Non-integrated Algorithm based on EDA and Tabu Search for Test Task Scheduling Problem

Hui Lu, Mengmeng Zhang
School of Electronic and Information Engineering
Beihang University, Xueyuan Road 37, Haidian District
Beijing, China
mluhui@163.com, txgczmm@163.com

Abstract—**The optimization of test task scheduling problem (TTSP) is an important issue in automatic test system (ATS). TTSP is a complex combination optimization problem and includes two sub-problems. They are test task sequencing and test scheme combination. According to the characteristic of TTSP, a non-integrated algorithm based on estimation of distribution algorithm and Tabu Search (EDA-TS) is proposed in this paper. EDA focuses on solving test task sequencing in global searching, and TS emphasizes on solving test scheme combination in local searching. In addition, we give a mathematical model for TTSP. We prove that TTSP is an NP-hard by using traveling salesman problem (TSP) based on the mathematical model. The statistical results of single objective TTSP suggest that our approach has a stronger searching ability and good convergence compared with other three popular algorithms. The experiments of the multi-objectives TTSP also illustrate that EDA-TS has a strong searching ability and can maintain a diversity of solutions.**

Keywords—test task scheduling problem; NP-hard; estimation of distribution algorithm; Tabu Search

I. INTRODUCTION

Optimizing test task scheduling problem (TTSP) is a kernel issue to the parallel test in automatic test system (ATS). TTSP allocates each test task to available instruments adequately for reducing test time, improving equipment utilization, enhancing throughput and driving down total costs significantly [1-3].

TTSP belongs to a complex combinatorial optimization problem and is a difficult nondeterministic polynomial (NP) [4] problem for optimization. In fact, the scheduling process of TTSP can be divided into two sub-problems. They are the test task sequencing and test scheme combination. On one hand, there are various arrangements of all tasks in TTSP. The number of arrangements will be dramatically increasing with the growth of the number of test tasks. On the other hand, each task may have more than one test schemes for a fix test task sequence. There are many test scheme combinations for a test task sequence. Therefore, selecting an optimal scheme from large test scheme combinations is another complex issue. The two sub-problems illustrate that the solution space of TTSP is tremendous, and it is difficult to obtain feasible solutions, especially for large scale TTSP.

In consequence, an algorithm with strong searching ability is necessary for solving TTSP. In current research, most of them focus on intelligent algorithms for solving scheduling problems. Owing to the inherent feature of potential parallelism and global searching ability, genetic algorithms (GA) becomes mainstream for solving scheduling problem. However, the GAs focus on solving TTSP in an integrated view. These GAs are difficult to solve TTSP because the solution space will expand exponentially with the problem size. Therefore, the performance of one single algorithm could be unsatisfactory. To improve the efficiency of the algorithms, some GAs emphasize the optimal design and improvement of the original algorithm. For example, Lu et al. [5] proposed a variable neighborhood multi-objectives algorithm (VNM) based on decomposition for TTSP. Others pay attention to adopting a hybrid algorithm by taking advantages of different algorithms. For example, a chaotic strategy was introduced into NSGA-II to enhance the local searching ability for solving TTSP [6]. The chaotic operation is used to help the algorithm to escape from the local optima of TTSP.

Although the methods above can solve TTSP based on an integrated perspective, they are more difficult to solve large scale problems. Besides, it is worth noting that the GAs solving TTSP are all based on microscopic genetic operation, like crossover and mutation between chromosomes. Therefore, considering the huge solution space and hierarchical structure of TTSP, we propose a non-integrate algorithm in this paper. It is designed based on estimation distribution algorithm (EDA) and Tabu Search (TS) to solve two sub-problems respectively in TTSP.

For EDA, it has recently been recognized as a prominent alternative to traditional evolutionary algorithms due to their increasing popularity. EDA generates new individuals in a macroscopic perspective. The new population is obtained through a probability model established from information of promising populations. In addition, EDA has been used to address other scheduling problems successfully [7-9]. Meanwhile, these scheduling problems have a common characteristic, namely they all solve permutation-based optimization problems with EDA. For TTSP, test task sequencing problem is also a permutation problem. Therefore, the EDA based on Bayesian statistical inference is designed to solve test task sequencing problem in TTSP, and Bayesian statistical inference is used to construct probability model. For TS, it is a kind of heuristic algorithms and has memory capacity. It adopts a tabu strategy to prevent the searching trapping into the local optimum and avoiding roundabout searching. Therefore, we employ TS to select optimal combination from massive test scheme combinations for a certain test task sequence. The proposed non-integrated algorithm considers both global searching ability and local searching ability in terms of EDA and TS. Therefore, it has a powerful searching ability for solving TTSP.

In addition, we give a mathematical model of TTSP based on integer programming with considering the constraint of resource in this paper. We define a maximal test completion time (makespan) according to the mathematical model of TTSP, and our work is to minimize the makespan.

