ISSN: 2251-9246
EISSN: 2345-6221

pp. 255:268

An Advanced Cost-Aware Mapping Algorithm Based on Frog Leaping and Tabu Search for Two-Dimensional Network-On-Chip

Azin Hatefi[1,2], Elham Yaghoubi[1,2*]

[1]Department of Computer Engineering, Na.C., Islamic Azad University, Najafabad, Iran, elhamyaghoubi@iau.ac.ir
[2]Big Data Research Center, Na.C., Islamic Azad University, Najafabad, Iran, azinhatefi@iau.ac.ir

Abstract

One of the main challenges in two-dimensional Network on Chip (NoC) architectures is mapping application graphs onto tile-based architectures. Without appropriate mapping algorithms, the system's performance can be significantly degraded. Various mapping algorithms have been proposed for NoC. However, most of these methods have not been able to address the main challenges in mapping algorithms, particularly the communication cost. To this end, a mapping algorithm named FTMA (Frog leaping and Tabu Search Mapping Algorithm) for two-dimensional mesh topology-on-chip base is proposed which utilizes the combination of frog leaping mapping algorithm and Tabu search for mapping operation. Using these two algorithms in the proposed method provides several advantages for the mapping algorithm. These advantages include finding the best mapping with minimum communication cost, utilizing smart memory to reduce extra costs, and being suitable for most application graphs. The algorithm is also appropriate for graphs with a large number of nodes and provides a final solution. Simulation results show that the proposed mapping algorithm reduces communication costs. The reduction percentages are 12.62, 7.45, 15.34, 11.60, 15.18, 13.805, 17.68, 15.64 and 13.66 compared to the ISFLA, IAM, ELIXIR, PSMAP, ACO, NMAP, LMAP, and BMA methods, respectively.

Keywords: network on two-dimensional chip, application graph, Tabu search algorithm, frog leaping algorithm
Article history: Received 2025/10/08; Revised 2025/11/02; Accepted 2025/11/13, Article Type: Research paper

https://doi.org/10.82234/IJSEE.2025.1220346

1. Introduction

Currently, two-dimensional network on chip (2D-NoC) is introduced as a high-performance communication structure for processing systems [1,2]. One of the most important topics in this area is the mapping of application graphs onto chips. The goal of this mapping is to minimize communication costs as much as possible. The lower the communication costs, the higher the system's performance increases [3,4].

On the other hand, the process of mapping graphs onto 2D chips is considered an interesting and dynamic challenge, attracting significant attention from researchers in this field. Various algorithms have been proposed for application graph mapping onto chips, which can be broadly categorized into two main types: dynamic (online) and static (offline) methods [5,6]. In the dynamic mapping method, the mapping is performed during runtime, leading to increased energy consumption, computational overhead, and time delays [7,8]. Additionally, implementing this type of mapping is challenging. Therefore, static mapping is considered the optimal solution. Given the increasing complexity of processing systems, there is a growing need for the design of efficient networks for data transmission and resource management [9,10]. Since NoC is a key technology in parallel processing and multi-core systems, examining the challenges and solutions related to mapping can lead to a better understanding of the needs of advanced architectures and offer new approaches to improving the performance of complex systems. Furthermore, achieving optimal designs at large scales requires this topic to be discussed and explored [11,12].

In static mapping, the allocation of components and communication paths in the network design is determined beforehand and remains fixed throughout the execution of the system. This approach can significantly impact reducing communication delay, optimizing energy consumption, and making better use of hardware resources such as bandwidth and power consumption. It also enables the design of scalable and more efficient systems by optimizing the distribution of processing loads and communications between cores. Therefore, this method can lead to algorithms that enhance the performance of NoC-based systems and address the challenges of complex processing systems [13,14].

Numerous algorithms have been proposed to address these challenges. These algorithms can improve network efficiency by effectively allocating communication resources such as bandwidth and memory capacity. Additionally, many of these algorithms target the reduction of communication delay, and by optimally distributing processing loads, they can decrease the data transfer time between cores. Reducing energy consumption is another significant benefit of these algorithms. Mapping algorithms, by selecting optimal paths and appropriately distributing processing loads, reduce the system's power consumption. Thus, using these algorithms can lead to system stability and overall performance improvement.

Despite the advantages of these algorithms, they also face certain drawbacks. One of the main issues is high computational complexity, which can increase execution time and computational costs, especially in systems with a large number of nodes (up to thousands of application nodes), where precise mapping requires substantial resources. Many of these algorithms have limited scalability and do not perform well in large-scale systems. Additionally, these algorithms are often dependent on specific topologies or hardware features, which makes them ineffective in diverse environments or with changing hardware configurations. Some algorithms also fail to optimize multiple objectives (such as delay, energy, and bandwidth) simultaneously and focus on just one specific goal. Moreover, in some algorithms, the reduction of processing load and optimal use of resources is not well-executed, leading to instability or decreased system performance [15,16].

This paper focuses on the challenge of the increasing number of mapping states as the number of application graph nodes grows, and we aim to propose an algorithm to address it.

In recent years, many authors have worked on the challenge of increasing mapping states as the number of application graph nodes increases. However, most of these methods have not been able to effectively reduce communication costs. Therefore, in this paper, a mapping algorithm called FTMA is introduced, which can significantly reduce communication costs in this network by effectively utilizing frog-leaping algorithms and Tabu search [17,18].

The results show that this algorithm achieves significant improvement compared to the previously proposed algorithms.

The structure of this paper is organized as follows. In the section 2, the basic concepts are introduced, which include the concept of application graph, mapping on chip network, frog mapping, and Tabu search mapping. In the section 3, previous works on mapping algorithms for network-on-chip architectures are reviewed. In the section 4, the problem statement is described. In the section 5, the FTMA mapping algorithm is introduced. In the section 6, the simulation results are presented and compared with the latest similar methods in the field. Finally, the conclusion is provided in the section 7.

2. Basic Concepts

In this section, the key concepts used in this article will be examined, and explanations will be provided regarding them.

A) The concept of usage graph

Each application consists of different components, and the connections between these components can be represented as a graph, known as the application graph. In this graph, the nodes represent the various components of the application, while the edges show the connections between them. Fig. 1 illustrates an example of an application graph. According to the figure, the graph contains four nodes labelled Memory, ARM CPU, DISPLAY, and FFT. Each of these nodes represents a part of the application, and the numbers on the edges indicate the bandwidth between each pair of nodes.

In fig. 2, four examples of common and well-known application graphs used in this paper for NoC mapping are shown. As observed in the figure, each application graph consists of several different nodes. Additionally, some of these graphs are directed, while others have bidirectional edges. As a result, they have different structures and can effectively represent the performance of the algorithm.

B) Mapping Concept in Network on chip

Each node in the application graph represents a part of the application and performs tasks that can be executed by the tiles in the NoC. The allocation of tasks to tiles is determined by the proposed algorithm, as shown in fig. 3, which provides an example of the mapping of application graph nodes onto the chip. As seen in the figure, the functions of the ARM CPU, DISPLAY, memory, and FFT are performed by tiles 2, 3, and 4, respectively. According to the figure, each application graph is represented as a graph G=G(V,E), where V refers to the nodes of the application graph and E represents the edges between them. Additionally, any connected graph is represented as G=G(T,P), where each vertex T refers to a tile, and each directed edge Pi,j represents a path from tile ti to tile tj [19,20].

