An Efficient Quantum-Behaved Particle Swarm Optimization for Multiprocessor Scheduling

Xiaohong Kong^{1,2}, Jun Sun¹, Bin Ye¹, and Wenbo Xu¹

¹ School of Information Technology, Southern Yangtze University,
Wuxi 214122, China

nancykong@hist.edu.cn, sunjun_wx@hotmail.com, xwb@sytu.edu.cn

² Henan Institute Of Science and Technology,
Xinxiang, Henan 453003, China

Abstract. Quantum-behaved particle swarm optimization (QPSO) is employed to deal with multiprocessor scheduling problem (MSP), which speeds the convergence and has few parameters to control. We combine the QPSO search technique with list scheduling to improve the solution quality in short time. At the same time, we produce the solution based on the problem-space heuristic. Several benchmark instances are tested and the experiment results demonstrate much advantage of QPSO to some other heuristics in search ability and performance.

1 Introduction

Multiprocessor scheduling problem (MSP) is popularly modeled by a weighted directed acyclic graph (DAG) or micro-dataflow graph, and the objective of MSP is minimizing the parallel completion time or schedule length by properly assigning the nodes of the graph to the processors without violating the precedence constraints. Generally, the multiprocessor scheduling problem is NP-hard[1] except for some cases for which an optimal solution can be obtained in polynomial time. In the past decades, a myriad of heuristic algorithms have been investigated[2][3][4][5][6][7][8][9], but the results are constrained by efficiency and complexity. In this paper, the multiprocessor scheduling problem is tackled by establishing the priority list of tasks utilizing the Quantum-Behaved Particle Swarm Optimization (QPSO).

A directed acyclic weighted task graph (DAG) is defined by a collection $G = \{V, E, C, W\}$, where $V = (n_j; j = 1 : v)$ is the set of task nodes and E is the set of communication edges. The weight $c(n_i, n_j) \in C$ corresponds to the communication cost incurred while the task n_i and n_j are scheduled, which is zero if both nodes are assigned on the same processor. The weight $w(n_i) \in W$ is the execution cost of node $n_i \in V$. The edge $e(i, j) = (n_i, n_j) \in E$ represents the partial order between tasks n_i and n_j . The target system is commonly assumed to consist of p processors connected by an interconnection network based on a certain topology in which a message is transmitted through bidirectional links with the same speed.

2 Quantum-Behaved Particle Swarm Optimization

2.1 The Standard PSO

Particle Swarm Optimization (PSO)[10][11][12], originally proposed by Kennedy and Eberhart in 1995, is a novel evolutionary algorithm and a swarm intelligence computation technique. In PSO system, a particle or an individual depicted by its position vector X and its velocity vector V , is a candidate solution to the problem. To solve an optimal problem, a population of initialized solutions search through a multidimensional solution space, and each member continually adjusts its position and velocity learning its own experience and the experience of other members. Considering $D(d = 1 : D)$ dimensions as an example, the particles are manipulated according to the following formula [10][11]:

$$V_{i,d} = V_{id} + c_1 * r_1 * (P_{id} - X_{id}) + c_2 * r_2 * (P_{gd} - X_{id}) \quad (1)$$

$$X_{id} = X_{id} + V_{id} \quad (2)$$

Where c_1 and c_2 are learning factors, r_1, r_2 are random number uniformly distributed in the range $[0, 1]$. In equation(1), the vector P_i is the best position (the position giving the best evaluate value) of particle i , vector P_g is the position of the best particle among all the particles in the population.

2.2 Quantum-Behaved PSO

In 2004, J. Sun et al. proposed a Delta potential well model of PSO in quantum time-space framework[13][14]. With the quantum particles, the exact values of X and V cannot be determined simultaneously, and the quantum state of a particle is described by wavefunction $|\psi(X, t)|$. In Quantum-Behaved Particle Swarm Optimization (QPSO), the particle can only learn the probability of its appearance in position X from probability density function $|\psi(X, t)|^2$, the form of which depends on the potential field the particle lies in. The updates of the particles are accomplished according to the following iterative equations[13][14]:

$$X(t+1) = P \pm \beta * |mbest - X(t)| * \ln(1/u) \quad (3)$$

$$mbest = \frac{1}{M} \sum_{i=1}^M P_i = (\frac{1}{M} \sum_{i=1}^M P_{i1}, \frac{1}{M} \sum_{i=1}^M P_{i2}, \dots, \frac{1}{M} \sum_{i=1}^M P_{id}) \quad (4)$$

$$P = \varphi * P_{id} + (1 - \varphi) * P_{gd}, \varphi = rand() \quad (5)$$

$mbest$ (Mean Best Position) is the mean value of all particles' best positions P_i , φ and u are a random number distributed uniformly on $[0, 1]$ respectively; The only parameter in QPSO algorithm is β , called Contraction-Expansion Coefficient. As a new method, QPSO has its advantages such as simple concept, immediate accessibility for practical applications, simple structure, ease of use, speed to get the solutions, and robustness, and parallel direct search method. QPSO has successful application in optimization problem[13][14].