We analyze the characteristic of TTSP in two aspects. First, a TTSP instance is used to illustrate TTSP is a combinational explosion problem in theoretical analysis. We also prove that TTSP is an NP-Hard based on TSP in mathematical proof.

For TTSP with minimizing makespan, the solution space will dramatically enlarge with the increasing of the scale. Therefore, EDA-TS is proposed to solve TTSP based on four different size of instances. We compare our proposed algorithm with other three intelligent algorithms, Genetic Algorithm and Scheme Choice Rule (GASCR) [10], Parallel Genetic Algorithm (PGA) [11] and Genetic Algorithm with Simulated Algorithm (GASA) [1] for two different sizes of TTSP instances. These three algorithms are based on microscopic genetic operation. The statistical results show that the searching ability of EDA-TS is stronger than other three approaches.

To verify the effectiveness of the searching ability of EDA-TS, we also investigate our approach for solving multi-objectives TTSP. The mean workload of instruments is considered together with makespan. For multi-objectives TTSP, there are many local optima. As a result, the difficulty of acquiring more Pareto solutions is increasing. The multi-objectives evolutionary algorithm based on decomposition (MOEA/D) [12] is compared with EDA-TS. The experiment results also suggest that our approach has a better searching ability and can explore more diverse solutions in objective space.

The remainder of this paper is structured as follows. Section II formulates the mathematical models for single objective TTSP. In Section III, TTSP is proved to be an NP-hard in detail. Section IV introduces the non-integrated algorithm based on EDA-TS. EDA based Bayesian inference and TS strategies are described in details. Section V gives the experiments results and the analysis according to the performance metrics of statistical data. Finally, Section VI concludes the paper.

II. MATHEMATICAL MODEL

A. *The Integer Programming Model for TTSP*

The aim of TTSP is to arrange n tasks on m instruments. In TTSP, there are a set of tasks $T=\{t_j\}_{j=1}^{n}$ and a set of instruments $R=\{r_i\}_{i=1}^{m}$. The notifications P_j^i, S_j^i and C_j^i represent the test time, the test start time and the test completion time of task t_j tested on r_i, respectively. Thereupon, an inequality $C_j^i \ge S_j^i + P_j^i$ is obtained. For the TTSP, one task can be tested on more than one instrument. In other words, some instruments collaborate for one test task. A variable O_j^i is defined to express whether the task t_j occupies the instrument r_i. The definition is as follows

$$O_j^i = \begin{cases} 1 & if\ t_j\ occipies\ r_i \\ 0 & others. \end{cases} \tag{1}$$

Generally speaking, some instruments could have redundancies. That is, task t_j may have several selectable test schemes. The set of the test scheme for t_j is defined as $W_j=\{w_j^k\}_{k=1}^{k_j}$, where k_j is the number of test schemes of t_j. The notation $P_j^k=\max_{r_i\in w_j^k} P_j^i$ is used to express the test time of t_j for w_j^i.

The constraint of resources is as following.

$$X_{jj^*}^{kk^*} = \begin{cases} 1 & if\ w_j^k \cap w_{j^*}^{k^*} \ne 0 \\ 0 & others. \end{cases} \tag{2}$$

In addition, the basic hypothesis includes three factors.

(1) At a given time, an instrument can only execute by one task.

(2) Each task must be completed without interruption once it starts.

(3) Assume $P_j^i = P_j^k$, $C_j^i = S_j^i + P_j^i$ to simplify the problem.

B. *Fitness Function*

The makespan is common criteria for the scheduling problem in practical industrial processing. Therefore, the makespan is minimized for TTSP. The notification $C_j^k = \max_{r_i\in w_j^k} C_j^i$ is the test completion time of t_j for w_j^i. Thus, the maximal test completion time of all tasks can be defined as follows:

$$f_1(x) = \max_{\substack{1\le k\le k_j \\ 1\le j\le n}} C_j^k \tag{3}$$

III. PROOF OF NP-HARD

From the integer programming model of TTSP, we know that task t_j may have several selectable test schemes, and there may be numerous test tasks. As a result, both the numerous tasks and various selectable schemes for each task will have a significant impact on the complexity of TTSP. We suppose that there are 20 test tasks of a task scheduling instance in ATS, and each task has three selectable test schemes. There will be $20! \approx 10^{18}$ orderings of the task sequence. The total number of the test scheme combinations is $3^{20} \approx 10^9$. Therefore, the size of searching space is up to 10^{27}. If a computer obtains one solution in 1 ns, the total time of all solutions being searched will be 10^{11} years. It is longer than the existence of our earth. This example helps illustrate that the size of TTSP will be exponential growth with the increasing of the number of the test task and test scheme.

Through theoretical analysis of the above example, it demonstrates that TTSP is an NP-hard problem. To further prove TTSP is an NP-hard, we give a mathematical proof based on TSP.

To prove TTSP, we suppose there are m test instruments and n tasks, and there is no sequence constraint between tasks. Every task has different test scheme and the corresponding set of instruments for each scheme is R. The test time set for every task on the instrument is T. The scheduling objective is test complete time $T_{\max}$. Given positive number L, does there exist a task schedule S whose $T_{\max}$ does not exceed L.

