Applied Soft Computing 36 (2015) 36–44

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A clonal selection algorithm for urban bus vehicle scheduling

Xinguo Shui [a], Xingquan Zuo [a,*], Cheng Chen [a], Alice E. Smith [b]

[a] School of Computer Science, Beijing University of Posts and Telecommunications, Beijing, China
[b] Industrial and System Engineering Department, Auburn University, AL, USA

ARTICLE INFO

Article history:
Received 25 March 2014
Received in revised form 23 May 2015
Accepted 2 July 2015
Available online 11 July 2015

Keywords:
Vehicle scheduling
Bus scheduling
Immune algorithm

ABSTRACT

The bus vehicle scheduling problem addresses the task of assigning vehicles to cover the trips in a timetable. In this paper, a clonal selection algorithm based vehicle scheduling approach is proposed to quickly generate satisfactory solutions for large-scale bus scheduling problems. Firstly, a set of vehicle blocks (consecutive trips by one bus) is generated based on the maximal wait time between any two adjacent trips. Then a subset of blocks is constructed by the clonal selection algorithm to produce an initial vehicle scheduling solution. Finally, two heuristics adjust the departure times of vehicles to further improve the solution. The proposed approach is evaluated using a real-world vehicle scheduling problem from the bus company of Nanjing, China. Experimental results show that the proposed approach can generate satisfactory scheduling solutions within 1 min.

1. Introduction

Public-transport operations consist of four phases in sequence, namely network design, timetabling, vehicle scheduling and crew scheduling. Each of these phases can be treated as an independent problem.

The vehicle scheduling problem of an urban public-transport system is to assign vehicles according to a given timetable, making departure times coincide with start times in the timetable, as well as minimizing some objectives, such as the number of vehicles used. Such scheduling is very significant for bus companies since good schedules can reduce operation costs and improve quality of service.

Generally speaking, there are two types of bus vehicle scheduling algorithms, i.e., exact algorithms and heuristic algorithms. Exact algorithms, such as mathematical programming, obtain an optimal schedule but the computation time grows unmanageable with the size of bus fleet. Freling et al. [1] used a linear programming model to describe the single depot vehicle scheduling problem and proposed an auction based algorithm to solve it. Ribeiro et al. [2] presented a column generation approach to solve the vehicle scheduling problem. Kliewer et al. [3] utilized a time–space network to model the multi-depot vehicle scheduling problem. Heuristic algorithms, such as dispatching rules or Lagrangian relaxation, can get approximate scheduling solutions within reasonable computation time. Pepin et al. [4] summarized five heuristic algorithms for the multi-depot vehicle scheduling problem, namely branch-and-cut method, Lagrangian heuristic, column generation heuristic, large neighborhood search heuristic and tabu search. These approaches were applied to real-world data and their advantages and disadvantages were discussed. Freling et al. [5] proposed a rule-based heuristic algorithm to solve vehicle scheduling problems with different vehicle types. Laurent et al. [6] combined an iterated local search algorithm with a neighborhood schema for the multiple depot vehicle scheduling problem. Vanitchakornpong et al. [7] proposed a bus fleet scheduling model with multi-depot and line change operations and developed a constrained local search method to solve this problem. Ceder [8] proposed a heuristic algorithm based on the deficit function theory for multiple vehicle-type scheduling problems. Hadjar et al. [9] used a branch and price approach to solve the multiple depot vehicle scheduling problem with time windows.

Some studies investigated the integration of vehicle scheduling and crew scheduling. Huisman et al. [10] combined column generation with Lagrangian relaxation to solve single depot integrated vehicle and crew scheduling. They also used a similar approach to solve multi-depot cases. Haase et al. [11] presented a set partitioning model for the crew scheduling problem. The model contains side constraints on the number of vehicles in order to generate feasible vehicle scheduling in polynomial time. de Groot et al. [12] split large vehicle and crew scheduling instances into smaller ones and solve them in integrated or sequential approaches respectively using a Lagrangian relaxation algorithm. Laurent et al. [13] used a greedy randomized adaptive search procedure (GRASP) to

* Corresponding author. Tel.: +86 10 62283406.
E-mail address: zuoxq@bupt.edu.cn (X. Zuo).

http://dx.doi.org/10.1016/j.asoc.2015.07.001
1568-4946/

solve vehicle and crew scheduling problems for the single depot case. Initial solutions were constructed using constraint programming techniques and then improved by a local search approach which embeds a neighborhood exploration mechanism. Rodrigues et al. [14] used conventional integer programming combined with a heuristic to solve vehicle and crew scheduling problems. Trip durations were stretched so as to provide a chance to adjust the trip departure times in the later phase to improve the final solution. Lin et al. [15] presented a multi-objective programming model of vehicle and crew scheduling problems, and used a branch and bound algorithm to solve it. Mesquitaa et al. [16] combined a multi-commodity model with a set partitioning/covering model to describe and solve the vehicle and crew scheduling problem.

An evolutionary algorithm (EA) is a kind of population based meta-heuristic. It starts from an initial population, and evolves the population by evolutionary operators until reaches a stopping criterion. Compared with traditional optimization algorithms, it can achieve global search capability and can effectively balance solution quality and computational time. Studies have shown that EA is very effective for NP-hard scheduling problems that cannot be solved by traditional algorithms within a reasonable timeframe [17–20]. Although EA has been used to solve vehicle routing problems [21,22], it has not been applied to solve realistic cases of urban bus vehicle scheduling.

Immune evolutionary algorithms emerged in recent years are inspired by the biological immune system, and have been applied successfully to a variety of optimization problems. Many researchers have shown that immune algorithms possess several attractive properties to avoid premature convergence and improve local search [23–25]. In our previous work [26], a culture immune algorithm was used to solve a small-scale bus vehicle scheduling problem. However, this approach was overly complex and only able to produce a feasible scheduling solution for a small-scale problem. We now propose an immune algorithm based vehicle scheduling approach to generate a practical and high quality vehicle scheduling scheme for a large-scale bus line scheduling problem in China.

