Parallel Test Scheduling based on Particle Swarm Optimization

Zhongwen Li [1, a], Xiangmiao Huang [2, b]

[1]College of Polytechnic,Hunan Normal University, Changsha, 410081, China
[2] The Third Xiangya Hospital, Central South University, Changsha, 410013, China

[a]email: lee_zw@163.com, [b]email:huangke1982@163.com

Keywords: Parallel Testing; Particle Swarm Optimization; Task Scheduling

Abstract. For the purpose of avoiding interference between each parallel testing tasks, this paper analyzes the testing process by dividing it into testing atoms, and makes the parameter set as the basic unit for each testing atom resource allocation so as to avoid interference. By means of modeling the parallel testing and with the object of minimizing the total testing time, it puts forward the parallel testing task scheduling algorithm on basis of improved particle swarm optimization. The experimental results verify that this method can be efficiently applied in parallel testing optimal scheduling.

Introduction

Parallel testing technology belongs to next generation testing technology, and it is one of the new technologies that support NxTestATS [1,2]. In the process of parallel testing , the concurrently performed testing tasks sending commands to the same tested object, may cause different changes to the same certain tested object state parameters, and in the follow-up testing program, the correctness of the parameter may be mistakenly judged, resulting in the situation of mutual interference.

Each testing task corresponds to a group of atoms, in which the running time of each atom is $t_1, t_2, t_3 \ldots tn$. Each atom is to use parameter resource in at least one or more parameter resource allocation sets. The parallel scheduling is to satisfy the above constraints, i.e., to get parallel scheduling sequence of testing atoms without arising conflicts in parameters resource allocation set, so to make overall testing in the shortest time.

For the above problem, undirected connected graph $G=(TA,E)$ is used to describe the relationship of testing atoms, where $TA=\{ta_1,ta_2,...,ta_n\}$ is the node set for graph G; each node represents a testing atom; E is the connection set of the graph, and the connection $e \in E$ connecting the two vertices tai and taj means that tai and tajdo not affect the same parameters.

Each connection of the graph G represents that there exist parameter resource allocation set competition between the two designated corresponding testing atoms. By means of graph G, the parallel scheduling problem is turned into the order of node coloring problem.

Graph node coloring problem is a typical combinatorial optimization problem widely used in combination analysis and everyday life. Because there is no exact algorithm for polynomial time, the graph coloring is a NP problem [3]. Due to this reason, the application of the heuristic algorithm for an approximate solution becomes a realistic solution. Gao Lin [4] puts forward the DNA algorithm of graph coloring on basis of properly coding node and color according to the experiment method of the molecular biology, and Dyeing P Galinier [5] puts forward the ant colony algorithm to solve the problem, which effectively avoid the defects that heuristic search easily fall into local minimum. Celia and Adam [6] put forward the improved genetic algorithm, which improves solving performance. The above algorithms have certain effect for ordinary graph coloring obtained, but it still needs further investigation for graph node order coloring.

Algorithm Design

- Particle Encoding

For graph G nodes set $TA=\{ta_1,ta_2,...,ta_n\}$, graph node coloring code is directly adopted as the coloring code, set coloring sets in a integer, the magnitude of which represents the order of the color, i.e. a number n of nodes are colored as x_1, x_2 ,…, xn respectively, where the integer x1 represents the color of node i. The results of coloring is represented by vector $(x_1,x_2,...,x_n)^T$. If coloring solution is $(1,2,3,2,1)^T$ that means the coloring sequence is $\{ta_1,ta_5\}\rightarrow\{ta_2,ta_4\}\rightarrow\{ta3\}$, and the corresponding parallel scheduling testing atoms ta1 and ta5 execute in parallel, followed by parallel execution of and ta2 and ta4, and finally the execution of ta3 testing atom.

- Fitness function

$$\tau=\sum_{i=1}^{n}\sum_{j=i+1}^{n}h(i,j)\text{, in which } h(i,j)=\begin{cases}1 & GN^{n\times n}(i,j)=1 \text{ and } x_i=x_j\\ 0 & \text{otherwise}\end{cases} \quad (1)$$

τ represents the number by which the violation of the coloring constraints occurs, therefore, τ = 0 means every node coloration satisfies the graph order coloring constraint 1.

$$\varsigma=\sum_{i=1}^{n}\sum_{j=i+1}^{n}g(i,j)\text{, in which } g(i,j)=\begin{cases}1 & AS^{n\times n}(i,j)=1 \text{ and } x_i>x_j\\ 0 & \text{otherwise}\end{cases} \quad (2)$$

ς represents the number by which the violation of the coloring order constraints occurs.