To illustrate the NP-hard of TTSP, we take one-instrument test task scheduling problem under consideration. The known NP-hard problem is the TSP. The definition of TSP is as follows. There are n cities and distances between each pair of cities. Given a positive number M, does there exists the shortest possible route $L'(L' \le M)$ that visits each city exactly once and returns to the origin city. Here, every sub-distance for the route is $d_i, i \in [1, n]$. The reduction process is as follows and Fig. 1 illustrates the reduction process of TSP and TTSP.

For an input of TSP, we have an input for TTSP. There are 1 instrument and n tasks. The test time for each task T_i is $t_i = d_i$. Here, $L = M$.

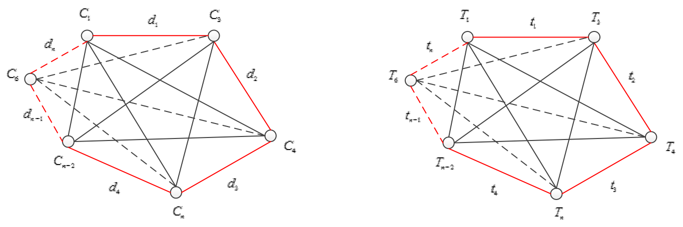


Fig. 1. The reduction process of TSP and TTSP

Firstly, we suppose for TSP the output is true. We have the shortest possible route $S = \{C_1, C_3, C_4, C_n, C_{n-2}, \cdots, C_6, \cdots, C_1\}$, as shown in the Fig.1. The distance of the route is $L' = d_1 + d_2 + d_3 + \cdots d_{n-1} + d_n$ and we have $L' \le M$. The corresponding input for TTSP is $t_i = d_i$. We have a schedule $S = \{T_1, T_3, T_4, T_n, T_{n-2}, \cdots, T_6, \cdots, T_1\}$ and the test complete time is $T_{\max} = t_1 + t_2 + t_3 + \cdots t_{n-1} + t_n$. Therefore, $T_{\max} \le L = M$.

Conversely, if we suppose the output of TSP is false. The shortest possible route does not exist. As a result, $L' > M$. It is easy to see that we can't get a possible schedule for TTSP that has $T_{\max} \le L = M$.

Note that, from the above process, TSP can be reduced to TTSP with 1 instrument and n tasks. As a result, TTSP with 1 instrument and n tasks is an NP-hard because TSP is an NP-hard.

IV. Algorithm Framework

A. EDA based on Bayesian Statistical Inference

For TTSP, the location of each task in test task sequence directly affects the individual's fitness. Therefore, the establishment of probability model reflected the location information of each task in test task sequence is important. It is the key problem in EDA for solving scheduling problem. Bayesian statistical inference provides us a tool to build a probability model by using the location information of test task sequences.

1) Bayesian Statistical Inference

The main feature of Bayesian statistical inference is the usage of the prior distribution. After a sample is chosen, the posterior distribution is obtained from observation values of sample and prior distribution information. In other words, the posterior distribution involves prior information and sample information. It is the basis of Bayesian statistical inference. The Bayesian formula is given below:

$$p(A_i/B) = p(A_i)p(B/A_i) \Big/ \sum_{j=1}^{n} p(A_j)p(B/A_j) \tag{4}$$

where $\{A_i, i = 1, 2, \cdots, n\}$ constitutes a complete set of mutually exclusive events. $p(A_i)$ is prior probability. $\{p(A_i), i = 1, 2, \cdots, n\}$ constitutes prior probability distribution. The occurrence of event B will affect the occurrence of event A_i and reflect by the posterior distribution $\{p(A_i/B), i = 1, 2, \cdots, n\}$. Therefore, Bayesian formula integrates a prior information and sample information provided by experiment to acquire a posterior information. The transformation of a prior distribution to a posterior distribution is a mathematical model of Bayesian statistical inference.

2) Establishing Probability Model

The Bayesian statistical inference is introduced into EDA to establish probability model. It directly uses the sample information to build a posterior probability. The posterior probability model contains prior distribution information and sample information. Therefore, we can use this posterior probability model to guide the generation of new population and achieve evolutionary process.

a) Prior probability model

The Bayesian statistical inference probability model is established based on superior individual and superior population. The superior individual and superior population is selected from the entire population according to the fitness values.

The first step is to establish the corresponding prior probability and conditional probability. For TTSP, test task sequence is made up of task numbers. The location information of tasks in test task sequence is used to build the probabilities.

The population TT can be expressed as follows,

$$TT = \{t_i(j),\ \ i = 1, 2, \cdots, N,\ \ j = 1, 2, \cdots, L\}$$

Where $t_i(j)$ is a task in the *j*th location of the *i*th individual (test task sequence). The length of an individual is L. In other words, L tasks correspond to L locations in a test task sequence. N is the number of population. First, we choose a superior individual $t^b = \left(t^b(1), t^b(2), \cdots, t^b(j), \cdots, t^b(L)\right)$ from population TT. The superior population is a sample selected from the entire population and the size of superior individuals is $M = S \times N$. S is the ratio of the number of sample to the number of population.

