International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 15, No. 2, July 2026, pp. 373∼385
ISSN: 2089-4864, DOI: 10.11591/ijres.v15.i2.pp373-385

Crow search algorithm for efficient IP placement in 2D and 3D network-on-chip architectures

Maamar Bougherara[1,2], Amina Guidoum[1], Rafik Amara[1,3]
[1]Department of Computer Science, High Normal School of Kouba, Algiers, Algeria
[2]LIM Laboratory, Faculty of Exact Sciences, Akli Mohand Oulhadj University of Bouira, Bouira, Algeria
[3]LTIR Laboratory, Faculty of Electronics and Computational Science, USTHB University, Algiers, Algeria

Article Info

Article history:

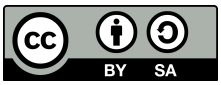
Received Dec 26, 2025
Revised Apr 12, 2026
Accepted May 31, 2026

Keywords:

Communication cost
Crow search algorithm
Network-on-chip
Optimization
Placement

ABSTRACT

The communication in system-on-chip (SoC) has evolved to meet the increasingly complex requirements of modern applications. To address connectivity challenges, the network-on-chip (NoC) has emerged as an efficient solution. While traditional NoCs are primarily based on 2D architectures, the inherent limitations of 2D designs have driven the adoption of 3D architectures, which offer enhanced space utilization and performance optimization. A key step in the design of NoC systems is the placement of cores, also known as the mapping phase, in which application tasks are assigned to the architecture's processing elements. This phase is considered an nondeterministic polynomial (NP)-complete problem due to its combinatorial complexity. Optimizing this phase is crucial, as it directly impacts the overall performance of the NoC. Various optimization algorithms have been employed to maximize the efficiency of 2D and 3D NoCs. In this paper, we adopt the crow search algorithm to find the places both 2D and 3D NoCs with minimal comunication. The goal is to evaluate its performance compared to other optimization algorithms in this crucial step.



Corresponding Author:

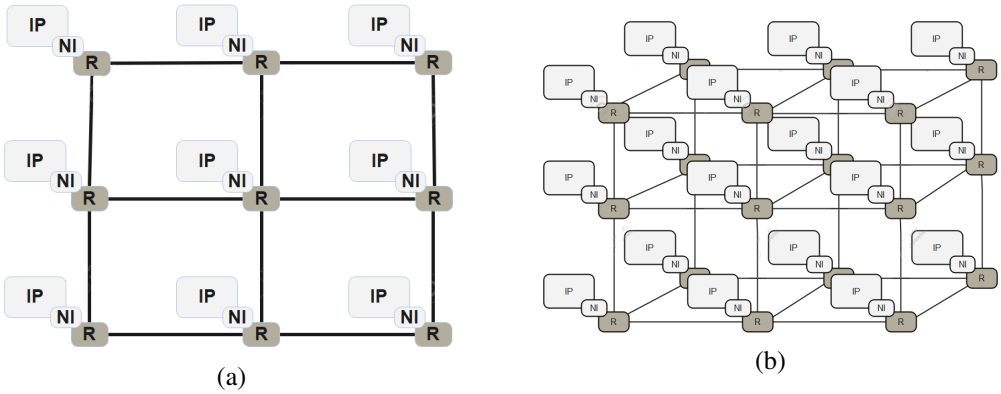
Maamar Bougherara
Department of Computer Science, High Normal School of Kouba
Algiers, Algeria
Email: bougherara.maamar@gmail.com

1. INTRODUCTION

The increasing complexity of modern system-on-chip (SoC) designs, characterized by the integration of numerous processing elements on a single chip, has made efficient on-chip communication a critical challenge. Traditional bus-based interconnection architectures suffer from scalability limitations, bandwidth constraints, and increased power consumption, making them unsuitable for large-scale systems. To overcome these limitations, network-on-chip (NoC) architectures have emerged as an efficient communication paradigm. Inspired by conventional computer networks, NoC provides a scalable and structured communication framework based on routers and packet-switched data transfer, enabling efficient communication among multiple on-chip components. In very large scale integration (VLSI) systems, NoC helps manage communication complexity, improves modularity, and reduces interconnection delays. Moreover, in embedded systems, NoC facilitates efficient data exchange between processing units while maintaining high performance and energy efficiency. Consequently, NoC has become a key technology for addressing communication bottlenecks and enabling the design of scalable and high-performance computing systems [1].

In NoC-based systems, applications are typically divided into several computational tasks, each en-

capsulated in an intellectual property (IP) block. These IPs must be integrated into the NoC infrastructure in a manner that maximizes performance while minimizing energy consumption and latency [2]. A typical NoC consists of four key components [3] IP cores, which include processors, memory units, and custom controllers, network interfaces (NIs), which serve as a bridge between the IP cores and the NoC communication fabric; routers, which handle packet forwarding and arbitration; and physical interconnects, which define the paths for data transmission. The topology of a NoC determines how tiles are connected [4]. Among the different topology options, the 2D mesh remains the most prevalent choice, largely attributed to its highly regular and scalable structure. With the increasing complexity and integration density of modern SoCs, 2D NoC architectures face growing limitations in terms of bandwidth, latency, and energy efficiency. To address these challenges, three-dimensional (3D) NoC architectures have been proposed [2]. By vertically stacking multiple 2D layers and employing through-silicon Via (TSV) technology [2], 3D NoCs significantly reduce the average communication distance, enhance bandwidth, and improve performance. In a 3D mesh topology, each router connects to up to six neighboring routers [4]: four in the same layer and two in adjacent vertical layers. This configuration enables both intra-layer and inter-layer communication, leading to better parallelism, reduced latency, and higher throughput. Figure 1 illustrates examples of NoC architectures topologies. The 2D mesh consists of 9 tiles arranged in a (3×3) grid (Figure 1(a)), whereas the 3D mesh architecture consists of 27 cores organized in a $(3 \times 3 \times 3)$ structure (Figure 1(b)).