Assign *k* as the number of colors used to render graph *G*, where the set of testing atoms marked by the *i*-th color is denoted as $CTA_i=\{ta_1^i,ta_2^i,...,ta_p^i\}$, and the testing time set that composed of testing time corresponding to each testing atom is denoted as $TT_i=\{t_1^i,t_2^i,...,t_p^i\}$, and the required testing time for parallel executing all the testing atoms in set CTA_I is $\sum_{i=1}^{k}\min(TT_i)$, and the total testing time required for sequence testing is $\sum_{i=1}^{n}t_i$, so define the speedup rate λ as:

$$\lambda=\sum_{i=1}^{n}t_i/\sum_{i=1}^{k}\min(TT_i) \quad (3)$$

According to equation (1) (2) (3), construct the fitness value function:

$$f=A(\tau+\varsigma)+1/\lambda \quad (4)$$

For fitness value *f*, the smaller the value of it, the better the location of the particle, where *A* is the penalty factor used for adjusting the proportion of graph node sequential coloring constraint 1 and constraint 2 in particle fitness value.

- Speed formula

Make use of *k* colors to render the graph *G*, then $k\le n$, in which *n* denotes the number of nodes, so that the worst situation is when $k=n$, all the testing atoms are executed in a manner of sequence testing. Therefore, it is necessary to make an appointment on the value of each element of the coloration vector. Assign the color set limited to *CSet= {0,2,…,n-1}* , when update the position in accordance with the speed formula, the value of *j*-dimensional component of particle *i* must be limited within the color set, so that assign $\mathrm{mod}(m,n)$ to represent modulus operation, and *abs()* to represent absolute value operation, *int()* to represent integer operation, in order to define the compound operation as $bind(x)=\mathrm{mod}(abs(\mathrm{int}(x)),n)$, where *n* is the number of nodes of graph *G*.

The speed formula of *j*-dimensional component of particle *i* is:

$$v_{ij}^{\kappa+1}=bind(\omega v_{ij}^{\kappa})+bind(c_1r_1(p_{ij}-d_{ij}))+bind(c_2r_2(p_{gj}-d_{ij})) \quad (5)$$

And the position update formula is:

$$x_{ij}^{\kappa+1}=bind(x_{ij}^{\kappa}+v_{ij}^{\kappa+1}) \quad (6)$$

In (5), *p* is the current particle's best experienced position, p_g is the global best position. About

the inertia weight ω,

$$\omega(t) = 0.9 - t / MI \times 0.5 \quad (7)$$

In (7), *MI* is the largest evolution generation, and *t* is the current evolution generation. Iteration of the particles according to (6) makes the group search have a good ability of space expansion at the beginning, and along with the execution of iteration, the pace of iterative search continues to decrease, which has effectively improved the algorithm convergence speed.

Learning factors $c_1 = c_2 = 2$.

- SA local optimization

The specific method of combining SA local optimization and PSO algorithm is to perform a simulated annealing operation upon the new locations of the particles again after the particles update their positions by the speed formula. And the initial temperature of annealing is denoted as ε_0, cooling coefficient is denoted as μ, $0 < \mu < 1$, and cooling operation is denoted as $\varepsilon_{k+1} = \mu\varepsilon_k$, so that assign the variation between fitness value of the particle's current position and the previous location to be ∇f, and for probability $p = \min(1, \exp(-\nabla f / \varepsilon_k))$, if $p \geq random[0,1]$, then accept the new position, otherwise reject the new position, and maintain the original position. From the annealing process it can be seen that the particle position accepts child individual as new individual according to *Metropolis* criterion, and in addition to optimal solution, a certain level of deterioration solution is also accepted in order to improve the diversity of particle swarm, moreover, with the continuous decrease of ε, the probability of accepting deterioration solution also decreases, and finally there is not any deterioration solution is accepted as ε tends to zero, and thus the algorithm escapes from the local optimum trap, and finally gets the global optimal solution.

- PSO-SA algorithm

In light of the above mentioned, the specific steps of algorithm is as follows:

Step 1 Assign initial parameters, including the maximum evolution algebra *MI*, group scale m, annealing initial temperature ε_0 and temperature reducing coefficient μ, etc.;

Step 2 Set $AS^{n\times n}$ according to the testing timing sequence constraint relation between atoms, and set $GN^{n\times n}$ corresponding to map G according to the competition relationship of parameter resource distribution sets;

Step 3 Make $\kappa = 1$, generate initial partial positions as many as m according to the above mentioned greedy algorithm, and set each particle a initial speed randomly;

Step 4 Set a fitness value for each particle according to (4) and update p and p_g according to the fitness value;

Step 5 Renew the speed of the particles according to (5), update the particle's position according to (6) , and color revise the new position of the particles.

