International Journal of Grid Distribution Computing
Vol.7, No.5 (2014), pp.139-148
http://dx.doi.org/10.14257/ijgdc.2014.7.5.13

BCC-DPSO Algorithm for Task Scheduling on NOC

Wei Gao[1], Yubai Li[2], Song Chai[3] and Jian Wang[4]

School of Computer and Information Engineering, University of Electronic, Science and Technology of China Chengdu 611731, China
[1]wgeaio@163.com, [2]ybli@uestc.edu.cn
[3]s.tschai@foxmail.com, [4]wangjian3630@uestc.edu.cn

Abstract

In this paper, a BCC-DPSO scheduling algorithm is proposed to solve multi-objective optimization problem for task scheduling on Network-on-Chip (NoC). In our proposal, the relative advantage of the solution is evaluated by calculating its efficiency using BCC model in Data Envelopment Analysis (DEA), and the referred-time method is introduced to rank the BCC-efficient solution. Moreover, a sub-swarm strategy is adopted to reduce the high computational requirement introduced by the DEA. There are four sub-swarms, each of which optimizes one of four observed metrics, namely makespan, energy, link load and workload balance. Meanwhile, the speed vector updating formulation is modified to comply with the sub-swarm strategy. By conducting comparative simulations, the results show that our proposal produces more efficient schedule solution than other multi-objective Particle Swarm Optimization (PSO).

Keywords: *Task Scheduling, Sub-swarm, Efficiency, Referred-time, NoC*

1. Introduction

Recently, the Network-on-Chip (NoC) presenting itself as a reliable, reusable, energy efficient and scalable solution for complex System-on-Chip (SoCs) implementation [1]. However, the emergence of NoC also brings new challenges to the task scheduling algorithm design. Most of previous researches on the NoC scheduling algorithm design focused on reducing power [2] or minimizing the execution time of the task set [3]. In recent years, it is a clear trend of scheduling algorithm to perform multi-objective optimization [4, 5]. Since multiple metrics are taken into consideration, it is desirable for a scheduling algorithm to produce an efficient schedule solution which makes reasonable trade-offs between these metrics.

To solve the above problem, Data Envelopment Analysis (DEA) [6] is introduced into the Discrete Particle Swarm Optimization (DPSO) algorithm. In this paper, four common metrics, including makespan, energy, link load and workload balance, are extracted from the NoC task scheduling problem, and are used to construct the BCC model of schedule solutions. In the scheduling algorithm design, the swarm of DPSO is divided into four sub-swarms, and each sub-swarm optimizes a single observed metrics. In the DPSO iteration process, the BCC-efficiency of each particle is calculated and the referred-time method is introduced to select the global best particle in the whole swarm to update the speed.

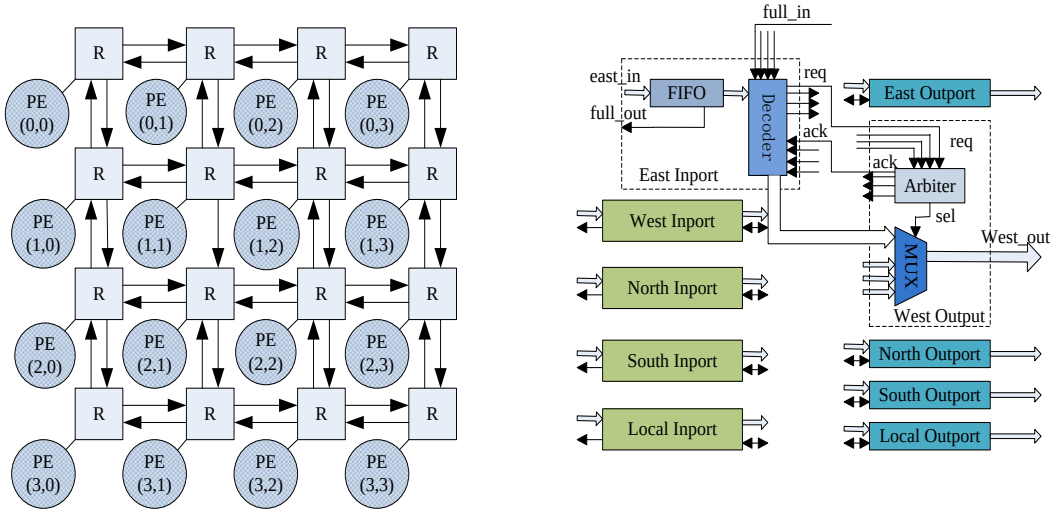
The rest of this paper is organized as follow: In section 2, the process of task scheduling on multiprocessor is formulated; Section 3 presents our proposal in detail; Section 4 shows experimental results and Section 5 concludes the paper.