(a) (b)

Figure 1. Mesh-based NoC: (a) 2D and (b) 3D

The design of a NoC-based system generally follows three main stages [3]: task assignment, which involves assigning application tasks to a predefined library of IP blocks, IP placement, where these IP blocks are mapped to physical nodes (routers) within the NoC and static routing, which determines the communication paths between the mapped IPs.

Each of these stages is supported by electronic design automation (EDA) tools, which explore the design space search to generate optimized hardware/software configurations. Figure 2 illustrates a standard embedded system design flow for NoC platforms. IP core placement in NoC architectures significantly affects system performance. Poor placement increases communication latency and power consumption due to longer distances between communicating cores. It may also cause network congestion and limits scalability in large many-core systems. Therefore, efficient placement strategies are essential to optimize communication cost and improve overall system performance. This paper introduces a novel model for solving the IP placement problem in both 2D and 3D NoCs. After modeling the problem, defining the main objective of the study, and formulating the cost function, the crow search algorithm (CSA) is employed for the first time as a new method to address the application placement problem. The structure of this paper is as follows: section 2 presentes the NoC topology and formulates the placement problem. Section 3 provides a review of prior research related to placement in NoC architectures. In section 4, we present the modeling of the placement problem prior to applying a metaheuristic approach. The CSA, used as the core placement strategy, is described in section 5. Section 6 analyzes the results obtained from the simulation experiments. Finally, section 7 presents the conclusion of the study along with directions for future work.

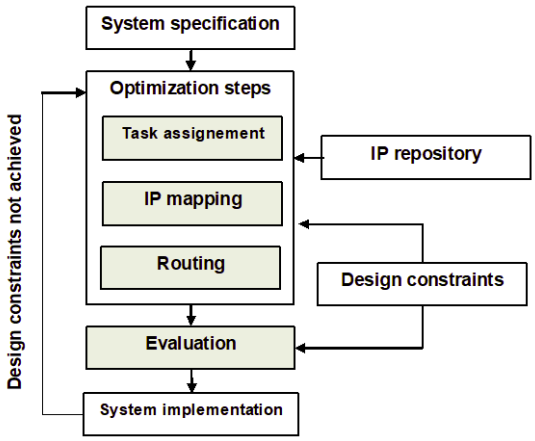


Figure 2. Typical embedded system design flow for NoC platform

2. NETWORK TOPOLOGY AND PLACEMENT PROBLEM

In conventional 2D NoCs, each IP is connected to a router via a NI, and these routers are arranged in a planar grid with wired links. Communication takes place using packet-switched data transfer, where messages are broken into packets and routed through the NoC [5]. Compared to traditional bus-based systems, 2D NoCs offer several advantages: higher communication bandwidth, reduced latency, improved scalability, and lower power consumption [6]. However, as more IPs are integrated into a single chip, the average number of hops (routers traversed by a packet) increases, leading to greater communication energy and degraded performance. This issue becomes increasingly problematic as system size grows [4]. The 3D NoC paradigm addresses these issues by enabling vertical communication through TSVs. This approach reduces the average hop count and significantly lowers latency and energy consumption. Additionally, it enhances area utilization, making it suitable for large-scale multicore systems.

Consequently, 3D NoCs have gained significant interest in both academia and industry as a compelling solution for future high-performance computing platforms. This work focuses on optimizing the placement phases of NoC design, aiming to minimize communication energy while maintaining high system performance. In a 3D NoC, the IP placement problem consists of assigning each IP core to a specific node within the 3D topology. This phase is critical, as efficient placement can greatly reduce communication cost, energy usage, and latency [4]. Due to its combinatorial nature, the IP placement problem is considered nondeterministic polynomial (NP)-hard and is closely related to the quadratic assignment problem (QAP). This complexity makes exact solutions impractical for large systems, especially within limited design time. As 3D NoCs enable the integration of a greater number of IPs across multiple layers using TSVs, efficient and scalable placement algorithms are becoming increasingly important. To tackle this problem, heuristic and metaheuristic techniques represent a practical solution, as they can efficiently generate high-quality approximations within a reasonable computation time. For example, when assigning m tasks to a NoC with n cores where $(m <= n)$, the number of possible placement configurations may grow as large as $n!/(n-m)!$.

3. RELATED WORK

Several approaches have been proposed to address the placement problem in NoC architectures, commonly referred to as the mapping problem. These mapping techniques can be classified into three principal categories: exact approaches, heuristic (or approximate) techniques, and metaheuristic methods [7]. This paper presents an overview of the most highly cited works that address the problem using a single-objective approach.

In exact mapping methods, such as exhaustive search, linear programming, and branch-and-bound algorithms, rely on mathematical rigor to guarantee globally optimal solutions. An integer linear programming (ILP) formulation for the IP mapping problem is presented in [8]. This approach models both objectives and constraints as linear functions, with the additional requirement that decision variables take integer values. The ILP model is solved using the FICO Xpress optimization tool, enabling precise solution computation within a mathematically defined space. Despite their accuracy, exact methods suffer from high computational complex-

ity and significant runtime requirements. As a result, they are typically feasible only for small-scale mapping scenarios involving a limited number of IP cores, where the size of the solution space remains manageable under current computing capabilities.

