

DEMAP: differential evolution mapping for network on chip optimization

Maamar Bougherara^{1,2}, Rafik Amara^{2,3}, Rebiha Kemcha^{2,4}

¹LIMPAF Laboratory, Bouira University, Bouira, Algeria

²Department of Computer Science, High Normal School of Kouba, Algiers, Algeria

³LTIR Laboratory, Faculty of Electronics and Computational Science, USTHB University, Algiers, Algeria

⁴LIMOSE Laboratory, Boumerdes University, Boumerdes, Algeria

Article Info

Article history:

Received November 4, 2022

Revised January 25, 2023

Accepted March 4, 2023

Keywords:

Communication cost

Differential evolution

Mapping

Network on chip

Optimization

ABSTRACT

Network-on-chip (NoC) is a new paradigm for system-on-chip (SoC) design, which facilitates the interconnection and integration of complex components. Since this technology is still new, significant research efforts are needed to accelerate and simplify the design phases. Mapping is a critical phase in the NoC design process, as a mismatch of application software components can significantly impact the final system's performance. Therefore, it is essential to develop automated tools and methods to ensure this step. The main objective of this project is to develop a new approach that can be used to map applications on the NoC architecture to reduce communication costs. To achieve this goal, we have opted for an optimization algorithm, specifically the differential evolution algorithm.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Maamar Bougherara

LIMPAF Laboratory, Bouira University

Bouira, Algeria

Email: bougherara.maamar@gmail.com

1. INTRODUCTION

The network-on-chip (NoC) concept originated from the need to accommodate the increasing size, complexity, and heterogeneity of applications running on system-on-chip (SoC). As the traditional communication bus between components failed to fulfill all the demands, the idea of establishing a network on a chip was conceived based on computer network principles. A critical stage in designing a NoC is the mapping of intellectual property (IP) cores to the architecture. This process has a significant impact on the performance of the system, affecting factors such as energy usage, latency, and load distribution. This process is considered an NP-hard problem, as more than one critical performance factor must be considered to develop the optimal mapping algorithm. Thus, several methods have been proposed in the literature to address this issue, often relying on heuristic algorithms. Differential evolution (DE) is one such algorithm that delivers better performance with lower complexity. However, in this paper, we aim to enhance DE's efficacy by coordinating it with other techniques. The remainder of this paper is organized as follows: section 2 provides an overview of NoC; section 3 highlights some relevant studies; section 4 discusses the application mapping problems, followed by a description of the Mapping problem with the differential evolution algorithm. In section 5, we showcase the experimental outcomes. In the last section, we conclude our work with a conclusion.

2. NOCS GENERALITIE

The idea of NoC is derived from the networks initially developed for supercomputers, comprising a group of interconnected devices on a single chip that communicate through packets sent over a scalable interconnection network. Compared to traditional bus architectures, NoC offers several advantages, such as energy efficiency, reliability, bandwidth scalability, and reusability [1]. The topology of a NoC is determined by how its components are interconnected to create the on-chip interconnect, which can take on various forms, such as 2D mesh, torus, and ring. In addition to the topology, other characteristics of NoC include communication mode, flow control mechanisms to prevent deadlock issues, and storage strategies Figure 1 presents an NoC with 9 tiles in 2D mesh topology.

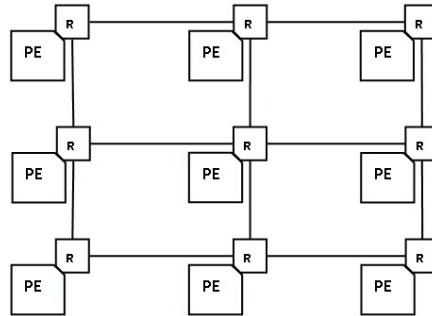


Figure 1. A 3×3 2D mesh NoC

To design a NoC, the system involves multiple stages. Initially, the application is decomposed into a set of communication tasks that can be executed concurrently. After that, each task is assigned to an available core that is selected and scheduled. In the end, these cores must be mapped onto the NoC to complete the system design [2].