ISSN: 2005-4262 IJGDC

2. Problem Formulation

2.1. Task Model

In the task scheduling, the program is represented by a Directed Acyclic Graph (DAG) expressed by $G = (V, E)$, which consists of a set V of nodes v and a set E of edges e. A node in the DAG represents a task, which must be executed sequentially without preemption in the same processor. Each edge $e_{ij} \in E$ is associated with a nonnegative c_{ij} representing the communication cost from task n_i to n_j task and note that task n_j cannot start execution until the result of task n_i has been transmitted. Moreover, the communication cost of a task $n_i \in V$ on the processor P_j is denoted as w_{ij}.



(a)a 4*4 mesh NoC based-on MPSoC (b)microstructure of NoC node

Figure 1. An Architectural Overview of an NoC-based System

2.2. NoC Structure

In our work, the hardware is a 4×4 mesh NoC-based MPSoC, as illustrated in Figure 1(a). In the graph, the processor (PE) is a homogenous processor core with local data cache and is connected to a router (R). Routers are interconnected with each other through bi-direction links.

In the router, there are five Inports and Outports corresponding to five directions of East, West, North, South and Local, In each inport module, there are five FIFOs and one decoder. The decoder performs XY routing algorithm, when it detects the head flit of a packet. Then a request signal is sent to the arbiter of the corresponding Outport. If the arbiter receives multiple request signals from different directions, it adopts Round-Robin algorithm [7] to solve the contention. Then the granted Inport forwards the packet to the ext router. Besides, the wormhole technology is adopted to minimize the buffer requirement and the packet ltency [8]. To further reduce end-to-end delay, we also employ the back pressure mechanism [9]. The microstructure of a router is shown in Figure 1(b).

2.3. Metrics

In this paper, four common metrics, namely makespan, energy [10], link load [11] and workload balance [12], are extracted from the scheduling problem on NoC, and considered as the observed metrics.

Makespan (the M metric) is the time span for the NoC to complete all tasks. Average link load (the L metric) monitors the traffic load on each link, and is defined to be the average value of traffic loads on all links.

Here, energy (the E metric) referring to the data routing energy is presented by the Bit Energy proposed in [10], and the average energy consumption of transmitting one bit from node i to node j is calculated by:

$$E_{bit}^{i,j} = n_{hops} \cdot E_{N_{bit}} + (n_{hops} - 1) \cdot E_{L_{bit}} \quad (1)$$

Where $E_{N_{bit}}$ and $E_{L_{bit}}$ represent the energy consumed on the processor and on the link respectively, and n_{hops} is the number of nodes the bit passes on its route from node i to node j.

The workload balance (the B metric) measures the inverse coefficient of the total workload on each processor. The larger load balance value suggests better balanced schedule and is calculated by:

$$workload\ balance = \frac{ave\_workload}{(\sum_{p_i \in P} (workload(p_i) - ave\_workload)^2)^{1/2}} \quad (2)$$

3. The Proposed Algorithm

3.1. The Input-oriented BCC Model

In our paper, a Decision Making Unit (DMU) represents a schedule solution which is measured by DEA. Relative efficiency among DMUs is defined as the ratio of total weighted output to total weighted input. Instead of assigning all weight coefficients manually, DEA allows each DMU to choose its own set of weight coefficients so as to maximize self-efficiency. To avoid confusion, only when referring to BCC-efficiency, the term ‘DMU’ is used to replace ‘schedule solution’.