The second category encompasses heuristic-based approaches, which aim to provide efficient and practical solutions to the application mapping problem on NoC architectures. The NMAP algorithm [9] is a widely used heuristic that maps application cores to tiles iteratively. At each step, a core is selected and assigned to a tile, repeating the process until all cores are mapped. While NMAP includes an iterative improvement mechanism, the quality of the final solutions often remains constrained by the initial mapping. BMAP [10] introduces a binomial mapping approach based on iterative two-way merging, taking into account the traffic load between cores to optimize communication. CastNet [11] enhances solution diversity by leveraging the symmetric characteristics of mesh topologies. It selects multiple initial tiles and generates several mapping options, ultimately choosing the best one based on the number of free neighboring tiles for each core. CHMAP [12] calculates a priority value for each core based on its communication requirements and its position within a communication spanning tree. The algorithm then determines the mapping order and assigns cores to appropriate tiles according to these computed priorities. Onyx [13] proposed in 2009, introduces a lozenge-shaped mapping path and defines four movement patterns to assign priorities to tiles. This method has shown improved communication cost compared to earlier heuristics. Spiral [14] starts by mapping the highest-priority task at the center of the mesh, then progressively maps remaining tasks outward in a spiral pattern from already mapped cores. To address hierarchical communication, V-CastNet3D is proposed in [15], a clustering-based mapping algorithm. Cores are grouped into clusters to reduce inter-cluster communication and the CastNet heuristic is then applied within each cluster for mapping onto a 2D NoC mesh. For 3D meshes, certain techniques developed for 2D meshes have been adapted. For example, NMAP 3D, ILP3D, and PSMAP3D. Furthermore, CastNet3D [8] extends this approach to 3D NoC architectures by optimizing the utilization of vertical links, thereby improving communication efficiency and reducing energy consumption.

Due to the high computational cost of exact methods and the limitations of current processing capabilities, the third category relies on metaheuristic techniques, which have emerged as practical and scalable alternatives for addressing the IP mapping problem in NoC architectures. Drawing inspiration from natural and biological phenomena, these approaches aim to efficiently explore large solution spaces and produce near-optimal solutions within reasonable computational time. In the context of 2D NoC architectures, several metaheuristic techniques have been proposed. In [16], the authors proposed CGMAP, a genetic algorithm-based approach that replaced the traditional random initialization with a chaotic mapping operator, enhancing the exploration capabilities of the algorithm. GBMAP [17] employs an evolutionary algorithm to efficiently map processing cores onto NoC tiles. Swarm intelligence-based approaches have also been explored. For example, a discrete particle swarm optimization (PSO) method is proposed in [18], utilizing deterministic initial solutions to improve performance. The same authors proposed several metaheuristic-based mapping techniques using PSO, ant colony optimization (ACO), artificial bee colony (ABC), and differential evolution (DE). These approaches aim to optimize task mapping in NoC architectures by reducing communication cost optimization in 2D NoC mapping problems.

While these techniques are primarily applied to 2D NoCs, recent research has extended metaheuristic strategies to 3D NoC architectures: in [19], a PSO-based algorithm is developed for mapping IPs onto a partially vertically-connected 3D mesh NoC. A PSO based mapping method is introduced in [18] to improve communication costs by applying bandwidth limitation. A similar approach is adopted in [20], but with the objective of minimizing energy consumption. In [19], a quantum PSO to the 3D NoC mapping problem s proposed, thereby reducing power consumption. An adaptive genetic algorithm based on logistic functions (LFAGA) is proposed in [21] to optimize the energy consumption of the network. An ABC-based method for 3D NoC mapping is introduced in [22], demonstrating improved results in communication optimization. A thermal-aware mapping technique is proposed in [23], which integrates genetic algorithms with fuzzy logic to minimize the peak temperature in 3D NoC systems. Finally, the same authors proposed a simulated annealing (SA)-based approach to improve communication efficiency in 3D NoC application mapping.

4. MODDELING PLACEMENT PROBLEM

In the context of IP placement for NoC, the application is typically modeled as an IP core graph, while the NoC infrastructure is represented as a topology graph. The IP placement problem consists of assigning

logical IP cores defined by specific communication relationships and bandwidth demands within the IP core graph to physical resource nodes in the NoC topology graph [5].

- NoC topology graph: the NoC topology is abstracted as a directed graph $T(R, L)$, where: R is the set of nodes, with each node $r_k \in R$ representing a tile in the NoC. L is the set of directed edges, where each edge $l_{k,l} \in L$ represents a physical communication link between tiles r_k and r_l [4].
- IP core graph: similarly to NoC topology, the IP core graph is a high level abstraction that represents the behavior of an application. It is denoted as a directed graph $G(C, A)$, where: C is the set of nodes, each node $c_i \in C$ representing an IP core and A is the set of directed edges, where each edge $a_{i,j} \in A$ represents communication from IP source core c_j to destination core c_i [5]. Each edge $a_{i,j} \in A$ is associated with a weight $w_{i,j}$ which denotes the bandwidth requirement between the IPs c_j and c_i.
- Placement function in NoC: the placement function defines the placement of IP cores from the IP core graph $G(C, A)$ to nodes in the NoC topology graph $T(R, L)$, with the objective of minimizing communication energy [13]. For each IP core $c_i \in C$, there exists a corresponding node $r_k \in R$ in the NoC topology such that: $map : V \rightarrow T\ map(c_i = r_k), \forall c_i \in C\ \exists r_j \in R$. Furthermore, each IP core must be mapped to a unique node, ensuring that for any two distinct IPs c_i and c_j: $map(c_i) \neq map(c_j)$. To satisfy this constraint, the number of nodes in the NoC topology graph must be greater than or equal to the number of nodes in the IP core graph. If necessary, dummy nodes can be added to the IP core graph to equalize the sizes of both graphs [3]. These dummy nodes are non communicating placeholders and do not participate in any data exchanges.
- Solution représentation: the solution is represented as a sequence with a permutation of the integers from 1 to n, where n is the number of nodes in the NoC topology graph [5]. Each position in the sequence corresponds to a node in the NoC topology. The value at that position indicates the ID of the IP core mapped to that node. An example of placement solution is illustrated in Table 1.

