Abstract—As a typical scheduling problem, the vehicle scheduling problem (VSP) plays a significant role in public transportation systems. VSP is difficult to solve since it classified as a high-dimensional combination optimization problem, which is well known as an NP-hard problem. Although existing studies on VSP usually assume that all factors in the problem are deterministic and known in advance, various uncertain factors are always present in practical applications, in particular uncertain processing time. In this paper, we consider the problem of VSP with uncertain processing time. In order to solve this problem, a hybrid cooperative co-evolution algorithm (hccEA) is proposed. First, we design two-phase encoding and decoding mechanisms with the aim to search a larger solution space and filter infeasible solutions for genetic algorithm (GA) and particle swarm optimization (PSO). Second, to overcome performance degradation due to high-dimensional variables, a modified PSO is embedded into the cooperative co-evolution framework, which is called ccPSO. Third, a self-adaptive mechanism for parameters of PSO is proposed to balance the uncertain factors. Then, GA and ccPSO work alternately in an iterative way. Finally, numerical experiments under an uncertain environment verify the superiority of the proposed hccEA based on comparisons with state-of-the-art algorithms.

Index Terms—Cooperative Co-evolution, Evolutionary Optimization, Vehicle Scheduling, Uncertainty.

I. INTRODUCTION

Intelligent Transportation Systems (ITS) and vehicles have recently attracted a great deal of attentions. The abundance of sensors and communication technologies has made a new possible design space for vehicular-based applications. As one of the leading scheduling problems, vehicle scheduling plays a significant role in public transportation systems [1], such as flight departure and arrival, interval times of two adjacent subways and school bus route planning. Given a set of vehicles and a set of tasks that consists of several operations, all operations must be processed on vehicles. An illustration of the vehicle scheduling problem (VSP) is shown in Fig. 1. The objective of VSP is to achieve minimal cost while taking the balance of workload and interval times into consideration.

A. Related work

VSP which was first proposed by Webber [2]. It has attracted the attention of researchers in the fields of operation research, applied mathematics, combinatorial mathematics and computer science [3]. Researchers have made great progress in theoretical research and experimental analysis. Actually, VSP can be divided into two categories: VSP under a single-vehicle environment and a multiple-vehicle environment.

For VSP under a single-vehicle environment (sVSP), several approaches can be found in the recent literature. Tarantilis et al. introduced a hybrid meta-heuristic methodology that combines the strategies of tabu search (TS) and guided local search (GLS) for a new practical problem version in a transportation logistics company [4]. Freling et al. proposed an integrated approach that combined two pieces to a line and implemented the round-trip-condition in the generation of problems for individual bus lines of the RET [5]. Laurent et al. and Mesquita et al. proposed a hybrid approach for simultaneous scheduling of drivers and vehicles, where the algorithm starts with a pre-processing phase and then uses a column generation scheme in the second period [6], [7]. The approaches mentioned above are applied for VSP in practical applications in ITS. However, VSP under a single-vehicle environment cannot satisfy the needs of users as the demands of real-world applications increase.

VSP under a multiple-vehicle environment (mVSP) has attracted much more attention. Haghani and Banhashemi pre-
presented a new approach for a multiple depot vehicle scheduling problem and multiple depot vehicle scheduling problem with route time constraints [8]. Laurent and Hao introduced an iterated local search algorithm for the multiple depot vehicle scheduling problem (MDVSP) [9]. Pepin et al. proposed a large neighborhood search heuristic algorithm combined with a tabu search approach for multiple vehicle scheduling problems [10]. Teng et al. proposed a novel approach for a vehicle scheduling model with multiple lines sharing vehicle resources [11].

In general, several areas may always exist uncertain factors such as processing time and due time in practical applications. Thus, uncertain VSP (uVSP) is an essential and urgent problem to solve. Recently, Huisman et al. presented a solution-based approach for the dynamic vehicle scheduling problem [12]. Hadiar and Soumis considered a comprehensive branch-and-price approach for multiple depot vehicle scheduling problems with time windows [13]. Wei et al. presented an improved particle swarm optimization by using an organic fusion with group search optimization for VSP with fuzzy travel times [14]. Shen et al. proposed a novel probabilistic approach for uVSP [15].

As a typical scheduling problem, VSP consists of two dependent sub-problems: task sequence and vehicle assignment [16] which can viewed as an NP-hard problem that is unable to find the optimal solution in polynomial time [17]. Evolutionary algorithms (EAs) have been widely applied for scheduling in recent years. For the scheduling problem under the determinate environment [18]. Zuo et al. presented a methodology to create a set of Pareto solutions for VSP [19]. Zheng et al. proposed a new quantum rotation angle updating strategy and the idea of adaptive genetic algorithm for VSP [20]. In addition to scheduling problems under a determinate environment, the literatures also address EAs for scheduling problems under an uncertain environment. Gu et al. proposed a novel competitive co-evolutionary quantum genetic algorithm for solving a stochastic scheduling problem aiming to minimize the expected value of makespan [21]. Zhang et al. proposed a hybrid approach importing GA into PSO for flexible scheduling with transportation constraints and bounded processing times [22].