In DEA, there are various models, such as CCR model [6], BCC model [13] and FDH model [14]. In our proposal, BCC model is adopted to analysis the efficiency of schedule solutions. The reason for choosing BCC model over CCR and FDH model is that: on one hand, CCR model is based on the assumption of constant returns to scale, which is not fit for the scheduling problem; on the other hand, there are more computing complexity in FDH model than BCC model.

In our task scheduling scenario, the input vector x for a DMU is consisting of M, E and L; B is regarded as the output. The classification of metrics follows the ‘rule of thumb’ described in [15]: if the value of a metric is larger-is-better, then it is an output; otherwise the metric is an input.

Combined with four metrics, the production possibility set P_B of the BCC model is defined by:

$$P_B = \{(x, y) \mid x \geq X\lambda, y \leq Y\lambda, e\lambda = 1, \lambda \geq 0\} \quad (3)$$

And the BCC model is presented by the following Linear Programming (LP):

$$\begin{aligned} &\min\ z = \theta_B \\ &s.t\begin{cases} \theta_B x_i - X\lambda \geq 0 \\ Y\lambda \geq y_i \\ e\lambda = 1 \\ \lambda \geq 0 \end{cases} \end{aligned} \tag{4}$$

Where is a row vector with all elements equal to 1; $X \in R^{3\times n}$, $Y \in R^{1\times n}$ are the input and output matrix of the DMU set; variable λ is a vector in R^n; n is the number of DMUs in the DMU set; $x_i = (x_{M,i}\, x_{E,i}\, x_{L,i})$, $y_i = y_{B,i}$ are respectively the input vector and output vector of DMU i, which is expressed by the notation (x_i, y_i); scalar θ_B is the efficiency of DMU i.

In equation (4), $X\lambda, Y\lambda$ are the input vector and output vector of the virtual DMU k_1 which is achieved by the linear combination of some DMUs in the DMU set. When the input of DMU i is scaled θ_B times, the virtual DMU k_2 $(\theta_B x_i, y_i)$ is produced. According equation (3), DMU k_2 exists in the P_B. Because $\theta_B = 1, \lambda_i = 1$ is a feasible solution for equation (4), θ_B is not greater than 1. Thus the virtual DMU k_2 is obtained by scaling down the input of DMU i by θ_B. In the constraint condition, the first two formulas describe the situation that the input of DMU k_1 is less than DMU i's but the output of DMU k_1 is more, which means DMU k_1 is more efficient than DMU i. If $\theta_B = 1, \lambda_i = 1$, namely $k_2 = i$, is the unique solution for equation (4), DMU i is BCC-efficient. However, if the solution $\theta_B < 1$ exists, namely $k_2 \neq i$, DMU i is BCC-inefficient and DMUs those form the DMU k_1 are defined as reference DMUs of DMU i.

3.2. Referred-time

Although the BCC model successfully discriminates the BCC-efficient and BCC-inefficient schedules, it does not further evaluate the relative advantages between BCC-efficient ones. Thus a ranking method between BCC-efficient schedules is desirable, which in our algorithm is the referred-time of a DMU.

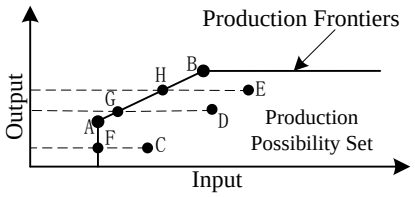


Figure 2. The BCC Model

In 1994, Sinuany proposed a technique that an efficient DMU is highly ranked if it is chosen as a reference DMU by many other inefficient DMUs [16]. The more times an efficient DMU is chosen to be a reference DMU, the better it is ranked. The concept of 'referred-time' provides us a way to further evaluate the BCC-efficient DMUs.

To be more specific, Figure 2 exhibits 5 DMUs, A, B, C, D and E, each with one input and one output. Only A and B are BCC-efficient DMUs. The efficient frontier of the BCC model consists of the bold lines connecting DMU A and B, which are the linear combination of BCC-efficient DMUs. As depicted in equation (3), the production possibility set is the area

consisting of the efficient frontier together with DMUs with an excess of input and/or a shortfall in output compared with the frontiers. In this Figure, G is the projection point on the frontier for the DMU D, when the input of DMU D is reduced. Meanwhile, the point G is also the linear combination of DMU A and DMU B. As described in 3.1, both DMU A and DMU B are forming the virtual DMU G, so DMU A and DMU B are reference DMUs for DMU D. In other words, DMU D's reference set consists of DMU A and B. It is clear that the referred-time of the DMU is the number of times it appears in each reference set. According to the above analysis, the referred-time of DMU A is 3 and B is 2. Therefore, DMU A is more efficient than B and is ranked better.