Table 1. Solution exemple

Core number	1	2	3	4	5	6	7	8	9
Node place	8	3	2	1	9	5	6	7	4

- Objective function: the objective function used in this paper aims to minimize communication cost by reducing the number of hops between tasks mapped onto the NoC. The total communication cost is calculated as (1) [4]:

$$commcost = \sum_{i} \sum_{j} w_{i,j} * nbhops(r_k, r_l), \tag{1}$$

where $c_i = source$, $c_j = distination$, $map(c_i) = r_k$, $map(c_j) = r_l$,$c_i \neq c_j$, and $r_k \neq r_l$. $nbhopt(.)$ is the number of hops between the source and destination, calculated using the Manhattan distance, defined by (2):

$$Hops(r_k, r_l) = |X_k - X_l| + |Y_k - Y_l| + |Z_k - Z_l| \tag{2}$$

where (X_k, Y_k, Z_k) and (X_l, Y_l, Z_l) are the coordinates of the nodes r_k and r_l on a 3D mesh NoC, respectively. In the case of a 2D mesh NoC, Z_k and Z_l are equal to 0.

5. PLACEMENT WITH CROW SEARCH ALGORITHM

5.1. Crow in nature

Crows are among the most intelligent animals in the animal kingdom. Biologists state that they have the most developed brains of all birds, with intelligence comparable to that of a child aged 2 to 5 years. In the wild, researchers have observed complex behaviors such as the use of tools like sticks to achieve goals, or even dropping stones on intruders approaching their nests [24]. Crows are also known for their cunning. When hiding food while being watched by another crow, they may pretend to hide it in one place, then actually hide it somewhere else to deceive the observer. Unlike most birds that rely mainly on vocal sounds to communicate, crows also use sophisticated non-vocal gestures a form of body language similar to sign language to convey more complex information. They are highly adaptable. They can live in a wide variety of environments, from

snow and deserts to forests and mountains. Their omnivorous diet allows them to eat a broad range of food sources: meat, fish, fruits, seeds, carcasses, and even garbage [25].

Due to its effective balance between exploration and exploitation, achieved through its memory-based search mechanism and awareness probability strategy, and compared to traditional approaches such as the genetic algorithm, it requires fewer control parameters and demonstrates strong performance in solving complex combinatorial optimization problems [25]. Therefore, it is adopted in this work for the first time to address the NoC mapping problem.

5.2. Crow search algorithm steps

The CSA is an optimization technique inspired by the intelligent behavior of crows. In nature, crows hide excess food in different locations and retrieve it when needed. They are opportunistic and may follow each other to find better food sources. However, if a crow realizes it is being followed, it may deceive the other by moving to a false location [25].

In CSA, the search space represents the environment, and each crow's position corresponds to a potential solution. The objective function evaluates the quality of the solution, similar to how food quality is assessed. The algorithm aims to find the global optimum—the best food source—by mimicking the crows' strategies of memory, deception, and exploration. CSA models the social behavior of crows using the following assumptions [24].

- Crows live in flocks and interact socially.
- Each crow remembers the best position it has found (its hiding spot).
- Crows follow others in the swarm to find better solutions.
- A crow can deceive another if it detects being followed, leading the follower to a random position instead of its true hiding place.

Consider a d dimensional search space represented by a number of crows. The total number of crows (flock size) is denoted by N. The position of crow i at iteration $iter$ is represented by a vector in this d-dimensional space [26].

$$X^{i,iter} = [x_1^i, x_2^i, x_3^i \dots x_d^i] \tag{3}$$

Each crow i is equipped with a memory that stores the location of its personal hiding placem^i. At iteration $iter$, this hiding place corresponds to the best position found by crow i while exploring the environment in search of better food sources (hiding places). Suppose that at iteration $iter$, crow j intends to visit its hiding place m^j. At the same time, crow i chooses to follow crow j in an attempt to approach its hiding place. In this context, two possible scenarios may occur:

- Crow j is not aware that it is being followed by crow i. Consequently, crow i moves closer to crow's j hiding place. The new position of crow i is then calculated as (4) [27]:

$$X^{i,iter+1} = X^{i,\text{iter}} + r_i \cdot fl^{i,\text{iter}} \cdot \left(m^{j,\text{iter}} - X^{i,\text{iter}}\right) \tag{4}$$

  where $m^{j,\text{iter}}$ is the memory of the crow j and fl is the flight length.
- Crow j is aware that crow i is following it. Consequently, to protect its hiding place from being discovered, crow j attempts to deceive crow i by moving to a different location within the search space, generating a feasible random solution [26].

The two scenarios described above can be expressed in (5) [28]:

$$X^{i,iter+1} = \begin{cases} X^{i,\text{iter}} + r_i \cdot fl^{i,\text{iter}} \cdot \left(m^{j,\text{iter}} - x^{i,\text{iter}}\right) & \text{if } r_j \geq AP^{i,iter} \\ a\ random\ position & otherwise \end{cases} \tag{5}$$

According to the equation, two key parameters play a crucial role in guiding the search process toward the global optimum: flight length and awareness probability. These two parameters significantly influence the performance of the algorithm. Proper tuning ensures better results, as they control the balance between intensification (exploitation of good solutions) and diversification (exploration of new regions in the search space).

a. Flight length: the flight length (fl) parameter plays a crucial role in controlling the balance between exploration and exploitation in the CSA. Small fl values promote local search behavior, where the new position of a crow remains close to either its current location or the memory of the crow it follows [27]. This behavior enhances solution refinement within a limited region of the search space. In contrast, large fl values encourage global search by allowing the crow to move farther from its current or target memory position, thus enabling the exploration of previously unvisited areas.
b. Awareness probability: the awareness probability (AP) controls also the balance between exploitation and exploration in the CSA. Low AP values promote exploitation by allowing followers to reach good solutions without being misled, while high AP values enhance exploration by causing leaders to deceive followers through random movements. Therefore, careful tuning of AP is crucial to achieve an effective search balance [26].

The Algorithm 1 summarizes the main steps of the proposed approach, providing a clear overview of its operational flow [24].