B. Motivation and Contribution

As mentioned above, many approaches have been proposed to solve VSP. However, the encoding and decoding mechanisms of EAs cannot be adapted for uVSP directly. Moreover, the increase of variables dimension leads to performance degradation of existing approaches. Therefore, we propose a hybrid cooperative co-evolutionary algorithm (hcceEA) adapted GA and PSO as the basic algorithms to solve uVSP.

The main contributions of this paper are summarized as follows:

1) Two-phase encoding and decoding mechanisms are proposed for GA and PSO. Two phases represent operation sequence and machine assignment, respectively. Multi-stage representation and random key-based mechanisms are adopted for GA operations and PSO operations, respectively.

2) To overcome high-dimensional issues, a modified PSO is embedded in the cooperative co-evolution framework, which is called ccPSO [23]. Each particle in the population of PSO is divided into several sub-particles through the random grouping technique. Each sub-particle only contains part of the dimensions of a corresponding particle. By the cooperative co-evolution framework, the proposed algorithm deals with a subproblem of lower dimensionality.

3) For balancing uncertain factors, a self-adaptive mechanism for the parameters of PSO is proposed. The parameters are adapted according to the performance of the proposed approach during the process of optimization.

This paper is organized as follows. In Section II, we present the formulation and model of uVSP. Our proposed algorithm is introduced in Section III. Section IV presents the numerical experiments, and Section V presents the final conclusion.

II. PROBLEM FORMULATION

In order to model uVSP, the probability distribution of the processing time is assumed to be known a priori. The realized outcome of a random processing time of an operation is only known at completion of processing. In this paper, we formulate a mixed integer programming model to transmute the processing times in terms of the stochastic variable. The uVSP can be formulated as an extended version of a precedence-constrained scheduling problem. Each operation $O_{ik}$ is carried out under uncertain random disturbance with a pre-given expected value $E[p_{ikj}]$, where $p_{ikj}$ is the processing time of $O_{ik}$ of the task $T_i$ on the vehicle $V_j$. The distribution of the variance can be predicted from the experimental data, e.g., Gaussian distribution, uniform distribution and exponential distribution. Therefore, the uncertainty can be constructed through the concept scenario $\xi$, which corresponds to an assignment of reasonable processing time, i.e., $\xi_{ikj}$.

The stochastic expected value model of uVSP may be formulated as follows: The makespan is the maximum completion time of vehicles, and the objectives are the expected value of makespan $C_{max}$ considering with the expected total workload $W$ and the expected total interval time $In$.

Indices:

\begin{align*}
  i & \quad \text{task ID} \quad (i = 1, 2, \ldots, n) \\
  j & \quad \text{vehicle ID} \quad (j = 1, 2, \ldots, m) \\
  k & \quad \text{operation ID} \quad (k = 1, 2, \ldots, n_t)
\end{align*}

Parameters:

\begin{align*}
  n & \quad \text{a number of tasks} \\
  m & \quad \text{a number of vehicles} \\
  n_t & \quad \text{a number of operations of task } T_i \\
  O_{ik} & \quad k - \text{th operation of task } T_i. \\
  \xi_{ikj} & \quad \text{an uncertain processing time of } O_{ik} \text{ on } V_j
\end{align*}

Decision variables:

\begin{align*}
  x_{ikj} & \quad 0 \text{ or } 1, \text{operation } O_{ik} \text{ assigned on vehicle } V_j \\
  \xi_{ikj}^S, \xi_{ikj}^L & \quad \geq 0, \text{start time and end time of operation } O_{ik} \text{ on } V_j
\end{align*}
The uVSP can be formulated as follows:

\[ \min_{C_{\text{max}}} E[C_{\text{max}}] = \max_{i} \{ \max_{k} \{ \max_{j} E[\xi t_{ikj}^T] \} \} \]  
(1)

where \( \xi t_{ikj}^T = \xi t_{ikj} + \xi p_{ikj} \)

\[ \min E[W] = \sum_{j=1}^{m} \{ E[\sum_{i=1}^{n} x_{ikj} \cdot \xi p_{ikj}] \} \]  
(2)

\[ \min E[|n|] = E[C_{\text{max}}] - E[W] \]  
(3)

\[ \sum_{j=1}^{m} x_{ikj} \leq 1, \forall i, k \]  
(4)

\[ \xi t_{ikj}^S - \xi t_{ikj}^T \geq 0, \forall i, (k > k') \]  
(5)