In the envelopment form of BCC model, $\lambda \in R^n$ is an variable in relation with "reference set", then we define it as follows:

Definition 1 (Reference Set). For a BCC-inefficient DMU i, we define its reference set E, based on an optimal solution λ^* by:

$$E = \{ j \mid \lambda_j^* > 0 \} (j \in \{1, \ldots, n\}) \quad (5)$$

When calculating the BCC-efficiency of the DMU i, if λ_j^* is non-zero, the DMU j is in the reference set of DMU i, and DMU j is a reference DMU for DMU i. If there are some DMUs with same referred times, in our implementation, the DMU with smaller energy cost is preferable.

3.3. Sub-swarm Strategy

In our implementation, the swarm size is 1000, so n is 1000 in (4). The LP that calculates the efficiency of one DMU has 1001 variables and 1005 constraint conditions. Considering the DPSO runs for 60 iterations, the entire computing work for one graph is extremely huge if the BCC efficiencies of all particles are calculated. Moreover, the simulation also shows that it requires about 30 seconds to calculate 1000 efficiencies for one time. Therefore, the time running 60 times is about half an hour.

To avoid this problem, the sub-swarm strategy is adopted. In our proposal, there are four sub-swarms with the same size corresponding to M, L, E and B respectively, and each optimizes a single metric. Through the evaluation and ranking, the top 10% of each sub-swarm are chosen to be DMUs for efficiency calculation, that is, only 100 particles are applied to BCC analysis. Comparing with 1000 particles, it reduces 900 constraint conditions and 900 variables when calculating the efficiency of one DMU. Therefore, the entire computing work for one graph sharply decreases by using the sub-swarm strategy.

3.4. The Algorithm Flow

The algorithm flow of BCC-DPSO is depicted in Algorithm 1. Firstly, the swarm is initialized, and the speed vector V and the location vector X for every particle are randomly generated. In DPSO, a particle's location vector represents one schedule solution, which is in forms of a binary array. Figure 3 demonstrates an example location vector of a particle of scheduling 4 tasks to 3 processors. The illustrated location vector is a 4*3 array, where each row represents a single task and each column corresponds to a processor. The element '1' in each row represents that the task is assigned to this processor. Thus, in this example, the task T1, T2, T3 and T4 are respectively allocated to P3, P2, P1 and P2.

$$\begin{array}{c} \\ T1 \\ T2 \\ T3 \\ T4 \end{array} \begin{array}{c} \begin{array}{ccc} P1 & P2 & P3 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{array}$$

Figure 3. The Example of the Particle Location

After initialization, the sub-swarm strategy is applied to DPSO, so there are four sub-swarms. Then, four metrics of each particle are evaluated and all particles in the same sub-swarm are ranked according to their corresponding metric value. As a result, the best particle in each sub-swarm are regarded as plbest of its sub-swarm and top 10% of each sub-swarm are chosen to be DMUs for efficiency calculation.

Moreover, the referred-time of the BCC-efficient DMU is also counted, and the particle with the largest referred-time is selected to be glbest (global best particle). To enhance the convergence speed, all efficient DMUs are used to replace the same number of worse particles in all sub-swarms.