This paper specifically concentrates on the final stage of application mapping, which is a critical but still unresolved search problem. The optimal mapping solution can yield energy savings of up to 51.7% compared to ad hoc implementations, as demonstrated in [3]. To achieve high performance, finding the optimal mapping solution is essential. For instance, if there are m tasks to be mapped onto an NoC consisting of n cores where ($m \leq n$), the number of potential solutions can reach up to $n!/(n-m)!$. Application mapping is a combinatorial optimization problem that is classified as NP-hard. In order to find a suboptimal solution, heuristic algorithms are typically used.

3. RELATED WORK

In NoC mapping, various approaches have been proposed, with particular attention given to the two-dimensional mesh topology. In this study, we review the most cited mapping techniques that consider mono-objective mapping. These techniques can be divided into two classes: meta-heuristic algorithms and heuristic approaches.

Meta-heuristic algorithms are widely used to solve NP-hard problems and strive to achieve a solution that is close to optimal. Examples of such algorithms include genetic algorithms, ant colony optimization, and particle swarm optimization. For example, GBMAP [4] and CGMAP [5] use genetic algorithms, and in [6], an ACO-based algorithm is proposed to minimize the bandwidth requirement. In reference [7], a technique for optimizing performance using deterministic initial solutions has been proposed. Specifically, a discrete multiple particle swarm optimization (PSO) based mapping technique was utilized, where the behavior of swarm intelligence serves as a meta-heuristic.

Unlike meta-heuristic algorithms, heuristic approaches are tailored to a specific problem and rely on specific cues to guide the search process. These cues are determined by the nature of the problem being addressed. For instance, NMAP [8] selects an application's core and maps them to tiles repeatedly, while BMAP [9] maps the cores according to traffic loads of cores. The CastNet algorithm described in reference [10] generates multiple solutions for mapping by using multiple tiles as initial tiles. The algorithm uses the

symmetric characteristics of a mesh to determine the optimal solution for each core, the number of available neighboring tiles for each tile is considered.

Chmap [11] determines the priority of each core by analyzing the communication needs and the data in the spanning tree. Next, the algorithm maps the chosen cores to the suitable tiles based on their priorities by establishing the mapping order of the cores. The ONYX algorithm [12] utilizes four moves to assign a core to the tiles on a lozenge-shaped path and achieves a lower communication cost than previous mapping techniques. The Spiral approach, described in [13], involves mapping the task with the highest priority at the center first, followed by the remaining tasks using a spiral path. Exact methods are also used in mapping but require a large amount of calculations over time, such as those based on integer linear programming (ILP) [14], or the branch and bound search method [15].

4. APPLICATION MAPPING PROBLEM

Our research focuses on mapping steps that involve two key inputs: NoC and its architecture, and the application that is supposed to run in the NoC. In our case, we used a NoC with a 2D mesh topology shown in Figure 2. An application is made up of multiple concurrent tasks that need to be placed onto a core on the NoC. The goal of the mapping process is to achieve the best placement with minimum communication cost, which is a crucial factor that we consider in our paper. The optimal mapping solution is one that achieves the best placement while minimizing communication costs. The resulting mapping solution is presented in the form of a table, where each task is represented by an index i , and the contents of the table represent the number of tiles assigned to each task during mapping.

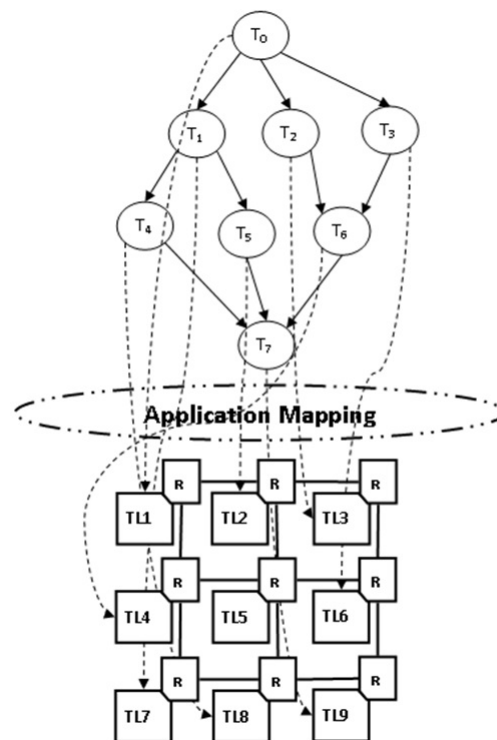


Figure 2. Application mapping problem

4.1. NoC Model

The problem is defined using three distinct definitions.