We establish prior probability vector based on each location in a superior population. The prior probability vector describes the probability of each task emerging in one location. All the prior probability vectors corresponding to each location compose a prior probability model. For example, the relationship between the values of tasks in location j and prior probability vector in the *j*th location can describe as follows.

$$\begin{array}{lcccc} \text{t}: & 1 & 2 & \cdots & L \\ \mathbf{p}_j: & p_{1j} & p_{2j} & \cdots & p_{Lj} \end{array}, \qquad j = 1, 2, \ldots, L$$

Probability vector $\mathbf{p}_j = \left(p_{1j}, p_{2j}, \cdots, p_{Lj}\right)^T$ describes the prior probability of all tasks appearing in the *j*th location. Therefore, all probability vectors in all locations construct a probability matrix $\mathbf{P} = \left(\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_j, \cdots, \mathbf{p}_L\right)$. $\mathbf{P}$ is the prior probability model.

b) Conditional probability model

In TTSP, a task can appear in any location in a test task sequence without constraints. Assume task $t_i(j)$ emerges in the *j*th location. Any other task can emerge before $t_i(j)$. It means we can use a conditional probability vector to indicate the information of the other tasks emerging before task $t_i(j)$. Therefore, the conditional probability vector can be expressed as $\mathbf{q}_j = \left(q\left(t^b(j)/1\right), q\left(t^b(j)/2\right), \cdots, q\left(t^b(j)/L\right)\right)^T$. This conditional probability vector is also based on superior individual and superior population. It describes the probability when other tasks emerge in location j, superior task $t^b(j)$ emerges before task $j, j = 1, 2, \cdots, L$ in the whole superior population. It means L locations represent L conditional probability vectors and composes a conditional probability matrix. In the *g*th generation in an evolutionary process, we assume that the superior individual is selected as $t^{gb} = \left(t^{gb}(1), t^{gb}(2), \cdots t^{gb}(j), \cdots, t^{gb}(L)\right)$. Therefore, the conditional probability matrix can be expressed as follows,

$$Q^g = \left\{ q\left(t^{gb}(j)/i\right),\ i, j = 1, 2, \cdots, L \right\}.$$

Here, $q\left(t^{gb}(j)/i\right)$ indicates the conditional probability of task $t^{gb}(j)$ before task $i, i = 1, 2, \cdots, L$. According to the conditional probability formula, we have $q\left(t^{gb}(j)/i\right) = q\left(t^{gb}(j) \cdot i\right) / q(i)$. Here, $q\left(t^{gb}(j) \cdot i\right)$ denotes the probability that $t^{gb}(j)$ and i emerge simultaneously, and $q(i)$ denotes the probability that i emerges. Therefore, conditional probability is transformed to a product probability. To compute the product probability in superior population, $t^{gb}(j)$ must be before i, $i = 1, 2, \cdots, L$. Therefore, the frequency of $t^{gb}(j) \cdot i$ is:

$$\begin{gathered} q\left(t^{gb}(j) \cdot i\right) = \frac{1}{M} \sum_{n=1}^{M} \delta_{ij}\left(t_n^g\right) \\ \delta_{ij}\left(t_n^g\right) = \begin{cases} 1 & t_n^g(l) = t^{gb}(l) \wedge t_n^g(l+1) = i \\ 0 & otherelse. \end{cases} \end{gathered} \tag{5}$$

Here, $i, j = 1, 2, \cdots, L,\ l \in [1, 2, \cdots, L-1]$. $\delta\left(t_n^g\right) = 1$ denotes that $t^{gb}(j)$ emerges before i, otherwise $\delta\left(t_n^g\right)$ equals to 0. For $q(i)$, it equals to 1. The reason is that i always emerges in a test task sequence.

c) Posterior probability model

If it is the *g*th generation of evolutionary process, posterior probability model R^g can be obtained by Bayesian formula with prior probability model and conditional probability model. The posterior probability model can be denoted as $R^g : \left\{ r_{ij} = r\left(i/t^{gb}(j)\right), i, j = 1, 2, \cdots, L \right\}$. Here, we have

$$r\left(i/t^{gb}(j)\right) = q\left(t^{gb}(j)/i\right) \cdot p_{ij} \Big/ \sum_{i=1}^{L} q\left(t^{gb}(j)/i\right) \cdot p_{ij} \tag{6}$$

The denominator is the sum of product of corresponding column in prior probability model and conditional probability model.

3) Updating Probability Model

The posterior probability model is composed of prior probability model and conditional probability model. Therefore, the probability model updating can be divided into two steps.

