IEICE TRANS. INF. & SYST., VOL.E102–D, NO.3 MARCH 2019

LETTER

BMM: A Binary Metaheuristic Mapping Algorithm for Mesh-Based Network-on-Chip

Xilu WANG[†], **Yongjun SUN**[†a)], ***Nonmembers***, ***and*** **Huaxi GU**[†], ***Member***

SUMMARY The mapping optimization problem in Network-on-Chip (NoC) is constraint and NP-hard, and the deterministic algorithms require considerable computation time to find an exact optimal mapping solution. Therefore, the metaheuristic algorithms (MAs) have attracted great interests of researchers. However, most MAs are designed for continuous problems and suffer from premature convergence. In this letter, a binary metaheuristic mapping algorithm (BMM) with a better exploration-exploitation balance is proposed to solve the mapping problem. The binary encoding is used to extend the MAs to the constraint problem and an adaptive strategy is introduced to combine Sine Cosine Algorithm (SCA) and Particle Swarm Algorithm (PSO). SCA is modified to explore the search space effectively, while the powerful exploitation ability of PSO is employed for the global optimum. A set of well-known applications and large-scale synthetic cores-graphs are used to test the performance of BMM. The results demonstrate that the proposed algorithm can improve the energy consumption more significantly than some other heuristic algorithms.

key words: *network-on-chip, application mapping, particle swarm algorithm, sine cosine algorithm*

1. Introduction

Network-on-Chip (NoC) is emerged to integrate hundreds to thousands of the intellectual property cores (IP cores) on a single chip. Because of the high frequent data transmission between cores, the energy consumption of NoC accounts for a majority portion of the overall chip energy budget [1]. Therefore, how to map IP cores of an application onto the adjacent location in a topology, namely application mapping, is one of the most important phases in NoC design. The mesh topology is the most widely used one for its superior properties such as the simple layout, the easy expansion and so on. Application mapping based on mesh NoC is generally known as a NP-hard problem, due to the exponentially increase of search space with growing mesh size. However, the traditional deterministic algorithms require a great amount of computation time to find an exact optimal solution, thus the metaheuristic algorithms (MAs) have been widely applied to the mapping problem.

Since the chromosome of Genetic Algorithm (GA) is integer string encoding, GA becomes one of the most popular population-based MAs for NoC mapping. Sun et al. [2] integrate the advantages of GA and Simulated Annealing (SA) Algorithm to reduce the energy consumption of NoC. A GA-based approach is introduced to carry out the mapping of application on NoC in [3]. Ant Colony Optimization (ACO) is an efficient method for discrete optimization problems. In [4], an ACO-based method is employed for NoC mapping. An ACO-based optimization algorithm is conductde for the mapping process [5]. Sahu et al. present a multistage PSO to map applications on both 2-D and 3-D mesh-based NoC [6]. A hybrid MA based on Tabu-search and PSO is presented in [7]. As shown above, just a few kinds of MAs have been applied to the mapping problem, because most MAs are designed for continuous problems. Although the binary MA is an alternative to satisfy the constraint, it is seldom used for application mapping.

Therefore, this study proposes a binary metaheuristic mapping algorithm (BMM) for 2D-mesh NoC based heterogeneous multi-processor systems. The binary encoding is used to make Sine Cosine Algorithm (SCA) and PSO suitable for the constraint mapping problem. An adaptive strategy is employed to allocate appropriate proportion for exploration and exploitation, respectively. For exploration phase, SCA finds the promising regions of the search space and jumps out of the local optimum. In addition, the global best solution is memorized and passed on to PSO. Guided by self-cognition and social-cooperation, PSO exploits for a better solution in the promising area.

2. Problem Formulation

2.1 Mapping Model

For a given application, an application can be viewed as an Application Core Graph (ACG). Similarly, the target NoC topology can be represented by a Topology Architecture Graph (TAG) [2]. Mathematically, the detailed definitions are shown as follows:

Definition 1: The Application Core Graph (ACG) is a directed weighted graph G(C, E), where each vertex $c_i \in C$ denotes an IP core used in application, the directed arc $e_{ij} \in E$ represents a communication trace from the IP core c_i to the IP core c_j, and the weight of each edge w_{ij} indicates the communication volume on edge e_{ij}.

Definition 2: The Topology Architecture Graph (TAG) is a directed graph G(T, L), where each vertex $t_i \in T$ represents one tile which includes a router r_i, a network interface and a position of IP cores, the directed arc $l_{ij} \in L$ represents the physical links (i.e., the routing path) between t_i and t_j.