In our approach, firstly, a set of vehicle blocks is generated based on the maximal wait time of bus drivers. A vehicle block is a set of consecutive trips by a single bus. Secondly, a subset of blocks is constructed by the clonal selection algorithm to produce an initial vehicle scheduling solution. Thirdly, two heuristics adjust the departure times of vehicles to further improve the solution.

The contributions of this paper includes: (1) a clonal selection algorithm based bus vehicle scheduling approach able to produce a good scheduling solution quickly (within 1 min); (2) a fitness function to evaluate the quality of a scheduling solution; (3) two heuristics to further improve the quality of the scheduling solution.

The remainder of this paper is organized as follows. The vehicle scheduling problem of urban bus lines is described in Section 2. Section 3 gives the proposed approach for this problem. In Section 4, results obtained by executing the approach on a real-world vehicle scheduling problem are given. Finally, concluding remarks are in Section 5.

2. Vehicle scheduling problem of urban bus lines

A bus line involving two control points is a typical case in practice, such that we consider a line with two control points (*CP*1 and *CP*2). In each of the two CPs, drivers can rest. A trip is the act of driving the vehicle between two CPs. Each trip has a duration and direction, starting from one CP to another. A vehicle block is a sequence of consecutive trips designated to one vehicle.

The initial departure time of a block means the departure time of its first trip. The timetable of a bus line consists of a large number of start times, each of which represents a trip. Given a timetable, the vehicle scheduling problem is to find a set of trips covering all start times (trips) in the timetable.

A constraint-based formulation of this vehicle scheduling problem is given as follows.

Let T be the set of trips in the timetable in chronological order. Let V be the set of vehicles and E be the set of drivers. For each driver $e \in E$, Sp_e represents the driver's maximum allowed spread time, which consists of the driver's working and resting time, and Wk_e is defined as the driver's maximum allowed working time.

Generally speaking, there are two types of blocks, namely long blocks and short blocks. A long block is completed by two drivers while a short block uses only one driver. The operation times of a short block and a long block can be denoted by Sp_e and $2Sp_e$, respectively. Let B^s and B^l be the set of short blocks and long blocks, respectively. For each block $b \in B^s \cup B^l$, $Tb(b)$ is defined as the set of trips that are covered by it.

For each driver $e \in E$, $Td(e)$ is defined as the set of trips that are assigned to the driver e. For each vehicle $v \in V$, $Tv(v)$ is defined as the set of trips that are assigned to vehicle v. For each trip $t \in T$, $st(t)$ and $et(t)$ represent the start time and end time of t, respectively. Let T_m be a set of time points and each element in it represents 1 min of a day.

In addition, we define

$$f(v,t) = \begin{cases} 1, & t \in Tv(v) \\ 0, & t \notin Tv(v) \end{cases}, \quad \forall v \in V, \quad t \in T$$

$$g_{CP1}(v,\delta) = \begin{cases} 1, & \text{vehicle } v \text{ is stationed at } CP1 \text{ at time } \delta \\ 0, & \text{vehicle } v \text{ is not stationed at } CP1 \text{ at time } \delta \end{cases} \quad \forall v \in V, \quad \forall \delta \in T_m$$

$$g_{CP2}(v,\delta) = \begin{cases} 1, & \text{vehicle } v \text{ is stationed at } CP2 \text{ at time } \delta \\ 0, & \text{vehicle } v \text{ is not stationed at } CP2 \text{ at time } \delta \end{cases} \quad \forall v \in V, \quad \forall \delta \in T_m$$

In order to assure service quality of the bus company, the following restrictions need to be satisfied:

(1) Each trip must be serviced by one vehicle.

$$\sum_{v \in V} f(v,t) \geq 1, \quad \forall t \in T$$

(2) The driver's spread time cannot exceed the allowed maximal spread time.

$$\left| st(t_1) - et(t_2) \right| < Sp_e, \quad \forall t_1 \in Td(e), \quad t_2 \in Td(e)$$

(3) The driver's working time cannot exceed the allowed maximal working time.

$$\sum_{t \in Td(e)} \left| st(t) - et(t) \right| < Wk_e, \quad \forall e \in E$$

(4) Trips that are assigned to each driver and vehicle must be feasible.

$$et(t_1) < st(t_2) \quad \text{or} \quad et(t_2) < st(t_1), \quad \forall t_1 \in Td(e), \quad t_2 \in Td(e)$$

$$et(t_1) < st(t_2) \quad \text{or} \quad et(t_2) < st(t_1), \quad \forall t_1 \in Tv(v), \quad t_2 \in Tv(v)$$

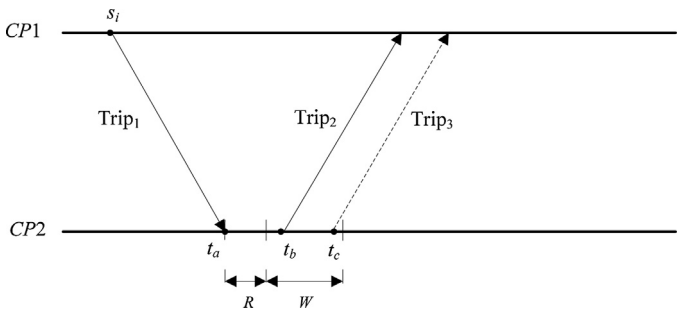

Fig. 1. Constructing a block set.

(5) For short vehicle blocks,

$$\left|et(t_1) - st(t_2)\right| < Sp_e, \quad \forall t_1 \in Tb(b), \quad t_2 \in Tb(b), \quad \forall b \in B^s$$

For long vehicle blocks,

$$\left|et(t_1) - st(t_2)\right| < 2Sp_e, \quad \forall t_1 \in Tb(b), \quad t_2 \in Tb(b), \quad \forall b \in B^l$$

The optimization objective is to generate a scheduling solution that is able to cover each time in the timetable by one trip in the solution and at the same time satisfies the above constraints.

