http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 13 (8): 1495-1504, 2014
ISSN 1812-5638 / DOI: 10.3923/itj.2014.1495.1504

Constraint Handling Technique in Test Task Scheduling Problem

Hui Lu, Xiaoteng Wang and Jing Liu
School of Electronic and Information Engineering, Beihang University, Beijing, 100191, China

Abstract: As the key element of automatic test systems, the Test Task Scheduling Problem (TTSP) is becoming an increasingly important issue. The TTSP often involves two kinds of constraints. The temporal constraints among tasks are often the net ones and the resource constraints are generated due to the lack of enough resources for each task. The current methods to handle the constraints can be classified into optimization and satisfaction. Most of them suffer much difficulty to solve the constraints in the TTSP. However, the optimization methods are abandoned since the infeasible solutions are not allowed in the TTSP to avoid the accidents. The current satisfaction methods are usually for the tree or chine constraints. In addition, there is seldom research on the temporal and resource constraints together. Hence, the Priority-based (PB) encoding is provided to solve the temporal constraints while the Scheme Choice Rule (SCR) is created to find the corresponding schemes to distribute certain resource to these tasks. The compounding of PB and SCR called PBSCR is embedded into a genetic algorithm and applied to two benchmarks. Empirical results suggest that PBSCR outperforms better than not only the optimization methods but also the satisfaction methods. The efficacy of PBSCR is thereby justified.

Key words: Temporal constraints, resource constraints, priority-based encoding, scheme choice rule

INTRODUCTION

Since the mid-1950s, automated test technology has been a necessary guarantee for complex systems and equipment in a variety of industries, such as the audio equipment, semiconductor, wireless, aeronautic and aerospace industries. With the rapid development of these industries, test requirements are changing and the test system weaknesses,such as high maintenance costs, limited application scope and deficiency of mutual systems, are becoming obvious. Thus, the Test Task Scheduling Problem (TTSP) is a key element of the automated test technology that has attracted research because of the demand for lower costs and improved test efficiencies and importability (Ross, 2003). TTSP means a finite set of tasks with various schemes to be processed on several unrelated resources. It aims to minimize the test completion time for all available resources by sorting all tasks to an optimal sequence and obtaining the optimal scheme for each task.

The temporal and resource constraints are main characteristics of the TTSP. The temporal constraints are temporal relations between tasks. Some tasks with higher priority should be executed first. For example, the power-on-self-test completes before others operates. They are generated due to the task decomposition according to the test requirements. This decomposition results in a temporal constraint net for the TTSP. Another problem is the resource constraints. The resources repeat to be used since only limited of them are provided for practical reasons, such as saving expenses, scarcity of resource or approaching deadline. Similarly, resource constraint in other scheduling problem (Lombardi and Milano, 2012) has been a hot issue.

The methods for constraints have two main approaches. The first branch is to optimize them while the other is to satisfy them. The penalty functions and the multiobjective optimization methods are the case of the optimization. The penalty function as an oldest method can transform the constrained problem into an unconstrained one by adding something to the objective function. The penalty function can range from completely rejecting the individual to decreasing its objective based on the degree of violation present in unfeasible individuals (Venkatraman and Yen, 2005). The static function can be expressed as the cost to repair the individuals, the amount of its violation or a fixed value (Coello, 2002) but they may result in too weak or too strong a penalty during different stages of the search process. In recent years, the function becomes a dynamic one with penalty parameters dependent on the generation number. It can make the penalty adaptive to the searching process (Haddad and Marino, 2007) but end up with the difficulty of selecting another set of factors. Then emphasis turns into multiobjective optimization to find an optimal feasible solution. It is another common method to

Corresponding Author: Hui Lu, School of Electronic and Information Engineering, Beihang University, Beijing, 100191, China

transform constrained problem by adding an objective function which is closely related to the constraint violation (Bartak *et al*., 2010). Coello (2000) calculates the rank of each individual based on their constraint violation as an objective and optimizes it by multiobjective genetic algorithm. Wang *et al*. (2007) chooses four different forms constraint violation as an objective and uses the evolutionary algorithm with the global and local search in various constrained optimization problems. However, the non-dominance solutions on the pareto-optimal front may mislead the search direction and make the search space limited in an inappropriate part of the feasible region from which is not be able to escape.

The optimization methods are effective but accompany the unfeasible solutions. However, the unfeasible solutions won't appear if the methods to satisfy a set of constraints are adopted. The repair operators and the backtracking methods belong to this second branch. They outperform better since they inference while search in the feasible solution space (Bartak *et al*., 2010). The repair operators are to fix an infeasible solution through a specific repair procedure without considering the objective functions (Harada *et al*., 2007). The repair operators perform well in a simple constrained situation such as simple bounds and chain constraints. Through the repair operators, the solutions outside the boundaries can return to the nearest boundary and the solution not satisfying the chain constraints can be rearranged and reunited. For example, Wang proposed special crossover operators to meet the time-limited constraints and combines with simulated annealing to repair a disrupted schedule with the minimum cost of resource conflicts (Wang, 2005). The weakness of the repair operators is the demand for priority knowledge of problem's characteristic to design certain repair procedure and provides at least one feasible solution for initialization (Chootinan and Chen, 2006). Backtracking is such a method more attached to the solutions without the priority knowledge. Its ability to check for satisfaction helps discard a huge number of junk solutions and reduce the parts of the search space that need to be visited (Comon *et al*., 1999). However, it suffers time consuming (Zahn and Hower, 1996), especially for the net-work constraints.