Each node in the application graph is mapped to a node in the on-chip network. Any algorithm used for this mapping results in the generation of multiple mappings, each of which can be considered as a potential solution to the problem. Among these solutions, the mapping with the lowest communication cost is selected as the final mapping. The communication cost of a mapping is measured by (1). Based on this equation, considering the weights of the edges between the nodes in the application graph and the distances between the source and destination nodes, the connection cost for each pair of source and destination nodes is calculated. Finally, by summing all the connection costs between the source-destination pairs, the total communication cost is determined [21]. Since determining the optimal mapping from the available mappings requires evaluating all possible mapping configurations, this problem is NP-hard. Therefore, an algorithm needs to be proposed that can solve this problem within a reasonable amount of time.

$$CommCost = \sum_{k=1}^{|E|} Vl(d^k) \times dist(source(d^k), dest(d^k)) \tag{1}$$

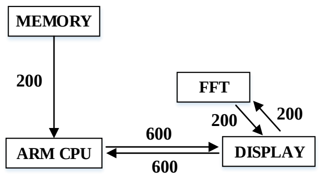

Fig. 1. An example of an application graph

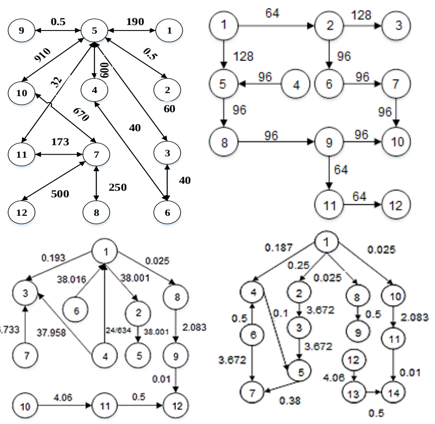

Fig. 2. An example of the usage graph [22], a) MPEG4 graph, b) MWD graph, c) 263enc mp3dec graph, d) 263enc mp3dec graph

where *dist(source(d), dest(d))* is the distance (hop number) between a source and a destination. *Vl (d)* is the weight of the edge between the source and the destination. This weight is multiplied by the distance between the source and the destination.

C) Frog Leaping Algorithm

The frog leaping algorithm is a nature-inspired optimization algorithm based on the behavior of frog groups. In this algorithm, each frog represents a solution to a problem [23]. The frog leaping Algorithm starts with an initial population of possible solutions, which essentially represent subsets of virtual machines and are divided into several groups. Some frogs have characteristics that can be influenced by the features of the entire group. This algorithm includes local search, which is similar to the particle swarm optimization (PSO) algorithm, where frogs improve their positions towards food sources, share information among each other, and then compare this information with the local search [24,25]. As a hybrid algorithm, the frog leaping algorithm is capable of solving many complex, nonlinear, and multimodal problems [26,27].

D) Tabu Search Algorithm

The Tabusearch algorithm is a meta-heuristic optimization algorithm. The general structure of this algorithm involves starting with an initial solution to achieve an optimal solution for a given problem. The algorithm then selects the best neighboring solution from the current solution’s neighbors. If this solution is not in the Tabu list, the algorithm moves to the neighbour’s solution. Otherwise, the algorithm checks a criterion called the "aspiration criterion". According to this criterion, if the neighboring solution is the best solution found so far, the algorithm will move forward even if the solution is in the Tabu list.

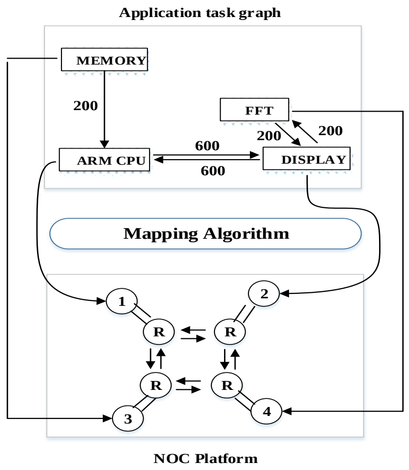

Fig. 3. An example of graph mapping on chip - based tiles

After moving to the neighbouring solution, the Tabu list is updated. This means that the previous move is added to the Tabu list to prevent repetition and avoid getting stuck in a local optimum. The Tabulist is essentially a tool used in the algorithm to avoid revisiting the same solutions [28,29]. After a certain period, moves previously placed on the tabu list are removed. The duration for which a move stays in the Tabulist is determined by a parameter known as tabu time. The movement from the current solution to the neighboring solution continues until a termination condition is met. Different termination criteria may be used for the algorithm, such as a limit on the number of moves to neighboring solutions. The advantage of this approach is that it uses intelligent memory to prevent the generation of repetitive solutions [30,31].

3. Previous Studies

As shown in fig. 4, there are two types of mapping algorithms: dynamic and static [32]. In the dynamic mapping method, the mapping occurs during execution, resulting in higher energy consumption, increased computational overhead, and longer mean time delay. Additionally, implementing this type of mapping can be challenging. Static mapping methods are divided into two categories: accurate mapping and search-based mapping. Additionally, accurate mapping algorithms are capable of finding the optimal solution, but when the number of nodes is high, this method performs poorly. As a result, the execution time increases exponentially. On the other hand, search algorithms are divided into two categories: deterministic search algorithms and heuristic algorithms, which can be optimized between these two sets of iterative search algorithms. In this paper, we use this method to map the graph nodes of NoC tiles. One of the key challenges in the domain of NoC is the mapping process, which has garnered significant attention from researchers. Since mapping algorithms directly affect the cost of the network, selecting the appropriate algorithm is crucial. The goal of this study is to introduce a low-cost mapping algorithm for NoC. Additionally, this paper will review some of the previously proposed mapping algorithms used to embed graph nodes onto the chip area.

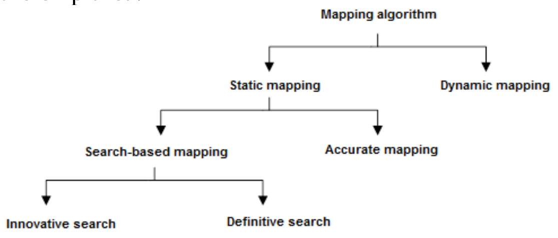

Fig. 4. Types of methods in order to map the application graph on 2 D chip

In [33], a mapping algorithm called NMAP is introduced. The method of this algorithm is as follows: first, a node of the application graph with the highest communication need is identified. Then, this node is mapped to the center of the network on the chip. Afterward, its neighbors, which have the most interactions with it, are identified and mapped in its vicinity. Finally, the Dijkstra algorithm is applied to find the shortest path. The advantage of this method is that it reduces communication costs. However, it cannot achieve the optimal communication cost for all application graphs.

In [34], a mapping algorithm called Tabu Search is introduced. In this method, a random mapping is generated, which is then stored in memory along with its associated communication cost. Two tiles are then randomly selected from the last mapping stored in memory, and tasks within these tiles are swapped. If a new mapping is generated and it has not been stored before, it is stored in memory with its corresponding communication cost. The algorithm continues iterating until the termination condition is met. The mapping with the lowest communication cost is selected as the final mapping. The advantage of this method is the use of memory to avoid repetitive mappings, but it is only suitable for application graphs with a small number of nodes.

In [35], a mapping algorithm called Elixir is presented. In this method, first, a set of mappings with low communication costs is generated using a tree search. Based on the obtained mappings, the mapping with the highest energy consumption and the lowest delay is selected as the final mapping. In addition to communication cost, energy consumption and delay are also considered as criteria. However, this method can result in multiple possible solutions.

In [36], a mapping algorithm called Ant Colony Optimization (ACO) is introduced. This method uses the ant colony algorithm for mapping tasks. Initially, several random mappings are generated. Then, the mapping with the lowest communication cost is selected. Other mappings attempt to adjust their layout to approach this mapping. In this way, the optimal mapping is found. The advantage of this method is that it reduces communication costs, but it performs well only for a limited number of application graphs.

In [37], a mapping algorithm called the frog leaping algorithm is introduced. In this method, several mappings are generated randomly, and then the mappings are sorted in ascending order based on their communication cost. In the next step, the sorted mappings are classified. Additionally, in the frog leaping algorithm, each cluster is followed by the worst mapping. If the communication cost of a new mapping is lower than that of the worst mapping, the new mapping replaces the worst one. Then, the clusters are recalculated based on communication costs. If the cost of the new mapping is higher than the worst mapping, the original worst mapping is retained. This process continues to find the optimal mapping. The advantage of this method is that it works well for application graphs with a large number of nodes. However, the best mappings are placed in the first cluster, leading to an unfair distribution.

In [38], a mapping algorithm called PSMAP is introduced. In this method, several random mappings are generated, and the total communication cost of all mappings is calculated. The mapping with the lowest communication cost is selected as the best and stored in memory. The algorithm repeats these steps until the termination condition is reached. The advantage of this method is that it reduces communication costs, but it is not suitable for graphs with a large number of nodes.

In [39], a mapping algorithm called BMAP (Binomial Mapping) is presented. This algorithm consists of three stages: rank computation, integration, and kernel collector update. The main advantage of this algorithm is that it can reduce communication costs for many graphs, but it has high complexity and adds extra load to the system in order to achieve optimal mapping.