The prior probability model updates as follows.

$$\begin{gathered} p_{ij}^{g+1} = \begin{cases} \left[1 - \alpha \cdot \frac{1}{M} \sum_{k=1}^{M} \left(\delta_j\right)_k\right] \cdot p_{ij}^g + \alpha \cdot \frac{1}{M} \sum_{k=1}^{M} \left(\delta_j\right)_k & t_k^g(j) = t_k^{gb}(j) \\ \left[1 - \alpha \cdot \frac{1}{M} \sum_{k=1}^{M} \left(\delta_j\right)_k\right] \cdot p_{ij}^g & t_k^g(j) \neq t_k^{gb}(j) \end{cases} \\ \delta_j = \begin{cases} 1 & t_k^g(j) = t_k^{gb}(j) \\ 0 & t_k^g(j) \neq t_k^{gb}(j) \end{cases} \end{gathered} \tag{7}$$

The conditional probability model updates as follows.

$$q_k^{g+1} = (1-\beta)\cdot q_k^g + \beta \cdot \frac{1}{M}\sum_{k=1}^{M}(\delta_j)_k \quad (8)$$

Here, $\alpha, \beta \in [0,1]$. The bigger values they have, the greater effect on the next generation will they have. $\delta_j\ (j = 1, 2, \cdots, L)$ is the comparison value between superior individual and superior population in each location. If the values are the same in a location, $\delta_j = 1$. Otherwise δ_j equals to 0. For each location, if the superior population is the same as the superior individual, the probability is increasing. Otherwise it decreases. The probability of the excellent individuals emerging in the next generation is increasing. It provides the information to further search and to achieve the optimal solution.

Similarly, posterior probability model updating is accomplished by prior probability model and conditional probability model based on Bayesian formula. The superior individual and superior population are taken into account during the updating process. Therefore, the posterior probability model has a better ability to guide the generation of new individuals during the evolutionary process.

B. Tabu Search

From the previous analysis of TTSP, there are huge test scheme combinations for a test task sequence with the increasing of the number of test tasks and test schemes. The TS is considered to obtain an optimal test scheme and avoid trapping into the local optimum. Generally speaking, TS includes four steps. They are initial solution, neighborhood solution, tabu list and termination criterion. In TTSP, the initial solution and neighborhood solution correspond to initial test scheme, neighborhood test schemes.

1) Initial Test Scheme

For TTSP, the test scheme needs to be selected from a huge test scheme combinations for a fixed test task sequence. According to the basic process of TS, the first step is to generate an initial scheme randomly. A test task instance 6×8 in TABLE I is used to illustrate TS process. $n\times m$ is used to represent the test task instance for TTSP. Here, n is the number of tasks and m is the number of instruments.

TABLE I. TEST TASK INSTANCE 6×8

task	scheme	resource	task	scheme	resource
t_1	w_1^1	r_1, r_7	t_4	w_4^1	r_4
	w_1^2	r_3, r_5		w_4^2	r_8
	w_1^3	r_6, r_8	t_5	w_5^1	r_7
t_2	w_2^1	r_2	t_6	w_6^1	r_1, r_4
	w_2^2	r_4		w_6^2	r_3, r_7
	w_2^3	r_6		w_6^3	r_6, r_8
	w_2^4	r_7			
t_3	w_3^1	r_3			
	w_3^2	r_5			

From TABLE I, each task has more than one test schemes. Assume that there is a certain arrangement of test task sequence $t = [t_3, t_2, t_1, t_6, t_5, t_4]$ in a population. An initial test scheme $ts = [w_3^1, w_2^2, w_1^2, w_6^3, w_5^1, w_4^1]$ of test task sequence t is randomly generated.

2) Neighborhood Test Schemes

Based on the current initial solution, neighborhood moving produces a number of new solutions according to a specific moving strategy. The new solution is called neighborhood solution. The number of new solutions is called neighborhood size.

For test scheme combination in TTSP, the neighborhood test scheme of ts can be expressed as $tn = [w_3^2, w_2^2, w_1^2, w_6^3, w_5^1, w_4^1]$ after initial test scheme ts is determined. The difference between ts and tn is that w_3^1 is placed by w_3^2 for task t_3. The exchanging between w_3^1 and w_3^2 is named as a moving and the inversive changing is regarded as the same moving.