In DPSO, the speed vector V and the location vector X of one particle are updated in every iteration. In traditional DPSO, the speed updating formula is

$$V_i^d = w * V_i^d + c_1 * rand_i^d * (pbest_i^d - X_i^d) + c_2 * rand_i^d * (\mathrm{glb}\, est_i^d - X_i^d) \quad (6)$$

In our proposal, the modified DPSO speed updating formula is presented by:

$$\begin{aligned} V_i^d = {} & w * V_i^d + c_1 * rand_i^d * (pbest_i^d - X_i^d) \\ & + c_2 * rand_i^d * (plbest_i^d - X_i^d) + c_3 * rand_i^d * (glbest_i^d - X_i^d) \end{aligned} \quad (7)$$

In equation (7), V_i^d , X_i^d respectively represent the d^{th} component of the particle i 's speed and location; w is a inertia weight; c_1 is a self acceleration factor; c_2 is a sub-swarm acceleration factor; c_3 is a meta-swarm acceleration factor; $rand_i^d$ is a random number between 0 and 1; $pbest_i^d$ is the d^{th} component of the best location in the particle i 's experience; $plbest_i^d$ is the d^{th} component of the best particle in the sub-swarm, in which the particle i is; $\mathrm{glb}\, est_i^d$ is the d^{th} component of the best particle in all swarm.

There are two parts in equations (7) that are different from equation (6). On one hand, *plbest* is the result of the sub-swarm strategy, which is different in different sub-swarms. On the other hand, *glbest* is up to the referred-time instead of the metric value. This modification aims to lead the evolution to the better direction.

With the speed updating, the location of the particle is also updated. Following equations illustrate the updating process of the location:

$$S(V_i^d) = 1 / (1 + e^{-V_i^d}) \quad (8)$$

$$X_i^d = \begin{cases} 1 \ if\ rand() < S(V_i^d) \\ 0 \ else \end{cases} \quad (9)$$

Equation (8) aims to map velocity V_i^d to the interval of [0,1], where $S(V_i^d)$ is the probability that the task is allocated to the processor d. Normally, $V_{\max} = 4$ is set to avoid $S(V_i^d)$ approaching 0 and 1 [17]. In equation (9), $rand()$ is used to generate a random number between 0 and 1. If this random number is not greater than $S(V_i^d)$, X_i^d is 1, which means that the d^{th} task is allocated to the processor d. Otherwise, this task is not allocated to the processor d. It is clear that the greater the speed, the greater the probability.

Following above formulas, particles continuously evolve. Until the end, the optimal solution is output.

Algorithm1: flow of BCC-DPSO

1. initialize the swarm and applying Sub-Swarm strategy to DPSO to divide the swarm into four sub-swarms
2. while iteration < 60 do
    1) evaluate each Sub-Swam and rank particles in each sub-swarm according to corresponding metric value
    2) choose the top of each Sub-Swam as their *plbest*s and pick out top 10% of each Sub-Swarm as DMUs to calculate their efficiencies
    3) count the referred-time of all BCC-efficient DMUs and choose the DMU with largest referred-time to be *glbest*
    4) replace worse particles in each sub-swarm using all BCC-efficient DMUs and update the location vector x and the speed vector v for each particle
    5) iteration ++
3. end while
4. output the best particle *glbest*

4. Experiment and Discuss

4.1. Test Bed

In our simulation, the scheduling algorithms are applied to 25 task graphs, which are randomly generated by TGFF [18]; task number varies from 32 to 51. The target hardware is a 4×4 2D mesh NoC, as described in Section 2.2. To evaluate the performance of our proposal, our BCC-DPSO scheduling algorithm is implemented in C++ under 32bit Win7 using VS2008. The LPs are calculated using GLPK4.47 [19], and the output schedules are simulated using a SystemC based cycle-accurate NoC simulator. The whole program is executed on the ASUS F8 PC with Intel Core(TM)2 Duo CPU, 1 GB RAM and 160 GB HDD.

4.2. Comparative Simulation

To show the efficient performance of our proposal, other four criterions are also applied to DPSO including Multiplication-and-Division (MD), Weighted Sum Method (WSM), Weighted Exponential Sum (WES) and Exponential Weighted Criterion (EWC) [20]. Combined with four metrics, these criterions are presented as follows:

$$
\begin{aligned}
fitness_{MD} &= M^{-1} \cdot L^{-1} \cdot E^{-1} \cdot B \\
fitness_{WSM} &= w_M \cdot M^{-1} + w_L \cdot L^{-1} + w_E \cdot E^{-1} + w_B \cdot B \\
fitness_{WES} &= w_M \cdot M^{-2} + w_L \cdot L^{-2} + w_E \cdot E^{-2} + w_B \cdot B^2 \\
fitness_{EWC} &= (e^{w_M} - 1) \cdot e^{M^{-1}} + (e^{w_L} - 1) \cdot e^{L^{-1}} + (e^{w_E} - 1) \cdot e^{E^{-1}} + (e^{w_B} - 1) \cdot e^{B}
\end{aligned} \tag{10}
$$

In Figure 4, there are four subplots respectively showing M, E, L and B performance, and five bars successively represent BCC-DPSO, MD-DPSO, WSM-DPSO, WES-DPSO, EWC-DPSO in each group.