\[ \{ \sum_{i=1}^{n} x_{ikj} \cdot \xi t_{ikj}^S \leq \sum_{i=1}^{n} x_{ikj} \cdot \xi t_{ikj}^T \} \wedge \]  
\[ \{ \sum_{i=1}^{n} x_{ikj} \cdot \xi t_{ikj}^S \leq \sum_{i=1}^{n} x_{ikj} \cdot \xi t_{ikj}^T \}, \forall j,k \]  
(6)

\[ x_{ikj} \in \{0, 1\}, \forall i, j, k \]  
(7)

\[ \xi t_{ikj}^S, \xi t_{ikj}^T \geq 0, \forall i, k, j \]  
(8)

where Eq. (1), Eq. (2) and Eq. (3) are the objective functions. Eq. (4) shows each task is only processed once. Eq. (5) ensures the task precedence constraint, i.e., the successive operation has to be started after the completion of its precedent operation of the same task. Eq. (6) ensures each vehicle only process one operation at the same time and the process cannot be interrupted. Eq. (7) and Eq. (8) represent the nonnegative restrictions.

### III. Proposed Algorithm

The hybrid cooperative co-evolutionary evolutionary algorithm (hccEA) aims to find a best solution for uVSP, and it focuses on the complex constraints and characteristics of VSP. The hccEA chooses both GA and PSO as its basic algorithms with the alternate strategy, and two-phase encoding and decoding mechanisms are proposed for uVSP. After the initialization stage, hccEA executes the GA process first. When the GA termination criteria is satisfied, hccEA goes into the ccPSO process. We refer to the two processes mentioned above as alternate process. If ccPSO termination criteria is also satisfied, hccEA goes to the next generation and repeats the alternate process until the final termination criteria are satisfied. A flowchart for hccEA is shown in Fig. 2. All details regarding hccEA are introduced separately as follow.

#### A. Representation

Encoding a schedule into an individual is a critical issue and an essential part of EAs; they should be considered because of the complex constraints of VSP. The performance of EAs can be improved with the help of an efficient encoding mechanism. In this paper, we propose two encoding mechanisms for GA and PSO, respectively.

1) **GA Encoding**: a multi-stage representation are adopted for task sequence and vehicle assignment, respectively. An example is shown in Fig. 3.

For a task sequence string in one individual, the total length is equal to all operations of all tasks \((\sum n_i)\). It consists of the integers belonging to \([1, n]\), and \(n\) stands for the number of total tasks. The total number of each integer is equal to the number of operations of this task. In other words, each task number \(i\) \((i = 1, \ldots, n)\) appears \(n_i\) times \((n_i\) is the total operations of task \(i\)). By scanning the task sequence string from left to right, the \(k\)th appearance of task number \(i\) indicates the \(k\)th operation of task \(i\). Obviously, any permutation of task sequence string can be decoded to a feasible solution. This feature can filter out a large number of infeasible solutions and improve the effectiveness of the algorithm. For example, by scanning from left to right for the task sequence string in Fig. 3, we get the task sequence:

\[ \{O_{11}, O_{31}, O_{12}, O_{21}, O_{13}, O_{32}, O_{22}, O_{13}\} \]

For a vehicle assignment string in one individual, the total length is also equal to \((\sum n_i)\) as well. It can be viewed as \(n\) parts, and each part consists of \(n_i\) integers among \([0, |V_{ij}|]\), where \(V_{ij}\) stands for the set of available vehicles for \(O_{ij}\) and \(|\cdot|\) stands for the size of the set. Each
The decoding procedure of both GA and PSO are described here. In order to filter out infeasible solutions, PSO decodes the real number to an integer number and uses the same decoding mechanism as GA. The vehicle assignment string can be arranged as \(\{(v_2, 5), (v_3, 4), (v_3, 3), (v_2, 5), (v_4, 7), (v_3, 9), (v_2, 1)\}\). Each individual consists of both the task sequence string and vehicle assignment string; i.e., a solution is formed by decoding both the task sequence string and vehicle assignment string.

1) **PSO Encoding**: hccEA adopts random key-based representation. An example is shown in Fig. 4. hccEA randomly establishes the initial individuals by real number according to the given lower and upper bounds. It should be noted that during the evolution process, it is necessary to check the validity of variables in individuals.

In order to filter out infeasible solutions, PSO decodes individuals by a network-based decoding mechanism for task sequence as shown in Fig. 5. The decoding process of the individual shown in Fig. 4 starts from node \(S\) and puts the nodes directly linked to \(S\) into available node set \(aval = \{O_{11}, O_{21}, O_{31}\}\). The node with the maximum value in \(\{O_{11}, O_{21}, O_{31}\}\) (1.7 < 2.6 < 3.0), i.e., \(O_{31}\), is put into ordered node set \(seq = \{O_{31}\}\) and is deleted from \(aval\). Then, the node linked to \(O_{31}\) is put into \(aval = \{O_{11}, O_{21}, O_{32}\}\). Repeat the above ordering process until \(aval\) is null and the size of \(seq\) equals the total operations of all tasks. The final sequence is: \(\{O_{31}, O_{21}, O_{32}, O_{33}, O_{23}, O_{11}, O_{12}, O_{13}\}\).