In [40], a static mapping algorithm called the improved shuffled frog leaping algorithm (ISFLA) for 2D mesh topology in network-on-chip is proposed. In this method, several random mappings are generated for the application graph, and the communication cost of all mappings is calculated. The mappings are then arranged in ascending order based on their communication cost. Additionally, the first and last members of each cluster are improved until the termination condition of the algorithm is met. The advantage of this method is that it reduces communication costs, but the method expands the search space, making it unsuitable for application graphs with a large number of nodes. Furthermore, the best mappings are placed in the last cluster, resulting in an unfair distribution.

In [41], a weed-based mapping algorithm is introduced. In this method, several solutions are randomly generated for the application graph. The total communication cost of all solutions is then calculated. Based on the merit of each solution, one or more neighboring solutions are constructed by manipulating two positions in the layout. The communication cost of the new layout is calculated. If the number of solutions exceeds a predefined limit, solutions with higher communication costs are discarded. This process is repeated several times to reach an optimal solution. The advantage of this method is that it reduces communication costs, but it can lead to multiple solutions.Table 1 summarizes the recent proposed mapping algorithms, highlighting their advantages, disadvantages, and differences from the mapping algorithm proposed in this study.

4. Problem Statement

One of the key topics in the domain of NoC is the selection of a suitable mapping algorithm. Choosing the right mapping algorithm can significantly enhance the performance of the network-on-chip. Various mapping algorithms have been proposed for 2D chip networks; however, most of them suffer from one or more of the following issues:

- Some methods provide optimal results only for a limited set of application graphs
- Some methods are only suitable for graphs with a very small number of nodes
- Some methods incur high communication costs, leading to increased network overload
- Some methods yield multiple final solutions

Therefore, it is necessary to propose an algorithm that:

- Is applicable to all types of graphs
- Has low communication cost

Given these issues, mapping algorithms need to be optimized and precise to reduce communication costs. Therefore, in Section 5, a mapping algorithm will be introduced that addresses the aforementioned problems.

5. FTMA Mapping Algorithm

In this paper, a static mapping algorithm based on an innovative search algorithm is proposed for embedding application graph nodes into NoC tiles. The proposed mapping algorithm is designed for a mesh topology NoC and demonstrates lower communication costs compared to the latest works in the field. The reason for selecting mesh topology for this study is its advantages, including simple implementation, smaller space requirements compared to other topologies, high scalability, and greater path diversity. These benefits have led to the widespread use of mesh topology over other topologies in most studies.

Additionally, the proposed method uses the XY routing algorithm to compute the number of steps from the source node to the destination, as this algorithm offers easy implementation and is free from issues like deadlocks and lovelocks. The proposed mapping algorithm combines the advantages of frog leaping and Tabu search algorithms. It begins by receiving the application graph as input. Next, several mappings are randomly generated for the input application graph. In the following step, the mapping with the least communication cost is stored as the best local mapping in memory, while the other mappings are discarded. The proposed algorithm then enters several classes, each of which is generated based on a specific pattern and stored in memory. These classes are repeated until a mapping is found whose cost is lower than that of the local mapping. At this point, the new mapping is stored as the best local mapping, and the existing mappings are discarded.

The algorithm then repeats the steps until the final condition is met. Fig. 5 illustrates the proposed mapping algorithm, which is discussed in section 5-A.

A) The Details of the Proposed Method

In this section, the details of the mapping algorithm, as shown in Fig. 5 in the previous section, are explained in full. Based on the data and information presented in that figure, the various steps of the algorithm and the methods used are examined. The aim of these explanations is to clarify how the components are mapped, introduce the techniques employed, and predict the outcomes of this process. Ultimately, this section helps provide a better understanding of the algorithm's functionality and its applications, offering a clear picture of how it is executed.

Table.1.
Recent proposed mapping algorithms

Algorithm	***Recommended method***	***Advantages***	***Disadvantages***	***Difference fromFTMA***	***Ref.***
NMAP	Considering the communication requirement of graph nodes, the mapping function is used.	Decrease the communication cost	Just for a limited number of application graph we have optimal results.	Optimal results have been used for most graph algorithms	[33]
Tabu-Search	First, the algorithm generates a random mapping, and then the tasks of two tiles use the last map stored in memory randomly.	Not producing repetitive graphs	It is only suitable for applications graph with low number of nodes	It is suitable for the application graphs with a large number of nodes	[34]
Elixir	In addition, we use a mapping tree search tree.	Consider two measures of consumption energy and delay	Has several final answers.	Has a final answer	[35]
ACO	There is a mapping procedure based on the ant colony optimization algorithm.	Decrease the communication cost	Only for a limited number of application graph we have optimal results.	Optimal results have been used for most graph algorithms	[36]
Frog-Leaping	First, a number of mappings are produced randomly. Then the mappings are classified according to a specific pattern. In the end, a mapping is generated randomly for each category and if its cost is lower than the cost of the worst mappings in the category, then it is replaced	Improving the communication cost	Unfair classification of mappings	Fair mapping of mappings	[37]
PSMAP	As long as the algorithm terminates, a number of mappings are generated by accident and the mapping which has the lowest link cost is selected as the best mapping and stored in memory.	It is suitable for the application graphs with a large number of nodes	It is only suitable for applications graph with low number of nodes	It is suitable for application graphs with a large number of nodes	[38]
BMAP	Through three steps, the algorithm performs the mapping and update of the core of the mapping function.	Decrease the communication cost	High complexity, high overhead	The simplicity of the algorithm	[39]
ISFLA	By improving the frogmapping algorithm, we perform a mapping function.	Decrease the communication cost	It is only suitable for the application graphs with a small number of nodes	It is suitable for application graphs of nodes with a large number of nodes	[40]
IAM	By using a weed algorithm, it performs a mapping function.	Decrease the communication cost	Unequal classification of mappings	The fair classification of mappings is a final solution.	[41]

In order to clearly illustrate the adaptive behavior of the proposed FTMA algorithm, the process in this section is represented in a state-oriented manner. Accordingly, each "Box" in the flowchart refers to a distinct decision stage rather than a fixed sequential step. Depending on the communication cost evaluation at each stage, the control flow may return to a previous box or move forward, which is consistent with the dynamic search mechanism of hybrid Tabu and frog-leaping metaheuristics. Furthermore, following the conventions used in shuffled frog-leaping literature, the terms "class" and "cluster" are used with the same meaning to denote groups of mappings that are processed in parallel during the exploration phase.

a) Inputs of FTMA mapping algorithm: In the proposed method, the required inputs for mapping the application graph onto NoC tiles are provided. These inputs include the application graph, mesh topology size, number of classes, number of members in each cluster, number of iterations, and the cost function. The following provides a more detailed explanation of each input for the mapping algorithm.

• Mesh Topology Size: In the proposed algorithm, the mesh topology size is directly determined based on the number of nodes in the application graph. For example, if the input application graph has 9 nodes, a 3×3 mesh topology with 9 nodes is chosen. This choice significantly affects the node allocation and the design of the network structure, directly influencing the system's performance.

• Number of Classes and Members in Each Cluster: Similar to the frog leaping algorithm, the generated mappings in this method are classified into clusters. The number of nodes and the number of mappings in each cluster must be provided as input to the simulator. These inputs help the algorithm perform more efficient classification and search, improving the overall optimization process and ensuring better results.

• Number of Iterations: One termination condition of the proposed mapping algorithm is based on a fixed number of iterations. The algorithm is executed for a certain number of iterations, with each iteration bringing the algorithm closer to an optimal solution. The number of iterations is given as input, effectively controlling the execution time of the algorithm and ensuring it converges within the desired range.

• Cost Function: Another termination condition is when the cost function of the best mapping generated during the algorithm execution becomes lower than the initial cost function. This cost function could represent communication costs or other performance metrics that are continuously updated throughout the process. When the cost reaches an acceptable level, the algorithm terminates. The cost function is introduced as another input to the algorithm, helping to optimize the mapping process and ensuring efficient results.

