31

(d) Continued.

C	Shifts				
	1	10	100	1000	MR
#26	82.2	81.5	81.8	**80.3**	80
#43	**54.1**	55.1	56.2	54.6	49
#467	37.5	37.5	**37.2**	**37.2**	36
#28	**47.8**	48.2	49.5	**47.8**	52
#34	47.4	**46.8**	47.9	47.8	48
#345	130.1	128.9	**128.4**	133.1	141
#322	110.4	110.7	**109.5**	109.6	139
Avg. RPD (%)	−5.12	−4.72	−3.91	−5.04	−5.12

Table 6: Minimum results of 20 runs for ten lines before and after using GRD method with $p_m = 0.05$, $p_c = 0.2$, and $g_{\text{gap}} = 0.8$.

Line	Without deadhead			With one deadhead			Max reducing shifts
	Shifts	RPD (%)	Running time(s)	Shifts	RPD (%)	Running time(s)	
#306	27	−18.18	253.78	26	−18.18	255.33	3
#695	62	−3.125	788.32	61	−4.6875	803.917	2
#348	22	−29.03	589.417	22	−29.03	593.833	2
#26	79	−0.0125	824.2	78	−0.025	837.2	5
#43	49	0	1316.633	48	−2.04	1331.2	5
#467	36	0	736.05	35	−2.78	739.7	3
#28	44	−15.3846	1293.233	42	−19.2308	1303.633	4
#345	120	−14.8936	5047.083	119	−15.6028	5992.683	13
#34	43	−10.42	1499.117	43	−10.42	1512.633	3
#322	103	−25.8993	10519.77	101	−27.3381	10942.5	5

constraints like GAGR for each node is also built. The rule of choosing next node depends on the heuristic function that defines the closeness between two nodes and the pheromone density in each iteration. Besides randomness comparison with exploration threshold is also reflected in the process of choosing node.

Step 4. Put an unselected node for a new shift into tabu list until the tabu list is full. If all ants finish their routes, estimate all solutions by calculating the total cost and store the best one. Otherwise, return to Step 3.

Step 5. Update the pheromone density in the pheromone matrix according to the updating rules. Then continue implementing the algorithm and searching for the obtained best solution.

Step 6. Stop.

The criterion used to terminate the iterative search process of GAGR and ACA is that when a maximum number of iterations have been exceeded or when no improvements are observed over a number of iterations (Algorithm 1). Then the algorithm can be considered to have converged. Each line runs for 20 times.

From the comparison results, the minimum results show a large gap between GAGR and ACA and the gap is larger on large pieces number (Table 7). The results on GAGR improve much more than ACA results, since all the ACA results are not better than the manual results seen from the positive RPD values. However, the running time in ACA is obviously shorter than GAGR for its simplicity.

6. Conclusion

This paper presents a GA-based evolutionary approach with gene recombination about crew-scheduling problem for bus drivers. Unlike existing methods with a potential shift set for crew scheduling, a new piece-based chromosome method associating with decomposition and recombination is proposed. Shifts in parent chromosomes are divided into several piece shifts and offspring shifts are combined into new shifts during the genetic iterative process. The global solution is achieved by the repeated removal and replenished procedures. In view of the effect of different parameters, hundreds of experiments of ten bus lines from Beijing Bus Group were tested to select the optimal parameters. What is more, the gene recombination methods with or without deadhead were also employed into the initial and final populations. First, two simple examples are applied on both GAGR and CPLEX. The performance shows little difference between the correct answers and the obtained results. Compared with the current manual results, the results

```
Begin:
  //m = 1: the number of iterations
  //i = 1: the total shift number
  Initialize the new shift set;
  Set parameter values;
  While (primary shift number ≠ new shift number)
   if m ≠ 1
      primary shift number = new shift number;
   end
   Presort the working time set in descending order;
   Select satisfied shifts and put them into new shift set;
   While (i < remaining shift set length)
     Set two joinable shifts into one shift;
     Put the new generated shift into new shift set;
      Delete the two primary shifts from the remaining shift set;
   end
   Put shifts in remaining shift set into new shift set;
   m++;
 end
End
Primary shifts from the remaining shift set;
   end
   Put shifts in remaining shift set into new shift set;
   m++;
```

Algorithm 1

Table 7: The minimum results compared with ant colony algorithm.

Line	GAGR			ACA		
	Shifts	RPD (%)	Running time(s)	Shifts	RPD (%)	Running time(s)
#306	26	−18.18	255.33	56	69.7	529.44
#695	61	−4.6875	803.917	84	31.25	253.726
#348	22	−29.03	593.833	33	6.45	306.294
#26	78	−0.025	837.2	87	8.75	370.4697
#43	48	−2.04	1331.2	66	34.69	392.52
#467	35	−2.78	739.7	40	11.11	419.7317
#28	42	−19.2308	1303.633	59	13.46	546.9484
#34	43	−10.42	1512.633	57	9.62	518.0322
#345	119	−15.6028	5992.683	184	30.496	1126.1212
#322	101	−27.3381	10942.5	153	18.71	2968.0859

obtained by our approach are better on both solution quality and speed, no more than 11 minutes, even for the schedule with large pieces such as 1108 for #322 in our test. Almost all the results improve in the tested ten lines especially the one achieving 36 reducing shifts, a very large number, and all the negative RPD values reflecting the solution quality. From the results, it is also found that the effectiveness becomes more significant while there are more pieces in a schedule. In addition, the comparison with ant colony algorithm also shows the validity of GAGR for a large gap in results under the identical conditions. For the future work, we will add more constraints stipulated in Bus Group to approximate the current need and apply this approach to more similar set covering problems.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This paper is sponsored by National Basic Research Program of China (no. 2012CB725403-5).