Definition 1: The core graph is constructed using a directional graph $G(V, E)$. In this graph, each vertex v_i represents a core, while a directional edge $e_{i,j}$ indicates the link between core v_i and core v_j . The weight of edge $e_{i,j}$ reflects the extent of communication between the two corresponding vertices.

Definition 2: A graph $A(T, L)$ represents the NOC architecture, where every vertex t_i denotes a tile

within the NOC architecture. Meanwhile, each edge $l_{i,j}$ represents a physical link originating from t_i and terminating at t_j .

Definition 3: The mapping function that associates each vertex v_i in the core graph with a vertex t_j in the NOC architecture is defined as follows.

$$map : V \rightarrow T \quad map(v_i = t_j), \forall v_i \in V \exists t_j \in T$$

In the core graph, each edge is considered as a flow of a single commodity, denoted as c^k . This value represents the required bandwidth and can be expressed as $vl\ c^k$. The set of all commodities is represented by

$$C = c^k : vl(c^k) = comm_{i,j} \quad k = 1 \dots |E| \quad comm_{i,j} \in E$$

with

$$source(c^k) = map(v_i) \text{ and } destination(c^k) = map(v_j)$$

To determine the quantity of communication between v_i and v_j , it is necessary to count the number of hops between $tile_i$ and $tile_j$. On a 2D mesh NoC, the X-Y routing algorithm is employed, and the number of hops can be determined using (1).

$$Hops(tile_i, tile_j) = |X_i - X_j| + |Y_i - Y_j| \quad (1)$$

In 2D mesh NoC the tiles i and j are represented by (X_i, Y_i) and (X_j, Y_j) , respectively.

4.2. Objective function

This paper aims to minimize communication costs when mapping two tasks onto the NoC. The primary strategy involves reducing the number of hops required for each communication within the application. The calculation of the communication cost formula involves utilizing (2).

$$commcost = \sum_{k=1}^{|E|} vl(C^k) * nbhops(src(C^k), dist(C^k)) \quad (2)$$

The source of a communication C^k is denoted by $src(C^k)$, while its destination is represented by $dist(C^k)$.

4.3. Differential evolution (DE)

In 1997, Price and Storn [16] proposed DE as an enhanced version of genetic algorithms. Like genetic algorithms, DE relies on an initial population and employs the same operator's crossover, mutation, and selection. However, DE utilizes these operators in a different order, as illustrated in Figure 3.

The primary distinction between genetic algorithms and DE lies in their respective approaches to building better solutions. Genetic algorithms utilize crossover, while DE relies on the operation of mutation. In DE, the mutation operation is the primary mechanism for searching and exploring potential regions in the search space, and the selection operator directs convergence toward those regions.

According to [16], the $DE/x/y/z$ notation is frequently employed to describe a DE strategy. The symbol x denotes the vector that will undergo mutation, which may be either a randomly selected population vector (*rand*) or the vector with the lowest cost in the current population (*best*). y specifies the number of differential vectors employed to perturb the target vector, and z denotes the crossover scheme, which could be either exponential or binomial.

