algorithms

MDPI

Article

IWO-IGA—A Hybrid Whale Optimization Algorithm Featuring Improved Genetic Characteristics for Mapping Real-Time Applications onto 2D Network on Chip

Sharoon Saleem, Fawad Hussain * and Naveed Khan Baloch

Department of Computer Engineering, University of Engineering & Technology, Taxila 47050, Pakistan; sharoon.saleem@uettaxila.edu.pk (S.S.); naveed.khan@uettaxila.edu.pk (N.K.B.)
* Correspondence: fawad.hussain@uettaxila.edu.pk

Abstract: Network on Chip (NoC) has emerged as a potential substitute for the communication model in modern computer systems with extensive integration. Among the numerous design challenges, application mapping on the NoC system poses one of the most complex and demanding optimization problems. In this research, we propose a hybrid improved whale optimization algorithm with enhanced genetic properties (IWOA-IGA) to optimally map real-time applications onto the 2D NoC Platform. The IWOA-IGA is a novel approach combining an improved whale optimization algorithm with the ability of a refined genetic algorithm to optimally map application tasks. A comprehensive comparison is performed between the proposed method and other state-of-the-art algorithms through rigorous analysis. The evaluation consists of real-time applications, benchmarks, and a collection of arbitrarily scaled and procedurally generated large-task graphs. The proposed IWOA-IGA indicates an average improvement in power reduction, improved energy consumption, and latency over state-of-the-art algorithms. Performance based on the Convergence Factor, which assesses the algorithm's efficiency in achieving better convergence after running for a specific number of iterations over other efficiently developed techniques, is introduced in this research work. These results demonstrate the algorithm's superior convergence performance when applied to real-world and synthetic task graphs. Our research findings spotlight the superior performance of hybrid improved whale optimization integrated with enhanced GA features, emphasizing its potential for application mapping in NoC-based systems.

Keywords: whale optimization algorithm; genetics algorithm; network-on-chip; real-time; parameter control

Citation: Saleem, S.; Hussain, F.; Baloch, N.K. IWO-IGA—A Hybrid Whale Optimization Algorithm Featuring Improved Genetic Characteristics for Mapping Real-Time Applications onto 2D Network on Chip. *Algorithms* **2024**, *17*, 115. https://doi.org/10.3390/a17030115

Academic Editors: Łukasz Knypiński, Ramesh Devarapalli and Marcin Kaminski

Received: 16 February 2024
Revised: 4 March 2024
Accepted: 6 March 2024
Published: 10 March 2024

1. Introduction

In the current era of multicore systems, core integration on System-on-Chip (SoC) devices has increased significantly due to ongoing research and development. However, this significant growth in the integration density of processing elements on System-on-Chip (SoC) devices raises significant performance and scalability concerns. The conventional bus-based architecture fails to satisfy the ever-increasing requirements for high-volume and high-speed communication imposed by the increasing number of cores. Therefore, looking for alternate approaches to overcome these constraints and raise the effectiveness of multicore systems is vital. Network-on-Chip (NoC) has surfaced as a viable solution [1,2] to meet the current communication needs of the very large scale integration (VLSI) paradigm at the deep nanoscale level. Network on Chip comprises integrated processor cores (IP), a network interface (NI), routers, and the links that connect them. In NOC, cores employ a packet-based switching technique for communication through the routers with the help of interconnection links. The NoC topology signifies the physical organization of the architecture, defining the arrangement of routers and cores. Various standard topologies have been designed and employed for NoC depending on the interconnected networks.

Out of these, the mesh topology is the most prominent [3,4], featuring shorter paths between interconnected cores and high bisection width. These features make it suitable for use in numerous application mapping techniques. Due to its built-in parallelism and concurrent communication features, NoC is a popular computing engine for future many-core real-time systems [5]. Of the many key concerns in NoC designs, such as router architecture, topologies, and routing algorithms, the NoC's IP mapping issue has gained substantial attention [6]. IP mapping is an NP-hard combinatorial optimization issue [7,8] that requires discovering the best way to map IPs onto a certain NoC architecture while attempting to meet predefined metrics [9]. The random placement of the IP core in NoC designs does not effectively influence the network's overall efficiency. Achieving high performance with low communication cost, latency, and throughput requires excellent or even optimal IP core mapping onto NoC systems.