These inputs effectively guide the mapping process, ensuring that the algorithm can produce the best possible results in the shortest time.

b) Generating random mapping and calculation of communication cost: In this method, a number of mappings are randomly generated based on the number of classes and members of each cluster. For example, if there are 5 classes and each cluster have 6 members, the algorithm will randomly generate 30 mappings ($30=6\times5$). Then, the communication cost for all 30 mappings is calculated. Afterward, these mappings are randomly distributed among the classes, with each class containing 6 mappings. As seen in the proposed method for solving the unfair distribution problem in the frog-leaping mapping algorithm, the mappings are randomly assigned to the clusters [Sections 1 to 3, Box 1 (Fig. 5)]. It is also important to note that the random mapping refers to the random assignment of the application graph nodes to network-on-chip tiles, without considering the communication cost or any other parameters. In this method, a set of numbers is randomly generated based on the number of nodes in the application graph. These numbers are then mapped to the processing elements of the network-on-chip. For instance, if the application graph has 9 nodes, the method will randomly generate 9 numbers between 1 and 9. These numbers are then mapped to the processing elements of the network-on-chip (Fig. 6).

c) Arrangement of mappings based on the communication cost: In this step, the generated mappings are sorted in ascending order according to communication cost. Mappings located at the top of the cluster have lower communication costs compared to those located lower down. The communication cost depends on factors such as the distance between components, the number of connections, and the resources required to establish these connections. The main goal is to minimize communication costs in order to optimize the establishment of connections. This cost is calculated using (1).

d) Best local mapping determination: In Section 5-1-3, the mappings within each cluster are arranged so that the best mappings in terms of communication cost are placed at the top of the stack in each cluster. Then, in the next step, a global search is performed to find the best mapping among the clusters. Through this process, only the first member of each cluster, which has the lowest communication cost, is compared with the mapping that has the highest communication cost among the clusters, and the mapping with the lowest cost is selected and stored as the best local mapping (see Box 5).

e) End condition for the FTMA mapping algorithm: As stated in Section 5-A, the FTMA mapping algorithm terminates in two modes:

• First mode: The algorithm is repeated for a specified number of iterations.

• Second mode: The communication cost generated during the execution of the algorithm is less than the total communication cost (Section 6-5).

If either of these two conditions is satisfied, the mapping with the most complex communication cost stored in memory is selected as the final mapping, and then the algorithm terminates.

f) The way of performing box 2 in the FTMA Mapping algorithm: In the FTMA mapping algorithm, after determining the best local mapping, if the termination condition is not satisfied (Section 6), the algorithm enters Box 2. In this phase, the tasks related to the last mapping stored in memory are carried out together. During this process, a new mapping is developed. Then, two modes may arise:

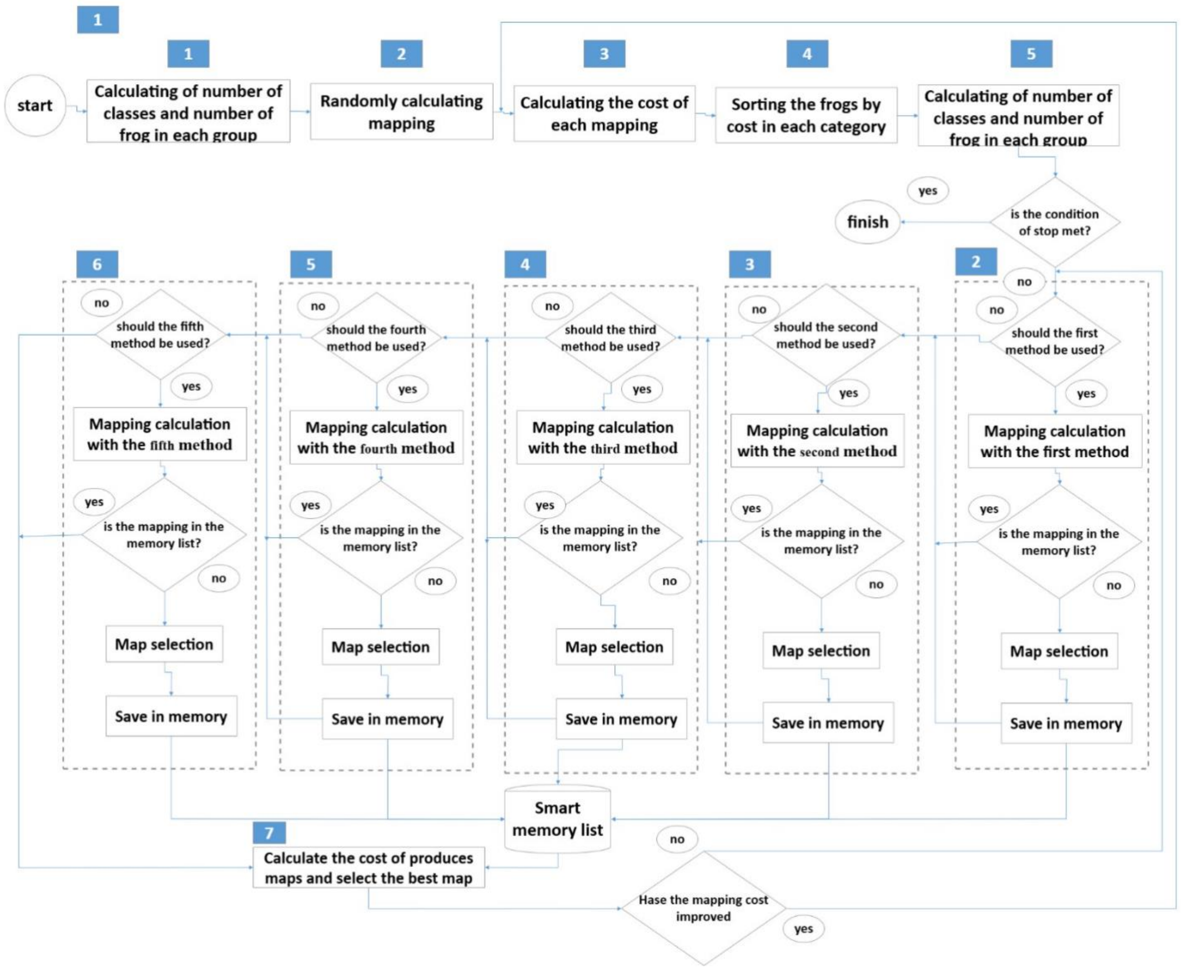
1
start
1
Calculating of number of classes and number of frog in each group
2
Randomly calculating mapping
3
Calculating the cost of each mapping
4
Sorting the frogs by cost in each category
5
Calculating of number of classes and number of frog in each group
yes
finish
is the condition of stop met?
no
6
should the fifth method be used?
Mapping calculation with the fifth method
is the mapping in the memory list?
Map selection
Save in memory
5
should the fourth method be used?
Mapping calculation with the fourth method
4
should the third method be used?
Mapping calculation with the third method
3
should the second method be used?
Mapping calculation with the second method
2
should the first method be used?
Mapping calculation with the first method
yes
no
Smart memory list
7
Calculate the cost of produces maps and select the best map
Hase the mapping cost improved
no
yes

Fig. 5. flowchart of the FTMA Mapping Algorithm

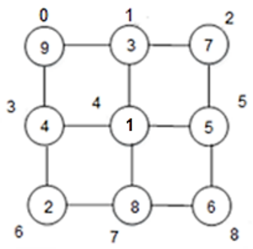

Fig. 6. An example of how to create a random mapping in the FTMA mapping algorithm

• First mode: If the new mapping does not exist in memory, the algorithm adds the mapping to memory.