3. Proposed scheduling approach

A block is a sequence of trips assigned to one vehicle and there is a time interval between any two adjacent trips in a block. Using a specific maximum allowed wait time W (determined by an algorithm parameter), a block set can be produced for each vehicle by making the wait time between any two adjacent trips less than W. The blocks of all vehicles compose the set of candidate blocks. Then we need to find a block subset (solution) from the set of candidate blocks to make each time in the timetable be covered by one trip in the subset, and at the same time make the solution satisfy all constraints. The clonal selection algorithm proposed in [28] relies on its mutation to achieve a good local search capability and is suitable to solve combinatorial problems. In the paper, we use the clonal selection algorithm to solve this problem.

3.1. Generating candidate blocks

Initial start times are defined as those start times in the timetable that can be chosen as the initial departure time of a block. Suppose the set of initial start times in the timetable are $S = \{s_1, s_2, \ldots, s_l\}$. For each start time $s_i \in S$, a block set $B_{s_i} = \left\{b_{s_i,1}, b_{s_i,2}, \ldots, b_{s_i,\left|B_{s_i}\right|}\right\}$ is constructed that contains all possible blocks starting from s_i, where $b_{s_i,j}$ means the jth block starting from time s_i. The set of candidate blocks is expressed by $B = \bigcup_{i=1}^{l} B_{s_i}$.

The process of constructing a block set for each initial start time is shown in Fig. 1. Let Trip1 be the first trip of a vehicle starting from $CP1$ at time s_i, and its arriving time to $CP2$ is time t_a. There are one or several start times in the interval $[t_a + R, t_a + R + W]$, where R is the driver's rest time. For each start time t in the interval, the next trip is chosen to continue this block. Each such possible continuation can produce a different vehicle block. For example, we can choose Trip1 and Trip2 as the first and second trips of a possible block, respectively, and we can also choose Trip1 and Trip3 for another possible block. The above process is then carried out continuously for each of Trip2 and Trip3 to select its next trip, such that a block set is constructed for each initial start time.

Fig. 2. Antibody encoding.

The W used to produce the block set is calculated by two start times t_b and t_c next to the arriving time t_a (see Fig. 1), namely $W = (t_c - t_b)\chi$. $\chi \geq 1$ is a given control factor to make sure there is at least one start time in the interval $[t_a + R, t_a + R + W]$ to continue a block.

During constructing a block, before the next trip is added, the total elapsed time T_w of this block is compared with the maximum allowed spread time Sp_e. If $T_w < Sp_e$ (for short blocks) or $T_w < 2Sp_e$ (for long blocks), the trip is added to this block; otherwise, the constructive process of the block is terminated.

3.2. Select a subset of candidate blocks

The number of blocks in block set B is usually huge. Choosing a block subset from those blocks is a complex combinatorial optimization problem.

(1) Representation of scheduling solution

Representation is a key issue of applying an evolutionary algorithm to solve a problem. We design an initial start time based encoding scheme for this problem. An antibody (solution) is expressed by an integer-valued vector, each gene of which corresponds to an initial start time. There are l initial start times in the set S, so the coding length is l.

The encoding of an antibody X is shown in Fig. 2. Gene x_i ($i = 1, 2, \ldots, l$) in X corresponds to the ith initial start time, s_i, in S. The value of each gene x_i is an integer in the range $\left[0, \left|B_{s_i}\right|\right]$. The expression $x_i = j$ means that the jth vehicle block $b_{s_i,j}$ in the set B_{s_i} is chosen. The expression $x_i = 0$ indicates that no block starting at time s_i is chosen. Therefore an antibody X represents a block subset B_X selected from B. Subset B_X contains $m_X = \sum_{i=1}^{l} z_{X,s_i}$ blocks, where $z_{X,s_i} = 1$ $\left(z_{X,s_i} = 0\right)$ if and only if a block belonging to B_{s_i} is (not) selected to construct B_X. Let B_X to be $\left\{b_{X,1}, b_{X,2}, \ldots, b_{X,m_X}\right\}$, where $b_{X,j}, j = 1, 2, \ldots, m_X$, denotes its jth vehicle block.

(2) Evaluation function

The following objective (evaluation) function is designed to evaluate an antibody (solution) X.

$$f(X) = \alpha \sum_{i=1}^{U} \left| \sum_{j=1}^{u_i} \left(o_{ij} - 1\right) \right| + \beta \sum_{i=1}^{n} y_{t_i}, \tag{1}$$

where n represents the number of start times in the timetable $T = \{t_1, t_2, \ldots, t_n\}$, and $y_{t_i} \in \{0, 1\}$. $y_{t_i} = 0$ means there is at least one trip in the solution X whose departure time is equal to t_i; otherwise, no such departure time exists in this solution. Start times in T are divided into several groups by a fixed small time interval, such that in each group the departure time of a trip in X can be adjusted to its adjacent start time in the timetable by the adjustment procedure (explained in Section 3.3). U denotes the number of groups; u_i represents the number of start times in the ith group; o_{ij} indicates the number of trips covering the jth start time in the ith group. Weights α and β are pre-specified positive integers.

The first term in Eq. (1) is used to evaluate the quality of a solution. If this term equals to zero, then the solution may cover each start time in T with one trip after the adjustment procedure. The second term is used to evaluate the feasibility of a solution (corresponding to constraint (1)). If this term is greater than zero, resulting in the increase of objective function value, then this solution tends to be deleted in the evolutionary process. Constraints (2)–(5) are satisfied in the process of creating the block set.

Antibodies in the population are sorted in ascending order according to their evaluation values. Then the sorted antibodies are divided into w groups, each of which contains the same number of antibodies. The affinity of an antibody in the ith group can be calculated by

$$A_i = h - i, \quad i \in \{1, 2, \ldots, w\} \tag{2}$$

where h is a given integer.

