Paper:

A Comparative Study on Evolutionary Algorithms for High-Speed Railway Train Timetable Rescheduling Problem

Shuxin Ding[*1,*2,†], **Tao Zhang**[*1,*2], **Rongsheng Wang**[*1,*2,*3], **Chunde Zhang**[*4], **Sai Lu**[*5,*6], **and Bin Xin**[*5,*6,†]

[*1]Signal and Communication Research Institute, China Academy of Railway Sciences Corporation Limited, Beijing 100081, China
[*2]The Center of National Railway Intelligent Transportation System Engineering and Technology, China Academy of Railway Sciences Corporation Limited, Beijing 100081, China
[*3]Postgraduate Department, China Academy of Railway Sciences, Beijing 100081, China
[*4]China Railway Beijing Group Corporation Limited, Beijing 100860, China
[*5]School of Automation, Beijing Institute of Technology, Beijing 100081, China
[*6]Key Laboratory of Intelligent Control and Decision of Complex Systems, Beijing Institute of Technology, Beijing 100081, China
E-mail: jackietindd@gmail.com, brucebin@bit.edu.cn
[†]Corresponding authors

In this paper, a high-speed railway train timetable rescheduling (TTR) problem is presented, addressing the adjustment of the train timetable with a complete blockage in the station according to the train operation constraints. Four competitive evolutionary algorithms (EAs), i.e., a dual-model estimation of distribution algorithm (DM-EDA), self-adaptive differential evolution (SaDE), comprehensive learning particle swarm optimizer (CLPSO), and Covariance Matrix Adaptation Evolution Strategy (CMA-ES), are adopted to solve the formulated problem. The individual of the EAs is represented as the permutation of trains' departure order in the disrupted station. The individual is decoded to a feasible schedule of trains using a rule-based method to allocate the running time in sections and dwell time in stations. For continuous space searching algorithms, the random key algorithm is used to obtain permutations from real vectors. Numerical experiments have been performed on 8 TTR instances. Experimental results demonstrate the superiority of SaDE in solving TTR.

Keywords: high-speed railway, train rescheduling, disruptions, evolutionary algorithm, combinatorial optimization

1. Introduction

High-speed railway (HSR) plays an important role in medium-to-long distance transportation service in China. HSR is operating according to the prescribed timetable. However, HSR may face inevitable emergencies, e.g., infrastructure failure, train failure, natural disasters [19]. Train operations may be disturbed/disrupted with delays. Therefore, train timetable rescheduling (TTR) is required for trains to recover to their regular operation.

A variety of studies have been analyzed for the TTR problem, which is proven to be NP-hard [1, 3]. In most studies, a mixed-integer linear programming (MILP) model is adopted, and the CPLEX solver is used to obtain solutions. A MILP model was proposed in [18] to deal with the real-time rescheduling of the timetable in case of a complete blockage in a railway segment. However, when the scale of the problem is getting larger, using the CPLEX solver will cost much time, which may exceed the time limit.

Metaheuristics are usually used for solving NP-hard problems [2]. Near-optimal solutions are obtained with limited time. Genetic algorithm-based particle swarm optimization had been used to reschedule the HSR timetable under primary delays [14]. Meng *et al.* [10] considered train rescheduling with track assignment and proposed an artificial bee colony algorithm to solve the problem. Departure time and arrival time of trains are used for solution representation. However, this may obtain constraint violated solutions during the process. Some related works considered other approaches to determine the train timetable. Wang *et al.* [15] adjusted train departure sequences based on Monte Carlo tree search. Metaheuristics have also been used to deal with permutation-based combinatorial optimization [16,17]. Wang and Wang [16] proposed an effective estimation of distribution algorithm (EDA) to solve the multi-track train scheduling problem. However, these permutation-based metaheuristics have not been used for train timetable rescheduling.

We summarize three contributions in this paper. First, the high-speed railway train timetable rescheduling problem with a complete station blockage is proposed and modeled as a MILP problem. Second, an effective permutation encoding method is proposed for the TTR problem, and a rule-based decoding method is designed to obtain a new schedule. These encoding and decoding methods can manage the entire constraints and guarantee the feasibility of the solution. Finally, several evolutionary algorithms are used for solving TTR. Experimental results show that SaDE can efficiently solve most of the test instances compared with other algorithms.