• Second mode: If the mapping is already stored in memory, the algorithm switches to Box 3. Box 3 is responsible for reviewing and selecting the final mapping from the mappings stored in memory and completes the optimization process. This part of the algorithm is explained in the next section.

g) The way box 3 performs in the FTMA Mapping algorithm: As shown in Section 5-6-1, in two control modes, the algorithm is directed to Box 3. In this box, the data of the last stored mapping is combined with the tasks of the first nodes and the results of the simulation function. In this process, a new mapping is generated. Then, the remaining steps of the FTMA mapping algorithm are implemented in the same dialog box.

h) The performance of Box 4 in the FTMA Mapping Algorithm: In Box 4, the tasks of the first node are combined with the intermediate (central) nodes in the memory of the last stored mapping. In this process, a new mapping is generated. Then, the remaining steps of the FTMA mapping algorithm in Box 4 are implemented in the same manner as in Boxes 3 and 2.

i) The performance way of Box 5 in the FTMA Mapping Algorithm: In Box 5, the last mapping stored in memory is applied alternately. In this process, a new mapping is obtained. Then, the remaining five FTMA mapping algorithms are performed in the same way as in Boxes 4, 3, and 2.

j) The performance of box 6 in the FTMA Mapping Algorithm: In Box 6 - Figure 5, a mapping is generated randomly. Then, the remaining six FTMA mapping algorithms are executed in the same manner as those in Boxes 5, 4, 3, and 2.

k) Calculating the communication cost of the generated mapping and the best mappings

In Section 7, the communication cost of all existing mappings in memory is calculated. Then, the communication cost of the lowest local mapping is compared with that of the best local mapping:

• First case: If the communication cost of the least application graph is lower than the cost obtained from the best local mapping, the algorithm control is transferred to Section 3 of Box 1 in Figure 5, and the algorithm steps are repeated.

• Second case: If the communication cost of the least application graph is not lower than the cost of the best local mapping, the algorithm control is transferred to Box 2 in Fig. 5, and the algorithm steps are repeated.

In this way, the steps of the FTMA mapping algorithm continue until the end condition of the algorithm is met.

B) *An example of the performance of the FTMA mapping algorithm*

To better understand the FTMA mapping algorithm in this section, an example is provided. It is assumed that the input application graph is the one shown in Figure 7, which contains nine nodes in the form of this graph. Thus, the mesh topology size is initialized as 3 × 3. In this example, the number of classes, the number of members in each cluster, the number of iterations, and the total communication cost are set to 2, 3, 4, and 5, respectively.

In the first step of the FTMA mapping algorithm, since the number of nodes is 2 and the number of members in each cluster is 3, 6 (6 = 3 × 2) mappings are generated randomly. The total cost of all these six mappings is then calculated, as shown in Figure 8.

After that, as shown in Figure 9, these mappings are randomly placed into the clusters, ensuring that each cluster contains three mappings. In the second step, the FTMA mapping algorithm is applied as shown in Figure 10. In each cluster, a local search is conducted, and the existing mappings are arranged in ascending order based on their communication cost. In the third stage, the FTMA mapping algorithm performs a global search between the two classes, and the first mapping from each cluster (mappings 4 and 2) are compared.

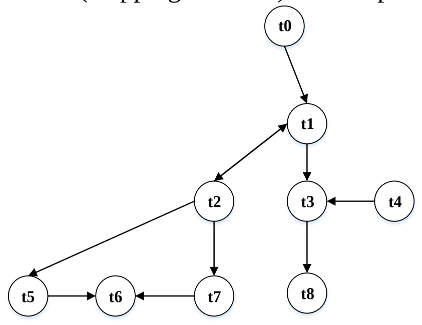

Fig. 7. An example of the input application graph in the FTMA mapping algorithm

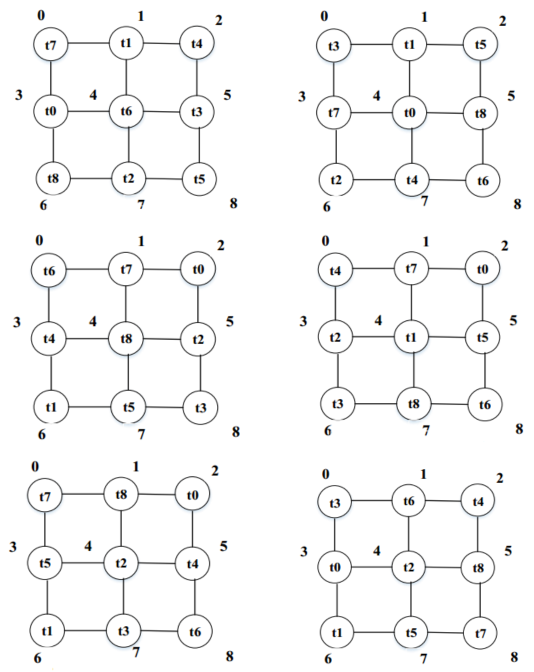

Fig. 8. An example of random mapping algorithm generating. (a): mapping no.1 with communication cost=12, (b): mapping no.2 with communication cost=9, (c): mapping no.3 with communication cost=24, (d): mapping no.4 with communication cost=8, (e): mapping no.5 with communication cost=14, (f): mapping no.6 with communication cost=26

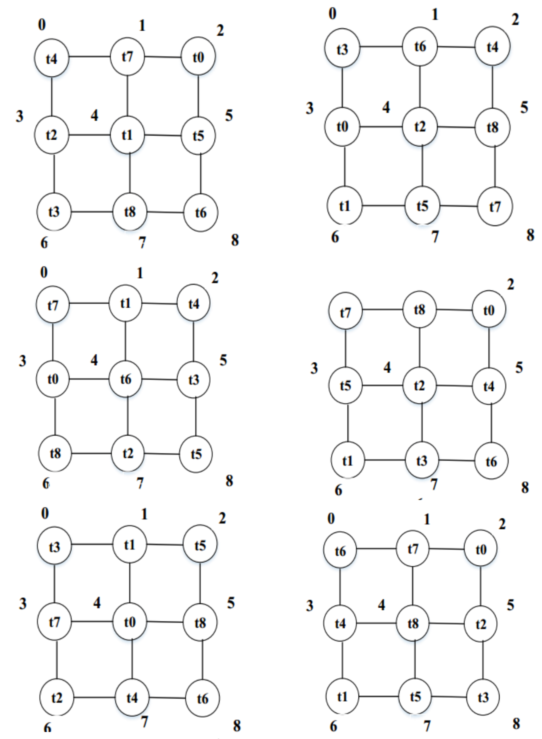

Fig. 9. An example of a mapping class in the FTMA mapping algorithm. First category contains figures (a), (b) and (c). second category contains figures (d), (e) and (f). (a): mapping no.6 with communication cost=26, (b): mapping no.4 with communication cost=8, (c): mapping no.3 with communication cost=24, (d): mapping no.5 with communication cost=14, (e): mapping no.2 with communication cost=9, (f): mapping no.1 with communication cost=12

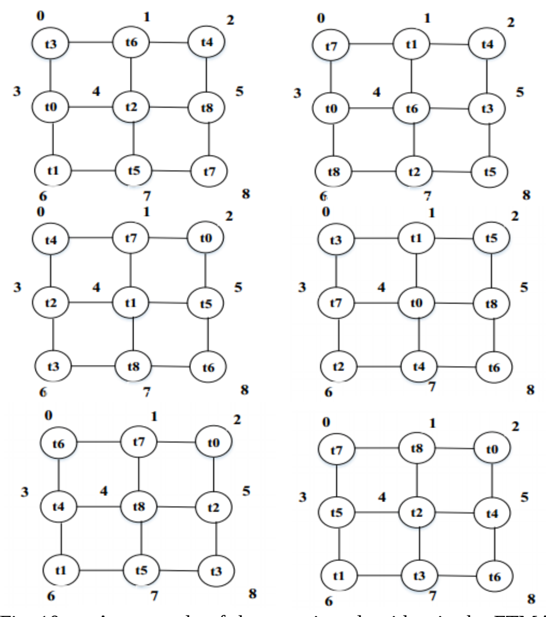

Fig. 10. An example of the mapping algorithm in the FTMA mapping algorithm. First category contains figures (a), (b) and (c). second category contains figures (d), (e) and (f). (a): mapping no.4 with communication cost=8, (b): mapping no.3 with communication cost=24, (c): mapping no.6 with communication cost=26, (d): mapping no.2 with communication cost=9, (e): mapping no.1 with communication cost=12, (f): mapping no.5 with communication cost=14

The mapping with the lowest communication cost (mapping 4) is then selected as the best local mapping and stored in memory. In the fourth stage, the algorithm checks the termination condition, and since it is not met, the algorithm proceeds to Box 2.

In the fifth stage, the tasks of the last mapping stored in memory (mapping 4) are alternately swapped (even & odd) with each other, as shown in Figure 11. Since this mapping does not already exist in memory, it is then added to memory. Finally, the control of the algorithm is transferred to box 3.