(3) Select candidate solutions using clonal selection algorithm

The clonal selection algorithm's steps are described as follows.

Step 1: Let g be the number of generations and $g = 0$. Initialize the population $P(g)$. The number of antibodies in $P(g)$ is N.

Step 2: Calculate the affinity of each antibody in $P(g)$ by Eq. (2), and sort all antibodies in descending order according to their affinities.

Step 3: Each antibody in $P(g)$ clones, and the number of clones for each antibody is proportional to its affinity. N clones are generated to compose the population $P_c(g)$.

Step 4: Each clone in $P_c(g)$ mutates. The mutation rate of each clone is inversely proportional to its affinity, i.e., a clone with a higher affinity will have a lower mutation rate. N mutated clones are produced to form population $P_m(g)$.

Step 5: n_c antibodies with the highest affinity are selected from $P(g)$ and $P_m(g)$ to compose the population $S(g)$.

Step 6: N–n_s randomly generated antibodies are added to $S(g)$. Let $P(g+1) \leftarrow S(g)$, $g \leftarrow g+1$. Return to Step 2 until the stopping criterion is satisfied.

Detailed operations are as follows.

(1) Population initialization

A good initial population can help an evolutionary algorithm quickly converge to a high quality solution. We generate the initial antibodies by three ways according to [26], i.e., (1) covering start times from beginning to end; (2) covering start times from end to beginning; (3) randomly covering start times. The number of blocks in each initial antibody is given as a random integer in the range $[m_{\min}, m_{\max}]$, where $m_{\min}$ and $m_{\max}$ are the lower bound and upper bound of the required number of blocks, respectively.

(2) Clone operation

All antibodies are sorted according to their affinities. Each antibody clones and the clone probability of the ith antibody Ab_i is calculated by

$$\frac{\text{Affinity}\,(Ab_i)}{\sum_{j=1}^{N}\text{Affinity}\,\left(Ab_j\right)}, \quad i \in \{1, 2, \ldots, N\} \tag{3}$$

where Affinity$(\cdot)$ is the affinity function. The roulette wheel method [27] is used to determine the number of clones for each antibody.

(3) Mutation operation

Each clone mutates and its mutation rate is inversely proportional to its affinity. An antibody with a higher affinity has a lower mutation rate. Let η and ρ be the highest and lowest mutation rate. The clones are divided into p groups according to their affinities, and the mutation rate of clones in the ith group is computed by

$$\rho + \frac{\eta - \rho}{p} \times i, \quad i \in \{1, 2, \ldots, p\} \tag{4}$$

If a gene x_i of a clone mutates, its value is replaced by a block number randomly chosen from the set B_{s_i} with probability of 0.7, and is set to zero (means no block starting at s_i is chosen) with probability of 0.3. The block number in a gene is replaced by another block number with greater probability (0.7) because it is possible to produce an improved solution. Setting the value of a gene to zero removes that block from the solution. This deletion may create a solution with a smaller number of blocks but often results in uncovered start times. Therefore, a small probability (0.3) is used for deletion.

3.3. The adjustment procedure

According to the objective function in Eq. (1), the clonal selection algorithm is used to produce a solution with a high affinity. To further improve the quality of this solution, an adjustment to try to make each start time in T be covered by exact one trip is made.

The adjustment procedure consists of two steps. The first step is to adjust the departure of trips to try to make each start time in the timetable be covered by one trip in the solution, and the second step is to avoid each start time in the timetable covered by more than one trip.

(1) Adjusting departure times of trips

Suppose there are n_1 start times in $CP1$ and n_2 start times in $CP2$. For a solution X, assume that the number of trips covering start times in $CP1$ is expressed by $C = \{c_1, c_2, \ldots, c_{n_1}\}$ and the number of trips covering start times in $CP2$ is expressed by $D = \{d_1, d_2, \ldots, d_{n_2}\}$. For each block $b_{X,j}(j = 1, 2, \ldots, m_{\boldsymbol{X}})$ in $B_X = \{b_{X,1}, b_{X,2}, \ldots, b_{X,m_X}\}$, suppose the relaxation times of its trips are denoted by $L_{X,j} = \{l_{X,j,1}, l_{X,j,2}, \ldots, l_{X,j,m_j^X}\}$. A trip's relaxation time in a block is defined as the time interval between its arrival time and the departure time of its next trip in the block. $r_{\min}$ is defined as the minimum rest time between two adjacent trips for a driver.

Firstly, the start times in $CP1$ are checked in chronological order, and then the start times in $CP2$ are checked. For each start time, such as the ith start time in $CP1$, if $c_i = 0$, then the forward adjustment and backward adjustment are executed sequentially to make the ith start time be covered.

Starting from $i-1$th start time and let $k = i-1$, the forward adjustment checks whether $c_k = 1$. If the answer is yes, we find the trip covering the kth start time and suppose that it is the qth trip in $b_{X,j}$. Then check whether its relaxation time $l_{X,j,q}$ is larger than the sum of $r_{\min}$ and the difference between the kth and $k+1$th start times. If the answer is yes, the adjustment of departure time of the trip from k to $k+1$ is feasible. Then for the $k-1$th start time, the above adjustment process is repeated until $c_k \neq 1$ or the relaxation time of the trip covering the kth start time is smaller than the sum of $r_{\min}$ and the interval between the kth and $k+1$th start times.