The size of neighborhood test scheme set is computed as formula (9). Here, $N(t(i))$ presents the number of the test schemes in the ith location for test task sequence t and L is the length of test task sequence.

$$Nei\_Size = \sum_{i=1}^{L} N(t(i)) - L \quad (9)$$

3) Tatu List

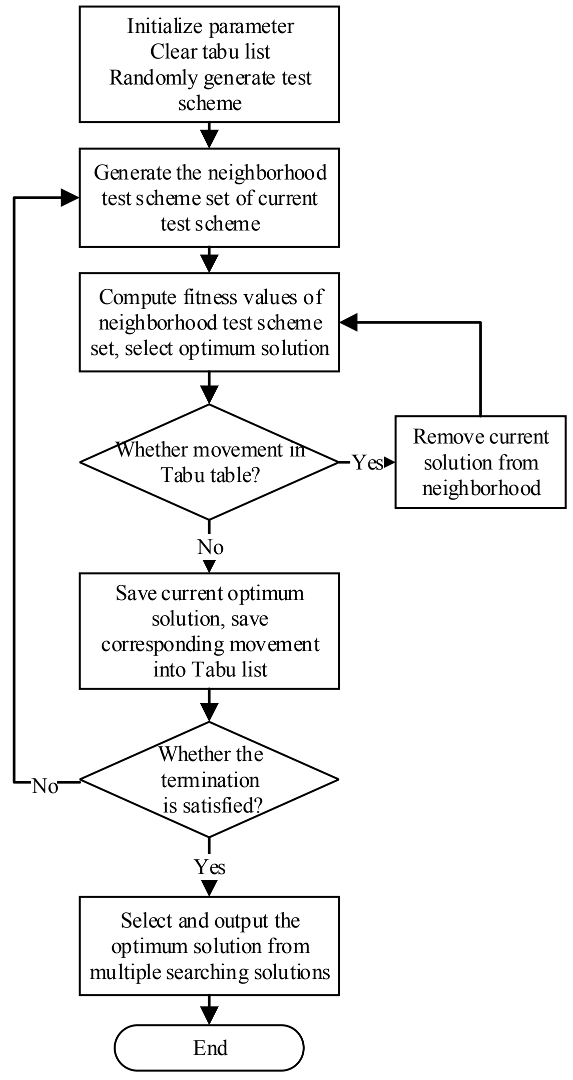
The tabu list is used to prevent the searching process appearing roundabout and avoid falling into local optimum. Tabu list records the recent movements. Within a certain number of iterations, the movement is forbidden to access. A movement will withdraw and can be accessed again after certain iterations. The tatu object can be designed flexibly according to problems. In this paper, the test task and its corresponding test schemes compose tabu objects. In other words, the tabu list records each movement for a certain test task and regarded them as tabu objects. The concrete form of tabu object is as formula (10). It indicates that the current solution in a certain iteration selects exchanging of test schemes w_i^a and w_i^b as a movement for test task t_i. The size of tabu list depends on a specific problem. For TTSP in this paper, the length of tabu list Len_Tabu is set to 10.

$$[t_i, w_i^a, w_i^b] \quad (10)$$

4) Tabu Search Flowchart

Fig. 2 illustrates the flowchart of using TS to obtain optimal test scheme for a fixed test task sequence.

In the flowchart, initializing parameter includes the length of tabu list Len_Tabu and maximum iteration $Iter\_Num$. The termination of iteration of TS is the maximum number of iteration.



Initialize parameter
Clear tabu list
Randomly generate test scheme
Generate the neighborhood test scheme set of current test scheme
Compute fitness values of neighborhood test scheme set, select optimum solution
Whether movement in Tabu table?
Yes
Remove current solution from neighborhood
No
Save current optimum solution, save corresponding movement into Tabu list
Whether the termination is satisfied?
No
Yes
Select and output the optimum solution from multiple searching solutions
End

Fig. 2. Flowchart of Tabu Search

V. Experiment and Analysis

In this section, we first conduct the simulations for single objective TTSP with the non-integrated algorithm we proposed. Four instances of TTSP are applied to verify our proposed algorithm. In addition, three other state-of-the-art genetic algorithms are also compared with EDA-TS. Secondly, to further validate the searching ability of EDA-TS, we also carry out the experiment for multiple objectives TTSP.

All these experiments are carried out on a personal computer running with an Intel(R) Core™ i7-4790 3.60GHz and 8.00GB RAM.

A. Single Objective TTSP

To validate the performance of EDA-TS for single objective TTSP, simulation results of four instances 6×8, 20×8, 30×12, 40×12 are illustrated. For each instance, the experiment runs ten times. The parameters set of EDA-TS is shown in TABLE II.

TABLE II. Parameters in EDA-TS

N	300	β	0.5
Evo_Num	20	Len_Tabu	10
S	0.8	$Iter\_Num$	10
α	0.5		

The statistical results of each instance are shown in TABLE III. For small size TTSP, the solution obtained is stable. Through enumeration method of the instance 6×8, we find that 23 is the optimal solution. Therefore, the statistical results of instance 6×8 show that our proposed approach has perfect searching ability for small scale TTSP. For larger scale instances of TTSP, the value of optimal solutions fluctuates.