Algorithm 1. The crow search algorithm

```
generate the population of the crow (the flock of crow)
evaluate each crow with the objective function
initialize the memory of each crow
iter ← 0
while iter < max of iteration do
  for each crow i do
    Randomly choose one of the crows to follow j
    Define an awareness probability AP^i
    Update position of the crow i using (5)
    Check the feasibility of new positions
    Evaluate the new position of the crows
    Update the memory of the crow
  end for
  iter ← iter + 1
end while
return  best solution
```

5.3. Modeling crow search algorithm for placement problem

In order to model the CSA so that it can effectively address the placement problem, we have detailed its steps by aligning them with the specific requirements of the problem.

- Step 1. Initialize the problem and algorithm parameters: at this stage, we define the parameters required to properly calibrate the algorithm in order to obtain optimal results. This includes specifying the optimization problem, the objective function and any constraints and setting the key parameters of the CSA.
  - N: number of crows (population size),
  - $maxt\_iter$: maximum number of iterations,
  - fl: flight length (step size),
  - AP: awareness probability.
- Step 2. Initialize the positions and memories of the crows: randomly position N crows in a d-dimensional search space, where each crow represents a feasible solution to the placement problem. Each crow has a memory in which it stores its best found position (initially set to its current position, as the crow has no experience at iteration 0).
- Step 3. Evaluate initial fitness: for each crow, evaluate the quality of its current position by inserting the corresponding decision variable values into the objective function.
- Step 4. Generate new positions: each crow attempts to generate a new position in the search space. To do so, crow i randomly selects another crow j and tries to move toward crow j's hiding place. This new position is calculated using (5) and this process is repeated for all crows.
- Step 5. Check feasibility of new positions: for each crow, the feasibility of the new generated position is evaluated based on the problem's constraints. If the position is valid, the crow updates its current position to the new one. Otherwise, it remains at its current location without making any changes.

- Step 6. Update memory: the crows update their memory according to (6).

$$m^{i,\text{iter}+1} = \begin{cases} X^{i,\text{iter}+1} & \text{if } f(X^{i,\text{iter}+1}) < f(m^{i,\text{iter}}) \\ m^{i,\text{iter}} & \text{otherwise} \end{cases} \tag{6}$$

  In other words, if the new position of crow i yields a better fitness value than the one currently stored in its memory, the memory is updated with the new position. Otherwise, it remains unchanged.
- Step 7. Check stopping criterion: the loop comprising steps 4 to 7 is repeated until the predefined maximum number of iterations (iter_{max}) is reached. Once the stopping criterion is met, the best solution stored in memory (in terms of the objective function value) is returned as the final solution to the optimization problem.

6. EXPERIMENT RESULT

To assess the effectiveness of the placement approach in a NoC architecture, a range of benchmarks is typically employed [13]. These benchmarks comprise interconnected tasks that mimic the behavior of real-world applications. Each task models a processing unit, along with the communication links between them, enabling the analysis of critical architectural characteristics such as energy consumption and communication overhead. In this work, we consider two main types of benchmarks:

a. Synthetic benchmarks: these benchmarks are generated automatically using the task graphs for free (TGFF) tool [29], which enables the creation of diverse task graphs with customizable parameters such as the number of tasks, communication volume, and execution time. Synthetic benchmarks are especially valuable for testing NoC architectures across a wide range of controlled and varied scenarios.
b. Real-world benchmarks: these benchmarks are extracted from practical multimedia applications and are widely used to evaluate NoC architectures. Figure 3 showcases four representative examples from this category:

- VOPD is a decoding application derived from MPEG-4 that processes video object planes (Figure 3(a)).
- MPEG-4 is an object-based video compression standard that enables efficient parallel processing of multimedia data (Figure 3(b)).
- PIP represents a sequence of image processing stages used to evaluate pipelined execution and parallelism (Figure 3(c)).
- MWD models realistic multimedia workloads, including audio and video processing tasks, to assess system performance under complex and heterogeneous applications (Figure 3(d)).

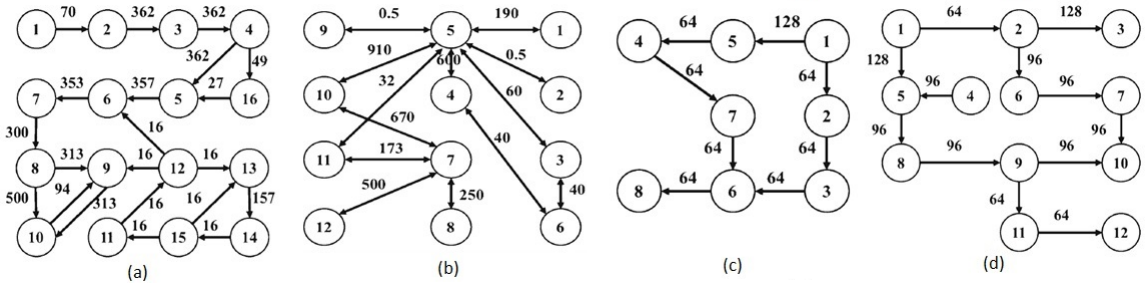


Figure 3. Benchmarks used in the experiments: (a) video object plane decoder (VOPD), (b) moving picture experts group (MPEG4), (c) picture in picture (PIP), and (d) multi-window display (MWD)

The benchmarks VOPD, MPEG4, PIP, and MWD are commonly used to evaluate NoC architectures. VOPD and MWD represent video decoding and signal-processing tasks with intensive inter-core communication, while MPEG4 simulates video compression workloads and PIP handles multiple video streams simultaneously. Together, they provide realistic and diverse communication patterns for testing NoC performance.

6.1. Setting

Before presenting the experimental results, we first detail the configuration parameters used in our study. For the real world benchmarks, the VOPD, MPEG4, and MWD applications were mapped onto a 4×4

in 2D NoC and a 2×4×2 in 3D architecture. As for the PIP benchmark, due to its smaller task graph, it was mapped to a 2×2×2 3D topology. Regarding the synthetic benchmarks, six benchemarks are used from [30], we employed a 2D mesh NoC with a symmetric configuration ($x = y$), as well as a 3D mesh NoC with full symmetry in all dimensions ($x = y = z$).