If $c_k > 1$ and the relaxation time of the trip covering the kth time is larger than the sum of $r_{\min}$ and the interval between the kth and $k+1$th start times, then we adjust the start time one by one according to the sequence $(k, k+1, \ldots, i)$ to make the ith start time in $CP1$ be covered by a trip. For example,

suppose the qth trip in $b_{X,j}$ covering the kth time needs to be adjusted, then that trip is adjusted to make it cover the $k+1$th start time instead of the kth start time and the relaxation times $l_{X,j,q}$ and $l_{X,j,(q-1)}$ are also updated. Suppose the interval between the kth and $k+1$th start times is a_k, then let $l_{X,j,q} = l_{X,j,q} - a_k$ and $l_{X,j,(q-1)} = l_{X,j,(q-1)} + a_k$. This process continues until the ith start time is reached, and then let $c_i = 1$ and $c_k = c_k - 1$. If there are more than one trip that cover the kth start time, then the trip with the maximal relaxation time is chosen to be adjusted.

If $c_k = 0$ or the relaxation time of the trip covering kth start time is smaller than the sum of $r_{\min}$ and the interval between the kth and $k+1$th start times, then the forward adjustment is terminated and the backward one starts. The backward adjustment is similar to the forward one, except that each start time after the ith time is checked in reverse chronological order.

(2) Remove overlapping trips

This step is used to remove overlapping trips in the solution. For each block $b_{X,j}(j = 1,2,\ldots, m_X)$ of a solution X, suppose start times covered by $b_{X,j}$ are $T_{X,j} = \left\{t_{X,j,1}, t_{X,j,2}, \ldots, t_{X,j,r_X^j}\right\}$, where r_X^j is the number of start times covered by $b_{X,j}$. Starting from the first start time in $T_{X,j}$, let $i = 1$ and check $t_{X,j,i}$ and $t_{X,j,(i+1)}$. If i is an odd number, it means start times $t_{X,j,i}$ and $t_{X,j,(i+1)}$ are in $CP1$ and $CP2$, respectively; otherwise they are in $CP2$ and $CP1$, respectively. If both $t_{X,j,i}$ and $t_{X,j,(i+1)}$ are covered by more than one trip, then only one randomly chosen trip is kept and all other trips are removed from each of the two start times. Set $i = i + 1$, and repeat the above process until all times in the block $b_{X,j}$ are considered. The above procedure is performed for each block in the solution X to remove overlapping trips.

4. Experiments and analysis

The proposed scheduling approach is applied to the real-world vehicle scheduling problem of a bus line of Nanjing city in China. The average travel time a trip takes is about 32 min for off-peak hours and 35 min for rush hours and normal hours. For this bus line, the rush hours include the periods of 6:30–9:00 and 17:00–20:00 while 4:30–6:30 and 21:00–01:05 are off-peak hours, and 9:00–17:00 and 20:00–21:00 are normal hours. The maximal spread time of a driver is 8 h (each driver needs to perform several trips in 1 day, and there is a rest time between any two adjacent trips). The operation information of this bus line is given in Table 1.

According to the actual operation management, the first trip of a vehicle should start before 11 o'clock. The average time interval of two adjacent start times in the timetable is about 3–4 min, such that the 120th start time in $CP1$ is just at 11:01. The first 120 start times in $CP1$ are considered as the initial start times, so the length of an antibody is 120. According to Section 3.2, the ith gene of an antibody represents a block in the set of B_{s_i}.

Table 1
Information of bus line 1 in Nanjing city.

Parameters	Values
Number of CPs	2
Distance between two CPs	10.7 km
Stations	17
Start time of the first trip	4:30 AM
Start time of the last trip	01:05 AM (next day)
Number of start times in $CP1$	397
Number of start times in $CP2$	397

The range of the number of blocks in an initial antibody is chosen as [42,46] by estimating the actual number of blocks required to produce a schedule solution. Each block corresponds to one vehicle, which is assigned to one (for a short block) or two drivers (for a long block). In the phase of generating the set of candidate blocks, the rest time R is set to 10 min. The control factor χ is set to 1.2 to guarantee that there is at least one start time to continue a block. The parameters of clonal selection algorithm are chosen as $N = 300$, $n_s = 240$, $\rho = 0.02$, $\eta = 0.08$, $p = 4$, $h = 17$, $w = 8$ after brief experimentation. The algorithm performance is not sensitive to its parameters except ρ and η, which are further analyzed in Section 4.2. In the adjustment phase, the minimum rest time $r_{\min}$ is set to 2 min according to practice.

4.1. Experimental results

The approach is programmed with Microsoft visual C++ and runs in Windows 7 on a 2.4 GHz Intel i5 PC with 2.3GB of RAM. 10 independent runs of the approach are done for the problem, and in each run the clonal selection algorithm is executed for 1000 generations to allow for convergence.

The scheduling solutions obtained in the 10 runs are given in Table 2. Each run is able to obtain a feasible solution able to cover start times in the timetable. The generations and time used by the clonal selection algorithm to obtain those solutions are also presented in Table 2. The time consumed for generating the set of all candidate blocks is <0.1 s, and the time for adjusting the solution is <0.01 s. We can see that our approach can obtain a scheduling solution quickly and the totally used time is <1 min.

Evolutionary curves of the clonal selection algorithm for the 10 runs are shown in Fig. 3. The horizontal axis represents the number of generations, and the vertical axis is the minimal evaluation (objective) function value for each generation.

The average numbers of vehicles and drivers in the scheduling solutions obtained in the 10 runs are 45.9 and 76.8, respectively.

Table 2
Experimental results of 10 runs of our approach.