References

[1] A. Wren, "Heuristics ancient and modern: transport scheduling through the ages," *Journal of Heuristics*, vol. 4, no. 1, pp. 87–100, 1998.

[2] A. Ceder, *Public Transit Planning and Operation: Theory, Modeling and Practice*, Elsevier, Butterworth-Heinemann, 2007.

[3] A. Ceder, B. Fjornes, and H. I. Stern, "OPTIBUS: a scheduling package," in *Computer-Aided Transit Scheduling: Proceedings of the Fourth International Workshop on Computer-Aided Scheduling of Public Transport*, vol. 308 of *Lecture Note in Economics Mathematical Systems*, pp. 212–225, Springer, Berlin, Germany, 1988.

[4] A. Wren and J. M. Rousseau, "Bus driver scheduling—an overview," in *Computer-Aided Transit Scheduling: Proceedings of the Sixth International Workshop on Computer-Aided Scheduling of Public Transport*, vol. 430 of *Lecture Notes in Economics and Mathematical Systems*, pp. 173–187, Springer, Berlin, Germany, 1995.

[5] D. Huisman, *Integrated and Dynamic Vehicle and Crew Scheduling*, Erasmus School of Economics (ESE), Rotterdam, The Netherlands, 2004.

[6] Y. Shen and J. Xia, "Integrated bus transit scheduling for the Beijing bus group based on a unified mode of operation," *International Transactions in Operational Research*, vol. 16, no. 2, pp. 227–242, 2009.

[7] S. Fores, L. Proll, and A. Wren, "An improved ILP system for driver scheduling," in *Computer-Aided Transit Scheduling: Proceedings of the Seventh International Workshop on Computer-Aided Scheduling of Public Transport*, vol. 471 of *Lecture Notes in Economics and Mathematical Systems*, pp. 43–61, 1999.

[8] S. Fores, L. Proll, and A. Wren, "TRACS II: a hybrid IP/heuristic driver scheduling system for public transport," *Journal of the Operational Research Society*, vol. 53, no. 10, pp. 1093–1100, 2002.

[9] Y. Shen, *Tabu search for bus and train driver scheduling with time windows [Ph.D. thesis]*, University of Leeds, 2001.

[10] S. Chen and Y. Shen, "An improved column generation algorithm for crew scheduling problems," *Journal of Information & Computational Science*, vol. 10, no. 1, pp. 175–183, 2013.

[11] A. S. K. Kwan, R. S. Kwan, M. E. Parker, and A. Wren, *Producing Train Driver Shifts by Computer*, Research Report Series, School of Computer Studies, University of Leeds, 1996.

[12] R. S. K. Kwan, A. S. K. Kwan, and A. S. K. Wren, "Evolutionary driver scheduling with relief chains," *Evolutionary Computation*, vol. 9, no. 4, pp. 445–460, 2001.

[13] J. Li and R. S. K. Kwan, "A fuzzy genetic algorithm for driver scheduling," *European Journal of Operational Research*, vol. 147, no. 2, pp. 334–344, 2003.

[14] Y. Shen and R. S. K. Kwan, "Tabu search for driver scheduling," in *Computer-Aided Transit Scheduling: Proceedings of the Seventh International Workshop on Computer-Aided Scheduling of Public Transport*, vol. 505, pp. 121–135, Springer, Berlin, Germany, 2001.

[15] Y. Shen and R. S. Kwan, *Tabu Search for Time Windowed Public Transport Driver Scheduling*, Research Report Series, School of Computer Studies, University of Leeds, Leeds, UK, 2002.

[16] P. Forsyth and A. Wren, "An ant system for bus driver scheduling," in *Proceedings of the 7th International Workshop on Computer-Aided Scheduling of Public Transport*, Boston, Mass, USA, 1997.

[17] K. Ghoseiri and F. Morshedsolouk, "ACS-TS: train scheduling using ant colony system," *Journal of Applied Mathematics and Decision Sciences*, vol. 2006, Article ID 95060, 28 pages, 2006.

[18] E. Mazloumi, M. Mesbah, A. Ceder, S. Moridpour, and G. Currie, "Efficient transit schedule design of timing points: a comparison of ant colony and genetic algorithms," *Transportation Research Part B: Methodological*, vol. 46, no. 1, pp. 217–234, 2012.

[19] M. F. Costa, N. C. Fiedler, and G. R. Mauri, "Clustering search and simulated annealing to solve the driver scheduling problem for timber transport," *Scientia Forestalis*, vol. 41, no. 99, pp. 299–305, 2013.

[20] R. S. K. Kwan, A. Wren, and A. S. K. Kwan, "Hybrid genetic algorithms for scheduling bus and train drivers," in *Proceedings of the Congress on Evolutionary Computation (CEC '00)*, vol. 1, pp. 285–292, July 2000.

[21] Y. Shen, K. Peng, K. Chen, and J. Li, "Evolutionary crew scheduling with adaptive chromosomes," *Transportation Research Part B: Methodological*, vol. 56, pp. 174–185, 2013.

[22] A. Wren and D. O. Wren, "A genetic algorithm for public transport driver scheduling," *Computers & Operations Research*, vol. 22, no. 1, pp. 101–110, 1995.

[23] J. Li and R. S. Kwan, "A self-adjusting algorithm for driver scheduling," *Journal of Heuristics*, vol. 11, no. 4, pp. 351–367, 2005.