3 Multiprocessor Scheduling Based on QPSO

Since 1995, Particle Swarm Optimization has been reported in literature to solve a range of optimization problem[15][16][17], and scheduling is the new field of PSO algorithm in discrete space. In general, there are two forms of solution representations, namely permutation-based representation and priority-based representation[17][18].

3.1 Particle-Based Solution Representation

For the priority-based form, every dimension $d(d = 1 : D)$ represents the task index number, and the corresponding value of each element in the vector X means the priority a node is scheduled to start. We assign a higher priority to a task with a smaller element value, i.e. each task is scheduled in ascending order of each element value [18]. Fig.1 illustrates the priority-based solution representation of target individual with five dimensions: the 4th element is the smallest, so the task 4 is scheduled firstly, and so on.

Priorities vector	1.3	4.5	3.7	0.6	2.3
Task index	1	2	3	4	5
Scheduling list	4	1	5	3	2

Fig. 1. Priority-based solution representation

3.2 Permutation-Based Representation

For the permutation-based form, every dimension $d(d = 1 : D)$ in the vector means the order or sequence the node is scheduled, and the corresponding value of each element means a node index number. Used to stand for the index number of a task, the parameter of each element in the particle represented permutation should be an integer limited to $[1, D]$. For example, a permutation of 5 tasks for a scheduling can be represented as the particle in Fig. 2.

Priorities vector	4	1	5	3	2
scheduling sequence	1	2	3	4	5
Scheduling list	4	1	5	3	2

Fig. 2. Permutation-based solution representation

3.3 QPSO for Multiprocessor Scheduling

When designing the QPSO algorithm, we combine the priority-based QPSO with list scheduling. The proposed QPSO approach are adopted searching for optimal combination of the priority values. A ready list preserves the ready tasks whose parent has finished. The highest-priority ready task is removed from the ready list and selected for scheduling during the first step. Next, the task list is

dispatched to processors using *the earliest start time* rule and satisfy precedence constraints. In case the earliest time of a task on two machines is the same, the algorithm breaks the tie by matching and scheduling the task to the first machine in the sequence. Finally, updating the ready task list, until all tasks are scheduled.

In addition, an initial population is a key factor for the solution quality so we produce an efficient initial set based on specific problem. b_level and t_level are two general attributes in DAG graph. b_level is total cost(including execution cost and communication cost)from the bottom node to specified node and t_level is total cost from top node to specified node[9]. As described above, the i th element of particle represents priority of a task n_i , and the X_{0i} of first particle X_0 is a set to the b_level of a task n_i . Every dimension of the rest of the initial population is generated by a random perturbation of the values of the first particle as follows:

$$X_{ji} = X_{0i} + \text{uniform}(t_level/2, -t_level/2) \quad (6)$$

Where X_{0i} is the priority of task n_i in the first particle, $i = (1; \dots; D); j = (1; \dots; Npop - 1)$; $\text{uniform}(t_level/2, -t_level/2)$ is a random number generated uniformly between $t_level/2$ and $-t_level/2$. This strategy has a good performance in the PSGA [19]. On the basis of above, the population evolves to search the optimal solution. After every iteration, we limit the position value range to interval $[b_level + t_level, b_level]$. Here, each dimension has a different priority value for each task in different particles, so each particle guides the heuristic to generate a different schedule in this case. The Proposed QPSO algorithm for multiprocessor scheduling is depicted in Fig .3.