For vehicle assignment, PSO first rounds the real number to an integer number and uses the same decoding mechanism as GA. The vehicle assignment can be arranged as \(\{(v_2, 5), (v_3, 4), (v_2, 5), (v_3, 3), (v_4, 7), (v_3, 9), (v_2, 1)\}\). The decoding procedure of both GA and PSO are described in Algorithm 1. It should be noted that there are two stages in hccEA and GA optimization begins after population initialization, thus, a special decoding process is required. The aim of this special decoding process is to decode the real number-based individual to an integer-based individual for GA operations. Similarly, it is also necessary to decode the integer-based individual to real number-based individual for the ccPSO operations.

**Algorithm 1** The decoding procedure.

**Input:** A individual and Problem data;

**Output:** A solution \(S\) of VSP;

1: Determine the operation schedule based on task sequence string;
2: Assign vehicle for each operation based on vehicle assignment string;
3: Combine operation schedule string and operation assignment string to form a solution;
4: return \(S\);

B. **GA Operations**

In this subsection, the GA operations are described in detail.

1) **Crossover and Mutation**: In this paper, we adopt two kinds of crossover operators for the task sequence string and one-cut crossover operator for vehicle assignment. The first crossover operation is precedence operation crossover (POX), and the second one is job-based crossover (JXB); both crossover operations can obtain feasible solutions after performing the crossover, and combining them can balance the local search and global search. Two parent individuals are denoted as \(P_1\) and \(P_2\), and two child individuals are denoted as \(C_1\) and \(C_2\). The pseudocode is shown in Algorithm 2.

The second crossover operator JOX is similar to POX, and the pseudocode is given in Algorithm 3. An example of POX
Algorithm 2 The procedure of POX.

**Input:** \( P_1 \) and \( P_2 \);
**Output:** \( C_1 \) and \( C_2 \);

1. Randomly divide the tasks \( T = \{T_1, T_2, ..., T_N\} \) into two nonempty subsets, \( TSet_1 \) and \( TSet_2 \);
2. Copy the elements of the tasks in \( TSet_1 \) from \( P_1 \) to \( C_1 \) at the same index and record down the indices of them as \( indices_1 \);
3. Copy the elements of the tasks in \( TSet_2 \) from \( P_2 \) to \( C_1 \) in sequence without the index in \( indices_1 \) and record down the indices of the elements of the tasks in \( TSet_1 \), as \( indices_2 \);
4. Copy the elements of the tasks in \( TSet_1 \) from \( P_2 \) to \( C_2 \) according to \( indices_2 \);
5. Copy the elements of the tasks in \( TSet_2 \) from \( P_1 \) to \( C_2 \) in sequence;
6. **return** \( C_1, C_2 \);

![Algorithm 2 Diagram](image-url)

Fig. 7. An illustration of JOX.

and JOX is shown in Fig. 6 and Fig. 7. hccEA adopts two mutation operators. The first is swapping mutation for a task schedule string, and the second is one-bit mutation for a vehicle assignment string. The detailed process is described in Algorithm 4.

2) Selection: Selecting individuals according to ranking fitness is widely used in GA. In hccEA, two selection operators are used. One is ranking selection, and the other is tournament selection. Each individual evaluates the fitness value according to the fitness function and then, to normalize fitness values, makes the sum of all fitness values equal 1. Assuming the population size is \( popSize \), the probability of ranking is \( p_r \); hccEA first selects the top \( p_r \times popSize \) individuals into the next generation. Then, tournament selection begins to work. In tournament selection, two individuals are selected randomly, and the one with better fitness is selected into the next generation. Repeat the above process until the size of the next generation reaches \( popSize \). This scheme performs a trade-off between the exploration and exploitation of the population.

Algorithm 3 The procedure of JOX.

**Input:** \( P_1 \) and \( P_2 \);
**Output:** \( C_1 \) and \( C_2 \);

1. Divide the tasks \( T = \{T_1, T_2, ..., T_N\} \) randomly into two nonempty subsets, i.e., \( TSet_1 \) and \( TSet_2 \);
2. Copy the elements of the tasks in \( TSet_1 \) from \( P_1 \) to \( C_1 \) at the same index and record down the indices of them as \( indices_1 \);
3. Copy the elements of the tasks in \( TSet_2 \) from \( P_2 \) to \( C_1 \) in sequence without the index in \( indices_1 \) and record down the indices of the elements of the tasks in \( TSet_1 \), as \( indices_2 \);
4. Copy the elements of the tasks in \( TSet_2 \) from \( P_2 \) to \( C_2 \) according to \( indices_2 \);
5. Copy the elements of the tasks in \( TSet_1 \) from \( P_1 \) to \( C_2 \) in sequence;
6. **return** \( C_1, C_2 \);

Algorithm 4 The procedure of mutation.