Runs	Vehicles	Drivers	Long blocks	Short blocks	Generations	Time (s)
1	45	77	32	13	205	43
2	47	77	30	17	230	49
3	46	79	33	13	292	44
4	46	76	30	16	234	44
5	45	76	31	14	131	44
6	46	77	31	15	297	45
7	47	78	31	16	234	45
8	46	76	30	16	428	44
9	46	76	30	16	209	44
10	45	76	31	14	225	44
Average	45.9	76.8	30.9	15	248.5	44.6

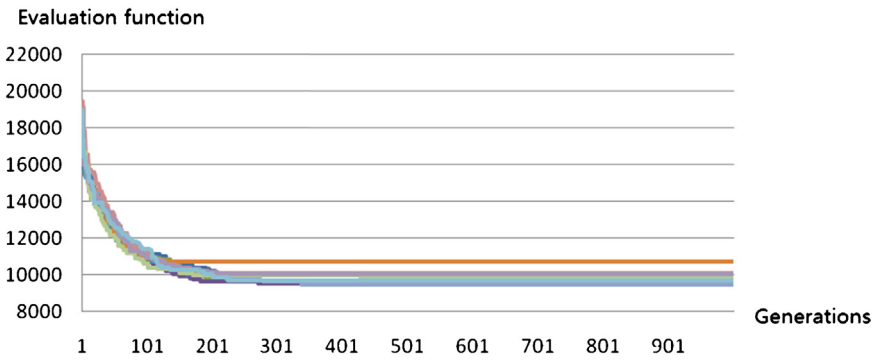

Fig. 3. Evolutionary curves of the clonal selection algorithm.

The 6th run produces a solution with 46 vehicles and 77 drivers, which are close to the average values, such that we choose this solution to draw its Gantt chart, as illustrated in Fig. 4. The horizontal axis displays the time in 1 day, and the vertical axis represents the vehicles in this solution. Each vehicle corresponds to a block containing a sequence of trips, each of which is represented by a powder blue rectangle. Each cyan rectangle in this figure represents the mealtime for drivers. A shorter white space between two adjacent trips means a rest time, while a longer one in a block represents the vehicle breaks for several trips before continuing to execute the ensuing trips.

To verify the effectiveness of our approach, it is compared with both the actually used experience-based scheduling scheme (which is devised using the experience of the human scheduler) and an iterated local search algorithm (ILS). We adapt the ILS in [6] to solve the problem. Two types of perturbation mechanisms are sequentially

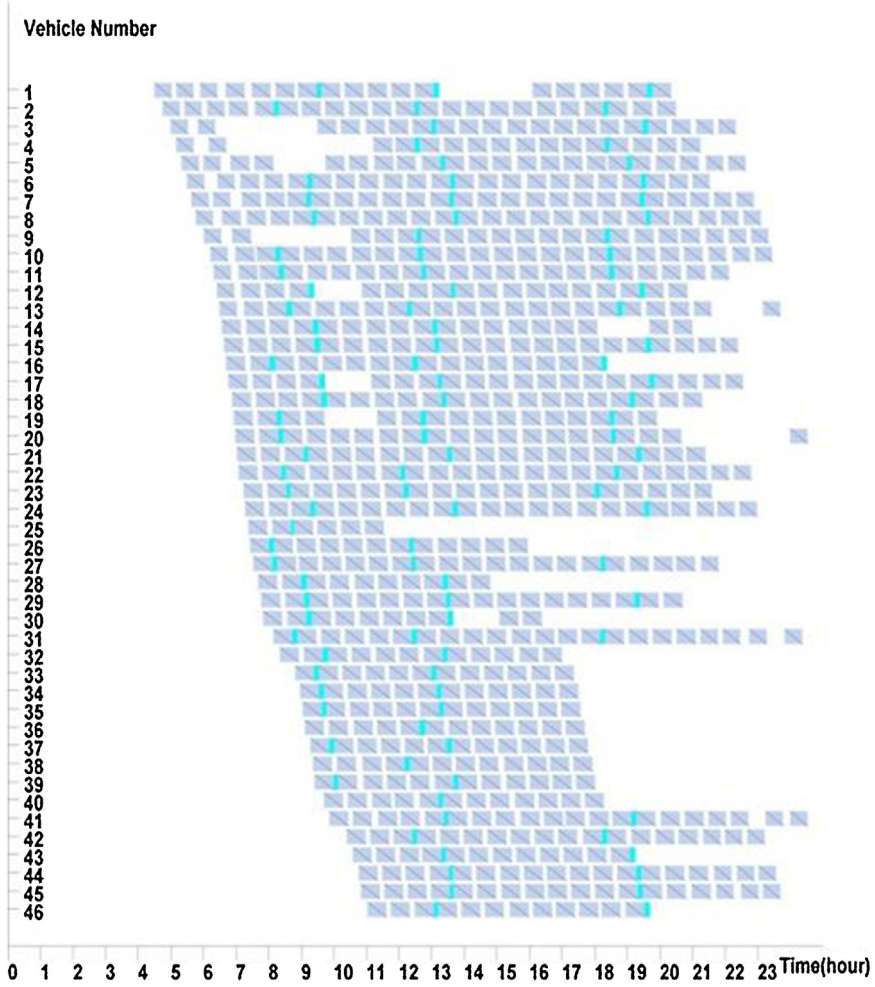

Fig. 4. A vehicle schedule solution produced by our approach.

Fig. 5. A vehicle schedule solution produced by ILS.

applied to the current solution. The first type is to randomly select a block of a vehicle and reassign it to another vehicle, and the second type is to swap two randomly chosen trips from two vehicles. Then, a neighbor is created by moving a randomly chosen block from one vehicle to another one. If the best neighbor among the neighborhood is better than the current solution, the current solution is updated as the best neighbor. The ILS stops when a maximum number of iterations are reached.

The actually used experience-based scheme is a reasonable scheduling solution designed by a dispatching engineer. Producing an experience-based scheme is time consuming and highly dependent on the expertise of the scheduler. The ILS is a heuristic able to quickly generate a scheduling solution. The parameters of ILS are given by: neighbor size is 2000 and the maximum number of iterations is 1000 to make it adequately converge. 10 independent runs of ILS are done for the problem, and solutions obtained in the 10 runs are given in Table 3. We choose the best solution in the 10 runs (that is, the solution obtained in the 5th run) to draw its Gantt chart, as shown in Fig. 5.