Initialize the population

Do {

Ngen ++

For i=1 to Npop(the particle population number)

β linearly decreases

Find out the mbest of the swarm

Obtain the update priority according to euqation(3)(4)(5)

Find possible permutation

End for

Re-evaluate the P_i and the P_g

While (the termination criterion is not met or generation j Ngen(the generation number)

Output results

Fig. 3. QPSO algorithm for MSP

4 Experimental Results

Firstly, we execute the QPSO and some other list scheduling to a set of random graphs with different parameter, including CCR , density ρ , task number n and machine number p . Some classical algorithms including HLFET [2], ISH [3], MCP

[4], ETF [5] and DLS [6] are also implemented. CCR is defined to be the ration of the mean communication to the mean computation which represents the weight of the communication to the whole computation[9]. Edge density ρ is defined as the number of edges divided by the number of edges of a completely connected graph with the same number of nodes [20]. There, $CCR \in \{0.1, 0.5, 1.0, 5.0, 10\}$
 $n \in \{20, 40, 60, 80, 100\}$

$p \in \{2, 4\}$

$\rho \in \{20, 40, 50, 60, 80\}$

For every combination (CCR, n, p, ρ) , there are 5 graphs, amount to $5(CCR) \times 6(n) \times 2(p) \times 5(\rho) \times 5(graphs) = 1500(graphs) = 1500(graphs)$. In order to evaluate the performance, we normalize the actual scheduling length according to the following formula(7)[8][9]:

$$NSL = \frac{SL}{\sum_{i \in CP} w(n_i)} \quad (7)$$

$\sum_{i \in CP} w(n_i)$ is the sum of computation time of all the CP nodes.

First, we select the graphs ($n = 100$) with different density to investigate the QPSO and other algorithms. The performance of QPSO algorithm is computed after 10 runs ($5CCR \times 5graphs$), and the results are depicted in Fig.4. In our algorithm, the parameter β is set changing from 1.2 to 0.5[13][14], which decreased linearly. From the Fig.4, an apparent conclusion is that more improvement is achieved for sparse-task graphs, while little enhancement in solution quality is observed for dense ones. It appears that dense graphs spent more time waiting for predecessor tasks to complete their execution and transferring data between related tasks scheduled on different processors.

At the same time, we test the performance of graphs with different tasks. The results are depicted in Fig.5. With more tasks, the QPSO has always the comparable performance among these algorithms, which is due to global search ability

Fig. 4. The NSL of QPSO and a few algorithms to $n=100$ with different density ($p = 2$, $N_{pon} = 20$, $N_{gen} = 30$)

of the QPSO to find near-optimal task list and the scalability. It appears that graphs with increasing tasks spent more time waiting for predecessor tasks to complete their execution and transferring data between related tasks scheduled on different processors.

Fig. 5. The NSL of QPSO and a few algorithms different tasks number ($p = 4$, $N_{pon} = 20$, $N_{gen} = 30$)

Finally, benchmark instances from website (<http://www.mi.sanu.ac.yu/tanjad>), and the results of a few algorithms, including algorithms CPES, LPTES, DC, LPTDC, LBMC, PPS are downloaded to test the algorithm. These algorithms are adjusted from other algorithms by T. Davidovic et al [20]. We select the $t200 - 80$ instances and investigate how the number of processors affects the scheduling results. The mean results of QPSO after 10 runs and other algorithms (download from <http://www.mi.sanu.ac.yu/tanjad>) are listed as Table.1 and Table.2.

From the above results, in most cases QPSO attains better results than the other algorithms whether $p = 2$ or $p = 4$, but with the increasing of machines,

Table 1. The comparison of results between QPSO and other algorithms to t200-80 instances ($p = 2$)

instance	CPES	LPTES	DC	LPTDC	LBMC	PPS	QPSO
t200-80-1	3420	3433	3464	3454	4526	3687	3351
t200-80-2	5353	5535	5630	5600	7146	5977	5359
t200-80-3	3232	3244	3363	3365	4125	3441	3226
t200-80-4	3586	3618	3804	3864	4754	3990	3580
t200-80-5	1968	1979	2057	2069	2783	2151	1934
t200-80-6	2498	2522	2525	2549	3396	2807	2485

Table 2. The comparison of results between QPSO and other algorithms to t200-80 problem ($p = 4$)

instance	CPES	LPES	DC	LPTDC	LBMC	PPS	QPSO
t200-80-1	3414	3427	3464	3453	5245	3875	3343
t200-80-2	5345	5475	5630	5581	7583	6225	5332
t200-80-3	3232	3244	3363	3365	4276	3413	3226
t200-80-4	3586	3618	3800	3684	4791	4040	3568
t200-80-5	1968	1979	2057	2069	3065	2705	1934
t200-80-6	2489	2522	2525	2549	3850	2995	2478

a little improvement is obtained. We analyze the reason behind the results: with more machines, more communications among machines are involved, the scheduling length is not proportional to machines.