Manuscript received October 4, 2018.
Manuscript publicized November 26, 2018.
[†]The authors are with State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, 710071, China.
a) E-mail: yjsun@mail.xidian.edu.cn
DOI: 10.1587/transinf.2018EDL8208

2.2 Energy Model

In this work, the mapping finds one-to-one connections between IP cores and the tiles to reduce the energy consumption. The energy consumption of NoC mainly comes from the routers and the communication links. Hence, the energy consumption model for single-data transmission is shown as follows:

$$E_{bit} = E_{Rbit} + E_{Lbit}, \tag{1}$$

where E_{Rbit} and E_{Lbit} represent the energy consumed by the routers and the links, respectively. Specifically, E_{Rbit} is suggested calculating as [8]:

$$E_{Rbit} = E_{Sbit} + E_{Bbit} + E_{Wbit}, \tag{2}$$

where E_{Sbit}, E_{Bbit} and E_{Wbit} represent the energy consumed by switch, buffering and interconnection wires, respectively. Since E_{Bbit} and E_{Wbit} are negligible, the energy consumed by sending one bit of data from tile to tile can be calculated as:

$$E_{bit}^{t_i,t_j} = (n_{hops} + 1) \times E_{Sbit} + n_{hops} \times E_{Lbit}, \tag{3}$$

where E_{Sbit} and E_{Lbit} represent the energy consumed by the switches and the links respectively, n_{hops} is evaluated by the Manhattan distance between the source tile and the destination tile.

Consequently, the total energy consumption is given by:

$$energy(map(c_i), map(c_j)) = \sum^{T} w_{ij} \times E_{bit}^{t_i,t_j}, \tag{4}$$

where T represents a set of tiles in NoC, w_{ij} denotes the communication volume over the communication links, $map(c_i)$ and $map(c_j)$ denote the corresponding tiles of IP cores c_i and c_j respectively.

Based on the models given above, the application mapping problem can be expressed as:

$$\min\{\sum_{\forall c_i,c_j \in C} w_{ij} \times (energy(map(c_i), map(c_j))\}, \tag{5}$$

$$\forall c_i \in C, map(c_i) \in T, \tag{6}$$

$$\forall c_i \neq c_j \in C, map(c_i) \neq map(c_j) \in T, \tag{7}$$

3. BMM Algorithm

The motivation of BMM algorithm is to make the hybrid MAs applicable to the constraint mapping problem. At the same time, the proposed algorithm is capable of saving energy consumption effectively. This section shows the details and analysis of BMM algorithm as follows.

3.1 Binary Encoding

For given inputs (ACG and TAG), as shown in Fig. 1, the

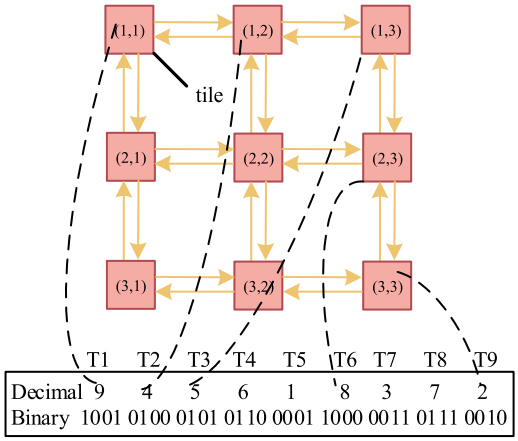


Fig. 1 An application mapping by binary encoding.

candidate solution $X_i = (x_1, x_2, \ldots, x_n)$, namely the position of search agent, is a binary vector. Similar to the binary PSO [9], the velocities of the search agents are mapped to probability values in [0, 1]. Based on the obtained probability values, the positions switch between 0 and 1 to achieve the updating. Note that each dimension of the vector has only two possible discrete values, 0 and 1, but the velocities are continuous and have no restriction. This strategy enables the universal MAs to solve discrete problems.