For most situations, the three input metrics achieved by BCC-DPSO are more superior. Comparing with the metric M, E and L yielded by MD-DPSO, the average reductions of the corresponding metric of BCC-DPSO are respectively 5.5%, 8.3% and 8.3%. Similar results are found between BCC-DPSO and other multi-objective optimization DPSOs: the M, E and L metric of BCC-DPSO are 9%, 22% and 22% smaller than the corresponding metric of WSM-DPSO; 8.1%, 23% and 23% smaller than that of WES-DPSO; and 9%, 22% and 22% smaller than that of EWC-DPSO.

However, the output metric (B metric) of most solutions produced by other four criterions are better than BCC-DPSO. Comparing with these four criterions, the reductions of the metric B of BCC-DPSO are respectively 106%, 155%, 154% and 155%. The most important reason why the performance of the input metric is better than the output metric is that we adopt the input-oriented BCC model in our paper, which aims to minimize inputs while satisfying at least the given output levels. In addition to the fact that we pay more attention on the optimizations of input metrics than the output metric, so the solution with the greater workload balance is likely to be eliminated.

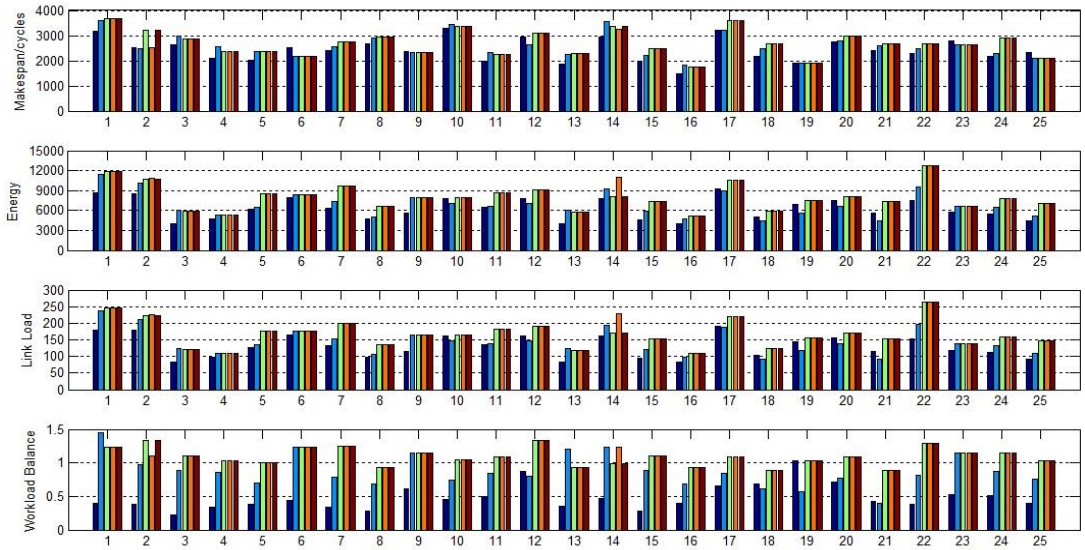


Figure 4. Four Metrics Comparison Result

Nonetheless, it is found that the average efficiency of all solutions produced by BCC-DPSO is more than others'. As a result of the above experiment, for one graph, there are 5 schedule solutions generated by 5 algorithms. Then, those 5 schedule solutions are regarded as a set of DMUs to be calculated their efficiencies. So, there are 25 sets of DMUs for 25 graphs and 25 efficiencies for each algorithm. In Figure 5, each bar represents the average

efficiency of each algorithm, which is the average value of 25 efficiencies of one algorithm. Comparing with MD-DPSO, WSM-DPSO, WES-DPSO and EWC-DPSO, the developments of the efficiency of BCC-DPSO are respectively 0.25%, 0.24%, 0.24% and 0.24%.