TABLE III. Statistical results of single objective TTSP

	6×8	20×8	30×12	40×12
Max	23	29	33	42
Min	23	28	31	40
Mean	23	28.7	32.3	41.2
Var	0	0.2333	0.4556	0.4

To inspect the searching ability of EDA-TS, GASCR, PGA and GASA are compared with our approach. The best value (BV), success rate (SR) and mean best fitness measure (MBF) are the common evolution metrics for the performance of these algorithms. BV is the optimum that an algorithm searches in multiple runs. SR measures the probability of searching the optimum of an algorithm for all runs. The metric MBF illustrates the mean value of optimum solutions for multiple runs. The statistical results of EDA-TS, GASCR, PGA and GASA for instances 20×8, 40×12 are shown in TABLE IV

TABLE IV. Statistical results of different algorithms

	20×8			40×12		
	BV	SR	MBF	BV	SR	MBF
EDA-TS	28	0.3	28.7	40	0.1	41.2
GASCR	28	0.5	29	40	0.4	40.6
PGA	31	0.2	33.8	42	0.1	44.8
GASA	32	0.2	33.9	47	0.1	49.2

The statistical results of EDA-TS have the same parameters shown in TABLE I. For instance 20×8, our approach is more competitive compared with PGA and GASA. For GASCR and EDA-TS, they both can search the same optimal solution. The metric MBF of EDA-TS is smaller than other algorithms. It suggests that, our proposed approach can obtain much better quality solutions for multiple runs. For larger size TTSP 40×12, EDA-TS also shows a better searching ability compared with PGA and GASA. Both EDA-TS and GASCR can obtain the same best value. However, the other two metrics of our approach are a little worse than that of GASCR. The reason is that with the scale of TTSP increasing, the solution space is enlarged.

To further verify the searching ability of EDA-TS for solving large scale TTSP, we carry out experiment for instance 40×12 by increasing the parameter $Iter\_Num$ to 20. The other parameters are the same as that given in TABLE II. The statistical results of metrics BV, SR and MBF are 39, 0.2 and 40.2 respectively. $Iter\_Num$ is the maximum iteration in TS. Increase of its value is equivalent to increase the searching range of test scheme combinations in TTSP. It is obvious that EDA-TS can obtain much better solution than that when $Iter\_Num=10$. Meanwhile, the metric MBF is becoming much better than that when $Iter\_Num=10$. It suggests that EDA-TS also has a perfect searching capacity for large scale TTSP.

B. *Multiple Objectives TTSP*

In order to verify the searching ability of EDA-TS for solving multiple objectives TTSP, the dimension of objective function is extended to two. The second function is a mean workload of instruments. For mean workload function, the notation Q is to describe the parallel steps. The initial value of Q is 1. Assign the instrument for all the tasks, if $X_{jj^*}^{kk^*}=1$, $Q=Q+1$. Therefore, the mean workload of the instrument is defined as follows:

$$f_2(x)=\frac{1}{Q}\sum_{j=1}^{n}\sum_{i=1}^{m}P_j^iO_j^i \qquad (11)$$

In multi-objectives TTSP, sometimes two solutions cannot be compared because both of them do not dominate each other. Therefore, we select the super individual and population based on the non-dominated sorting [13] of the entire population.

The experiments of multi-objectives TTSP based on EDA-TS are carried out with the same parameters in TABLE II. In addition, another popular genetic algorithm MOEA/D is used to compare with our approach. Four instances 6×8, 20×8, 30×12 and 40×12 are also applied to verify the role of our approach for solving multi-objectives problems.

Fig.3 is the four Pareto fronts of different instances. The diversity of the solutions of our algorithm is better than that of MOEA/D. For small size instance 6×8, there are four true Pareto points, $(23,70/3)$, $(28,19.5)$, $(31,18)$, $(36,14.6)$. Our approach can search all these solutions exactly. MOEA/D only searches three solutions, and one of them is not the true Pareto solution. For other three different size instances, the EDA-TS can also search more solutions in the entire solution space than that of MOEA/D. Meanwhile, the distribution of the solutions is relatively uniform. In fact, these four Pareto fronts suggest that EDA-TS can enlarge the searching range and further increase the diversity of the solutions for multi-objectives TTSP. However, the convergence of solutions obtained by these two algorithms is similar according to Pareto fronts.

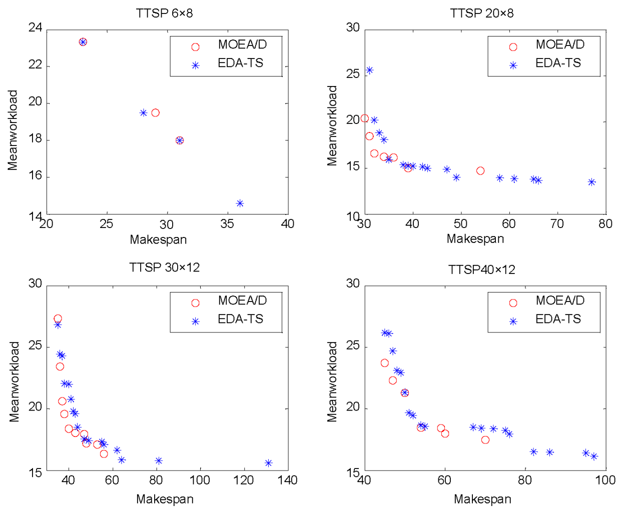


Fig. 3. Pareto fronts for different TTSP

C. *Summary of Experiments*

Overall, the above experiments are divided into two parts: single objective TTSP for minimizing makespan and multi-objectives TTSP for optimizing makespan and mean workload together.