The rest of this paper is organized as follows. The TTR problem is described in Section 2. Section 3 presents several evolutionary algorithms to solve TTR. The performances of the proposed algorithms are evaluated in Section 4. Finally, conclusions and future work are provided in Section 5.

2. Problem Formulation

Punctuality is an important factor in railway operations. However, in the case of disruption, the railway system may fall into disorder. Trains may not be able to arrive or depart at stations.

In this section, we introduce a MILP model to formulate the TTR problem. We need to determine the new arrival and departure time of the trains at stations in order to recover the railway operations.

There are six assumptions: (1) All trains should follow their original schedules before disruption happens. (2) No trains are canceled in the train timetable rescheduling problem. (3) A macroscopic model is presented without considering the signaling systems and the station capacity. (4) The disruption considered is a complete blockage in the first station. All affected trains should depart after the disruption ends. (5) There is only one disruption whose duration is a known value. (6) Train reordering is not allowed except for departure trains in the first station.

For clarity, the notations of the proposed model are shown in Table 1.

The TTR problem can be stated as the following mathematical formulation:

$$\min \sum_{i\in T}\sum_{s\in S} w_i(t^a_{i,s}-T^a_{i,s}+t^d_{i,s}-T^d_{i,s}) \tag{1}$$

s.t.

$$t^d_{i,s}-t^a_{i,s}\geq d_{i,s}\ \forall i\in T; s\in S \tag{2}$$

$$t^a_{i,s+1}-t^d_{i,s}\geq r^{min}_{i,(s,s+1)}+r^s_{i,(s,s+1)}y_{i,s}+r^e_{i,(s,s+1)}y_{i,s+1}$$
$$\forall i\in T; s\in S\backslash D(i) \tag{3}$$

$$t^d_{j,s}-t^d_{i,s}\geq h_{(s,s+1)}q_{i,j,(s,s+1)}-M(1-q_{i,j,(s,s+1)})$$
$$\forall i,j\in T; i\neq j; s\in S\backslash D(i) \tag{4}$$

$$t^a_{j,s+1}-t^a_{i,s+1}\geq h_{(s,s+1)}q_{i,j,(s,s+1)}-M(1-q_{i,j,(s,s+1)})$$
$$\forall i,j\in T; i\neq j; s\in S\backslash D(i) \tag{5}$$

$$q_{i,j,(s,s+1)}+q_{j,i,(s,s+1)}=1\ \forall i,j\in I; i\neq j; s\in S\backslash D(i) \tag{6}$$

$$t^a_{i,s}=T^a_{i,s}\ \forall i\in T; s\in S: T^a_{i,s}\leq H^s_{dis} \tag{7}$$

$$t^d_{i,s}=T^d_{i,s}\ \forall i\in T; s\in S: T^d_{i,s}\leq H^s_{dis} \tag{8}$$

$$t^a_{i,s^*}\geq H^s_{dis}+D_{dis}\ \forall i\in T: H^s_{dis}\leq T^a_{i,s^*}\leq H^s_{dis}+D_{dis} \tag{9}$$

$$t^a_{i,O(i)}=t^d_{i,O(i)}\ \forall i\in T \tag{10}$$

$$t^a_{i,s}\geq T^a_{i,s}\ \forall i\in T; s\in S \tag{11}$$

$$t^d_{i,s}\geq T^d_{i,s}\ \forall i\in T; s\in S \tag{12}$$

$$q_{i,j,(O(i),O(i)+1)}=q_{i,j,(s,s+1)}$$
$$\forall i,j\in T; i\neq j; s\in S\backslash\{O(i),D(i)\} \tag{13}$$

$$y_{i,s}\leq t^d_{i,s}-t^a_{i,s}\ \forall i\in T; s\in S\backslash\{O(i),D(i)\} \tag{14}$$

$$y_{i,s}\geq \frac{t^d_{i,s}-t^a_{i,s}}{M}\ \forall i\in T; s\in S\backslash\{O(i),D(i)\} \tag{15}$$

$$y_{i,s}\geq Y_{i,s}\ \forall i\in T; s\in S\backslash\{O(i),D(i)\} \tag{16}$$

$$y_{i,s}=Y_{i,s}\ \forall i\in T; s\in\{O(i),D(i)\} \tag{17}$$

$$t^a_{i,s},t^d_{i,s}\geq 0\ \forall i\in T; s\in S \tag{18}$$

$$q_{i,j,(s,s+1)}\in\{0,1\}\ \forall i,j\in T; i\neq j; s\in S\backslash D(i) \tag{19}$$