To calibrate the right parameters of the algorithm, many tests are executed varying the population size is setted to 500 and the maximum number of iterations is 1000. As highlighted befor, the flight length (fl) is a critical parameter that significantly influences the performance of the algorithm. To ensure a balanced trade-off between exploration and exploitation capabilities, we considered two scenarios: one with $fl < 1$ and another with $fl > 1$. This choice enables the algorithm to effectively capture both global search (exploration) and local refinement (exploitation) behaviors. The fl parameter was set to $fL < 1 = 0.2$ and $fL > 1 = 2$, However, both r and AP are randomly generated in the interval $[0, 1]$ for each crow and at every iteration, introducing dynamic variability in the search process.

6.2. Result and discussion

Table 2 presents the performance evaluation of the CSA placement approach under two configurations defined by the fl parameter, for both 2D and 3D NoC topologies across all selected benchmarks. The obtained values correspond to the communication cost, which is considered as the objective function to be minimized in this paper.

Table 2. Comparison of obtained result with fl

Benchmarks	# Task	# Edge	$fl < 1$		$fl > 1$	
		CSA_2D	CSA_3D	CSA_2D	CSA_3D	
TG0	12	13	24800	22100	24700	22000
TG1	16	16	35600	32400	36600	31700
TG2	27	27	88000	74000	87500	71800
TG3	30	34	113600	92000	113000	92000
TG4	50	59	260500	198400	253100	194500
TG5	62	70	363000	283700	384000	287300

The results clearly show that the 3D NoC configuration consistently outperforms the 2D design, underscoring the benefits of leveraging the third dimension the placement problem. This performance gain is particularly evident in both setups when applying the CSA algorithm, as reflected by the fl value. By introducing the extra dimension, 3D NoCs reduce communication distances, which translates into more efficient resource utilization and overall better performance than 2D architectures. The visual representations in Figure 4 provide a comparative analysis of the placement outcomes obtained with CSA in both 2D and 3D NoC architectures, considering both parameter values of the flow latency factor (fl). Similarly, also evaluated under the two fl parameter settings. Together, these analyses highlight the significant impact of the fl factor on the quality of task-to-core placement.

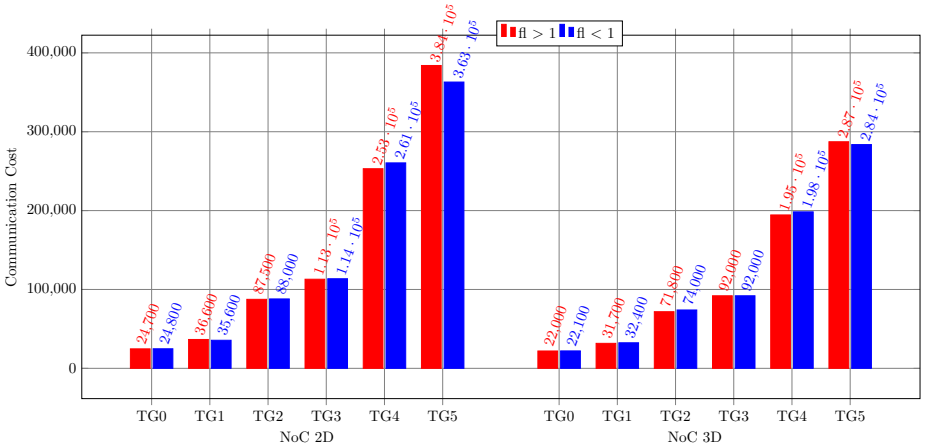


Figure 4. Comparison between placement solution

A closer examination of the results shows that when the flow latency factor exceeds one ($fl > 1$), the outcomes systematically improve. As previously discussed, a smaller value of fl favors local search intensification, while a larger value encourages diversification in the exploration process. In our case, we observed that higher fl values consistently yield better results compared to smaller ones, confirming that diversification in the search process leads to superior outcomes. Before comparing our CSA placement approach with existing methods from the literature, we first evaluate its effectiveness by benchmarking it against extended algorithms proposed by the same authors for 3D NoC architectures, namely ABC and simulated annealing, and then compare the optimal results it produces with those obtained from other state-of-the-art approaches. Table 3 reports the outcomes related to the 2D and 3D NoC topology.

Table 3. Comparison of obtained result in 2D and 3D

Benchmarks	$2D$			$3D$		
	SA	ABC	CSA	SA	ABC	CSA
TG0	32400	49300	24700	27800	31700	22000
TG1	39700	63400	36600	37200	40600	31700
TG2	94700	115100	87500	78000	73000	71800
TG3	-	179900	113000	-	121100	92000
TG4	-	326400	253100	-	199900	194500
TG5	-	419600	384000	-	273200	287300

Figure 5 illustrates the scalability of the proposed approaches with respect to the number of cores in both 2D and 3D NoC architectures. It shows the variation of the objective function values obtained by the two proposed algorithm versions and the ABC algorithm, highlighting their comparative performance under different network sizes.

The results presented in the Figures 5(a) and 5(b) indicate that the proposed versions generally achieve better objective function values as the number of cores increases. However, in the 3D NoC architecture, the ABC algorithm tends to perform better for larger numbers of cores. Table 4 presents the results related including a comparative analysis with other placement methods from the literature related to 2D and 3D NoC.