For single objective TTSP, four instances are solved based our approach. The solutions acquired by EDA-TS suggest that our algorithm can search more stable solutions in objective space. We compare EDA-TS with GASCR, PGA and GASA for solving 20×8 and 40×12. From the statistical results of metrics BV, SR and MBF, it shows that EDA-TS has a stronger searching ability than the others. The metric MBF indicates that our approach can obtain the stable optimum in multiple runs. In addition, EDA-TS can search better solution for larger size TTSP.

For multi-objectives TTSP, from the Pareto fronts of four instances, the solutions obtained by EDA-TS are widely spread in the solution spaces. It shows that EDA-TS also has a perfect searching ability for solving multi-objectives TTSP and further maintain diversity of the solutions. However, the convergence of EDA-TS is not apparently superior compared with MOEA/D.

VI. Conclusions

TTSP is an important issue in automatic test system. In this paper, we introduce a mathematical model based on integer programming for TTSP. We prove that TTSP is an NP-hard problem. For TTSP, it is decomposed into two sub-problems, test task sequencing operation and test scheme combination operation. According to the characteristic of these two sub-problems, we propose a non-integrated algorithm based on EDA and TS. The hybrid algorithm has a stronger searching ability for solving both the single objective TTSP and the multi-objectives TTSP. For single objective TTSP, our approach has a good performance in convergence. In addition, it also has a strong searching ability for larger scale TTSP. To further explore the feasibility and searching ability of our approach for solving multiple objectives TTSP, the dimension of objective function is extended. The experiment results show that our approach also has a strong searching ability and can maintain the diversity of solutions in entire objective space for different sizes of TTSP. However, the convergence of our approach for solving multi-objectives TTSP does not improve. Therefore, our future work will focus on improving the convergence.

Acknowledgment

This research is supported by the Industrial Foundation Technology Development Program under No.J1320130001 and the National Natural Science Foundation of China under Grant No.61101153.

Reference

[1] R. Xia, M. Xiao, J. Cheng, and X. Fu, "Optimizing the multi-UUT parallel test task scheduling based on multi-objective GASA," International Conference on Electronic Measurement and Instruments, vol. 4, pp. 839-844, Aug 2007.

[2] D. Zhou, P. Qi, and T. Liu, "An optimizing algorithm for resources allocation in parallel test," in IEEE International Conference on Control and Automation, vols 1-3, Christchurch, 2009, pp. 1997-2002.

[3] R. Xia, M. Xiao, and J. Cheng, "Parallel TPS design and application based on software architecture, components and patterns," in 2007 IEEE Autotestcon, vols 1 and 2, New York , 2007, pp. 234-240.

[4] A. Rădulescu, C. Nicolescu, A. J. C. van-Gemund, and P. P. Jonker, "CPR: mixed task and data parallel scheduling for distributed systems," in Parallel and Distributed Processing Symposium, San Francisco, 2001.

[5] H. Lu, Z. Zhu, X. Wang, and L. Yin, "A variable neighborhood MOEA/D for multiobjective test task scheduling problem," Mathematical Problems in Engineering, vol. 2014, pp. 14, 2014.

[6] H. Lu, R. Y. Niu, J. Liu, and Z. Zhu, "A chaotic non-dominated sorting genetic algorithm for the multi-objective automatic test task scheduling problem," Applied Soft Computing, vol. 13, pp. 2790-2802, May 2013.

[7] D. Pandolfi, A. Villagra and G. Leguizamon, "Hybrid estimation of distribution algorithms for the flow shop scheduling problem," in IEEE Congress on Evolutionary Computation, Cancun , 2013, pp.1694-1701.

[8] Q. K. Pan and R. Ruiz, "An estimation of distribution algorithm for lot-streaming flow shop problems with setup times," Omega, vol. 40, pp. 166-180, Apr 2012.

[9] S. Wang, L. Wang, M. Liu, and Y. Xu, "An effective estimation of distribution algorithm for solving the distributed permutation flow-shop scheduling problem," International Journal of Production Economics, vol. 145, pp. 387-396, Sep 2013.

[10] H. Lu, J. Liu, R. Y. Niu, and Z. Zhu, "Fitness distance analysis for parallel genetic algorithm in the test task scheduling problem," Soft Computing, vol. 18, pp. 2385-2396, Dec 2014.

[11] F. M. Defersha and M. Y. Chen, "A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups," International Journal of Advanced Manufacturing Technology, vol. 49, pp. 263-279, Jul 2010.

[12] Q. F. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," IEEE Transactions on Evolutionary Computation, vol. 11, pp. 712-731, Dec 2007.

[13] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," IEEE Transactions on Evolutionary Computation, vol. 6, pp. 182-197, Apr 2002.