Based on the consideration of the safety of test systems and related operators, the infeasible solution for the TTSP is not allowed. Hence, the satisfaction methods here are proposed to obtain a solution closer the ideal optimal one. The disadvantages of the backtracking and repair operators motivate another method for the constraints. What is more, the constraints in TTSP are the net ones which are harder than the chain and tree ones. Then the repair operators and backtracking methods are ignored and priority-based encoding is proposed to solve the net-constraints among tasks in the TTSP.

The resource constraints in the TTSP are reflected by multiple schemes for a task since the scheme choices of all the tasks determine the final resource arrangements. Then, the resource constraints become a combination problem instead of the common constraint methods. The dispatching rule is the cheapest way to limit resource constraints to the "reasonable" ones. Sels provide a summary of 28 dispatching rules including the basic dispatching rules and the hybridizations of them (Sels *et al*., 2011). These rules, as Kim indicates, are general priority rules that control which job should be selected among candidate jobs to be allocated in the flow job-shop scheduling problems (Kim *et al*., 2008). Here, we propose a new dispatching rule called the Scheme Choice Rule (SCR) to be a decision maker for the scheme choice to select the resources for tasks.

PROBLEM STATEMENT

The TTSP has a finite set of tasks with various schemes to be processed on several unrelated resources. There are two major considerations in this problem. First, the resources applied to complete a task are selected from multiple schemes for this task. Different schemes for a task include different resources and there is no relationship between the schemes for different tasks. Second, a task may be accomplished with several resources which are not only used for that particular task. Some tasks may not be able to start until previous tasks are finished, because tasks may have resource conflicts. These conflicts may results in idle and waiting time. In summary, the task sequence interacts with the scheme choices which determine the resources. They are the main characteristics of TTSP and the aim of TTSP is to arrange a sequence of tasks, scheduled with certain schemes, to minimize the total time by decreasing the idle time of equipment. A base model is given here which lays a foundation for further research.

A TTSP problem with instance N×M has N independent tasks and M resources. A task sequence $TP = \{t_1, t_2, \ldots, t_N\}$ has a particular scheme sequence $SC = \{s_1, s_2, \ldots, s_M\}$. The task t_i which is the I_{ti}-th($1 \le I_{ti} \le N$) task in TP has w_i schemes. A scheme $s_i (1 \le s_i \le w_i)$ of task t_i can be processed on equipment set

R_{t_i,s_i} with a processing time of T_{t_i,s_i}. Some additional indices are introduced for the TTSP. The collection of tasks requiring the same equipment as t_i under scheme s_i is:

$$\Re(t_i, s_i) = \{(t_j, s_j) \mid R_{t_j,s_j} \cap R_{t_i,s_i} \neq \varnothing,\ j \neq i, 1 \leq j \leq N\} \quad (1)$$

The collection of tasks which are scheduled before t_i is:

$$P(t_i, s_i) = \{(t_j, s_j) \mid I_{t_j} < I_{t_i},\ j \neq i, 1 \leq j \leq N\} \quad (2)$$

The collection of tasks which has the precedence constraint relationship with t_i is:

$$L(t_i, s_i) = \{(t_j, s_j) \mid t_j \succ t_i,\ 1 \leq j \leq N\} \quad (3)$$

and hence, the collection of tasks satisfying precedence constraints with t_i and taking place prior to t_i which have a resource conflict with t_i is:

$$\Psi(t_i, s_i) = P(t_i, s_i) \cap \Re(t_i, s_i) \cap L(t_i, s_i) \quad (4)$$

The completion time of task t_i is:

$$C_{t_i,s_i} = \max\left(C_{t_j,s_j} + T_{t_i,s_i}\right), t_j \in \Psi(t_i, s_i) \quad (5)$$

The objective of the TTSP is to minimize the maximum processing time of all tasks:

$$Z = \min C_{max} = \min\left\{\max\left\{C_{t_i,s_i}\right\},\ t_i \in TP,\ 1 \leq i \leq N\right\} \quad (6)$$

subject to the following conditions:

- Different tasks should not be scheduled to use the same resource at the same time:

$$R_{t_i,s_i} \cap R_{t_j,s_j}|_c = \varnothing \quad (7)$$

where, c is a time constant.

- The start time of each task must be greater than zero:

$$C_{t_i,s_i} \geq T_{t_i,s_i} \quad (8)$$

METHODS FOR CONSTRAINTS IN TTSP

The Priority-based (PB) encoding and the Scheme Choice Rule (SCR) are applied, respectively to solve the temporal constraints and resource constraints for the constrained TTSP. In the meantime, the task sequence and the scheme choice are decided during the process. They compound together called the PBSCR.

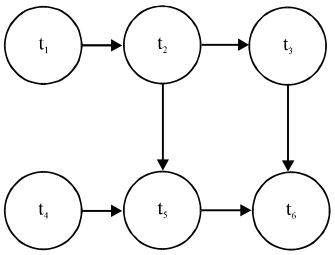

Fig. 1: Directed graph for the tasks

Table 1: Matrix for the net-constraints of tasks

	Task					
Task	t_1	t_2	t_3	t_4	t_5	t_6
t_1	0	1	0	0	0	0
t_2	0	0	1	0	1	0
t_3	0	0	0	0	0	1
t_4	0	0	0	0	1	0
t_5	0	0	0	0	0	1
t_6	0	0	0	0	0	0

Priority-based encoding for task constraints: In order to produce feasible solution, the temporal constraints among tasks are the first concern. The temporal constraint is usually illustrated by the directed graph G = (υ, ε). The vertices collection v in the G are all the processing tasks while the edge collection ε concludes all the precedence relation among them. To represent the precedence relations between task t_i and t_j in TTSP:

$$d_{ij} = \begin{cases} 1, & \text{if } t_i \text{ is prior before } t_j \\ 0, & \text{otherwise.} \end{cases}$$

is used. They form a precedence matrix D = [D_{ij}]. Then, the precedence relationship with six tasks is shown in Fig. 1 and the precedence matrix is shown in Table 1. This is such a simple constraint that we can give all its feasible by enumerating. For complex constraints, it is impossible to enumerate due to the huge number of all feasible. Hence, the priority-based encoding is expected to produce feasible solutions in the net-constraints.