The Sigmoid function in Eq. (8) is widely used to transfer real values of velocities to probability values. However, it obtains poor convergence and cannot truly reflect the actual search process. Hence, the modified Sigmoid function [10] in Eq. (9) is used in our work.

$$Sigmoid(V_i) = \frac{1}{1 + e^{-V_i}}, \tag{8}$$

$$S'(V_i) = 2 \times |Sigmoid(V_i) - 0.5|, \tag{9}$$

Consequently, the position-updating function can be expressed as:

$$X_i(t+1) = \begin{cases} exchange(X_i(t)), & p1 < S'(V_i(t+1)) \\ X_i(t), & p1 \geq S'(V_i(t+1)) \end{cases} \tag{10}$$

where t is the current iteration, exchange operator denotes the transformation between 0 and 1.

3.2 Exploration and Exploitation Phases

In the view of exploration and exploitation, a better trade-off between the two components is required to help MA achieve good performance. Hence, an adaptive parameter A is adopted to choose smoothly between exploration and exploitation. Mathematically, A is calculated as follows:

$$A = \gamma \times (2 \times r_1 - 1), \tag{11}$$

$$\gamma = (2 - t \times 2/MaxIter), \tag{12}$$

where $Maxiter$ is the maximum number of iterations, $r1$ is a random number in [0, 1].

When $|A| > 1$, the exploration phase is exhibited. The

binary Sine Cosine Algorithm [11] is chosen to globally investigate the search space. Based on sine and cosine functions, SCA creates a set of random solutions. It has been proved that SCA can explore different regions of the search space effectively. In this regard, the proposed algorithm employs the binary SCA to improve the global search capability and increase the diversity. As discussed in Sect. 4.1, the velocity of binary SCA is defined for constrained mapping problem and updated by:

$$V_i(t+1) = \begin{cases} V_i(t) + \gamma \times sin(r_2) \times |r_3 \times X^* - X_i|, r_4 < 0.5 \\ V_i(t) + \gamma \times cos(r_2) \times |r_3 \times X^* - X_i|, r_4 \geq 0.5 \end{cases} \quad (13)$$

where X^* is the global best solution obtained so far, X_i denotes the current position, $|\cdot|$ indicates the absolute value operation, $r_2/r_3/r_4$ are random numbers in $[0, 1]$.

Correspondingly, when $|A| < 1$, the exploitation phase is performed to search the promising area by the binary PSO. In PSO, the self-cognition and social-cooperation enable the swarm to move toward the best solution and search the neighborhood of the current best solution. This specific strategy contributes to the powerful exploitation ability of PSO.

Based on this background, the binary PSO is executed for local search. The velocity of each particle is given as

$$V_i(t+1) = wV_i(t) + c_1 \times r_5 \times (p\_best - X_i) + c_2 \times r_6 \times (X^* - X_i) \quad (14)$$

where r_5/r_6 are random values in $[0, 1]$, c_1 and c_2 are self-cognition and social-cooperation parameters respectively, and w is the inertia weight. The larger value of w emphasizes the global search and a smaller one enhances the local search. Therefore, a linearly decreasing w is adopted to balance the exploration and exploitation.

$$w = w_{max} - \frac{w_{max} - w_{min}}{w_{max}} \times t \quad (15)$$

where w_{max} and w_{min} are the lower and upper boundaries of w respectively. The pseudo code of BMM algorithm is presented in Algorithm 1.

Algorithm 1 : A binary metaheuristic mapping algorithm

Input: The weighted matrix ${}^{r}mG(C, E)$ of ACG
Output: The optimal mapping results, the minimum energy consumption

Initialization(){
  Initialize the parameters *gamma*, A, w and $MaxIter$
  Initialize the population X_i and velocity V_i, $(i = 1, 2, \ldots, n)$
  Calculate the fitness (energy consumption) of each search agent
  X^* = the best solution}
Main loop(){
While ($t <$ *MaxIter*)
  For each search agent
  Update γ, A and w
    If($|A| < 1$)
      The exploration phase is exhibited by SCA
      Update V_i by Eq. (13)
      Update X_i by Eq. (10)
      Evaluate the solutions and update X^*
    Else if($|A| \leq 1$)
      The exploitation phase is performed by PSO
      Update V_i by Eq. (14)
      Update X_i by Eq. (10)
      Evaluate the solutions and update X^*
    End if
  End for
  $t = t + 1$
End while}
Return X^* and the fitness of X^*

4. Results and Analysis

To evaluate the effectiveness of the proposed BMM algorithm, some typical real multimedia ACGs, namely VOPD, MPEG-4, MMS, MWD and PIP, are served as benchmarks. For larger scale NoC, five synthetic ACGs with the number of IP cores from 7×7 to 11×11 are selected. The energy consumption and performance of BMM algorithm also are compared with ACO, GA, binary PSO and Random Algorithm (RAND).