$$y_{i,s}\in\{0,1\}\ \forall i,j\in T; i\neq j; s\in S \tag{20}$$

where Eq. (1) is to minimize the total delay time, including the delay arrival and departure time of each train at all the stations. Eq. (2) is the minimum dwelling time constraint. Eq. (3) is the minimum running time constraint. Eqs. (4) and (5) are the headway constraints for departure headway and arrival headway, respectively. Eq. (6) is the traverse order constraint of two trains in a section, which means that either train i traverses on section$(s,s+1)$ before train j or later than train j. Eqs. (7) and (8) guarantee the arrival and departure times for the unaffected trains are equal to the original timetable, respectively. Eq. (9) guarantees that no trains are allowed to arrive at stations during the disruption. Eq. (10) means the arrival time and departure time are the same for the origin station. Eqs. (11) and (12) are the timetable constraints that restrict trains are not allowed to arrive and depart from stations before the original arrival and departure time, respectively. Eq. (13) guarantees that the actual traversing orders of all trains are equal to the traversing orders in their first section. Eqs. (14) to (17) are the train stop indicator constraints. Eqs. (18) to (20) restrict the decision variables to be real numbers and binary numbers.

3. Evolutionary Algorithms for TTR

Since the TTR problem is NP-hard, there is no polynomial-time algorithm to obtain the exact solution. In this section, several evolutionary algorithms (EAs) are presented for solving TTR. First, encoding and decoding are introduced to transform the original MILP problem into a permutation-based combinatorial optimization problem. Then, several EAs are provided for solving the problem.

3.1. Encoding and Decoding

For TTR, most studies use the real-coded encoding scheme. The arrival and departure times are used as the solution. However, it is easy to obtain constraint violations during the evolutionary computation process. However, suppose the traversing order of trains in each section is determined. In that case, we only need to figure out the arrival and departure times that satisfy the operation constraints, e.g., dwelling time, running time, headway constraints, etc. In this section, we propose a permutation-based encoding method for solving TTR. The integer number in the solution determines the rescheduling order of the trains. For example, a solution $\mathbf{p}=(1,2,4,3,5)$ represents the order of 5 trains, where train 4 is scheduled first before train 3. The order for the other trains remains the same. Since before disruption happens, trains follow their original schedules, the set of affected trains

Table 1. Summary of notations.

Symbol	Description
	Indices
i, j	the index of train, $i, j \in T$
s	the index of station, $s \in S$
$(s, s+1)$	the index of section, which is between stations s and $s+1$, $(s, s+1) \in K$
s^*	the index of the disrupted station, $s^* \in S$
$O(i), D(i)$	the index of origin station and destination station of train i, respectively
	Parameters
T	the set of trains
S	the set of stations
K	the set of sections
$T^a_{i,s}$	the arrival time of train i at station s in the orignal schedule
$T^d_{i,s}$	the departure time of train i at station s in the orignal schedule
$d_{i,s}$	the minimum dwell time at station s for train i
$Y_{i,s}$	the train stop indicator in the orignal schedule, 1 if train i stops at station s; 0 otherwise
$r^{min}_{i,(s,s+1)}$	the minimum running time at section $(s, s+1)$ for train i
$r^s_{i,(s,s+1)}$	the additional time caused by starting for train i in section $(s, s+1)$
$r^e_{i,(s,s+1)}$	the additional time caused by stopping for train i in section $(s, s+1)$
$h_{(s,s+1)}$	the minimal headway between two consecutive trains of the same direction on section $(s, s+1)$
w_i	the weight value for train i
M	a large positive number
H^s_{dis}	the start time of the disruption
D_{dis}	the duration of the disruption
	Decision variables
$t^a_{i,s}$	the actual arrival time of train i at station s
$t^d_{i,s}$	the actual departure time of train i at station s
$q_{i,j,(s,s+1)}$	the actual traversing order, 1 if train i traverses on section $(s, s+1)$ before train j; 0 otherwise
$y_{i,s}$	the actual train stop indicator, 1 if train i stops at station s; 0 otherwise

T_{dis} can be determined if the arrival time in the first station is after H^s_{dis}. Therefore, $|T_{dis}|$ is the dimension of the permutation-based optimization problem.