As it is known, a number in a sequence is characterized by two factors: locus, the position of the number within the structure of sequence and allele, the value the number takes (Gen *et al*., 2006). Here, the locus represents the number of tasks and the value performs the priority for each task. As is shown in Table 2, the task t_1 has the priority value 3. Since the priority value of each task is given, the task arrangement which satisfies the

constraints can be obtained. It uses topology sorting to generate all the possible solutions for the given problem. The task with the highest priority and no precedence constraints is select to TP and the successor constraints of this task is deleted. The pseudo code of the priority-based encoding is shown in Fig. 2.

For example, the tasks in Table 2 obey the rules and scheduling in the procedure in Fig. 3. For the constraints shown in Fig. 3a, t_1 and t_4 have no predecessors and their corresponding priority values are 3 and 5, respectively. t_4 is selected firstly for a feasible sequence because its priority is higher than that of task 1. After selecting t_4, we can remove it along with all outgoing arcs as shown in the situation (b) of Fig. 3. t_1 becomes the only one which has no predecessors and following behind t_4. Accordingly, the feasible sequence for all tasks TP = [4 1 2 5 3 6] is obtained.

The priority-based encoding avoids blind trails and ensures feasible solutions for the net-constraints. In addition, the general operators in heuristic algorithms can be directly applied since the encoding hides the constraints in certain aspect. For example, the genes of each chromosome in the Genetic Algorithm (GA) for the TTSP are the priority values for the tasks. The only modification for the classical GA is the calculation of the fitness values for each chromosome.

Table 2: Priority for each task

Task	1	2	3	4	5	6
Priority	3	6	2	5	4	1

K = 1, S $\{t_i|L(t_i, s_i) = \phi,\ I = 1,...,N\}$
while $S \neq \phi$
select the $t_i \in S$ with the highest priority into the TP delete t_i from S and clear the constraints from the t_i
k = k+1
Endwhile

Fig. 2: Pseudocode of the priority-based encoding

Dispatching rule for resource constraints: The resource constraints for the TTSP are caused by the limited resources for multiple tasks. In general, few tasks can monopolize certain resource during the whole test process. For each task the resource distribution is determinate by the scheme choice. Hence, the resource constraint problem is equal to the scheme choice problem in the TTSP. To address the scheme choice of a fixed task sequence, the SCR is created.

The SCR utilizes scheduling knowledge to achieve a good performance. It is designed to find the proper scheme for the current scheduling task with the smallest completion time of all scheduled tasks. Before the rule is applied, each scheme combination should be considered in theory to obtain a proper scheme choice for a fixed task sequence and the numbers is as much as the multiplication of all scheme numbers. The amount of computation is 2^{20} if the total number of tasks is 20 and each task has 2 schemes. If the SCR rule is adopted, the amount decrease to 2×20. It clearly shows that the rule decreases the work load. At the same time, it also helps to solve the code problem of the TTSP. We can ignore scheme choice when finding a task sequence due to the existence of the SCR. The scheme choice becomes relatively independent of the task sequence. It is a big process since the problems like TTSP which has to

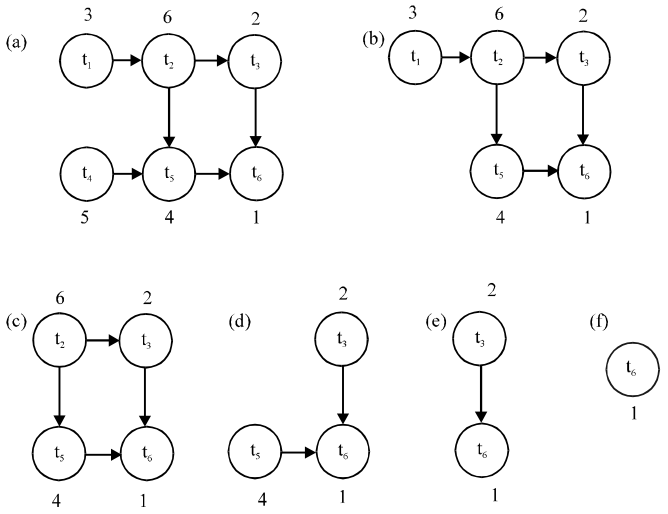

Fig. 3(a-f): Priority-based representation procedure, (a) Initial graph, (b) t_4 removed, (c) t_1 removed, (d) t_2 removed, (e) t_5 removed and (f) t_3 removed

Table 3: Test schemes for test tasks in 4×4

Task	Scheme	Resource	Time
t_1	1	r_1, r_2	5
	2	r_2, r_4	3
t_2	1	r_1	6
	2	r_3	2
t_3	1	r_4	8
t_4	1	r_1, r_3	4
	2	r_2, r_4	11
	3	r_2, r_3	4

consider quite different sub-problems commonly use a two-dimensional code such as the packing problem (Matayoshi, 2010). They have to design special operators suitable for the heuristic algorithm. Instead of quite complex improvement of algorithm, the independence makes little change of the algorithms. The problem-specific rule to determine a SC for a TP is shown below:

- The first task in TP is to choose the scheme which has the minimum time
- The following task t_i in TP has scheme s_i which ensures that:

$$z = \max\left\{C_{t_j,s_j} \mid_{t_i,s_i}, 1 \le I_{t_j} \le I_{t_i}, 1 \le s_j \le w_j, 1 \le s_i \le w_i\right\}$$

  has a minimum value. The first scheme satisfying the general limitation is chosen if there exist alternatives which also have minimum z value

Following the above criteria a proper scheme choice for a task sequence can be obtained. To demonstrate how the SCR works, the instance 4×4 is given in Table 3. A proper scheme sequence is found for a fixed task ordering TP = {1 4 2 3} in the following steps.

The first task to be executed is t_1. According to the second criterion, since the second scheme has a shorter processing time than the first scheme, the scheme of task t_1 is 2. At this point, SC = {2 / / /}. t_4 selects one scheme from among the three possibilities. If the first scheme is chosen, t_4 can start at zero and its completion time is 4 units of time. If we select the second scheme, t_4 must wait until t_1 release the resources r_2, r_4. It then finishes at 14 units of time. The completion of t_4 becomes 7 units of time if the last scheme is adopted. Accordingly, the first scheme is the best according to the third criterion. Therefore, SC = {2 1 / /}. The other task t_2 operates in an identical manner to t_4. t_3 has only one choice of scheme. From the fourth criterion, the scheme number of t_3 is therefore, 1. We therefore, finally conclude that:

SC = {2 1 3 1}

The resource for the TTSP is circuitous obtained by the scheme choice of tasks, hence the resource constraints is solved by the problem-specific SCR. Then all the constraints of the TTSP are finished by the cooperation of the Priority-Based (PB) encoding and the SCR, PBSCR in short. The method introduced here can be used on most single objective optimization methods, such as evolutionary algorithm, the taboo algorithm, the simulated annealing.

EXPERIMENTS

To evaluate the efficacy of PBSCR, we experimentally compare PBSCR to a number of state-of-the-art approaches in this section. We first provide two benchmarks and then the comparisons with the optimization branch and satisfaction methods to verify the effectiveness of the PBSCR for the constraints in the TTSP.

In the following experiments, the PBSCR is embedded into Genetic Algorithm (GA) called GAPBSCR. In the aforementioned introduction, the GA to handling the constraints is briefly introduced. The gene of the chromosome is the priority value of each task. After the initialization, they continue to crossover and mutate. When calculating the fitness of each individual to select the next generation, the SCR is applied to get certain resources. The fixed task sequence must have their corresponding scheme choice. The genetic operators stay the same with the classical GA.

Test problem: The 20×8 in Lu *et al.* (2013) is abstracted from a missile system. Low pressure test, high pressure test, frequency test and time test are applied to analysis the technical indicators in the missile system. The parameters measured contain voltage, current, frequency, power and time and so on. They are decomposed into 20 tasks based on the test requirements while the test equipments are presented by the resources. The temporal constraints with the total number of 25 among these tasks are shown in Fig. 4.

In general, the number of tasks is at most 40. The more detailed decomposition affects the independence of tasks and produce unnecessary constraints. Hence, the instance 40×12 is produced as a large bench mark for the TTSP in Table 4 and constraints shown in Fig. 5. In Table 4, J, S, S and T respresent the tasks, the scheme, resource and time respectively in instance 40×12. The number of constraints is up to 44. These instances are used for experiments.

Experiments with the optimization methods: The optimization methods such as the penalty function and

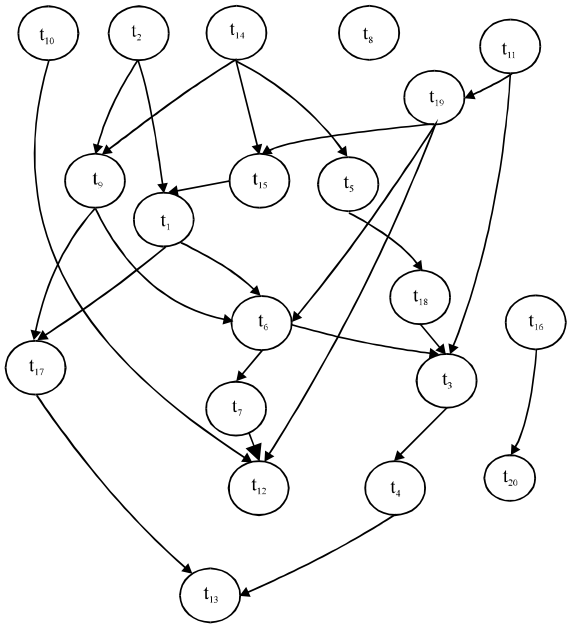

Fig. 4: Temporal constraints among tasks in instance 20×8

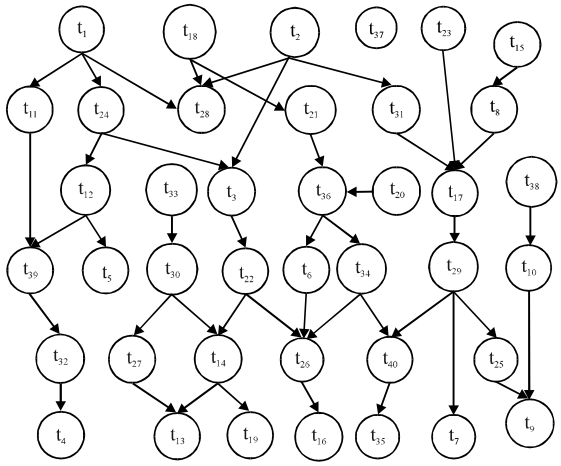

Fig. 5: Temporal constraints among tasks in instance 40×12

multiobjective optimization methods have been compared with the GAPBSCR. The penalty function is set due to the problem specialty and the NSGA-II is a representation of multiobjective optimization algorithm:

- The comparison with the penalty function

The Penalty Function (PF) is embedded into the same GA as the GAPBSCR and called GAPF but the fitness function is different due to the effect of penalty function:

Table 4: Test schemes for test tasks in 40×12

J	S	R	T	J	S	R	T
t_1	1	r_1, r_7	5	t_{21}	1	r_1, r_4	2
	2	r_3, r_5	5		2	r_3, r_5	5
	3	r_6, r_{10}	4		3	r_6, r_8	3
t_2	1	r_2, r_{11}	5	t_{22}	1	r_2	3
	2	r_4, r_9	4		2	r_4	4
	3	r_5, r_6	6		3	r_6	3
	4	r_3, r_7	4		4	r_{10}	4
t_3	1	r_3	7	t_{23}	1	r_3	5
	2	r_{12}	5		2	r_{12}	5
t_4	1	r_9	25	t_{24}	1	r_4	14
	2	r_{10}	22		2	r_{11}	17
t_5	1	r_{12}	14	t_{25}	1	r_7	19
t_6	1	r_1, r_4	7	t_{26}	1	r_1, r_4	7
	2	r_3, r_7	8		2	r_3, r_7	8
	3	r_6, r_8	8		3	r_6, r_8	10
t_7	1	r_1, r_2	4	t_{27}	1	r_1, r_2	2
	2	r_3, r_8	2		2	r_1, r_7	2
	3	r_7, r_{11}	3		3	r_3, r_8	4
t_8	1	r_1, r_3	5	t_{28}	1	r_1, r_3	5
	2	r_6, r_{10}	4		2	r_4, r_5	4
	3	r_7, r_{12}	7		3	r_7, r_{12}	2
t_9	1	r_1, r_4	11	t_{29}	1	r_3, r_4	11
	2	r_7, r_9	13		2	r_3, r_4	15
	3	r_8, r_{11}	12		3	r_7, r_8	12
t_{10}	1	r_2	9	t_{30}	1	r_1	9
	2	r_4	10		2	r_4	12
	3	r_{10}	10		3	r_{12}	10
t_{11}	1	r_2, r_7	6	t_{31}	1	r_2, r_3	6
	2	r_3, r_{12}	9		2	r_5, r_{11}	8
	3	r_8, r_9	8		3	r_6, r_9	8
t_{12}	1	r_2	11	t_{32}	1	r_2	11
	2	r_5	13		2	r_5	13
	3	r_{11}	15		3	r_6	17
t_{13}	1	r_2	4	t_{33}	1	r_2	6
	2	r_8	5		2	r_6	5
	3	r_9	7		3	r_{11}	4
t_{14}	1	r_3	7	t_{34}	1	r_3	7
	2	r_{11}	10		2	r_7	8
	3	r_{12}	8		3	r_{12}	10
t_{15}	1	r_{12}	2	t_{35}	1	r_9	2
t_{16}	1	r_2	9	t_{36}	1	r_2	9
	2	r_5	7		2	r_5	7
	3	r_8	6		3	r_{10}	6
t_{17}	1	r_1, r_{10}	10	t_{37}	1	r_1, r_2	10
	2	r_5, r_9	12		2	r_7, r_{11}	7
	3	r_{11}, r_{12}	11		3	r_5, r_{12}	11
t_{18}	1	r_6	15	t_{38}	1	r_{10}	15
t_{19}	1	r_2	8	t_{39}	1	r_4	8
	2	r_5	7		2	r_6	7
	3	r_{10}	7		3	r_9	7
	4	r_{12}	6		4	r_{10}	6
t_{20}	1	r_3	6	t_{40}	1	r_3	6
	2	r_6	4		2	r_6	5
	3	r_9	5		3	r_9	5

$$f=\begin{cases} Z, \text{ feasible solution} \\ Z'+\sum_{i=1}^{N}\alpha(Vio_i), \text{otherwise.} \end{cases} \quad (9)$$

where, Z is actual objection function value if the solution is feasible, Z' is the objection function value under no constraints if the solution is unfeasible, Vio_i is the constraint violation of task t_i from the feasible region and