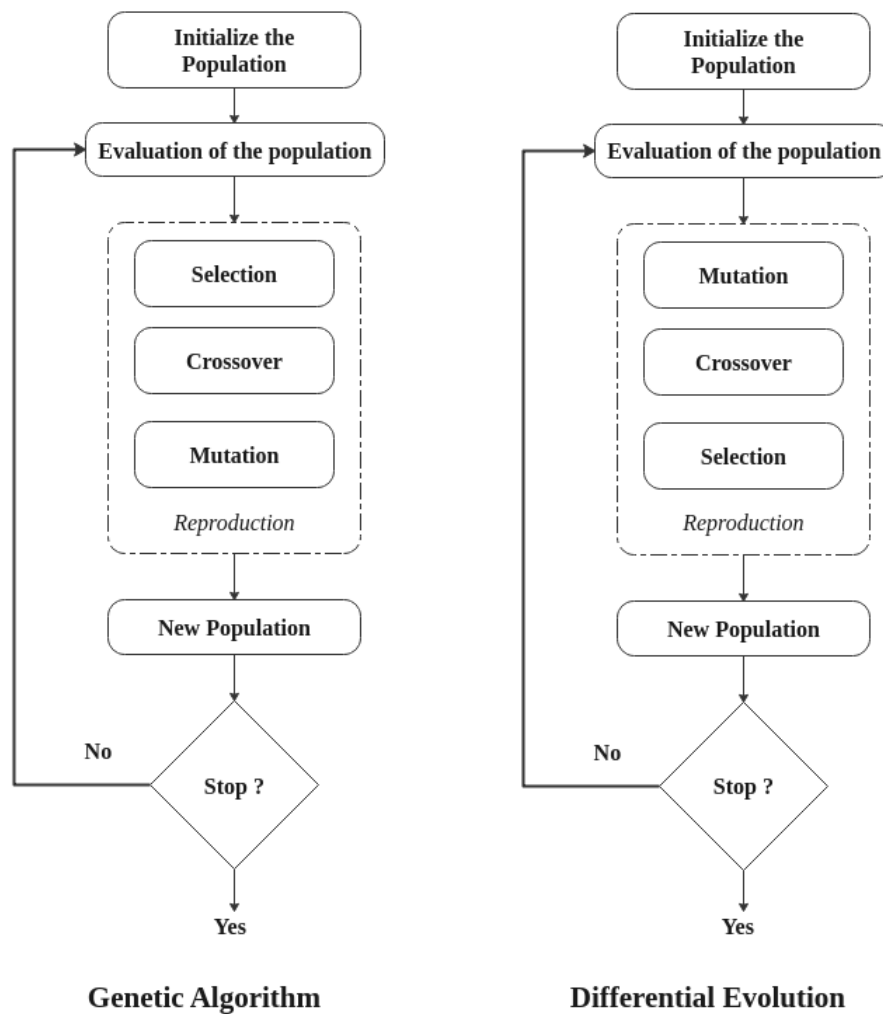


Figure 3. Genetic algorithm vs differential evolution

4.4. Differential evolution algorithm steps

DE is a global optimization algorithm that operates at the population level. At the outset, it creates a population of NP individuals, each of dimension D , where each individual encodes a potential solution, denoted by $X_{i,G} = X_{i,G}^1, \dots, X_{i,G}^D$, with $i = 1, \dots, NP$ and G indicating the generation to which the population belongs [16].

As noted in reference [16], DE is a population-level global optimization algorithm. Initially, DE creates a population consisting of NP individuals, each with a dimension of D . Each individual in the DE population represents a possible solution to the optimization problem being addressed represented by $X_{i,G} = X_{i,G}^1, \dots, X_{i,G}^D$, where $i = 1, \dots, NP$ and G represents the generation to which the population refers. The DE algorithm starts by generating an initial population of individuals, which are randomly distributed across the search space. Subsequently, the algorithm follows a set of primary steps [17].

4.4.1. Mutation Operation:

In the DE algorithm, during the generation G , the population is perturbed by the mutation operator which modifies each individual $X_{i,G}$ using a corresponding mutant vector $V_{i,G}$. The mutation operator can be generated using different strategies, among which the most frequently employed ones are listed in [18].

– DE/rand/1 :

$$V_{i,G} = X_{r_1,G} + F \cdot (X_{r_2,G} - X_{r_3,G}) \quad (3)$$

– DE/best/1:

$$V_{i,G} = X_{best,G} + F.(X_{r_1,G} - X_{r_2,G}) \quad (4)$$

– DE/best/2:

$$V_{i,G} = X_{best,G} + F.(X_{r_1,G} - X_{r_2,G}) + F.(X_{r_3,G} - X_{r_4,G}) \quad (5)$$

– DE/rand/2:

$$V_{i,G} = X_{r_1,G} + F.(X_{r_2,G} - X_{r_3,G}) + F.(X_{r_4,G} - X_{r_5,G}) \quad (6)$$

– DE/current-to-best/2:

$$V_{i,G} = X_{i,G} + F.(X_{best,G} - X_{r_1,G}) + F.(X_{r_2,G} - X_{r_3,G}) \quad (7)$$

– DE/current-to-rand/2:

$$V_{i,G} = X_{i,G} + F.(X_{r_1,G} - X_{r_2,G}) + F.(X_{r_3,G} - X_{r_4,G}) \quad (8)$$

The variable $V_{i,G}$ represents the mutant vector that is being created. The integers r_1, r_2, r_3, r_4 , and r_5 are randomly generated constants within the range $[1, NP]$ and different from the index j . The variable $X_{best,G}$ corresponds to the best individual in the population at generation G . The scale factor F is a real constant that is typically selected in the range $[0, 1]$ and determines the degree of amplification of the difference variation.