**Input:** \( P; r \);
**Output:** \( C \);

1. Copy \( P_1 \) to generate \( C \);
2. Select two random indices \( index_1 \) and \( index_2 \) in the task schedule string in \( P \);
3. Swap the elements in the position with index \( index_1 \) and \( index_2 \) in \( O \);
4. Randomly select \( r \) indices as \( indices \) for each element with the index in \( indices \) to change the value of the selected position to the other vehicle in the vehicle set corresponding operation;
5. **return** \( C \);

Algorithm 5 The procedure of PSO.

C. ccPSO Operations

In practical applications, the scale of the problem always increases, thus, to improve the ability of the proposed algorithm for overcoming the increase in scale of the problem becomes challenging. The divide and conquer (D&C) is usually considered as an effective strategy for large scale problems. The core idea of D&C is that to decompose one problem into two or more sub-problems of the same or related type until sub-problems become simple enough to be solved directly. However, VSP, a high-dimension constraints optimization problem, could not be divided into several simple sub-problems directly. Fortunately, Xin Y. et al. proposed a cooperative co-evolution framework for combinational optimization problems and gave the mathematical proof of rationality [23]. We import the CC framework into PSO with a self-adaptive mechanism as the second part of the proposed hccEA. The pseudocode is given in Algorithm 5. where \( gen \) and \( isAdjust \) in Algorithm 5. \( gen \) is the total generation size of ccPSO, and \( isAdjust \) is used to decide whether the parameters within the update process of PSO need to be adjusted.

1) Grouping and Dynamic re-grouping: As shown in Fig. 8, there exist \( popSize \) individuals in the ccPSO population, and each individual consists of \( n \) variables. In our proposed algorithm, \( n \) is double the total operations of all tasks. At the
Algorithm 5 The procedure of ccPSO.

Input: Data sets; \( gen; is\text{Adjust}; \)

Output: The global optimal solution of ccPSO;

1: get \( sub_p(t) \) by random grouping, \( s = n/k; \)
2: \( g(t) \leftarrow 0 \)
3: while \( (g(t) < gen) \) do
4: if \((is\text{Adjust}) \) then
5: adjust the grouping status, \( s' = n/k'; \)
6: end if
7: for each group \( j \) do
8: for each individual \( sub\_ind(t) \) in group \( j \) do
9: if \( b(j,sub\_ind(t),p\text{best}(t)) \) then
10: replace \((j,p\text{best},sub\_ind(t))\); end if
11: end for
12: if \( b(j,p\text{best}(t),g\text{best}(t)) \) then
13: replace \((j,g\text{best},p\text{best}(t))\); end if
14: for each individual do
15: updateLBest \((l\text{best}(t))\); end for
16: for each \( sub_p(t) \) do
17: for each individual do
18: updatePSO \((position,velocity)\); end for
19: end for
20: Adjust parameters of PSO;
21: \( g(t) \leftarrow g(t + 1) \); end while
22: return \( g\text{best}(t) \)

initialization phase, ccPSO randomly groups all \( n \) variables into \( k \) sub-individuals; i.e., there exist \( s = n/k \) variables in each sub-individual. \( pop\text{Size} \) sub-individuals with the same grouping status form sub-populations, i.e., \( sub_p \) in Algorithm 5 (line:1). It should be noted that if \( n \) is not able to be divided by \( k \), the remain variables are put in the last sub-individual. Obviously, \( k \) is a key value for ccPSO because it has a significant impact on performance. Thus, it is necessary to vary \( k \) during the optimization process. Yang et al. gave a detailed study and proposed a scheme to probabilistically choose a group size from a set of potential group sizes [25]. It selects a new group size according to the probability based on the performance of each individual. Since it may cost much time, ccPSO adopts a simpler version. The group size set is given in advance, and the group size set is \( K = \{2, 5, 10, 50, 100\} \).

In the process of optimization, if there is no improvement in performance using current group size \( k \), we think \( k \) is tired. Then, ccPSO randomly selects a new group size \( k' \) in \( K \) and re-groups all variables randomly in the population according to \( k' \) (line:3-6). When random grouping is applied, the indices of all variables are recorded and randomly permuted. These permuted indices are then used to construct \( k' \) sub-individuals. This is achieved by simply taking out every \( s = n/k' \) variables from the left to right to form a new sub-individual.