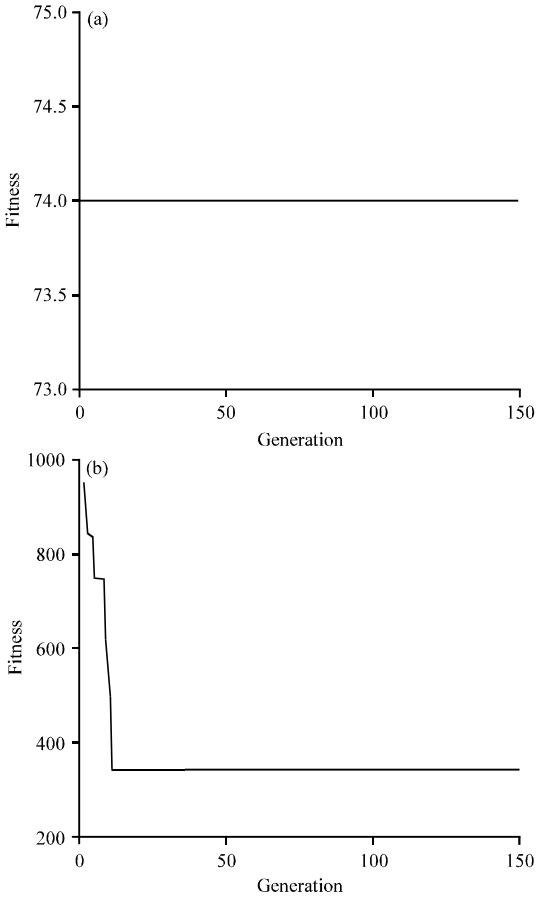

Fig. 6(a-b): Convergence curve of (a) GAPBSCR and (b) GAPF in instance 20×8

α = 100 is constant variable. The constant variable is set based on magnitude level of the maximum processing time.

The new objective considers the computation of objective function for the unfeasible solutions. The original objective for the unfeasible solution cannot be achieved since the constraint violation is in conflicts with the starting time for each task. For example, six tasks have the constraints shown in Fig. 1. The task sequence of one unfeasible solution is TP = [5 1 2 4 3 6]. How to calculate the starting time of t_4 since t_5 violates the constraints and executes before t_4? Based on this consideration, Z is replaced by Z'. Then the new objective function becomes the sum of the maximum processing time without constraints and those violations with proper weight vectors.

The comparisons of GAPBSCR, GAPF are shown in Table 5 and the convergence curves of them are shown in Fig. 6, the left for the PBSCR and the right for the penalty function.

The instance 20×8 has the only critical path $t_{14}\rightarrow t_5\rightarrow t_{18}\rightarrow t_3\rightarrow t_4\rightarrow t_{13}$ and t_5 and t_{18} are the bottleneck in this path. It is obvious in the Gantt chart for the optimal solution in Fig. 7. Hence, the solutions which tasks have the relationship have great possibility to become optimal ones. The existence of the critical path makes the problem quite easy to deal with. That is the reason why the GAPBSCR functions well and find the optimal solutions in each run when dealing with the instance 20×8. However, the example is still hard for the penalty functions. From the results, the penalty function does not find the feasible solutions for the instance 20×8.

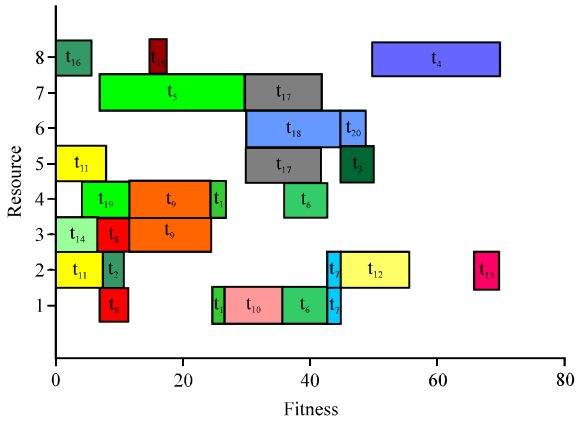

Fig. 7: Gantt chart for the instance 20×8

Table 5: Comparison of GAPBSCR, GAPF

	Instance 20×8		Instance 40×12	
Variables	GAPBSCR	GAPF	GAPBSCR	GAPF
f	74	337	69	1147
Vio_i	0	3	0	10

The instance 40×12 is a general example without the only critical paths. The convergence curves of GAPBSCR and GAPF for instance 40×12 are shown in Fig. 8a and b, respectively. The fitness for the GAPF is quite larger than that of the GAPBSCR since all solutions are infeasible ones.

- Comparison with the multiobjective optimization methods

The multiobjective optimization method compared to the GAPBSCR is NSGA-II which is proposed by Deb *et al.* (2002). Later it is adopted to deal with the constraints (Chitra *et al.*, 2011). The random key is used as the encoding methods to express the task sequence and scheme choice at the same. The encoding method turns the solution space of TTSP from a discrete space to a continuous one and makes it possible to apply the NSGA-II to the TTSP. The two objectives of the NSGA-II in the TTSP are the maximum processing time and the number of constraint violation. The basic framework for the NSGA-II is of no change.