4.4.2. Crossover operation

Following the mutation phase, the diversity of the population is increased through the application of the crossover operation in DE. This operation utilizes the mutant vector $V_{i,G}$ created during the mutation phase to exchange its components with the target vector $X_{i,G}$, thereby producing the test vector $U_{i,G}$. The crossover operation can be expressed using the formulation presented in [19].

$$U_{i,G}^j = \begin{cases} V_{i,G}^j & \text{If } (rand_j[0, 1] \leq CR) \text{ or } (j = j_{rand}) \\ X_{i,G}^j & \text{Otherwise} \end{cases} \quad (9)$$

In the aforementioned expression, j is an integer that varies between 1 and D . The variable $rand_j$ corresponds to the j th evaluation of a uniform random number generator that produces values within the interval $[0, 1]$, as described in [20]. The crossover rate, denoted by CR , is a constant specified by the user and takes values within the range $[0, 1]$. Additionally, j_{rand} is a random integer selected from the range $[1, D]$, as stated in [20].

4.4.3. Selection operation

Once the test vector $U_{i,G}$ has been created through the crossover operation, the selection operation is carried out to maintain the population size for the next generation. To accomplish this, the objective function is evaluated for both the target vector $X_{i,G}$ and the test vector $U_{i,G}$. The vector that yields a better fitness value is retained in the population for the subsequent generation. More specifically, if the fitness value of the test vector is superior to that of the target vector, then the test vector replaces the target vector. On the other hand, if the fitness value of the target vector is better, it remains in the population. The selection procedure is mathematically represented as shown in [21].

$$X_{i,G+1} = \begin{cases} U_{i,G} & \text{If } f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G} & \text{Otherwise} \end{cases} \quad (10)$$

The three steps (mutation, crossover, and selection) are repeated for each generation up to a termination criterion.

The DE algorithm is characterized by the interaction between the different particles. The mechanism responsible for generating new potential solutions is the imitation of the global behavior of the neighborhood. Algorithm 1 presents a classic version of the DE [16] algorithm.

The parameters used in this algorithm 1 are:

- D The dimension of the problem.
- N The number of individuals.
- F The values of the scale factor.

- Cr The crossover rate.
- Max_{it} The maximum number of iterations Max_{it} .
- The choice of mutation strategy.

Algorithm 1 Differential evolution algorithm

```

Initialize the individuals of the population
Evaluate all individus
Initialize the best solution (Best)
iteration := 0
while iteration < max_iteration do
  for each individu do
    (a) Generate a mutant vector using the mutation operation4.4.1.
    (b) Generate the test vector using the crossover operation4.4.2.
    (c) Evaluate the try vector
    (d)If the test trial is better than the individual, replace it 4.4.3.
  end for
  Update The Best solution
  iteration := iteration + 1
end while
Return the best solution
  
```

5. EXPERIMENT RESULT

To assess the performance of the NoC, a group of benchmarks, as described in [14], are employed. A benchmark comprises a series of tasks that communicate with one another. Figure 4 (a) to (c) represents three commonly used benchmarks in testing, which are the Video Object Plane Decoder (VOPD) benchmark, (b) Moving Picture Experts Group (MPEG4) benchmark, and (c) Multi-Window Display (MWD) benchmark, respectively.

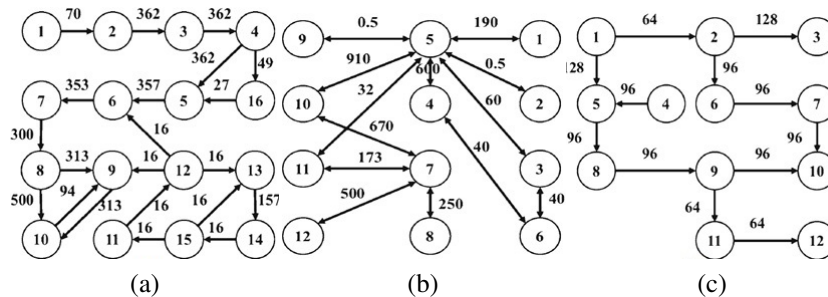


Figure 4. The benchmarks utilized in the study include (a) VOPD benchmark, (b) MPEG4 benchmark, and (c) MWD benchmark

Before presenting the result, it is crucial to specify the parameters that were employed in our study. A 4×4 NoC was utilized, and the parameters are listed in Table 1.