In the sixth step, in the last mapping stored in memory (shown in Figure 11), the tasks of the first nodes are swapped with the tasks of the last manipulated nodes, as shown in Figure 12. Since this mapping does not already exist in memory, it is added to memory. Additionally, because the communication cost of this mapping is lower than the global communication cost (5), the algorithm's termination condition is met. In this case, the mapping with the lowest communication cost (the mappings from Figure 12) is selected as the final mapping, and the algorithm terminates.

In this section, the details of the proposed mapping algorithm are presented with an example to better illustrate the FTMA mapping algorithm. To demonstrate the superior performance of the proposed method compared to other methods, the simulation results of the FTMA mapping algorithm and its comparison with the latest similar methods will be presented in the next section.

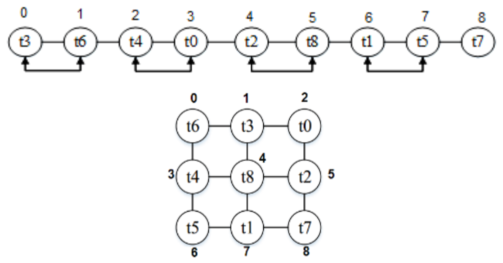

Fig. 11. An example of creating a mapping via Box 2 in the FTMA Mapping Algorithm. (a): Swapping even and odd elements, (b): The mapping created through class 1 with the cost of 36

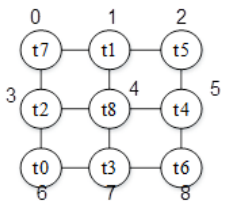

Fig. 12. An example of generating a mapping through class 2 in the FTMA mapping algorithm with cost of 3

6. Simulation Results

To evaluate the communication cost of the proposed mapping algorithm, MATLAB software has been used. To investigate the communication cost, four well-known application graphs—MPEG4, 263, mp3dec, and 263 vehicles—were utilized. The reason for using different application graphs is to demonstrate that the proposed method performs well for application graphs with varying numbers of nodes. Figure 3 shows the application graph used to simulate the proposed method. Additionally, based on the number of nodes, the mesh topology sizes are set to 3×3 and 4×4. It should be noted that if the number of nodes in the graph is less than the number of nodes in the mesh topology, zeros are placed in the empty mesh topology nodes. Furthermore, the number of classes and the number of members in each cluster for the MWD, 263, mp3dec, and 263 vehicles graphs are considered as 5 and 2, 2 and 4, and 4 and 4, respectively.

Another point is that to ensure consistent simulations, the proposed method is implemented on a system with the same specifications. The basic features of the system are: a dual-core i5 CPU @ 2.4 GHz with 4 GB of RAM.

Fig. 13 presents the evaluation results of the proposed method in this paper compared to recent works. Each of the plots in this figure shows the evaluation results for one of the application graphs. As seen in the figure, the mapping generated by the proposed algorithm has a lower communication cost compared to other algorithms. This improvement is the result of using the Tabu and Frog Leaping algorithms to find the optimal mapping among the available mappings for the network.

This result is also observed in Figure 13(b). In this figure, the communication cost of the mapping created by the proposed algorithm is 1086, while the best communication cost among previous works belongs to PSMA with a communication cost of 1120. As shown in the figure, the use of a combination of Tabu and Frog Leaping optimization algorithms leads to a significant reduction in the communication cost of the network mapping.

The evaluation results of the FTMA algorithm on the 263encmp3enc graph are shown in Figure 13(c). These results demonstrate the improvement in the communication cost of the network mapping by the proposed method compared to other algorithms. The communication cost of the mapping by the FTMA algorithm on this graph is 189.31, while for other algorithms, this value is higher. In this figure, the communication costs of the NMAP, PSMA, LMAP, and BMA algorithms are close to each other and significantly higher than FTMA. Although the ISFLA and IAM algorithms have lower communication costs than other algorithms, their communication costs are still higher than FTMA.

These results indicate that the Tabu and Frog Leaping algorithms have played an important role in finding the best network mapping with the lowest cost. Fig. 13(d) shows the evaluation results based on communication cost for the 263decmp3enc graph. This plot illustrates that the use of the proposed algorithm for network-on-chip mapping results in a reduction in communication cost. In this case, the performance of the algorithm shows a significant improvement compared to other algorithms. In order to gain a better understanding of the performance of the proposed algorithm in this paper, we next discuss and examine the average improvement in communication cost compared to other algorithms. Table 2 shows the percentage improvement in the communication cost of the FTMA mapping algorithm of the proposed method compared to the latest methods in this area. The table clearly illustrates how the proposed method performs better in reducing communication costs compared to other methods.

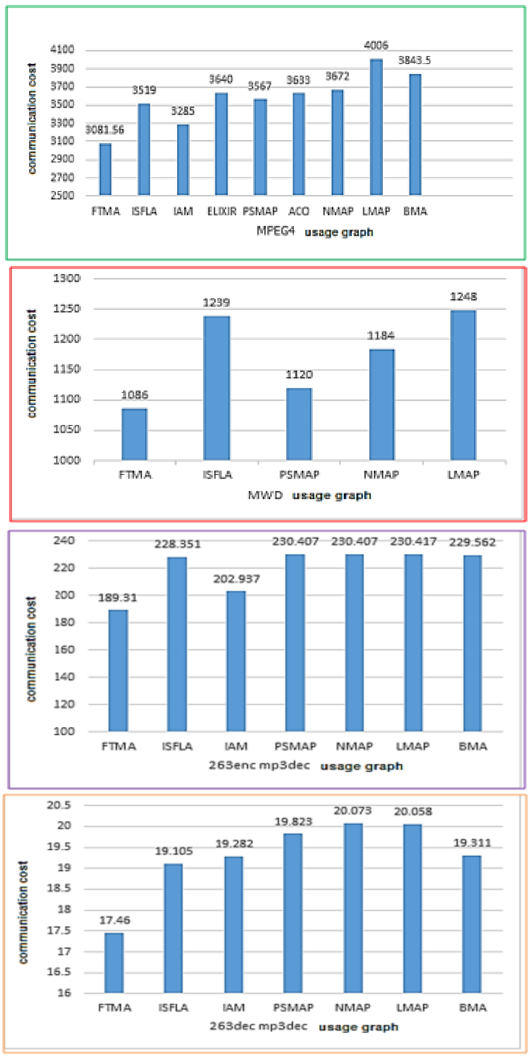

Fig. 13. Comparison of the communication cost between the proposed method and the latest similar methods, a) MPEG4 graph, b) MWD graph, c) 263encmp3enc graph, d) 263decmp3enc graph

According to the data, it is evident that the proposed method incurs lower communication costs than the alternatives. This indicates that the proposed method is more effective in reducing communication overhead, which in turn reduces the overall system load and enhances its overall performance. The last row of Table 2 shows the average improvement in communication cost of the proposed algorithm compared to other algorithms. As seen in this row, on average, the proposed algorithm achieves a 13.66% reduction in communication cost compared to the other algorithms, indicating the effective role of the Tabu and Frog Leaping algorithms in searching for the best mapping among the available mappings for the network-on-chip mapping problem. In general, the results presented in this table show that the proposed method outperforms other methods in terms of reducing communication costs and can have a positive impact on real-world applications and complex systems. Fig. 14 illustrates the changes in communication costs for the proposed method and three other methods, based on the number of nodes in the application graph. In this figure, the Y-axis on the left side represents the communication cost for the MPEG4 application graph, one of the common graphs used in this study, particularly in multimedia and video processing applications. On the right side of the Y-axis, the communication cost for the 263dec mp3de application graph is shown. As shown in the figure, the proposed method maintains an acceptable communication cost even as the number of nodes in the application graph increases. This is particularly important, as in complex systems that require extensive communication between nodes, keeping communication costs low directly impacts performance and processing speed.

In other words, an increase in the number of nodes should not result in a disproportionate increase in communication costs, and the proposed method effectively mitigates this issue. These results demonstrate the advantage of the proposed method in optimizing communication costs across different types of graphs and its impact on improving the performance of complex systems under various conditions.