Each algorithm is coded in Matlab R2015b, and all of the experiments are performed on Intel Core i7 platform with 8 GB RAM and 2.9 GHz clock frequency in the Windows 10 environment. The parameters in energy consumption model, E_{Sbit} and E_{Lbit}, are set to $0.43pJ$ and $5.445pJ$ respectively. Specifically, E_{Sbit} is calculated using $0.18\mu m$ CMOS technology with 1GHz clock frequency, and E_{Lbit} is determined from the following parameters: capacitance of wire ($0.5\mu F/\mu m$), voltage swing (3.3V) and length of link (2mm). Each mapping algorithm is conducted on target application for 20 times and the average and the minimum (best) energy consumption are obtained. Mean execution time (in second) obtained by each algorithm on five synthetic ACGs are noted in Table 1. Stability is one of the most important performance metrics for stochastic algorithms, and the gap between the average and the best results is introduced to describe this metric [12]

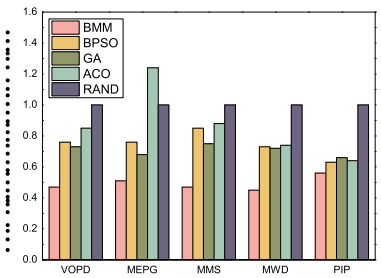
Table 1 Mean execution time (in second) on five synthetic ACGs.

Synthetic ACGs	BMM	BPSO	GA	ACO	RAND
7×7	16.01	15.53	18.64	420.88	0.23
8×8	25.02	24.07	30.48	1012.83	0.41
9×9	36.35	35.78	48.39	2169.54	0.63
10×10	53.75	53.34	79.03	4475.71	1.04
11×11	74.97	74.42	115.66	9013.12	1.43

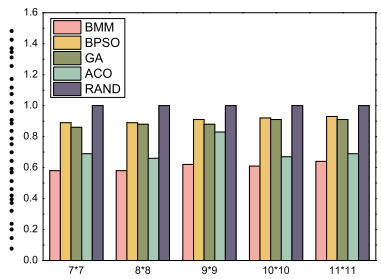
$$GAP = \frac{\frac{1}{k} \times \sum_{i=1}^{k} E(X_i) - E(X^*)}{E(X^*)} \quad (16)$$

where k is equal to 20, and $E(X_i)$ represents the energy consumption of the NoC with the mapping solution X_i, $E(X^*)$ denotes the minimum energy consumption obtained from 20 times. The normalized energy consumption of different mapping algorithms is shown in Fig. 2. Table 2 presents the GAPs and the reduction of energy consumption (RD) of BMM over the compared algorithms, respectively.

For small-size real ACGs, the proposed algorithm outperforms the traditional algorithms in terms of stability and energy consumption. The GAPs of BMM ranges from 0 to 3.1%, while GAPs of GA and BPSO are bigger than 10%.



(a)



(b)

Fig. 2 Energy consumption normalized to RAND.

Table 2 Comparisons of *GAP*(%) and *RD*(%).

	BMM	BPSO		GA		ACO		RAND	
	GAP	*RD*	*GAP*	*RD*	*GAP*	*RD*	*GAP*	*RD*	*GAP*
VOPD	1.1	37.6	10.1	35.0	26.3	44.0	2.9	52.6	14.6
MEPG	2.1	31.3	23.3	24.0	18.9	56.5	3.2	48.2	6.5
MMS	3.1	44.7	18.3	37.1	14.4	58.1	7.08	52.9	7.9
MWD	2.9	33.0	15.0	32.6	14.3	33.7	2.5	51.1	8.4
PIP	0	9.3	10.2	14.7	17.2	12.1	4.4	43.3	5.5
7×7	3.4	34.0	6.3	32.1	5.1	15.0	4.9	41.5	7.7
8×8	2.8	34.7	13.0	33.9	4.2	11.7	5.2	41.8	3.2
9×9	4.8	31.2	5.3	28.9	5.3	24.9	5.7	37.5	3.6
10×10	6.3	33.5	6.6	32.4	6.5	8.3	6.9	38.5	2.9
11×11	4.1	30.6	5.0	28.9	8.8	6.4	7.2	35.5	2.8

Figure 2 (a) indicates that the proposed algorithm achieves the minimum energy consumption, followed by GA and ACO. BMM saves 35% to 60% energy consumption for VOPD, MEPG, MMS and MWD, and the most significant reduction of energy consumption occurs on MMS application. For five synthetic ACGs, BMM reduces around 30% energy costs compared with BPSO, GA and RAND. Although BMM saves the energy consumption insignificantly (about 5%) compared with ACO, as seen in Table 1, ACO has the highest consuming time due to its computation complexity. The mean execution time of BMM and BPSO are very close to each other and both are better than GA and ACO. In conclusion, the proposed algorithm outperforms other four methods considering energy consumption, stability and execution time.