5 Summary

As to be seen from the previous results of the performance-testing experiment, the performance of the Quantum Particle Swarm Optimization method in coping with scheduling problems is better than some classical algorithms in most cases. Our future work will focus on hybrid strategies of the QPSO algorithm with other optimization algorithms.

References

1. M.R. Garey, D.S. Johnson: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company (1979)
2. T. L. Adam, K. M. Chandy, J. Dickson: A comparison of list scheduling for parallel processing systems. Communication of the ACM **17** (1974) 685-690
3. B. Kruatrachue and T. G. Lewis: Duplication scheduling heuristic (DSH): A new precedence task scheduler for parallel processor systems. Technical Report, Oregon State University, Corvallis, OR 97331, (1987)
4. M.-Y. Wu, D. D. Gajski: Hypercool: a Programming Aid for Message-passing Systems. IEEE Trans. Parallel Distrib. Systems **1** (1990) 330-343
5. J. J. Hwang, Y. C. Chow, F. D. Anger, C. Y. Lee: Scheduling Precedence Graphs in Systems with Interprocessor Communication Times. SIAM Journal on Computing **18** (1989) 244-257
6. G. C. Sih, E. A. Lee: A Compile-time Scheduling Heuristic for Interconnection-constrained Heterogeneous Processor Architectures. IEEE Transactions Parallel and Distrib. Systems **4** (1993) 175-187
7. T. Yang, A. Gerasoulis: DSC: Scheduling Parallel Tasks on an Unbounded Number of processors. IEEE Transactions on Parallel and Distributed Systems **5** (1994) 951-967
8. Y.K. Kwok, I. Ahmad: A Static Scheduling Algorithm Using Dynamic Critical Path for Assigning Parallel Algorithms onto Multiprocessors. Proceedings of the 1994 International Conference on Parallel Processing (ICPP'94) **2** (1994) 155-159

9. Y.K Kwok, I.Ahmad: Benchmarking and Comparison of the Task Graph Scheduling Algorithms. *Journal of Parallel and Distributed Computing* **59** (1999) 381-422
10. J. Kennedy, R.c. Eberhart: Particle Swarm Optimization. *Proc. IEEE Conf. On Neural Network* **IV** (1995)1942-1948
11. R.C.Eberhart,J. Kennedy : A New Optimizer Using Particles Swarm Theory. *Proc. Sixth Intemational Symposium on Micro Machine and Human Science* (1995) 39-43
12. Y. Shi, R.C. Eberhart: Empirical Study of Particle Swarm Optimization. *Proc. Congress on Evolutionary Computation* (1999) 1945-1950.
13. J.Sun, B.Feng,W.B. Xu: Particle Swarm Optimization with Particles Having Quantum Behavior. *Proc. 2004 Congress on Evolutionary Computation* (2004) 325-331
14. J. Sun, W.B. Xu: A Global Search Strategy of Quantum-Behaved Particle Swarm Optimization. 2004 IEEE Conference on Cybernetics and Intelligent Systems (2004) 111-116
15. F. van den Bergh, A.P Engelbrecht: Cooperative Learning in Neural Networks using Particle Swarm Optimizers. *South African Computer Journal* **26** (2000) 84-90
16. S. C.Esquivel, C. A.Coello: On the Use of Particle Swarm Optimization with Multimodal Functions. *Proceedings of IEEE Congress on Evolutionary Computation 2003 (CEC 2003)* 1130-1136
17. H. Zhang, X.D.Li, H.Li, F.L. Huang: Particle Swarm Optimization-based Schemes for Resource-constrained Project Scheduling. *Automation in Construction* **14** (2005) 393-404
18. M.F.Tasgetiren, Y.C. Liang, M.Sevkli, G. Gencyilmaz: Differential Evolution Algorithm for Permutation Flowshop Sequencing Problem with Makespan Criterion. 4th Inter-national Symposium on Intelligent Manufacturing Systems (IMS 2004)
19. M. K. Dhodhi, Imtiaz Ahmad, A.Yatama,Ishfaq Ahmad: An Integrated Technique for Task Matching and Scheduling onto Distributed Heterogeneous Computing Systems. *Journal of Parallel and Distributed Computing* **62** (2002) 1338-1361
20. T. Davidovic, T. G. Crainic: Benchmark-problem Instances for Static Scheduling of Task Graphs with Communication Delays on Homogeneous Multiprocessor Systems. *Computers and Operations Research* **33** (2006) 2155-2177