Table 1. Parameter of DE used in experiment

Parameter	Value used
CR	0.001 ... 1.0
F	0.01 ... 1.0
Pop size	200; 500; 1,000
Max it	500; 1,000; 2,000

After conducting multiple test attempts, we concluded that each application requires its own set of parameters to achieve the best results. Therefore, we have adopted the parameters presented in Table 2.

Table 2. Parameter used in DEMAP algorithms

Noc Size	CR	F	Pop Zise	Max it
4×4	0.0001	0.85	1,000	2,000

We utilized multiple mutation operations to determine which one would lead us to the optimal outcome. The results obtained from the five mutation operations are shown in Table 3.

Table 3. Result obtained

Application	Rand/1	Rand/2	Current to best/1	Best/1	Best/2
VOPD	4701	4743	4965	4119	4119
MWD	1280	1216	1248	1184	1184
MPEG4	3660	3666	3667	3470	3470

Figure 5 enables a visual assessment of the mapping performance, based on the five mutation strategies utilized during the experiment, as executed by the DEMAP algorithm. The outcomes indicate that utilizing the best strategy consistently yields the optimal results, surpassing the alternative strategies employed in DEMAP. To further establish the effectiveness of our approach, we compared the results achieved using our best approach with some other techniques, as shown in Table 4. Figures 6 to 8 provide a visual comparison between the communication cost results of DEMAP mapping and the compared approach, allowing for an easy analysis of the best results.

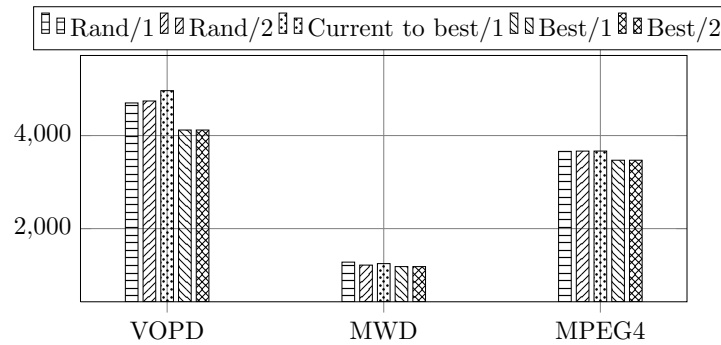


Figure 5. Comparison of DEMAP with the mutation strategy

Table 4. DEMAP result compared to other approach

Results	VOPD	MWD	MPEG4
DEMAP	4119	1184	3470
GBMAP	4217	-	3572
ONYX	4242	-	3612
CGMAP	4300	-	3600
NMAP	4265	1344	3852
CHMAP	4167	1344	3852
ILP	4119	1120	3567
Castnet	4135	1280	3852