We obtain the actual arrival time and departure time through the decoding procedure shown in **Algorithm 1** for a permutation encoded solution $\mathbf{p} = (p_1, p_2, \ldots, p_{|T|})$. Besides, the feasibility of the solution decoded from the permutation can be guaranteed since the constraint handling technique is implied during encoding and decoding. It follows a rule that trains may arrive and depart at stations once they are allowed as soon as possible.

Remark 1: In **Algorithm 1**, constraints for TTR are satisfied. In line 22, the condition will be met when an addition stop may be added in station s. It may add to the total running time for section $(s-1, s)$ because of additional time caused by stopping. Therefore, the arrival time should be updated. If the arrival time is larger than the departure time, adding stop is canceled, and the arrival time is set to the departure time.

3.2. EAs for TTR

3.2.1. A Dual-Model Estimation of Distribution Algorithm

EDA was first introduced in 1996 [11]. It estimates the overall distribution of the parent solutions and updates a probabilistic model with the superior individuals. New solutions are sampled from the model. Therefore, it is vital to select a suitable probability model of EDA. For a permutation-based optimization problem, node histogram model (NHM) and edge histogram model (EHM) are frequently used models [8]. Therefore, a dual-model estimation of distribution algorithm (DM-EDA) is proposed to solve TTR. Readers can refer to [8, 9] for more detail informations. The main components of DM-EDA are introduced as follows:

Selection Operator. Truncation selection is used to select the top N_{adv} solutions from parent solutions based on the objective values.

Modeling. Both NHM and EHM are used. The node histogram matrix and edge histogram matrix are initialized by equal probability. Both models are equally chosen during the process.

Sampling Operator. The sequence of each individual is obtained by the roulette wheel.

Restart Strategy. The determinants of the matrices are 0 after initialization. The determinant of the histogram matrix is approaching 1 during the process. This matrix is reinitialized if its determinant is less than $1 - \varepsilon$.

3.2.2. Self-adaptive Differential Evolution

Differential evolution (DE) is first proposed by Storn and Price [13]. There are three operations, including mutation, crossover, and selection. Different trial vector generation strategies can be selected, as well as three control parameters: crossover rate, scaling factor, and population size. Self-adaptive differential evolution (SaDE)

Algorithm 1 Decoding Procedure

Input: The original timetable information; The disruption information; The set of affected trains T_{dis}; Scheduling order of the trains $\mathbf{p} = [p_i]_{1\times|T|}$

Output: The actual arrival time $t^a_{i,s}$ and departure time $t^d_{i,s}$

for $i = 1$ to $|T| - |T_{dis}|$ **do**
  for $s = O(i)$ to $D(i)$ **do**
    $t^a_{i,s} = T^a_{i,s}$; $t^d_{i,s} = T^d_{i,s}$;
  end for
end for
for $i = |T| - |T_{dis}| + 1$ to $|T|$ **do**
  if $i = |T| - |T_{dis}| + 1$ **then**
    $t^a_{p_i,O(p_i)} = H^s_{dis} + D_{dis}$;
    $t^d_{p_i,O(p_i)} = t^a_{p_i,O(p_i)}$;
  else
    $t^a_{p_i,O(p_i)} = \max(t^d_{p_{i-1},O(p_{i-1})} + h_{(O(p_i),O(p_i)+1)}, T^a_{p_i,O(p_i)})$;
    $t^d_{p_i,O(p_i)} = \max(t^a_{p_i,O(p_i)} + d_{p_i,O(p_i)}, T^d_{p_i,O(p_i)})$;
  end if
  $y_{p_i,O(p_i)} = Y_{p_i,O(p_i)}$;
  for $s = O(i) + 1$ to $D(i)$ **do**
    $y_{p_i,s} = Y_{p_i,s}$;
    $t^a_{p_i,s} = \max(t^d_{p_i,s-1} + r^{min}_{p_i,(s-1,s)} + y_{p_i,s-1} r^s_{p_i,(s-1,s)} + y_{p_i,s} r^e_{p_i,(s-1,s)}, T^a_{p_i,s})$;
    $t^a_{p_i,s} = \max(t^a_{p_i,s}, t^a_{p_{i-1},s} + h_{(s-1,s)})$;
    $t^d_{p_i,s} = \max(t^a_{p_i,s} + d_{p_i,s}, T^d_{p_i,s})$;
    if $s < D(p_i)$ **then**
      $t^d_{p_i,s} = \max(t^d_{p_i,s}, t^d_{p_{i-1},s} + h_{(s,s+1)})$;
      if $\mathrm{sgn}(t^d_{p_i,s} - t^a_{p_i,s}) > y_{p_i,s}$ **then**
        $t^a_{p_i,s} = \min(t^d_{p_i,s-1} + r^{min}_{p_i,(s-1,s)} + y_{p_i,s-1} r^s_{p_i,(s-1,s)} + r^e_{p_i,(s-1,s)}, t^d_{p_i,s})$;
        $y_{p_i,s} = \mathrm{sgn}(t^d_{p_i,s} - t^a_{p_i,s})$;
      end if
    end if
  end for
end for
return