2) Evaluation under CC framework: In Algorithm 5, function \( b(j,trial,target) \) returns a boolean value. If the performance of the individual formed by \( n \) variables consisted of \( target \) with its \( j \)th component replaced by \( trial \) is better than the performance of \( target \), \( b \) returns true; otherwise, it returns false. If \( b(j,sub\_ind(t),p\text{best}(t)) \) returns true, ccPSO generates a new individual consisting of \( p\text{best}(t) \) with its \( j \)th component replaced by \( sub\_ind(t) \) (line:7-12). After evaluations of all sub-individuals in the current sub-population, if \( b(j,p\text{best},g\text{best}) \) returns true, ccPSO generates a new individual consisting of \( g\text{best} \) with its \( j \)th component replaced by \( p\text{best}(t) \) (line:13-15). After evaluating all individuals, ccPSO begins to update the local best individual \((l\text{best}) \) in Algorithm 5), which is defined as the individual with the best performance among the individuals before it (the \((i-1)\)th), the individual after it (the \((i+1)\)th) is considered an individual itself (the \((i)\)th) (line:16-18).

3) Self-adaptive mechanism: The update strategy of classical PSO is shown as follows:

\[
v_i(k + 1) = \omega v_i(k) + c_1 rand_1 [p\text{best}(k) - x_i(k)] + c_2 rand_2 [g\text{best}(k) - x_i(k)], \tag{9}
\]

where \( \omega \) is the inertia weight, and it determines the impact on the velocity in the next generation of the global and local optimal solutions. \( c_1 \) and \( c_2 \) are constants, and we set \( c_1 = c_2 = 1.49 \) in this paper. \( rand_1 \) and \( rand_2 \) are random numbers generated among \([0, 1]\). \( v_i(k) \) represents the velocity of individuals, and \( x_i(k) \) represents the position of individuals in the \( k \)th generation. \( p\text{best}(k) \) and \( g\text{best}(k) \) represent the personal best and global best in the \( k \)th generation. In order to search a larger solution space, ccPSO makes an improvement to Eq. (9), and the new update strategy is shown as follows:

\[
v_i(k + 1) = \omega v_i(k) + c_1 rand_1 [l\text{best}(k) - x_i(k)] + c_2 rand_2 [g\text{best}(k) - x_i(k)], \tag{10}
\]

where \( l\text{best}(k) \) represents the neighborhood best in the \( k \)th generation. For the balance between personal search ability and local search ability, ccPSO combines two update strategies
In the next generation when generated by Eq. (9) and Eq. (10) are all offspring, the number of offspring successfully entering the generation is recorded in an array \( CR_m \), respectively, and the numbers of offspring discarded when generated by Eq. (9) and Eq. (10) because they determine the influence of the local best and global best. They are confirmed to the normal distribution with average value \( CR_m \) and standard deviation 0.1., i.e.,

\[
r_{CR_1} \text{ or } 2 = N_i(CR_m, 0, 1). \tag{13}
\]

\( CR_m \) is first set as 0.5. These \( CR \) values for all individuals remain the same for 5 generations, and then a new set of \( CR \) values is generated using the same equation. During every generation, the \( CR \) values of offspring successfully entering the next generation are recorded in an array \( CR_{rec} \). After 25 generations, \( CR_m \), is updated:

\[
CR_m = \frac{1}{|CR_{rec}|} \sum_{k=1}^{CR_{rec}} CR(k). \tag{14}
\]

The inertia weight \( \omega \) is adapted by Eq. (15); we first set the selection probability \( f_p \) as 0.5; if the random value is smaller than \( f_p \), \( \omega \) is confirmed to the normal distribution with average value 0.5 and standard deviation 0.3, and otherwise, \( \omega \) is confirmed to the Cauchy distribution with parameter 1. The inertia weight \( \omega \) is adapted every 15 generations.

\[
\omega = \begin{cases} N(0.5, 0.3), & \text{if } U(0, 1) < f_p; \\ \xi, & \text{otherwise}, \end{cases} \tag{15}
\]

where \( f_p \) stands for selection probability. After evaluation of all offspring, the number of offspring successfully entering the next generation when generated by Eq. (9) and Eq. (10) are recorded as \( ns_1 \) and \( ns_2 \), respectively, and the numbers of offspring discarded when generated by Eq. (9) and Eq. (10) are recorded as \( nf_1 \) and \( nf_2 \), respectively. These two pairs of numbers are accumulated within a specified 50 generations, which is called the learning period. Then, the \( f_p \) is updated as:

\[
f_p = \frac{ns_1(ns_2 + nf_2)}{ns_2(ns_1 + nf_1) + ns_1(ns_2 + nf_2)).} \tag{16}
\]

D. Evaluation Criteria

The evaluation criteria of each individual reflects the performance of individuals based upon its achievement of objectives. In this paper, we choose the weighted value as the evaluation criteria, which considers total workload of vehicle \( W \), total interval time \( In \) and the makespan \( C_{max} \). The evaluation criteria equation in this paper is written as follows:

\[
f_{eval} = \sum_{i=1}^{q} \alpha_i n(f_i), \tag{17}
\]

where \( q \) is the number of sub-objective functions, \( n(f_i) \) is a normalized function and \( \alpha_i \) is a weighting coefficient. In this paper, there are 3 sub objective functions, i.e., \( q = 3 \). The values are 0.8, 0.15 and 0.05, respectively [24]. The objectives are listed as follows:

1) The completion time of the last task abandoning the system is called the makespan \( C_{max} \). The objective function is widely used in the performance indices of the vehicle scheduling research fields:

\[
f_1 = E[C_{max}]. \tag{18}
\]

2) The total expected vehicle load is:

\[
f_2 = E[W]. \tag{19}
\]

3) The total expected interval time of each vehicle is:

\[
f_3 = E[In]. \tag{20}
\]

IV. Numerical Experiments

In order to verify the effectiveness and efficiency of hccEA for uVSP under an uncertain environment, hccEA is performed on the benchmark under a determined environment with uniform distribution and Gaussian distribution, respectively. We compare hccEA with classic GA [26], binary GA (bGA) [26], PSO [27], differential evolution (DE) [28] and self-adaptive neighborhood search DE (SaNSDE) [29] and the co-evolution PSO (ccPSO) [23] on uVSP. All algorithms are implemented in Java, and the target machine is a PC with Intel(R) Core(TM) i7-4790 CPU @3.60GHZ and 12GB RAM. Detailed descriptions of numerical experiments are presented in the following sections.

A. Benchmark Description

For uVSP, we consider 6 instances under a determinate environment as the basic benchmark, denoted from VSP1 to VSP6. The number of tasks ranges from 10 to 20, the number of vehicles ranges from 4 to 15, the number of operations for each task ranges from 5 to 15, and the number of operations for all tasks ranges from 55 to 240 [30]. In order to ensure the reliability of the experiments and avoid the shock of results, all experiments are repeated 30 times, and the mean value is used as the final experimental results. It is observed that a vehicle scheduling with more than six vehicles is regarded as a large-scale uVSP [31] since the computational complexity is high enough. In this paper, a uVSP consisting of fifteen vehicles \((m = 15)\) is simulated to evaluate the performance of the methods.

B. Parameters Settings

The detailed settings of all algorithms and experiments are described in the following parts.
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2018.2797268, IEEE Access

1) PSO configuration: Two kinds of PSOs are tested in this section. They are classical PSO and ccPSO. The values of the parameters, i.e., $c_1$, $c_2$, and $w$ are selected based on suggestions in the literature [27] and given in Table II.

- PSO: The classical version PSO with two-phase encoding and decoding mechanisms.
- ccPSO: The classical PSO with two-phase encoding and decoding mechanisms embedded into the co-evolution framework.

2) GA configuration: Two GAs with different encoding and decoding mechanisms are tested in this section to verify the effectiveness of the proposed encoding and decoding mechanisms.

- GA: The classical GA with two-phase encoding and decoding mechanisms. The first phase uses job-based encoding and decoding mechanisms, and the second phase uses integer-based encoding and decoding mechanisms.
- bGA: The classical GA with binary-based encoding and decoding mechanisms.

The initial parameters of two GAs are given in Table II.

3) DE configuration: Except for GA and PSO, DE performs well in solving continuous optimization problems as well. Two kinds of DEDs are tested in the experiments.

- DE: The classical differential evolution algorithm.
- SaNSDE: A differential evolution algorithm with parameters self-adaptive mechanism.

4) Common parameters: The common parameters, i.e., population size and generations, for all algorithms (proposed algorithm and compared algorithms) are given in Table III.

C. Experiment I

In this section, we perform experiments focusing on the benchmark with uniform distribution. For each instance $VSP_i$, 30 scenarios are generated in which the processing time of operation on the vehicle is in line with the uniform distribution and referred to as $VSP_{i0}$. We execute hccEA and all compared algorithms on $VSP_{i0}$ and record the makespan, total workload and total interval time of all vehicles for 30 scenarios. The comparative results calculated by Eq. (17) are shown in Table IV. The Gantt chart of the best solution of $VSP_{i0}$ with uniform distribution is shown in Fig. 10. We can get conclusion that although hccEA and ccPSO have the same performance on $VSP_{i0}$, our proposed hccEA almost performs well on other 5 instances with uniform distribution.

D. Experiment II

Gaussian distribution is considered in this section. 30 scenarios are generated for each instance $VSP_i$. The processing time of operation on vehicle in all scenarios is in line with Gaussian distribution and referred to as $VSP_{iG1}$-$VSP_{iG50}$. We execute hccEA and all compared algorithms on $VSP_{iG1}$-$VSP_{iG50}$ and record the makespan, total workload and total interval time of all vehicles for 30 scenarios. The evaluation values calculated by Eq. (17) are shown in Table V. The Gantt chart of the best solution of $VSP_{i0}$ with Gaussian distribution is shown in Fig. 10. Among 6 instance with Gaussian distribution, our proposed hccEA has better performance especially on for $VSP_{i0}$.