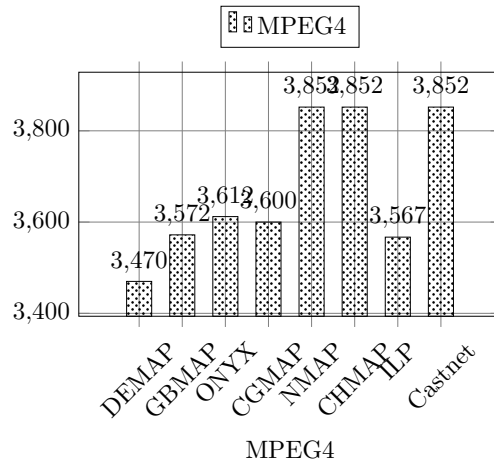


Figure 6. The MPEG4 benchmark with DEMAP and other approaches

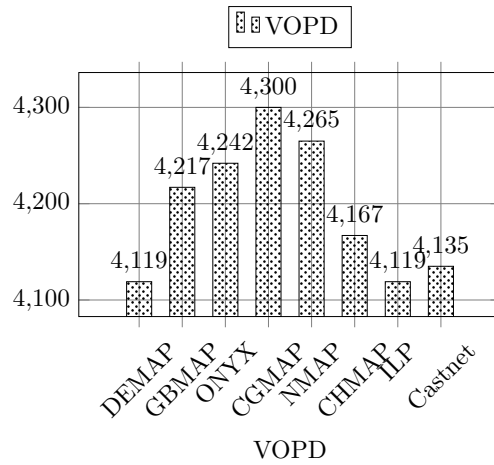


Figure 7. The VOPD benchmark with DEMAP and other approaches

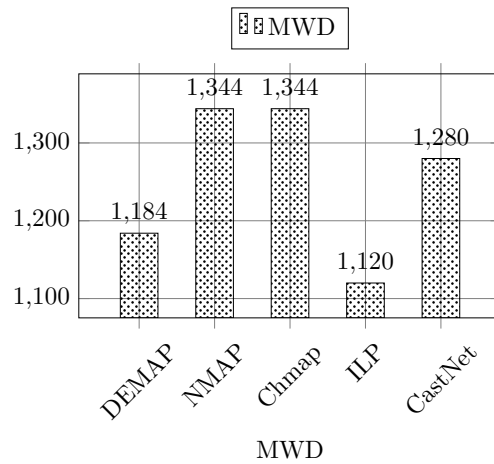


Figure 8. The MWD benchmark with DEMAP and other approaches

Based on the obtained outcomes, we can infer that DEMAP produces superior results compared to other meta-heuristic approaches. The results are also comparable to the best outcomes achieved using heuristic approaches, even though these approaches are typically more effective in optimizing a single objective.

6. CONCLUSION

The process of mapping applications onto a network-on-chip (NoC) is a complex task that is known to be NP-hard. To address this challenge, we introduce a novel approach based on evolutionary strategy, specifically the differential evolution (DE) algorithm. Our approach, named DEMAP, employs a mono-objective strategy that seeks to minimize the communication cost. The experiments were carried out on an actual Cores Graph, and the results indicate that DEMAP can produce better outcomes when the algorithm parameters are appropriately tuned. As a part of our future work, we plan to investigate the effectiveness of collaborative techniques in multi-objective optimization mapping and compare it with other established methods.