Step 6 Calculate fitness value of the particle in new position so to reach fitness variation ∇f between the old and the new position, $p = \min(1, \exp(-\nabla f / \varepsilon_k))$, if $p \geq random[0,1]$, the new position is accepted, if not, keep the old position unchanged;

Step 7 If $\kappa > MI$ or the position of each particle remain unchanged after 10 consecutive generations, turn to Step 8; Otherwise make $\kappa = \kappa + 1$ and $\varepsilon_{k+1} = \mu\varepsilon_k$, turn to Step 4;

Step 8 Stop operations, return to p_g as algorithm output.

Test results

For checking the effectiveness of the scheduling algorithm, corresponding data for a particular issue are given in [7] .

As for the choices of experiment parameters, the swarm size is 20; the maximum evolution algebra is 50, the annealing initial temperature ε_0 is 1; the temperature reducing coefficient μ is 0.95; penalty coefficients A is 2.

Three experimental algorithm are used. They are the TaskScheduler-T algorithm described in [7], the standard particle swarm optimization (PSO) and PSO-SA algorithm proposed by this paper.

Each algorithm runs 20 times.

As shown in table 1, the experiment results indicates that TaskScheduler-T algorithm randomly generates feasible testing scheduling sequence in parallel, and gets the optimal solution by exhausting them. Although this method can find optimal solution each time, the time consumption by it is intolerable when the size of the problem to be solved reaches certain extent. As for the other two algorithms, the PSO-SA is superior to PSO in time consumption and precision. It can be seen from Fig. 1 that PSO-SA method finds the optimal solution in 13th generation. By contrast, the PSO method finds it in the 28th generation, which indicates that the searching capability of PSO-SA is higher than that of PSO.

Table 1. Experimental results

Algorithm	Total Testing Time	Fitness Value	Average Resolving Time (sec.)	Success rate
TaskScheduler-T	62	0.5167	3	1
PSO	59	0.4916	2.3	0.92
PSO-SA	59	0.4916	1.2	0.96

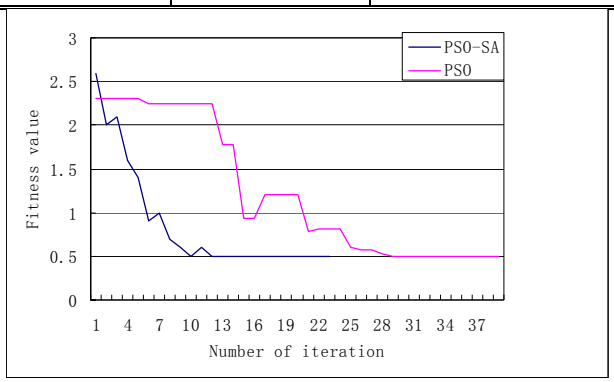

Fig.1. Fitness value tendency of PSO and PSO-SA

Conclusion

For the purpose of avoiding interference between each parallel testing tasks in test progress, this paper analyzes the testing process by dividing it into testing atoms, and makes the parameter set as the basic unit for each testing atom resource allocation so as to avoid interference. By means of modeling the parallel testing and with the object of minimizing the total testing time, it puts forward the parallel testing task scheduling algorithm on basis of improved particle group algorithm. The experimental results verify that this method can be efficiently applied in parallel testing optimal scheduling.

Acknowledgement

The work was supported by Hunan Provincial Natural Science Foundation of China (NO. 13JJ6029).

References

[1] William A Ross. The Impact of Next Generation Testing Technology on Aviati on Maintenance[A]1 AUTOTESTCON pr ceedings, IEEE, 2003.

[2] Anderson J L Jr. High performance missile testing [A]. AutotestCon Proceedings [C]. IEEE, 2003 , Page(s): 19 - 27:

[3] R M Karp. Reducibility among Combi natorial Problems[M] .Raymond E Miller and James W Thather: eds. Complexity of Computer Computations, Plenum Press, 1972. 85- 103.

[4] Gao Lin, Xu Jin. A DNA algorithm for graph node co loring problem[J] . Acta Electronica Sinica, 2003, 31(4) : 494- 497.

[5] P Galinier, J K Hao. Hybrid evolutionary algorithms for graph coloring[J] . Journal of Combinatorial Optimization, 1999, 3:379- 397.

[6] Anna M, Adam P B, Celia A G. Improve graph colouring with linear programming and genetic algorithms[A] . Proceeding of the 2000 Genetic and Evo lutionary Computation Conference [C] . Las Vegas, Nevada, USA, 2000. 240- 245.

[7] Hu Yu . Colored Petri net based modeling of parallel automatic testing systems[D]. Chengdu: School of Automation, University of Electronic Science and Technology of China, 2003.