5. Results and Analysis

Mapping the IP cores onto NoC architecture plays an important role. To reduce the energy consumption, a modified MA is used to address the constraint mapping problem. The original SCA and PSO are extended to discrete problem by introducing the binary encoding and the transfer function. The combination of different advantages of MAs improves the performance in terms of convergence rate, solution precision and efficiency. An adaptive strategy is proposed to switch smoothly between the exploration and exploitation. In our future work, more metrics will be considerd, such as the temperature and latency.

Acknowledgments

This work was supported by the National Science Foundation of China under Grants 61634004, and Grant 61472300, the Fundamental Research Funds for the Central Universities Grant No.JB180309 and No.JB170107, the key research and development plan of Shaanxi province No.2017ZDCXL-GY-05-01, and the General Armament Department and Ministry of Education United Fund under Grant No.6141A0224-003.

References

[1] T. Maqsood, K. Bilal, and S.A. Madani, "Congestion-aware core mapping for Network-on-Chip based systems using betweenness centrality," Futur. Gener. Comput. Syst., vol.82, pp.459–471, 2018.
[2] G. Sun, S. Lin, D. Jin, Y. Li, L. Su, Y. Zhang, and L. Zeng, "Performance-aware hybrid algorithm for mapping IPS onto mesh-based network on chip," IEICE Trans. Inf. Syst., vol.E94-D, no.5, pp.1000–1007, 2011.
[3] J.V. Bruch, E.A. Da Silva, C.A. Zeferino, and L.S. Indrusiak, "Deadline, energy and buffer-aware task mapping optimization in NoC-based SoCs using genetic algorithms," Brazilian Symp. Comput. Syst. Eng. SBESC, pp.86–93, 2017.
[4] M. Farias, E. Barros, A. Filho, A. Araujo, A. Silva, and J. Melo, "An Ant Colony metaheuristic for energy aware application mapping on NoCs," Proc. IEEE Int. Conf. Electron. Circuits, Syst., no.485829, pp.365–368, 2013.
[5] J. Wang, Z. Chen, J. Guo, Y. Liet, and Z. Lu, "ACO-Based Thermal-Aware Thread-to-Core Mapping for Dark-Silicon-Constrained CMPs," IEEE Trans. Electron Devices, vol.64, no.3, pp.930–937, 2017.
[6] P.K. Sahu, T. Shah, K. Manna, and S. Chattopadhyay, "Application Mapping Onto Mesh-Based Network-on-Chip Using Discrete Particle Swarm Optimization," IEEE Transactions on Very Large Scale Integration Systems, vol.22, no.2, pp.300–312, 2014.
[7] M. Obaidullah and G.N. Khan, "Application mapping to mesh NoCs using a Tabu-search based swarm optimization," Microprocess. Microsyst., vol.55, pp.13–25, 2017.
[8] T.T. Ye, L. Benini, and G. De Micheli, "Analysis of power consumption on switch fabrics in network routers," Design Automation Conference, pp.524–529, 2002.
[9] G. Zhang, R. Yang, Z. Su, F. Yue, Y. Fan, M. Qi, and J. Jiang, "Using binary particle swarm optimization to search for maximal successful coalition," Appl. Intell., vol.42, no.2, pp.195–209, 2015.
[10] H. Nezamabadi-pour, M. Rostami-Shahrbabaki, and M. Maghfoori-Farsangi, "Binary particle swarm optimization: Challenges and new solutions," CSI J. Comput. Sci. Eng., vol.6, no.1-A, pp.21–32, 2008.
[11] S. Mirjalili, "SCA: A Sine Cosine Algorithm for solving optimization problems," Knowledge-Based Syst., vol.96, pp.120–133, 2016.
[12] X. Wang, H. Liu, and Z. Yu, "A novel heuristic algorithm for IP block mapping onto mesh-based networks-on-chip," J. Supercomput., vol.72, no.5, pp.2035–2058, 2016.