REFERENCES

- [1] F. Moraes, A. Mello, L. Möller, L. Ost, and N. Calazans, "A low area overhead packet-switched network on chip: architecture and prototyping," in *International Conference on Very Large Scale Integration of System-on-Chip*, 2003, pp. 318–323.
- [2] B. Yang, L. Guang, T. Säntti, and J. Plosila, "t(k)-SA: accelerated simulated annealing algorithm for application mapping on networks-on-chip," in *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, Jul. 2012, pp. 1191–1198, doi: 10.1145/2330163.2330327.
- [3] Jingcao Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 551–562, Apr. 2005, doi: 10.1109/TCAD.2005.844106.
- [4] M. Janidarmian, A. Khademzadeh, and M. Tavanpour, "Onyx: a new heuristic bandwidth-constrained mapping of cores onto tile-based network on chip," *IEICE Electronics Express*, vol. 6, no. 1, pp. 1–7, 2009, doi: 10.1587/elex.6.1.
- [5] M. Tavanpour, A. Khademzadeh, S. Pourkiani, and M. Yaghobi, "GBMAP: an evolutionary approach to mapping cores onto a mesh-based NoC architecture," *Journal of Communication and Computer*, vol. 7, no. 3, 2010.
- [6] J. Wang, Y. Li, S. Chai, and Q. Peng, "Bandwidth-aware application mapping for NoC-based MPSoCs," *Journal of Computational Information Systems*, vol. 7, no. 1, pp. 152–159, 2011.
- [7] 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 (VLSI) Systems*, vol. 22, no. 2, pp. 300–312, Feb. 2014, doi: 10.1109/TVLSI.2013.2240708.
- [8] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, pp. 896–901, doi: 10.1109/DATE.2004.1269002.
- [9] W. T. Shen, C. H. Chao, Y. K. Lien, and A. Y. Wu, "A new binomial mapping and optimization algorithm for reduced-complexity mesh-based on-chip network," in *Proceedings NOCS 2007: First International Symposium on Networks-on-Chip*, 2007, pp. 317–322, doi: 10.1109/NOCS.2007.5c.
- [10] S. Tosun, O. Ozturk, E. Ozkan, and M. Ozen, "Application mapping algorithms for mesh-based network-on-chip architectures," *The Journal of Supercomputing*, vol. 71, no. 3, pp. 995–1017, Mar. 2015, doi: 10.1007/s11227-014-1348-x.
- [11] C. H. Cheng and W. M. Chen, "Application mapping onto mesh-based network-on-chip using constructive heuristic algorithms," *Journal of Supercomputing*, vol. 72, no. 11, pp. 4365–4378, 2016, doi: 10.1007/s11227-016-1746-3.
- [12] F. Moein-Darbari, A. Khademzadeh, and G. Gharooni-Fard, "CGMAP: a new approach to Network-on-Chip mapping problem," *IEICE Electronics Express*, vol. 6, no. 1, pp. 27–34, 2009, doi: 10.1587/elex.6.27.
- [13] A. Mehran, S. Saeidi, A. Khademzadeh, and A. Afzali-Kusha, "Spiral: a heuristic mapping algorithm for network on chip," *IEICE Electronics Express*, vol. 4, no. 15, pp. 478–484, 2007, doi: 10.1587/elex.4.478.
- [14] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *Journal of Systems Architecture*, vol. 59, no. 1, pp. 60–76, 2013, doi: 10.1016/j.sysarc.2012.10.004.
- [15] L. Zhou, M. Jing, L. Zhong, Z. Yu, and X. Zeng, "Task-binding based branch-and-bound algorithm for NoC mapping," in *2012 IEEE International Symposium on Circuits and Systems*, May 2012, pp. 648–651, doi: 10.1109/IS-CAS.2012.6272115.
- [16] K. Ullensvang, K. P. Lehre, J. Storm-Mathisen, and N. C. Danbolt, "Differential developmental expression of the two rat brain glutamate transporter proteins GLAST and GLT," *European Journal of Neuroscience*, vol. 9, no. 8, pp. 1646–1655, Aug. 1997, doi: 10.1111/j.1460-9568.1997.tb01522.x.
- [17] K. Fleetwood, "An introduction to differential evolution," in *Proceedings of Mathematics and Statistics of Complex Systems (MASCOS) One Day Symposium*, 2004, pp. 785–791.





- [18] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, Feb. 2011, doi: 10.1109/TEVC.2010.2059031.
- [19] V. Feoktistov, *Differential evolution*, vol. 5. Boston, MA: Springer US, 2006.
- [20] M. F. Ahmad, N. A. M. Isa, W. H. Lim, and K. M. Ang, "Differential evolution: a recent review based on state-of-the-art works," *Alexandria Engineering Journal*, vol. 61, no. 5, pp. 3831–3872, May 2022, doi: 10.1016/j.aej.2021.09.013.
- [21] T. Robič and B. Filipič, *DEMO: differential evolution for multiobjective optimization*. 2005.

BIOGRAPHIES OF AUTHORS







Maamar Bougherara     is an assistant professor at Ecole Normale Supérieure de Kouba, Algeria and pursuing his Ph.D. at LIMPAF Laboratory, Bouira University, Algeria. He is currently working on networks on chip (NoC). He holds an M.Sc. and an Engineering degree in Computer Science from the University of Blida, Algeria. He can be contacted at bougherara.maamar@gmail.com.



Rafik Amara     is an assistant professor at the Computer Science Department, Ecole Normale Supérieure de Kouba, Algiers. He obtained a computer engineering degree in 2001 from the University of Science and Technology (USTHB) in Algiers. After professional experience in the air navigation sector, he continued his studies to obtain in 2008 a master's degree in computer science, specializing in image processing and GIS. He is currently a doctoral student at the Image Processing and Radiation Laboratory (LTIR) at USTHB and more precisely in the GIS team. He works especially on linear networks such as air route networks and air traffic simulation and optimization.



Rebiha Kemcha     is currently an assistant professor at the Computer Science Department, Ecole Normale Supérieure de Kouba, Algiers. She received her M.Sc. degree in Networks and Distributed Systems from the University of Bejaia, Algeria in 2012. She worked on interconnection networks, their topologies, and routing. She is a Ph.D. candidate at the University of Boumerdes at LIMOSE Laboratory. She works on evolutionary circuit design and design space exploration using multiobjective optimization algorithms.