Table 4 gives solutions obtained by our approach, the experience-based one and the ILS approach. Since the cost of a bus line depends on the number of vehicles and drivers used, we use these two performance indices to evaluate the scheduling approaches. The number of vehicles used is equal to the sum of the numbers of long blocks and short blocks in the scheduling solution. In Table 4, "best" means the best solution found in 10 runs, namely the solution with a smaller number of vehicles and drivers, and "worst" represents the worst solution in the 10 runs. "mean" is the average values of those indices over the 10 runs.

From Table 4, we can see that solutions obtained by our approach and the actually used experience-based one have the similar number of vehicles, but solutions obtained by our approach

Table 3
Experimental results of 10 runs of ILS.

Runs	Vehicles	Drivers	Long blocks	Short blocks	Time (s)
1	51	83	32	19	1
2	52	89	37	15	1
3	53	84	31	22	1
4	52	85	33	19	1
5	49	82	33	16	1
6	52	89	37	15	1
7	49	82	33	16	1
8	52	90	38	14	1
9	49	89	40	9	1
10	51	85	34	17	1
Average	51	85.8	34.8	16.2	1

Table 4
Comparison of the three approaches.

	Our approach			ILS			Experience solution
	Best	Worst	Mean	Best	Worst	Mean	
Rest time	8.66	8.87	8.77	8.78	9.5	9.12	8.99
Vehicles	45	47	45.9	49	52	51	46
Long blocks	31	31	30.9	33	37	34.8	34
Short blocks	14	16	15	16	15	16.2	12
Drivers	76	78	76.8	82	89	85.8	80

Table 5
Influence of mutation rate on algorithm performance.

$[\rho,\eta]$	Average evaluation values	Average vehicles	Average drivers
[0.01,0.04]	10,291	46.5	78.6
[0.02,0.08]	9611	46	76.1
[0.03,0.12]	9848	46	77.2
[0.04,0.16]	9904	46.3	77.6
[0.05,0.20]	10,038	46.9	77.9

require a smaller number of drivers than the experience-based approach. This is because the solutions found by our approach have more short blocks and less long blocks than the manual approach. Since about 50% of the cost comes from wages and related items, a smaller number of drivers can produce sizable savings.

From Table 3, we can see that ILS is able to find a solution quickly, taking only one second. However, its solutions are worse than our approach, and even worse than the experience-based one. This may be because the initial solution has an influence on the performances of ILS, and it is not easy to assign a good enough initial solution for this problem.

In practice, an appropriate rest time of a driver in a CP after finishing one trip should be 8–10 min. In our approach, the rest time is give as a parameter in the phase of generating candidate blocks, but it may be changed in the following adjustment phase. The average drivers' rest time in the solutions produced by our approach and other two approaches are also given in Table 4. We can see that the rest time of our approach is similar to those of other two approaches. This means that our approach is able to produce a better scheduling solution without decreasing the drivers' rest time.

4.2. Algorithm parameter analysis

To observe the influence of mutation rate of the clonal selection algorithm on performance, we consider several different ranges of mutation rate. For each range, 10 runs are conducted, and the results are shown in Table 5. We can see that the approach cannot get good scheduling solutions with either a very large or a very small mutation rate. The reasoning is intuitive – with too much mutation, a random search is carried out and with too little mutation, not enough of the search space is explored. The best results are with relatively modest mutation rates (the second row in Table 5).

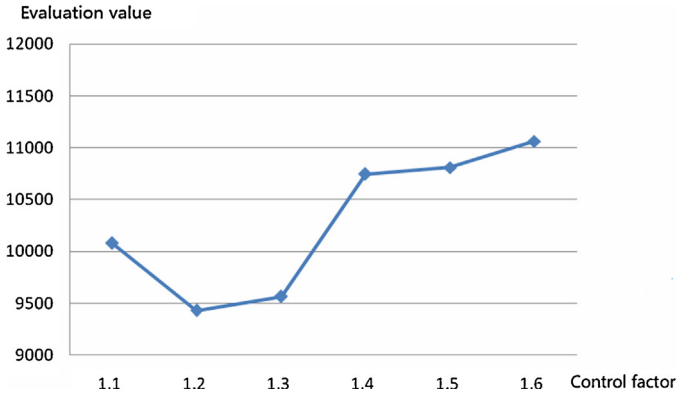

Fig. 6. Influence of control factor χ on algorithm performance.

The maximal wait time W, determined by the control factor χ, has influences on the number of candidate blocks. The factor is considered for six values, and for each value the algorithm runs 10 times. Table 6 shows the number of produced candidate blocks, used time, average evaluation function value, average number of vehicles and drivers over the 10 runs. It can be seen that with the increase of χ, the number of candidate blocks and used time increase sharply. χ also influences algorithm performance. From Table 6, we can see that different values of χ will result in different evaluation values and different number of vehicles and drivers in the final solution. This is also illustrated in Fig. 6, in which the horizontal axis represents the control factor χ and the vertical axis indicates the objective (evaluation) function value. We observe that if χ is too small, the number of candidate blocks is not enough to construct a satisfactory solution; if χ is too large, the number of candidate blocks is huge, such that the search space becomes very large and it is not easy to find a satisfactory solution.

Table 6
Influence of control factor χ on algorithm performance.

χ	Candidate blocks	Time (s)	Average evaluation values	Average vehicles	Average drivers
1.1	446	<0.1	10,080	45.7	77.4
1.2	534	<0.1	9432	46	76.1
1.3	774	<0.2	9566	46	76.3
1.4	4935	1	10,746	46.2	77.4
1.5	129,136	31	10,812	46.1	77.6
1.6	165,237	37	11,064	46.1	77.9

5. Conclusions

In this paper, we propose a bus vehicle scheduling approach. The approach consists of three steps. Firstly, a set of candidate vehicle blocks is constructed, which includes all feasible blocks that satisfy the constraints of the problem. Then a clonal selection algorithm based block selection approach is designed to choose the most suitable block subset from the candidate block set to produce a scheduling solution. Finally, two adjustment procedures are used to adjust the departure times of trips in the solution to further improve its quality.