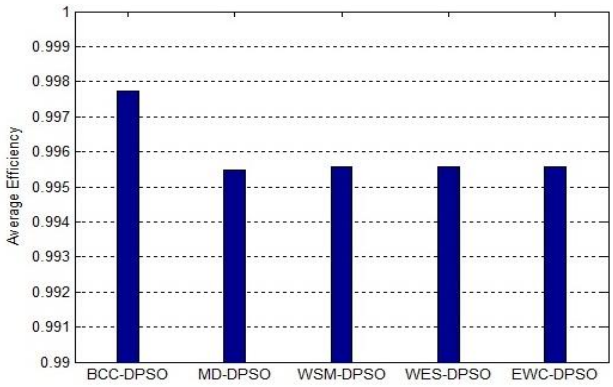


Figure 5. The Average Efficiency of each Algorithm

Finally, the actual solving time of our BCC DPSO as well as other four multi-objective DPSOs are shown in Figure 6. As observed in the figure, MD-DPSO, WSM-DPSO, WES-DPSO and EWC- DPSO require almost the same level of solving time, while BCC-DPSO consumes more. The average solving time of BCC-DPSO is 42.48 seconds in 25 task graphs, which is 169% more than the average solving time of MD-DPSO (15.8 seconds), 181% more than WSM-DPSO (15.12 seconds), 174% more than WES-DPSO (15.48 seconds) and 181% more than EWC-DPSO (15.12 seconds). Though the solving time of BCC-DPSO is longer, the differences between BCC-DPSO and other criterions are little. Moreover, comparing with the resolving time of 1000 particles, namely half an hour, it confirms that the sub-swarm strategy can effectively reduce the calculation in BCC model.

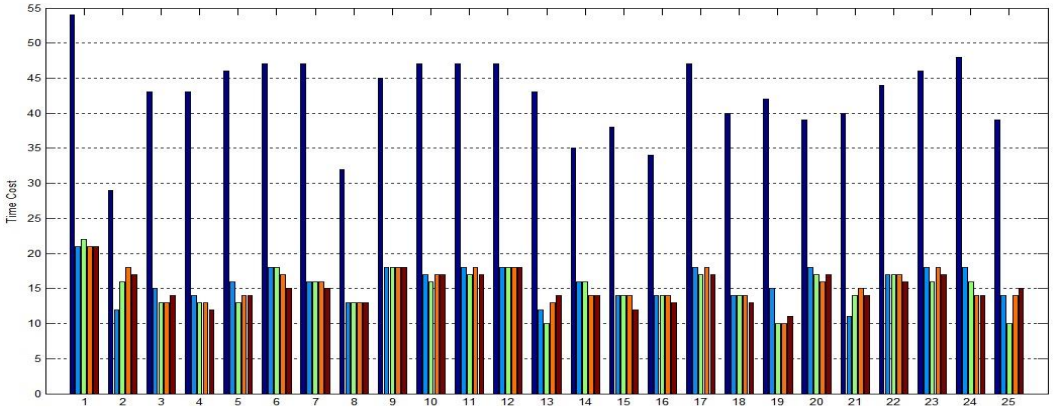


Figure 6. Time Cost Result

5. Conclusion and Future Study

In this paper, a modified DPSO algorithm using BCC model for solving the multi-objectives task scheduling problem in NoC has been proposed. On one hand, we adopt the Sub-Swarm strategy to reduce the calculation about BCC-efficiency. On the other hand, referred-time is introduced into our proposal to define the global best solution. In

experimental results, it shows that the output schedule solution of BCC-DPSO is more efficient than others'.

Acknowledgements

Thanks to the support of the National Natural Science Foundation of China [61201005], the Fundamental Research Funds for the Central Universities [ZYGX2012J001] and the National Major Projects [2011ZX03003-003-04].