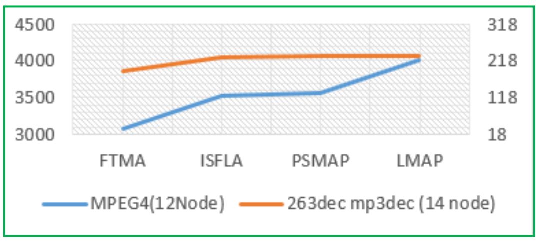

Fig. 14. Chart of increasing of the number of nodes and communication costs

Reducing communication costs is especially important for complex systems that require extensive processing and communication. Specifically, this improvement can lead to faster processing times and greater system efficiency. By employing the proposed method, not only are communication costs minimized, but system performance is also significantly improved, making it a suitable choice for practical applications and systems with specific needs. The reason for this is that the FTMA algorithm is a combination of two swarm optimization algorithms: The Tabu search algorithm and the frog leaping algorithm. The Frog Leaping algorithm is capable of finding mappings with the minimum communication cost for application graphs with a small number of nodes. On the other hand, the Tabu Search algorithm, using its intelligent memory, can identify the best mapping with the lowest communication cost. Therefore, due to the combination of these two algorithms in the proposed method, its associated communication cost has been significantly improved compared to other methods.

Table.2.
Improvement in the communication cost of the FTMA mapping algorithm to other mapping algorithms.

Task Graph / *Method*	*MPEG4*	*MWD*	*263enc mp3dec*	*263dec mp3dec*	*Average of improvement percentage*
ISFLA [39]	12.43%	12.35%	17.097%	8.61%	12.62%
IAM [42]	6.19%	-	6.71%	9.45%	7.45%
ELIXIR [35]	15.34%	-	-	-	15.34%
PSMAP [37]	13.61%	3.036%	17.84%	11.92%	11.60%
ACO [36]	15.18%	-	-	-	15.18%
NMAP [34]	16.08%	8.28%	17.84%	13.02%	13.805%
BMA [38]	19.82%	-	17.53%	9.58%	15.64%
		Average			13.66%

7. Conclusion

In this paper, a mapping algorithm based on the Frog Leaping Algorithm and Tabu Search is proposed, aimed at reducing communication costs compared to recent similar methods in this field. The proposed mapping algorithm uses a two-dimensional mesh topology-on-chip method, offering advantages such as high performance for application graphs with a large number of nodes by limiting the state space. This method utilizes smart memory to prevent repetitive costs and can find the best mapping in the shortest possible time. Compared to the latest methods in this area, the proposed method was able to reduce communication costs by 7.45%, 15.34%, 15.18%, 13.805%, 17.68%, 17.68%, and 15.64% for ISFLA, IAM, ELIXIR, PSMAP, ACO, NMAP, LMAP, and BMA, respectively. For future work, it is recommended to apply the proposed mapping algorithm for the mapping of application graphs onto on-chip tiles. Additionally, the proposed method is currently applicable only to two-dimensional network-on-chip systems, but it can be improved in the future to support three-dimensional network-on-chip systems. Furthermore, while the proposed mapping algorithm is designed for mesh topology, it could be extended in the future to work with other types of topologies as well.

References

[1] A. Cakin, S. Dilek, S. Tosun, "Energy-aware application mapping methods for mesh-based hybrid wireless network-on-chips", The Journal of Supercomputing, vol. 80, pp. 15582–15612, July 2024, doi: 10.1007/s11227-024-06062-4.

[2] M. Wang, K.C.M. Lee, B.M.F. Chung, S.V. Bogaraju, H.C. Ng, J.S.J. Wong, "Low-latency in situ image analytics with FPGA-based quantized convolutional neural network", IEEE Trans. on Neural Networks and Learning Systems, vol. 33, no. 7, pp. 2853-2866, July 2022, doi: 10.1109/TNN-LS.2020.3046452.

[3] S. Shafaghi, R. Sabbaghi-Nadooshan, "Optimal path diagnosis by genetic algorithm for NoCs", International Journal of Smart Electrical Engineering, vol. 1, no. 2, pp. 131-136, June 2012, dor: 20.1001.1.22519246.2012-.01.02.8.6.

[4] C. Xu, Y. Liu. P. Li, Y. Yang, "Unified multi-objective mapping for network-on-chip using genetic-based hyper-heuristic algorithms", IET Computers and Digital Techniques, vol. 12, no. 4, pp. 158-166, March 2018, doi: 10.1049/iet-cdt.2017.0156.

[5] C. Marcon, E. Moreno, N. Calazons, F. Moraes, "Comparison of network-on-chip mapping algorithms targeting low energy consumption", IET Computers & Digital Techniques, vol. 2, no. 6, pp. 471-482, Nov. 2008, doi: 10.1049/iet-cdt:20070111.

[6] P. Sharma, S. Biswas, P. Mitra, "Energy efficient heuristic application mapping for 2-D mesh-based network-on-chip", Microprocessors and Microsystems, vol. 64, no. 9, pp. 88-100, Feb. 2019, doi: 10.1016/j.micpro.2018.10.008.

[7] M.Z. Dageleh, M.A. Jamali, "V-CastNet3D: A novel clustering-based mapping in 3-D Network on chip", Nano Communication Networks, vol. 18, no. 5, pp. 51-61, Dec. 2018, doi: 10.1016/j.nancom.2017.11.002.

[8] N. Ghorbani, E. Babaei, S. Laali, P. Farhadi, "Per unit coding for combined economic emission load dispatch using smart algorithms", International Journal of Smart Electrical Engineering, vol. 5, no. 1, pp. 11-21, March 2016, dor: 20.1001.1.22519246.2016.05.01.3.7.

[9] M. Hemmati, E. Yaghoubi, S.M.A. Zanjani, M. Dolatshahi, "Providing a routing algorithm in network on chip to reduce energy consumption and increase reliability with fuzzy neural network and genetic programming", Journal of Novel Researches on Electrical Power, vol. 10, no. 2, pp. 43-51, June 2021, dor: 20.1001.1.23222468.1400.10.2.5.2.

[10] M.R. Hemmati, S.M.A. Zanjani, E. Yaghoubi, "A new model for enhancing efficiency in on-chip optical networks based on adaptive routing algorithm", Journal of Southern Communication Engineering, vol. 13, no. 51, pp. 13-22, June 2024, doi: 10.30495/jce.2023.1992434.1216.

[11] A. Liu, X. Zhang, Z. Liu, Y. Li, X. Peng, X. Li, Y. Qin, C. Hu, Y. Qiu, H. Jiang, Y. Wang, Y. Li, J. Tang, J. Liu, H. Guo, T. Deng, S. Peng, H. Tian, T.L. Ren, "The roadmap of 2D materials and devices toward chips. nano-micro let", vol. 16, Article Number: 119, Feb. 2024, doi: 10.1007/s40820-023-01273-5.

[12] N. Taherkhani, R. Akbar, F. Safaei, M. Moudi, "A congestion-aware routing algorithm for mesh-based platform networks-on-chip", Microelectronics Journal, vol. 114, Article Number: 105145, Aug. 2021, doi: 10.1016/j.mejo.2021.105145.

[13] P.K. Sahu, S. Chattopadhyay, "A survey on application mapping strategies for Network-on-Chip design", Journal of Systems Architecture, vol. 59, no. 1, pp. 60-76, Jan. 2013, doi: 10.1016/j.sysarc.2012.10.004.

[14] [2024-277] S.P. Kaur, M. Ghose, A. Pathak, R. Patole, "A survey on mapping and scheduling techniques for 3D Network-on-chip", Journal of Systems Architecture, vol. 147, Article Number: 103064, Feb. 2024, doi: 10.1016/j.sysarc.2024.103064.

[15] S.M.A. Zanjani, G. Shahgholian, A. Fathollahi, S.M.H. Zanjani, "Coordination design of power system stabilizer and FACTS controllers using nature-inspired metaheuristics optimization algorithms– A brief review", International Journal of Smart Electrical Engineering, vol. 13, no. 1, pp. 89-106, Dec. 2024, doi: 10.30495/ijsee.2023.1992877.1278.

[16] M. A. Honarvar, G. Shahgholian, H. Mahmoodian, S. Yaghoubi, A. Mosavi and A. Fathollahi, "Reviewing power system stabilizer (PSS) parameters optimization using evolutionary meta-heuristic algorithms for power system stability", Proceeding of the IEEE/SISY, pp. 481-486, Pula, Croatia, Sept. 2023, doi: 10.1109/SISY60376.2023.10417875.