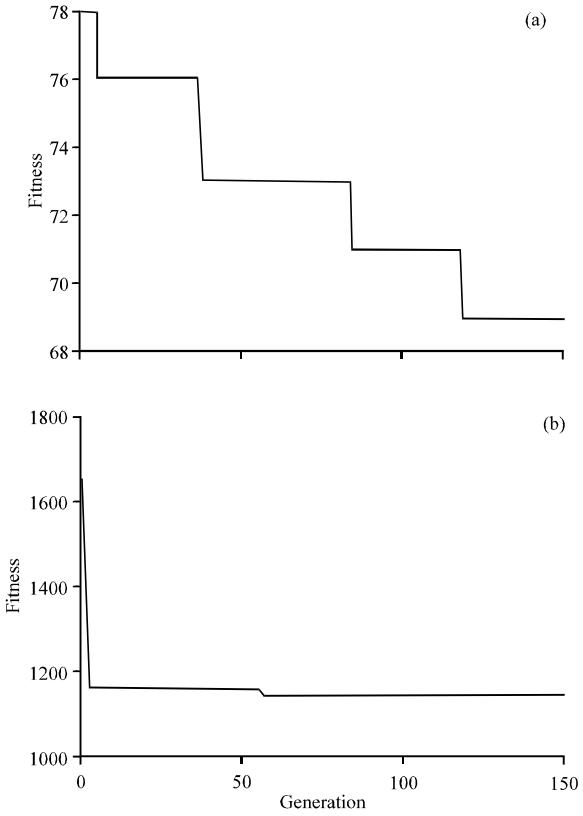

Fig. 8(a-b): Convergence curve of (a) GAPBSCR and (b) GAPF in 40×12

The results of GAPBSCR are given before when compared with the GAPF. The pareto-optimal fronts of NSGA-II for the instance 20×8 in Fig. 9a and instance 40×12 in Fig. 9b, respectively shown. The penalty function could not find the optimal solutions. The pareto-optimal fronts consist of several points instead of a curve in continuous optimization problem. The maximum processing time for the optimal solutions on the pareto-optimal front in NSGA-II are far greater than that in GAPBSCR. The solutions on the front are all infeasible. For the special instance 20×18, the best solution of NSGA-II is still has 6 constraints violations.

The optimal solution for the GAPBSCR has the Gantt chart in Fig. 10 with the maximum processing time equal to 69 for intance 40×12. The objective obtained by GAPBSCR is far more similar than that by the NSGA-II. In addition, the population and generation in the NSGA-II are at least 5 times upon those in the GAPBSCR.

Just as we suppose, the optimization methods has difficulty in finding the optimal solutions. From the experiments, GAPBSCR outperforms far better than the NSGA-II and GAPF to deal with the constraints in TTSP. Then, the branch of optimization methods for the TTSP is ignored.

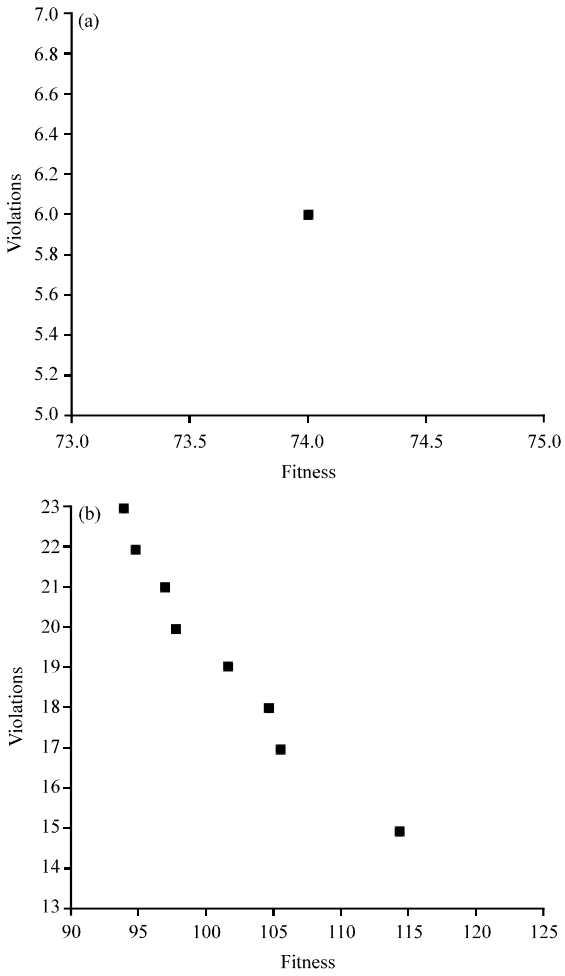

Fig. 9(a-b): Pareto-front curves of the NSGA-II for two instances, (a) 20×8 and (b) 40×12

Comparisons with the satisfaction methods: The repair operators and the backtracking method are analyzed in theory when comparing with the PBSCR. The theoretical analysis is enough to verify the competitiveness of PBSCR.

The repair operators are excluded at the concept stage since the priority knowledge is not obvious. In addition, from the experiments of the optimization methods, it is hard to get the feasible solution by randomly generated as an initial population. The multiobjective optimization methods are such convincing examples.

The backtracking method can be realization for the net constraints among tasks in the TTSP while the SCR is still used to handle the resource constraints. In the worst situation, its complexity for searching of feasible solutions can turn to the index level. However, the priority-based encoding method is only O(n+e) while n is the number

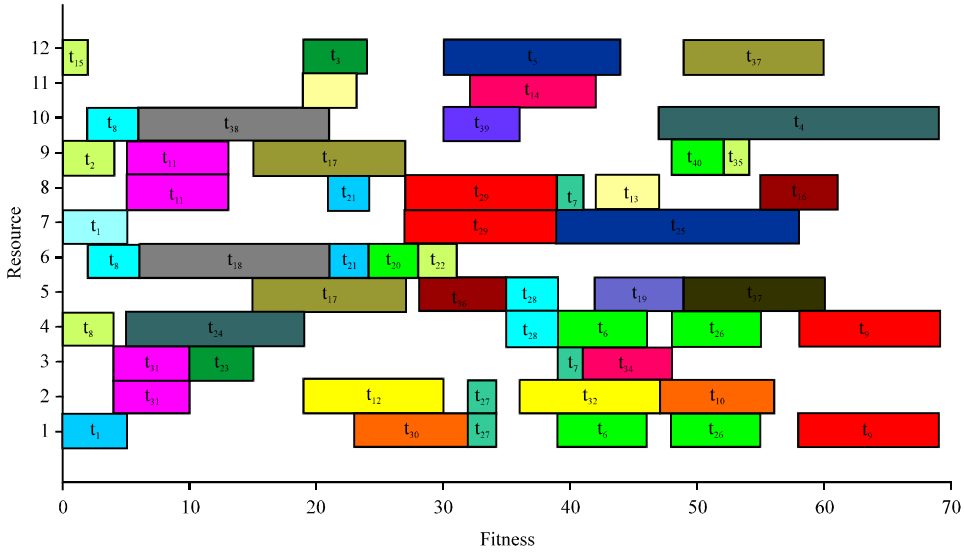

Fig. 10: Gantt chart for the instance 40×12

of all tasks and e is the number of constraints among tasks. Only in the comparison of complexity, the backtracking method fails to the PBSCR. In addition, the backtracking method is often used in tree constraints while the priority-based encoding is good at net constraints.