uses a self-adaptive method to choose trial vector generation strategies and control parameter values [12]. Readers can refer to [12] for more detail informations.

3.2.3. Comprehensive Learning Particle Swarm Optimizer

Particle swarm optimization (PSO) is a popular evolutionary computation technique [5]. Each particle in the swarm updates its position using its own experience and other particles' experiences. Comprehensive learning particle swarm optimizer (CLPSO) is a variant of PSO [6]. Each dimension of a particle learns from the best corresponding dimension of the particle. Readers can refer to [6] for more detail informations.

3.2.4. Covariance Matrix Adaptation Evolution Strategy

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a state-of-the-art optimizer for single-objective continuous functions first introduced in [4]. CMA-ES also uses a probability model to obtain new solutions. It works by sampling solutions from a multivariate normal distribution. Readers can refer to [4] for more detail informations.

3.2.5. Random Key Algorithm

Since SaDE, CLPSO, and CMA-ES are algorithms designed to search in continuous space, the random key algorithm is applied to transform the real-valued vector to a permutation. Given a real vector (3.5, 2.4, 1.6, 0.5, 4.1), the permutation obtained is the ranking of the real vector, which is (4, 3, 2, 1, 5). The range for each element in the real vector is also the dimension of the vector.

4. Computational Experiments

This section presents the performance investigation of the proposed algorithms. At first, we present the test instances for TTR. Then, we solve the problem under different methods, including exact solutions by CPLEX. All experiments were carried out on a PC with an Intel Xeon Gold 5218 CPU 2.30GHz and 32 GB internal memory. Exact solutions for TTR problems were implemented in MATLAB R2019b using YALMIP as the modeling language, and CPLEX 12.10 with default parameter settings [7]. EAs for TTR problems were implemented in MATLAB R2019b.

4.1. Test Instances for TTR

Due to the lack of benchmark instances with disruptions for TTR in literature, we first develop the test instances. The Beijing–Tianjin intercity railway timetable from Beijing South to Tianjin is considered in this paper. There are altogether 6 stations and 5 sections. 40 trains downstream from 6:00 to 12:00 are considered for the railway timetable, which is shown in **Fig. 1**. The minimum dwell time for train stops at stations is set to 2 min and no dwell time for pass-through stations, the origin stations, and destination stations. The minimum running time of each section is shown in **Table 2**. The additional times caused by starting and stopping are set to 2 min and 3 min, respectively. The minimal headway is set to 4 min. The start time of the disruption H^s_{dis} is set to 6:40. s^* is set to 1, which means the disruption is in the first station. M is set to 1440 min.

We categorize the generation of w_i into the following two cases:

Case 1: The weight values w_i of trains are set to 1.

Case 2: The weight values w_i of trains are generated as uniformly distributed random integers in a range between 1 to 10.