[17] Y. Asadi, "A comprehensive study and holistic review of empowering network-on-chip application mapping through machine learning techniques", Discover Electronics, vol. 1, Article Number: 22, Oct. 2024, doi: 10.1007/s44291-024-00027-w.

[18] R. Phosung, K. Areerak, K. Areerak, "Design and optimization of control system for more electric aircraft power systems using adaptive tabu search algorithm based on state-variables-averaging model", IEEE Access, vol. 12, pp. 76579-76588, May 2024, doi: 10.1109/ACCESS.2024.3406855.

[19] S. Saleem, F. Hussain, W. Amin, R. Ahmed, Y.B. Zikria, F. Ishmanov, "A survey on dynamic application mapping approaches for real-time network-on-chip-based platforms", IEEE Access, vol. 11, pp. 122694-122721, Nov. 2023, doi: 10.1109/ACCESS.2023.3329233.

[20] M. El-Azazy, A.I. Osman, M. Nasr, Y. Ibrahim, N. Al-Hashimi, K. Al-Saad, M.A. Al-Ghouti, M.F. Shibl, A.H. Al-Muhtaseb, D.W. Rooney, A.S. El-Shafie, "The interface of machine learning and carbon quantum dots: From coordinated innovative synthesis to practical application in water control and electrochemistry", Coordination Chemistry Reviews, vol. 517, Article Number: 215976, April 2024, doi: 10.1016/j.ccr.2024.215976.

[21] W. Amin, F. Hussain, S. Anjum, S. Saleem, N.K. Baloch, Y.B. Zikria, H. Yu, "Efficient application mapping approach based on grey wolf optimization for network on chip", Journal of Network and Computer Applications, vol. 219, Article Number: 103729, Oct. 2023, doi: 10.1016/j.jnca.2023.103729.

[22] J. Hu, R. Marculescu, "Energy and performance aware mapping for regular NOC architectures", IEEE Trans. on computer Aided Design of Integrated Circuit and Systems, vol. 24, no. 4, pp. 551-562, 2005, doi: 10.1109/TCAD.2005.844106.

[23] Z. Tang, Y. Wu, J. Wang, T. Ma, "IoT service composition based on improved shuffled frog leaping algorithm", Heliyon, vol. 10, no. 7, Article Number: e28087, April 2024, doi: 10.1016/j.heliyon.2024.e28087.

[24] G. Shahgholian, S. Fazeli-Nejad, M. Moazzami, M. Mahdavian, M. Azadeh, M. Janghorbani, S. Farazpey, "Power system oscillations damping by optimal coordinated design between PSS and STATCOM using PSO and ABC algorithms", Proceeding of the IEEE/ECTI-CON, Chiang Mai, Thailand, pp. 1-6, July 2016, doi: 10.1109/ECTICon.2016.7561458.

[25] G. Shahgholian, M. Mahdavian, M. Noorani-Kalteh, M.R. Janghorbani, "Design of a new IPFC-based damping neurocontrol for enhancing stability of a power system using particle swarm optimization", International Journal of Smart Electrical Engineering, vol. 3, no. 2, pp. 73-78, April 2014, dor: 20.1001.1.22519246.2014.03.02.2.4.

[26] H.P. Hsu, C.N. Wang, "A hybrid approach combining improved shuffled frog-leaping algorithm with dynamic programming for disassembly process planning", IEEE Access, vol. 9, pp. 57743-57756, 2021, doi: 10.1109/ACCESS.2021.3072831.

[27] X. Guo, C. Fan, M. Zhou, S. Liu, J. Wang, S. Qin, "Human–robot collaborative disassembly line balancing problem with stochastic operation time and a solution via multi-objective shuffled frog leaping algorithm", IEEE Trans. on Automation Science and Engineering, vol. 21, no. 3, pp. 4448-4459, July 2024, doi: 10.1109/TASE.2023.3296733.

[28] A.K. Sangaiah, R. Khanduzi, "Tabu search with simulated annealing for solving a location–protection–disruption in hub network", Applied Soft Computing, vol. 114, Article Number: 108056, Jan. 2022, doi: 10.1016/j.asoc.2021.108056.

[29] M.A. Abido, Y.L. Abdel-Magid, "Eigenvalue assignments in multimachine power systems using tabu search algorithm", Computers & Electrical Engineering, vol. 28, no. 6, pp. 527-545, Nov. 2002, doi: 10.1016/S0045-7906(01)00005-2.

[30] R. Phosung, K. Areerak, K. Areerak, "Design and optimization of control system for more electric aircraft power systems using adaptive tabu search algorithm based on state-variables-averaging model", IEEE Access, vol. 12, pp. 76579-76588, May 2024, doi: 10.1109/ACCESS.2024.3406855.

[31] H. Lotfi, "A new hybrid algorithm for multi-objective distribution feeder reconfiguration considering reliability", International Journal of Smart Electrical Engineering, vol. 8, no. 3, pp. 83-92, Sept. 2019, dor: 20.1001.1.22519246.2019.08.03.1.0.

[32] J.P. Matos-Carvalho, F. Moutinho, A.B. Salvado, T. Carrasqueira, R. Campos-Rebelo, D. Pedro, L.M. Campos, J.M. Fonseca, A. Mora, "Static and dynamic algorithms for terrain classification in UAV aerial imagery", Remote Sensing, vol. 11, no. 21, Article Number: 2501, Oct. 2019, doi: 10.3390/rs11212501.

[33] [21] S. Murali, G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures", Proceeding of the IEEE/TATE, pp. 896-901, Paris, France, France, Feb. 2004, doi: 10.1109/DATE.2004.1269002

[34] [22] C. Marcon, E. Moreno, N. Calazons, F. Moraes, "Comparison of network-on-chip mapping algorithms targeting low energy consumption", IET Computers and Digital Techniques, nol. 2, no. 6, pp. 471-482, Nov. 2008, doi: 10.1049/iet-cdt:20070111.

[35] [24] M. Rashedi, A. Khademzadeh, A. Reza, "Elixir: A new band width constrained mapping for Networks-on-chip", IEICE Electronics Express, vol. 7, no. 2, pp. 73-79, Jan. 2010, doi: 10.1587/elex.7.73.

[36] Y. Xie. Y. Liu, "A research on NOC mapping with quantum ant colony algorithm", Proceeding of the IEEE/WiSPNET, pp. 874-877, Chennai, India, March 2017, doi: 10.1109/WiSPNET.2017.8299886.

[37] P. Kaur, S.H. Mehta, "Resource provisioning and work flow scheduling in clouds using augmented shuffled frog leaping algorithm", Journal of Parallel and Distributed Computing, vol. 101, no. 4, pp. 41-50, March 2017, doi: 10.1016/j.jpdc.2016.11.003.

[38] P. Sahu, P. Venkatesh, S. Gollapalli, S. Chattopadhyay, "Application mapping onto mesh structured network-on-chip using particle swarm optimization", Proceeding of the IEEE/ISVLSI, pp. 335-336, Chennai, India, July 2011, doi: 10.1109/ISVLSI.2011.21.

[39] M. Keley, A. Khademzadeh, M. Hosseinzadeh, "Efficient mapping algorithm on mesh-based NoCs in terms of cellular learning automata", International Arab Journal of Information Technology, vol. 16, no. 2, pp. 312-322, March 2019.

[40] B. Broumand, E. Yaghoubi, B. Barekatain, "An enhanced cost-aware mapping algorithm based on improved shuffled frog leaping in network on chip", The Journal of Supercomputing, vol. 77, pp. 498-522, Jan. 2021, doi: 10.1007/s11227-020-03271-5.

[41] P.M. Kalahroudi, E. Yaghoubi, B. Barekatain, "IAM: an improved mapping on a 2-Dnetwork on chip to reduce communicationcost and energy consumption", Photonic Network Communications, vol. 41, pp. 78-99, Feb. 2021, doi: 10.1007/s11107-020-00911-x, 2020.

[42] T. Maqsood, K. Bilal, S. Madani, "Congestion-aware core mapping for Network-on-Chip based systems using betweenness centrality", Future Generation Computer Systems, vol. 82, no. 5, pp. 459-471, May 2018, doi: 10.1016/j.future.2016.12.031.