References

[1] L. Benini and G. De Micheli, "Networks on chip: a new paradigm for systems on chip design", in Design, Automation and Test in Europe Conference and Exhibition, Proceedings, **(2002)**, pp. 418-419.
[2] D. Shin and J. Kim, "Power-aware communication optimization for networks-on-chips with voltage scalable links", in Proceedings of the 2nd IEEE/ACM/IFIP international conference on Hardware/software code sign and system synthesis, **(2004)**, pp. 170-175.
[3] T. Lei and S. Kumar, "A two-step genetic algorithm for mapping task graphs to a network on chip architecture", in Digital System Design, Proceedings Euromicro Symposium on, **(2003)**, pp. 180-187.
[4] H. Jain and K. Deb, "An Improved Adaptive Approach for Elitist Non-dominated Sorting Genetic Algorithm for Many-Objective Optimization", in Evolutionary Multi-Criterion Optimization, **(2013)**, pp. 307-321.
[5] R. Garg and A. K. Singh, "Enhancing the Discrete Particle Swarm Optimization based Workflow Grid Scheduling using Hierarchical Structure", International Journal of Computer Network and Information Security (IJCNIS), vol. 5, **(2013),** pp. 18.
[6] A. Charnes, W. W. Cooper and E. Rhodes, "Measuring the efficiency of decision making units", European journal of operational research, vol. 2, **(1978),** pp. 429-444.
[7] P. P. Pande, C. Grecu, M. Jones, A. Ivanov and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures", Computers, IEEE Transactions on, vol. 54, **(2005),** pp. 1025-1040.
[8] F. Samman, T. Hollstein and M. Glesner, "New theory for deadlock-free multicast routing in wormhole-switched virtual-channelless networks-on-chip", Parallel and Distributed Systems, IEEE Transactions on, vol. 22, **(2011),** pp. 544-557.
[9] E. Bolotin, I. Cidon, R. Ginosar and A. Kolodny, "QNoC: QoS architecture and design process for network on chip", Journal of Systems Architecture, vol. 50, **(2004),** pp. 105-128.
[10] T. T. Ye, L. Benini and G. De Micheli, "Analysis of power consumption on switch fabrics in network routers", in Design Automation Conference, Proceedings 39th, **(2002)**, pp. 524-529.
[11] N. Nikitin, S. Chatterjee, J. Cortadella, M. Kishinevsky and U. Ogras, "Physical-aware link allocation and route assignment for chip multiprocessing", in Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on, **(2010)**, pp. 125-134.
[12] Y. Fang, F. Wang and J. Ge, "A task scheduling algorithm based on load balancing in cloud computing", in Web Information Systems and Mining, ed: Springer, **(2010)**, pp. 271-277.
[13] R. D. Banker, A. Charnes and W. W. Cooper, "Some models for estimating technical and scale inefficiencies in data envelopment analysis", Management science, vol. 30, **(1984),** pp. 1078-1092.
[14] D. Deprins, L. Simar and H. Tulkens, "Measuring labor-efficiency in post offices", in Public goods, environmental externalities and fiscal competition, ed: Springer, **(2006)**, pp. 285-309.
[15] A. J. Ruiz-Torres and F. J. López, "Using the FDH formulation of DEA to evaluate a multi-criteria problem in parallel machine scheduling", Computers & Industrial Engineering, vol. 47, **(2004),** pp. 107-121.
[16] N. Adler, L. Friedman and Z. Sinuany-Stern, "Review of ranking methods in the data envelopment analysis context", European journal of operational research, vol. 140, **(2002),** pp. 249-265.
[17] A. Y. Zomaya, "Handbook of nature-inspired and innovative computing", Integrating classical models with emerging technologies: Springer, **(2006)**.
[18] R. P. Dick, D. L. Rhodes and W. Wolf, "TGFF: task graphs for free", in Proceedings of the 6th international workshop on Hardware/software codesign, **(1998)**, pp. 97-101.
[19] A. Makhorin, "GNU linear programming kit", Moscow Aviation Institute, Moscow, Russia, vol. 38, **(2001)**.
[20] R. T. Marler and J. S. Arora, "Survey of multi-objective optimization methods for engineering", Structural and multidisciplinary optimization, vol. 26, **(2004),** pp. 369-395.