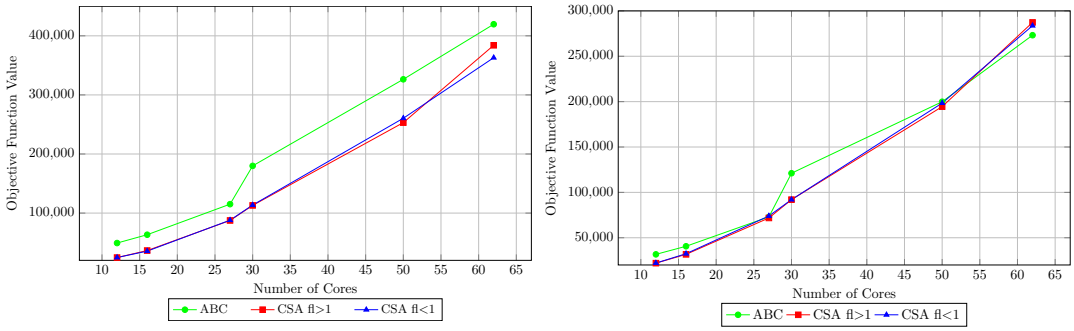


Figure 5. Scalability results: (a) 2D and (b) 3D

The comparison on real benchmarks indicates that the CSA delivered superior results for the VOPD application in 3D. However, for the remaining benchmarks, the performance exhibited certain limitations. These shortcomings can be largely attributed to the use of a uniform, generalized configuration applied across all benchmarks, without adjusting the parameters to the specific characteristics of each case. Notably, the number of iterations was fixed at 1000 for all experiments and population size was 500, which may not have been sufficient or optimal for every benchmark. This observation highlights the importance of adaptive parameter tuning particularly with respect to iteration count and population size which could significantly enhance the algorithm's performance and lead to more consistent improvements across diverse applications.

Table 4. Comparison results with other approaches in the 2D and 3D NoC

2D	VOPD	MWD	PIP	MPEG4	3D	VOPD	MWD	PIP	MPEG4
CSA	4320	1248	640	3774	CSA	4103	1216	640	3576
SA	4119	1184	640	3834	SA	4103	1138	640	3567
PSO	4141	1216	640	3757	NMAP 3D	4119	1184	640	3672
GBMAP	4217	-	-	3572	PSMAP 3D	4119	1120	640	3567
ONYX	4242	-	-	3612	DPSOMAP3D	4119	1120	640	3567
CGMAP	4300	-	-	3600	ILP 3D	4103	1120	640	3567
NMAP	4265	1344	640	3852	TSBM	4103	1120	640	3567
CHMAP	4167	1344	640	3852					
ILP	4119	1120	640	3567					
Castnet	4135	1280	-	3852					

7. CONCLUSION

his study introduces an algorithm for NoC architectures that is based on the CSA, targeting both 2D and 3D mesh topologies. We address the application core placement problem, which is known to be NP-hard and combinatorial in nature. To evaluate the proposed approach, experiments were carried out using both real applications and synthetic benchmark core graphs. A single-objective optimization strategy was adopted, where communication cost served as the primary evaluation criterion. The experimental results demonstrate that the CSA achieves superior performance compared to other standard algorithms when appropriate parameter tuning is applied. However, for real benchmarks, only the VOPD case yielded the best performance, while the outcomes for other benchmarks revealed certain limitations. These limitations mainly stem from the use of a generalized configuration across all benchmarks, whereas each benchmark requires specific parameter tuning, particularly in terms of iteration count and population size. For future work, we aim to extend our approach to a multi-objective optimization framework and explore hybridization with other metaheuristic algorithms in order to further improve performance.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Maamar Bougherara	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Amina Guidoum					✓					✓	✓			
Amara Rafik					✓					✓	✓			

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal Analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject Administration
Fu : **Fu**nding Acquisition

DATA AVAILABILITY

The authors confirm that the data supporting the findings of this study are available in [30].

REFERENCES

[1] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proceedings of the ASP-DAC Asia and South Pacific Design Automation Conference*, 2003, pp. 233–239, doi: 10.1109/ASPDAC.2003.1195022.

[2] W. R. Davis *et al.*, "Demystifying 3D ICs: The pros and cons of going vertical," *IEEE Design and Test of Computers*, vol. 22, no. 6, pp. 498–510, 2005, doi: 10.1109/MDT.2005.136.

[3] M. Bougherara, N. Nedjah, L. D. M. Mourelle, R. Rahmoun, A. Sadok, and D. Bennouar, "IP assignment for efficient NoC-based system design using multi-objective particle swarm optimisation," *International Journal of Bio-Inspired Computation*, vol. 12, no. 4, pp. 203–213, 2018, doi: 10.1504/IJBIC.2018.096483.

[4] L. S. Junior, N. Nedjah, and L. D. M. Mourelle, "Efficient routing in network-on-chip for 3D topologies," *International Journal of Electronics*, vol. 102, no. 10, pp. 1695–1712, 2015, doi: 10.1080/00207217.2014.989545.

[5] N. Nedjah, M. V. C.s Da Silva, and L. de M. Mourelle, "Preference-based multi-objective evolutionary algorithms for power-aware application mapping on NoC platforms," *Expert Systems with Applications*, vol. 39, no. 3, pp. 2771-2782, 2012, doi: 10.1016/j.eswa.2011.08.137.

[6] Q. Chen, W. Huang, Y. Zhang, and Y. Huang, "An IP core mapping algorithm based on neural networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 1, pp. 189–202, 2021, doi: 10.1109/TVLSI.2020.3033658.

[7] 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.

[8] Y. Nalci, P. Kullu, S. Tosun, and O. Ozturk, "ILP formulation and heuristic method for energy-aware application mapping on 3D NoCs," *The Journal of Supercomputing*, vol. 77, no. 3, pp. 2667–2680, 2021, doi: 10.1007/s11227-020-03365-0.

[9] S. Murali and G. De Micheli, "Bandwidth constrained mapping of cores onto NoC architectures," in *Design, Automation and Test in Europe Conference and Exhibition Proceedings*, Paris, France, 2004, pp. 896–901, doi: 10.1109/DATE.2004.1269002.