E. Experiment III

Exponential distribution is considered in this section. For each instance $VSP_{i}$, 50 samples are generated in which each processing time of operation on vehicle is in line with the exponential distribution and referred to as $VSP_{iE}$-$VSP_{iE50}$.
Fig. 9. A Gantt chart for uVSP6 with uniform distribution.

Fig. 10. A Gantt chart for uVSP6 with Gaussian distribution.

Fig. 11. A Gantt chart for uVSP6 with exponential distribution.
We execute hccEA and all compared algorithms on VSP\textsubscript{1E1}, VSP\textsubscript{1E50}. Then, we record all 50 evaluation values and calculate the mean value as the final objective value of VSP\textsubscript{1}. The results are shown in Table VI. The Gantt chart of the best solution of uVSP\textsubscript{6} is shown in Fig.11. The results shown that, besides uVSP\textsubscript{2} and uVSP\textsubscript{4}, our proposed hccEA performs better than other state-of-the-art algorithms.

V. CONCLUSION

In this paper, we proposed hccEA which combines GA and ccPSO with the parameter self-adaptive mechanism to solve uVSP effectively and increases the robustness of optimal solutions under uncertainty. We first designed a two-phase encoding mechanism with the complex constraints of VSP in mind. Secondly, we considered to overcall the increase of problem scale by CC framework and finally, we balanced the uncertain factors using self-adaptive mechanism in the process of optimization. The proposed algorithm is experimented on the benchmark under uncertain environment with three kinds of probability distributions. For the total 30 instances, hccEA obtained 30 optimal solutions than compared algorithms. The results proves the effectiveness and efficiency of hccEA. Some work remains to be done to improve the performance of the algorithm, and future research will apply the algorithm on multiple objectives uVSP.

ACKNOWLEDGMENT

This work is partly supported by the National Natural Science Foundation of China under Grant 61572100 and in part by the Grant-in-Aid for Scientific Research (C) of Japan Society of Promotion of Science (JSPS) No. 15K00357.

REFERENCES


Lu Sun received BS and MS degrees at the Dalian University of Technology in 2013 and 2015 respectively and is currently a Ph.D. student at School of Software, Dalian University of Technology. She focuses on computational intelligence, deep learning, probabilistic graphical models, and their applications in combinatorial optimization problems.

Lin Lin received the Ph.D. degree from Graduate School of Information, Production and Systems, Waseda University, Japan, in 2008. He is a Professor with the DUT-RU International School of Information Science and Engineering, Dalian University of Technology, China, and a Senior Researcher with Fuzzy Logic Systems Institute, Japan. He was a Research Assistant and a Visiting Lecturer with Information, Production and Systems Research Center, Waseda University from 2006 to 2010. His research interest includes computational intelligence and their applications in combinatorial optimization and pattern recognition.

Haojie Li received the B.E. degree from Nankai University, Tianjin, China, in 1996 and the Ph. D. degree from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China, in 2007. He is a Professor with the DUT-RU International School of Information Science and Engineering, Dalian University of Technology, China. From 2007 to 2009, he was a Research Fellow with the School of Computing, National University of Singapore, Singapore. He has coauthored more than 60 journal and conference papers, including the IEEE TRANSACTION ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE TRANSACTIONS ON IMAGE PROCESSING, ACM Multimedia, and ACM ICMR. His research interests include social media computing, computer vision and multimedia information retrieval. Prof. Li is a Member of ACM.

Mitsuo Gen received the B.E., M.E., and Ph.D. degrees in electronic engineering from Kogakuin University in 1969, 1971, and 1975, respectively and the Ph.D. degree in informatics from Kyoto University, Japan, in 2006. He is a Senior Research Scientist with Fuzzy Logic Systems Institute and a Visiting Professor at Tokyo University of Science, Japan. He was a Professor with the Graduate School of Information, Production and Systems, Waseda University, Japan from 2003 to 2010. He was a Visiting Professor with Dept. of Industrial Eng. & Eng. Management, National Tsing Hua University in Taiwan from 2012 to 2014; Dept. of Industrial and Management Eng., a Hanyang Chair Professor with Hanyang University in Korea from 2010 to 2012; a Visiting Professor with Dept. of IE & OR, the University of California, Berkeley, CA from 1999 to 2000 and with Dept. of Industrial Eng., Texas A&M University in College Station, TX in 2000. His research interest includes evolutionary algorithms, manufacturing scheduling, logistics network, and decision making. He has coauthored five books such as Introduction to Evolutionary Algorithms, Springer in 2010, Network Models and Optimization: Multiobjective Genetic Algorithm Approach, Springer in 2008 and Genetic Algorithms & Engineering Optimization, John Wiley & Sons in 2000. He is an Area Editor of Computers & Industrial Engineering and an Editorial Board Member of several international journals.