To validate the performance of the EAs, we produce 8 test instances. The first four instances (No. $1 \sim 4$) are from Case 1, and the last four instances (No. $5 \sim 8$) are from Case 2. The settings of the two basic parameters T and D_{dis} are listed in **Table 3**. For instance with the number of trains T less than 40, e.g., instance No. 1, the first train is the same train starting from 6:00. We do not need to adjust the schedule of all 40 trains when the duration of the disruption is only 30 min. As a result, T for different

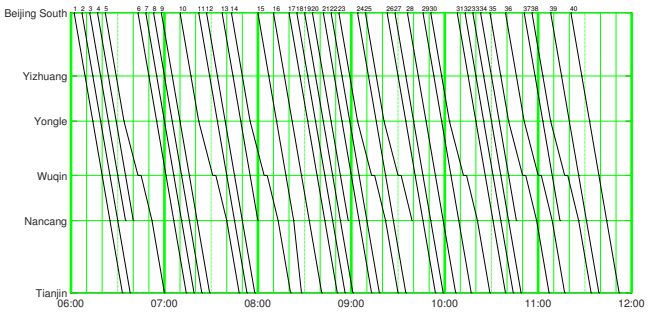

Fig. 1. Original timetable for Beijing–Tianjin intercity railway with 40 downstream trains within 6-h time horizon.

Table 2. The minimum running time in each section between two stations.

No.	Section	Time (min)
1	Beijing South - Yizhuang	5
2	Yizhuang - Yongle	5
3	Yongle - Wuqin	6
4	Wuqin - Nancang	5
5	Nancang - Tianjin	5

instances are generated according to the duration of the disruption D_{dis}.

4.2. Parameter Settings

For all the algorithms, the particle/population size is set to $10 \cdot D$, where D is the dimension of the searching space. For DM-EDA, the subpopulation size N_{adv} is set to D, which is 10% of the population size. The learning rate μ_n and μ_e are both set to 0.2. The predefined threshold ε is set to 0.01. For CLPSO, the acceleration constant c is set to 1.49445. The inertia weight w is selected linear decreasing from 0.9 to 0.4. Termination criterion and number of independent runs: Each algorithm is terminated when $10000 \cdot D$ fitness evaluation is reached (i.e., $MaxFEs = 10000 \cdot D$). The independent runs for each algorithm on each instance are set to 20.

The parameter settings of the four algorithms are kept the same as the original papers. Besides, for the CPLEX solver, the termination time is set to 3600s.

4.3. Results and Analysis

This section provides the experimental results of the four algorithms and CPLEX in solving TTR. **Table 4** shows the result of 20 independent runs of each algorithm with mean values and standard deviations. For CPLEX, it only runs once.

4.3.1. The performance of EAs

It can be drawn from **Table 4** that SaDE outperforms other methods. In five instances (No. 1, 2, 4, 5, and 6), the results of SaDE equal that of CPLEX. Moreover, for instances No. 3 and 7, the results of SaDE are only slightly larger (0.07% and 0.01%) than that of CPLEX (within one hour). In instance No. 8, the result of SaDE is better than that of CPLEX (within one hour).

Table 3. Setting of the two basic parameters for the test instances.

No.	$\|T\|$	D_{dis} (min)	No.	$\|T\|$	D_{dis} (min)
1, 5	15	30	2, 6	20	50
3, 7	30	70	4, 8	40	90

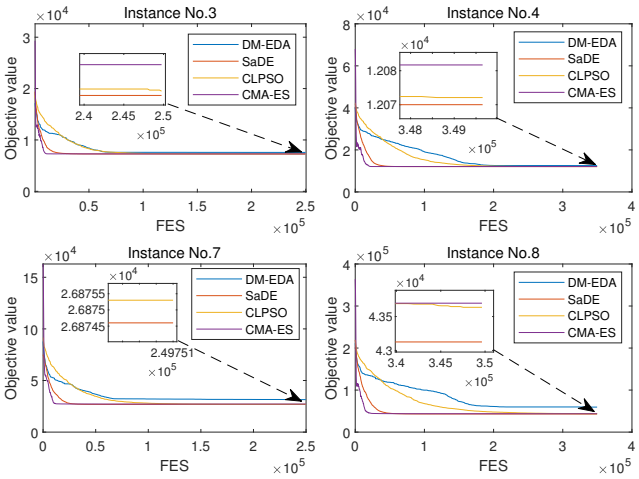

Fig. 2. Convergence curves of the proposed DM-EDA, SaDE, CLPSO, and CMA-ES for several test instances.

In instances No. 1 and 2, all four EAs converge to the optimal value. It is because the size of the instance is small, and the algorithms can cover nearly all feasible solutions. For instances No. 5 and 6 with different train weights, only SaDE and CLPSO converge to the optimal value. CMA-ES converges to optimal in instance No. 5, whereas DM-EDA cannot converge to the optimal.