The proposed approach is applied to a real-world vehicle scheduling problem. The experiments show that the approach is effective and can find satisfactory scheduling solutions quickly.

Acknowledgement

This work was supported in part by National Natural Science Foundation of China under Grant 61374204.

References

[1] R. Freling, A.P.M. Wagelmans, J.M. Pinto Paixão, Models and algorithms for single-depot vehicle scheduling, Transport. Sci. 35 (2) (2001) 165–180.
[2] C.C. Ribeiro, F. Soumis, A column generation approach to the multiple-depot vehicle scheduling problem, Oper. Res. 42 (1) (1994) 41–52.
[3] N. Kliewer, T. Mellouli, L. Suhl, A time–space network based exact optimization model for multi-depot bus scheduling, Eur. J. Oper. Res. 175 (2006) 1616–1627.
[4] A. Pepin, G. Desaulniers, A. Hertz, D. Huisman, Comparison of heuristic approaches for the multipledepot vehicle scheduling problem, Econometric Institute Report from Erasmus University Rotterdam, Econometric Institute, EI 2006-34.
[5] R. Freling, D. Huisman, A.P.M. Wagelmans, Applying an integrated approach to vehicle and crew scheduling in practice, *Computer-Aided Scheduling of Public Transport of Lecture Notes in Economics and Mathematical Systems*, 2001, 73–90.
[6] B. Laurent, H. Jin-Kao, Iterated local search for the multiple depot vehicle scheduling problem, Comput. Ind. Eng. 57 (2009) 277–286.
[7] K. Vanitchakornpong, N. Indra-Payoong, A. Sumalee, P. Raothanachonkun, Constrained local search method for bus fleet scheduling problem with multi-depot with line change, Appl. Evol. Comput. 4974 (2008) 679–688.
[8] A. Ceder, Optimal multi-vehicle type transit timetabling and vehicle scheduling, Procedia Soc. Behav. Sci. 20 (2011) 19–30.
[9] A. Hadjar, F. Soumis, Dynamic window reduction for the multiple depot vehicle scheduling problem with time windows, Comput. Oper. Res. 36 (2009) 2160–2172.
[10] D. Huisman, R. Freling, A.P.M. Wagelmans, Multiple-depot integrated vehicle and crew scheduling, Transport. Sci. 39 (4) (2005) 491–502.
[11] K. Haase, G. Desaulniers, J. Desrosiers, Simultaneous vehicle and crew scheduling in urban mass transit systems, Transport. Sci. 35 (3) (2001) 286–303.
[12] S.W. de Groot and D. Huisman, Vehicle and crew scheduling: Solving large real-world instances with an integrated approach, *Computer-aided System in Public Transport of Lecture Notes in Economics and Mathematical System*, 2008, 43–56.
[13] B. Laurent, J.K. Hao, Simultaneous vehicle and crew scheduling for extra urban transports, in: Proceedings of the 21st international conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems: New Frontiers in Applied Artificial Intelligence, 2008, pp. 466–475.
[14] M.M. Rodrigues, C.C. de Souza, A.V. Moura, Vehicle and crew scheduling for urban bus lines, Eur. J. Oper. Res. 170 (3) (2006) 844–862.
[15] Y. Lin, S. Pan, L. Jia, N. Zou, A Bi-level multi-objective programming model for bus crew and vehicle scheduling, in: The 8th World Congress on Intelligent Control and Automation, 2010, pp. 2328–2333.
[16] M. Mesquitaa, A. Paias, Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem, Comput. Oper. Res. 35 (2008) 1562–1575.
[17] S. Elloumi, P. Fortemps, A hybrid ant-based evolutionary algorithm applied to multi-mode resource-constrained project scheduling problem, Eur. J. Oper. Res. 205 (2010) 31–41.
[18] T. Hanne, S. Nickel, A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects, Eur. J. Oper. Res. 167 (2005) 663–678.
[19] P. Chitraa, R. Rajarama, P. Venkatesh, Application and comparison of hybrid evolutionary multiobjective optimization algorithms for solving task scheduling problem on heterogeneous systems, Appl. Soft Comput. 11 (2011) 2725–2734.
[20] I.T. Tanev, T. Uozumi, Y. Morotome, Hybrid evolutionary algorithm-based real-world flexible job shop scheduling problem: application service provider approach, Appl. Soft Comput. 5 (2004) 87–100.
[21] K.C. Tan, Y.H. Chew, L.H. Lee, A hybrid multi-objective evolutionary algorithm for solving truck and trailer vehicle routing problems, Eur. J. Oper. Res. 172 (2006) 855–885.
[22] N. Jozefowiez, F. Semet, E.G. Talbi, An evolutionary algorithm for the vehicle routing problem with route balancing, Eur. J. Oper. Res. 195 (2009) 761–769.
[23] G.C. Luh, C.H. Chueh, A multi-modal immune algorithm for the job-shop scheduling problem, Inf. Sci. 179 (2009) 1516–1532.
[24] X. Zuo, H. Mo, J. Wu, A robust scheduling method based on a multi-objective immune algorithm, Inf. Sci. 179 (2009) 3359–3369.
[25] M. Mobini, Z. Mobini, M. Rabbani, An artificial immune algorithm for the project scheduling problem under resource constraints, Appl. Soft Comput. 11 (2011) 1975–1982.
[26] X. Shui, X. Zuo, C. Chen, A cultural clonal selection algorithm based fast vehicle scheduling approach, IEEE Congress on Evolutionary Computation, Brisbane, Australia, 2012.
[27] D.E. Goldberg, Genetic algorithms in search optimization and machine learning, Addison Wesley, 1989.
[28] L.N. de Castro, F.J. Von Zuben, Learning and optimization using the clonal selection principle, IEEE Trans. Evol. Comput. 6 (3) (2002) 239–251.