[10] W. Shen, C. Chao, Y. Lien, and A. Wu, "A new binomial mapping and optimization algorithm for reduced-complexity mesh-based on-chip network," in *Proceedings of the International Symposium on Networks-on-Chip (NOCS)*, 2007, pp. 317–322, doi: 10.1109/NOCS.2007.5.

[11] S. Tosun, O. Ozturk, and M. Ozen, "Application mapping algorithms for mesh-based network-on-chip architectures," *The Journal of Supercomputing*, vol. 71, no. 3, pp. 995–1017, 2015, doi: 10.1007/s11227-014-1348-x.

[12] C.-H. Cheng and W.-M. Chen, "Application mapping onto mesh-based network-on-chip using constructive heuristic algorithms," *The Journal of Supercomputing*, vol. 72, no. 12, pp. 1–14, 2016, doi: 10.1007/s11227-016-1746-3.

[13] 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.

[14] 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.

[15] M. Z. Dageleh and M. A. J. Jamali, "V-CastNet3D: A novel clustering-based mapping in 3-D network on chip," *Nano Communication Networks*, vol. 18, pp. 51–61, 2018, doi: 10.1016/j.nancom.2017.11.002.

[16] 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.

[17] 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, pp. 1–7, 2010.

[18] 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, 2014, doi: 10.1109/TVLSI.2013.2240708.

[19] C. Huang, D. Zhang, and G. Song, "Low-power mapping algorithm for three-dimensional network-on-chip based on diversity-controlled quantum-behaved particle swarm optimization," *Journal of Algorithms and Computational Technology*, vol. 10, no. 3, pp. 176–186, 2016, doi: 10.1177/1748301816649070.

[20] P. K. Hamedani, S. Hessabi, H. Sarbazi-Azad, and N. E. Jerger, "Exploration of temperature constraints for thermal aware mapping of 3D networks on chip," in *20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing*, Munich, Germany, 2012, pp. 499–506, doi: 10.1109/PDP.2012.68.

[21] J. Wang, L. Li, Z. Wang, R. Zhang, and Y. Zhao, "Energy-efficient mapping for 3D NoC using logistic function based adaptive genetic algorithms," *Chinese Journal of Electronics*, 2014, doi: 10.23919/CJE.2014.10851854.

[22] P. Kullu, S. Yigit-Sert, M. Ozkan-Okay, and Y. Ar, "An artificial bee colony based mapping method for three-dimensional network-on-chip," in *2023 International Conference on Emerging Trends in Networks and Computer Communications*, 2023, pp. 1–6, doi: 10.1109/ETNCC59188.2023.10284966.

[23] F. Asadzadeh, A. Reza, M. Reshadi, and A. Khademzadeh, "Thermal-aware application mapping using genetic and fuzzy logic techniques for minimizing temperature in three-dimensional network-on-chip," *The Journal of Supercomputing*, vol. 80, no. 8, pp. 11214–11240, 2024, doi: 10.1007/s11227-023-05869-x.

[24] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Computers & Structures*, vol. 169, pp. 1–12, 2016, doi: 10.1016/j.compstruc.2016.03.001.

[25] A. Y. Abdelaziz and A. Fathy, "A novel approach based on crow search algorithm for optimal selection of conductor size in radial distribution networks," *Engineering Science and Technology, an International Journal*, vol. 20, no. 2, pp. 391–402, 2017, doi: 10.1016/j.jestch.2017.02.004.

[26] K.-W. Huang, A. S. Girsang, Z.-X. Wu, and Y.-W. Chuang, "A hybrid crow search algorithm for solving permutation flow shop scheduling problems," *Applied Sciences*, vol. 9, no. 7, 2019, doi: 10.3390/app9071353.

[27] A. Saha, A. Bhattacharya, P. Das, and A. K. Chakraborty, "Crow search algorithm for solving optimal power flow problem," in *2017 Second International Conference on Electrical, Computer and Communication Technologies*, 2017, pp. 1–8, doi: 10.1109/ICECCT.2017.8118028.

[28] A. G. Hussien *et al.*, "Crow search algorithm: Theory, recent advances, and applications," *IEEE Access*, vol. 8, pp. 173548–173565, 2020, doi: 10.1109/ACCESS.2020.3024108.

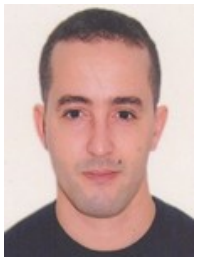
[29] R. P. Dick, D. L. Rhodes, and W. Wolf, "TGFF: Task graphs for free," in *Proceedings of the Sixth International Workshop on Hardware/Software Codesign*, IEEE, Seattle, WA, USA, 1998, pp. 97–101, doi: 10.1109/HSC.1998.666245.

[30] "TGFF Benchemarks NoC 2D 3D," ResearchGate Dataset, Jul. 2025, doi: 10.13140/RG.2.2.20887.07842.

BIOGRAPHIES OF AUTHORS

Maamar Bougherara is an Associate Professor at the École Normale Supérieure of Kouba, Algeria. He holds a Ph.D. in Computer Science from the LIM Laboratory at Bouira University, Algeria. His current research focuses on networks-on-chip (NoC) and optimization algorithms. He also holds both an M.Sc. and an Engineering degree in Computer Science from the University of Blida, Algeria. He can be contacted at email: bougherara.maamar@gmail.com.

Amina Guidoum an Associate Professor at the Higher School kouba, Algiers, Algeria, she holds Master of Science in Computer Science from the University of Sidi Bel Abbès; Doctor of Philosophy (Ph.D.) in Network, Architecture, and Multimedia from the University of Sidi Bel Abbès. Her research interests include images processing, multimedia systems, machine learning, and emerging technologies in multimedia. She can be contacted at email: Amina.guidoum@g.ens-kouba.dz.



Rafik Amara is an assistant professor at Department of Computer Science, 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 in 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. He can be contacted at email: rafik.amara@g.ens-kouba.dz.