4.3.2. Convergence Analysis

Fig. 2 provides the converge curves of the four EAs in instances No. 3, 4, 7, and 8. The curves are zoomed in some areas for better visualization. The horizontal axis and the vertical axis represent the number of fitness evaluations and the mean of the objective function of 20 runs, respectively. It can be drawn from the figure that CMA-ES converges faster than other algorithms. SaDE converges second but provides better results.

4.3.3. Running Time Analysis

Table 5 shows the running time of the four EAs and CPLEX. It also shows the mean values and standard deviations of 20 independent running for EAs. The result shows that DM-EDA takes the longest time compared with the other EAs. It can be seen that all instances can be solved within one minute. However, the running time for CPLEX increases a lot with the increase of the problem size. For instance No. 7, the total running time is around 2862s, and for instances (No. 3, 4, and 8), the total running time is more than 3600s. This result implies the efficiency of the proposed framework with permutation-based encoding and the rule-based decoding methods.

5. Conclusion

The high-speed railway TTR problem is formulated as a MILP problem. Four EAs are designed to solve TTR.

Table 4. Results of the comparison between DM-EDA, SaDE, CLPSO, and CMA-ES.

No.	DM-EDA	SaDE	CLPSO	CMA-ES	CPLEX
1	**1628.0000 ± 0.0000**‡	**1628.0000 ± 0.0000**‡	**1628.0000 ± 0.0000**‡	**1628.0000 ± 0.0000**‡	1628.0000‡
2	**3874.0000 ± 0.0000**‡	**3874.0000 ± 0.0000**‡	**3874.0000 ± 0.0000**‡	**3874.0000 ± 0.0000**‡	3874.0000‡
3	7570.8000 ± 34.5522	**7272.8000 ± 7.5226**	7274.4000 ± 7.6116	7284.3000 ± 0.7327	7268.0000†
4	12539.2000 ± 55.0154	**12070.0000 ± 0.0000**‡	12072.1000 ± 3.3388	12081.7000 ± 13.2709	12070.0000†
5	6462.0000 ± 0.0000	**6126.0000 ± 0.0000**‡	**6126.0000 ± 0.0000**‡	**6126.0000 ± 0.0000**‡	6126.0000‡
6	15386.0000 ± 0.0000	**14810.0000 ± 0.0000**‡	**14810.0000 ± 0.0000**‡	15060.6000 ± 695.6067	14810.0000‡
7	31475.0500 ± 684.5033	**26874.6000 ± 8.0026**	26875.3000 ± 8.3168	27177.0000 ± 330.6453	26872.0000‡
8	59492.1000 ± 1055.7585	**43125.0000 ± 10.7508**	43636.0000 ± 157.0169	43697.0000 ± 599.0109	43128.0000†

† CPLEX stopped after running for one hour.

‡ Optimal value.

Table 5. Runtime performance of different algorithms (sec.).

No.	DM-EDA	SaDE	CLPSO	CMA-ES	CPLEX
1	5.7372 ± 0.4578	8.1800 ± 0.6489	3.7526 ± 0.2992	2.2386 ± 0.3614	10.3855
2	10.0875 ± 0.6872	11.9927 ± 0.6156	5.5700 ± 0.3859	3.0539 ± 0.2743	64.7492
3	24.8920 ± 0.9677	19.9135 ± 1.2381	11.2691 ± 1.8045	6.0701 ± 1.0042	–
4	47.5454 ± 1.8224	30.0148 ± 2.1255	17.3618 ± 0.6499	9.6739 ± 0.1859	–
5	5.1246 ± 0.5452	8.1143 ± 1.1712	3.7058 ± 0.6896	1.8761 ± 0.3053	10.5488
6	10.2779 ± 1.2930	12.3090 ± 2.0349	6.0593 ± 0.7121	2.7641 ± 0.1174	30.5911
7	24.8921 ± 0.8240	20.0972 ± 1.1450	11.3079 ± 1.3316	6.2739 ± 1.1048	2861.8612
8	49.8737 ± 3.1044	31.1891 ± 2.9282	17.4095 ± 0.8151	10.4551 ± 1.6275	–

– CPLEX cannot find optimal value after running for one hour.