CONCLUSION

In this study, we proposed a constraint handling method called PBSCR for the TTSP. PBSCR is motivated by the weakness of current methods to deal with the constraints when applied to the TTSP. PBSCR tries to obtain the task sequences which satisfy the temporal constraints by priority-based encoding and arrange the corresponding scheme choice for each task according to the SCR. PBSCR employs a genetic algorithm to find optimal solution for the TTSP. To verify the efficacy of PBSCR, we embedded it in GA and tested the new algorithm on two benchmarks. Experimental results showed that the PBSCR provided very competitive results in comparison to a number of state-of-art approaches for both temporal and resource constraints together. Furthermore, the PBSCR is independent of the method generating the feasible solutions and can be easily embedded in any method utilizing the hierarchical approaches with integrated encoding. Therefore, we believe that the PBSCR would be an effective supplementary when dealing with the temporal and resource constraints together.

ACKNOWLEDGMENTS

This study is supported by the National Natural Science Foundation of China under Grant No. 61101153 and the National 863 Hi-Tech R and D Plan under Grant 2011AA110101.

REFERENCES

Bartak, R., M.A. Salido and F. Rossi, 2010. New trends in constraint satisfaction, planning and scheduling: A survey. Knowl. Eng. Rew., 25: 249-279.

Chitra, P., R. Rajaram and P. Venkatesh, 2011. Application and comparison of hybrid evolutionary multiobjective optimization algorithms for solving task scheduling problem on heterogeneous systems. Applied Soft Comput., 11: 2725-2734.

Chootinan, P. and A. Chen, 2006. Constraint handling in genetic algorithms using a gradient-based repair method. Comput. Oper. Res., 33: 2263-2281.

Coello, C.A.C., 2000. Constraint-handling using an evolutionary multiobjective optimization technique. Civ. Eng. Environ. Syst., 17: 319-346.

Coello, C.A.C., 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Comput. Methods Applied Mech. Eng., 191: 1245-1287.

Comon, H., M. Dincbas, J.P. Jouannaud and C. Kirchner, 1999. A methodological view of constraint solving. Constraints, 4: 337-361.

Deb, K., A. Pratap, S. Agarwal and T. Meyarivan, 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput., 6: 182-197.

Gen, M., F. Altiparmak and L. Lin, 2006. A genetic algorithm for two-stage transportation problem using priority-based encoding. OR Spectrum, 28: 337-354.

Haddad, O.B. and M.A. Marino, 2007. Dynamic penalty function as a strategy in solving water resources combinatorial optimization problems with honey-bee mating optimization (HBMO) algorithm. J. Hydroinform., 9: 233-250.

Harada, K., J. Sakuma, I. Ono and S. Kobayashi, 2007. Constraint-Handling Method for Multi-objective Function Optimization: Pareto Descent Repair Operator. In: Evolutionary Multi-Criterion Optimization, Obayashi, S., K. Deb, C. Poloni, T. Hiroyasu and T. Murata (Eds.). Springer, Berlin, Heidelberg, pp: 156-170.

Kim, I., J. Watada and I. Shigaki, 2008. A comparison of dispatching rules and genetic algorithms for job shop schedules of standard hydraulic cylinders. Soft Comput., 12: 121-128.

Lombardi, M. and M. Milano, 2012. Optimal methods for resource allocation and scheduling: A cross-disciplinary survey. Constraints, 17: 51-85.

Lu, H., R. Niu, J. Liu and Z. Zhu, 2013. A chaotic non-dominated sorting genetic algorithm for the multi-objective automatic test task scheduling problem. Applied Soft Comput., 13: 2790-2802.

Matayoshi, M., 2010. Double chromosome GA with Corner Junction for solving the 2D strip packing problem. Proceedings of the 36th Annual Conference on IEEE Industrial Electronics Society, November 7-10, 2010, Glendale, Arizona, pp: 1110-1116.

Ross, W.A., 2003. The impact of next generation test technology on aviation maintenance. Proceedings of the IEEE Systems Readiness Technology Conference, September 22-25, 2003, Anaheim, CA, USA., pp: 2-9.

Sels, V., N. Gheysen and M. Vanhoucke, 2011. A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions. Int. J. Prod. Res., 50: 4255-4270.

Venkatraman, S. and G.G. Yen, 2005. A generic framework for constrained optimization using genetic algorithms. IEEE Trans. Evol. Comput., 9: 424-435.

Wang, J., 2005. Constraint-based schedule repair for product development projects with time-limited constraints. Int. J. Prod. Econ., 95: 399-414.

Wang, Y., Z. Cai, G. Guo and Y. Zhou, 2007. Multiobjective optimization and hybrid evolutionary algorithm to solve constrained optimization problems. IEEE Trans. Syst. Man Cybern. B, 37: 560-575.

Zahn, M. and W. Hower, 1996. Backtracking along with constraint processing and their time complexities. J. Exp. Theor. Artif. Intell., 8: 63-74.