A novel encoding and decoding method are specially designed for TTR, transferring the original problem to an unconstrained one. This avoids a large amount of ineffective search in the solution space. After being tested in 8 test instances, SaDE outperforms other algorithms and shows its efficiency compared with CPLEX. The results can be obtained within one minute which is suitable for real-time rescheduling. In the future, we will consider situations with more types of trains (e.g., trains with different prefixes including G, C, D) and consider reordering in other stations based on the feature of the timetable. Meanwhile, considering the uncertainties in the dynamic environment will make the model more practical.

Acknowledgements

This work was supported in part by the National Key R&D Program of China (2018YFB1308000), in part by the National Natural Science Foundation of China under Grants U1934220, 62088101, 61822304, 61673058, and in part by the Center of National Railway Intelligent Transportation System Engineering and Technology (Contract No. RITS2019KF03), China Academy of Railway Sciences Corporation Limited.

References:

[1] V. Cacchiani, D. Huisman, M. Kidd, L. Kroon, P. Toth, L. Veelenturf, and J. Wagenaar, "An overview of recovery models and algorithms for real-time railway rescheduling", Transp. Res. Part B Methodol., 63:15–37, 2014.

[2] S. Ding, C. Chen, Q. Zhang, B. Xin, and P. M. Pardalos, "Metaheuristics for resource deployment under uncertainty in complex systems", CRC Press, Boca Raton FL, USA, 1 edition, 2021.

[3] W. Fang, S. Yang, and X. Yao, "A survey on problem models and solution approaches to rescheduling in railway networks", IEEE Trans. Intell. Transp. Syst., 16(6):2997–3016, 2015.

[4] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation", In Proc. IEEE Int. Conf. Evol. Comput., pp. 312–317. IEEE, 1996.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization", In Proc. ICNN'95 - Int. Conf. Neural Networks, volume 4, pp. 1942–1948. IEEE, 1995.

[6] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions", IEEE Trans. Evol. Comput., 10(3):281–295, 2006.

[7] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB", In Proc. CACSD Conf., Taipei, Taiwan, 2004.

[8] S. Lu and B. Xin, "Tabu-model-based estimation of distribution algorithm framework for permutation optimization problems", In 9th Int. Symp. Comput. Intell. Ind. Appl., 2020.

[9] S. Lu, B. Xin, L. Dou, and L. Wang, "A comparative study on evolutionary algorithms for the agent routing problem in multi-point dynamic task", Int. J. Autom. Control, 14(5-6):571–592, 2020.

[10] X. Meng, Y. Wang, W. Xiang, and L. Jia, "An integrated model for train rescheduling and station track assignment", IET Intell. Transp. Syst., 15(1):17–30, 2021.

[11] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters", In Int. Conf. Parallel Probl. Solving from Nat., pp. 178–187. Springer, 1996.

[12] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization", IEEE Trans. Evol. Comput., 13(2):398–417, 2008.

[13] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces", J. Glob. Optim., 11(4):341–359, 1997.

[14] M. Wang, L. Wang, X. Xu, Y. Qin, and L. Qin, "Genetic algorithm-based particle swarm optimization approach to reschedule high-speed railway timetables: A case study in China", J. Adv. Transp., 2019, 2019.

[15] R. Wang, M. Zhou, Y. Li, Q. Zhang, and H. Dong, "A timetable rescheduling approach for railway based on monte carlo tree search", In 2019 IEEE Intell. Transp. Syst. Conf., pp. 3738–3743. IEEE, 2019.

[16] S. Wang and L. Wang, "An effective estimation of distribution algorithm for multi-track train scheduling problem", In Int. Conf. Intell. Comput., pp. 697–708. Springer, 2014.

[17] Y. Wang, B. Xin, and J. Chen, "An adaptive memetic algorithm for the joint allocation of heterogeneous stochastic resources", IEEE Trans. Cybern., 2021, in press.

[18] S. Zhan, L. G. Kroon, L. P. Veelenturf, and J. C. Wagenaar, "Real-time high-speed train rescheduling in case of a complete blockage", Transp. Res. Part B Methodol., 78:182–201, 2015.

[19] M. Zhou, H. Dong, X. Liu, H. Zhang, and F.-Y. Wang, "Integrated timetable rescheduling for multidispatching sections of high-speed railways During large-scale disruptions", IEEE Trans. Comput. Soc. Syst., 2021, in press.