Because a mapping problem with n number of IPs leads to n! possible solutions, obtaining optimal solutions through exhaustive enumeration of all permutations is not practical. Consequently, more efficient methods must be discovered to overcome this issue. As it is an NP-Hard problem, there are no exact approaches to finding the solution in polynomial time. Even small-scale cases may need significant computing time [10]. Therefore, heuristic approaches are a practical and effective strategy for finding high-quality solutions [6,11]. In addition, rapid growth in the amount of data represents a challenge to researchers and data scientists in analyzing and extracting relevant information. Swarm intelligence is increasingly used for many optimization tasks, including feature selection, a complex task essential for reducing data dimensionality in high-dimensional datasets [12]. Although various heuristic-based techniques have been employed to solve the NoC application mapping problem, they still solve the problem at the cost of certain factors, and all have certain drawbacks. The well known Ant Colony optimization method has disadvantages such as parameter sensitivity, sluggish convergence rate, and precociousness. Certain evolutionary algorithms suffer from longer computation times in certain cases, along with poor stability. Although heuristic searches are extensively employed to solve the IP mapping issue, they often have the drawback of quickly settling on a local optimum. However, efficient local and global search capabilities on the part of heuristic algorithms can help to achieve optimum results. Hence, an optimization algorithm that balances exploration and exploitation while avoiding local optima may solve application mapping problems for both small-scale and large-scale applications in an ideal manner. Metaheuristic algorithms use a mathematical model of social evolution to efficiently solve optimization problems by promoting high-level methods and local improvement strategies.

Based on the factors and characteristics of the algorithm, an improved version of a nature-inspired meta-heuristic mapping technique is presented to achieve better results under allowed bandwidth limitations. The modified whale optimization algorithm (IWOA) features strong exploration and exploitation capabilities and a competitive convergence rate for a given set of problems. IWOA is proposed as the first step to finding an optimal solution to NoC application mapping, followed by implementing a hybrid IWOA featuring Improved Genetic Mechanism (IWOA-IGA). The genetic-based technique incorporates crossover and mutation to optimize IWOA-based results for optimal NoC mapping. The integrated tweaked GA features assist the IWOA in achieving the optimal mapping with faster convergence, allowing it to avoid local optima with enhanced search ability. These capabilities of the proposed algorithm make it superior to conventional techniques.

To analyze the NoC efficiency metric, current research seeks to model power, latency, and communication cost. Successful mapping strategies generate optimum NoC mapping with low communication costs, fewer iterations, and fewer processing resources. X–Y routing simplifies mapping problems. In a regular mesh design, network performance is measured by the cost of communication:

$$C_T = \sum_{i=1}^{size(ACG)} \sum_{j=0}^{te} [CB_{(te_i)}] \times HopCount_{(te_i)} \quad (1)$$

where t_e represents a collection of edges within a core graph, $B_{(te_i)}$ represents the communication bandwidth on edge i, and the number of hops that separate two communicating cores, referred to as $HopCount_{(te_i)}$, is used to quantify delay and communication costs [13,14]. Thus, as suggested by Equation (1), lowering communication costs indirectly reduces latency and energy usage.

IWOA-IGA offers an effective trade-off between performance measures and faster mapping onto 2D NoCs. The hybrid mechanism of the suggested method optimizes mapping results, reducing communication, energy, power, and latency costs. The contributions of the planned research are as follows:

- Based on an improved whale framework incorporating GA characteristics, a novel and effective solution to NoC application mapping is proposed.
- Improved Initial Mapping for IWO and direction-based crossover and mutation ability are introduced based on a ranked selection-based method during GA evolution.
- The final mapping technique delivers superior performance in terms of total communication costs. The proposed IWOA-IGA improves power, energy, and communication costs compared to existing bio-inspired algorithms.

The rest of this paper is arranged in the following manner: Section 2 presents the literature review; Section 3 discusses the mathematical formulation and performance metrics; Section 4 presents the framework of the proposed IWOA algorithm; Section 5 shows the enhanced version of the WOA algorithm featuring the improved genetic mechanism; our computational findings and analyses are presented in Section 6; Finally, Section 7 contains concluding remarks and discusses future research directions.

2. Related Work

In the current section, we briefly look into the current approaches to application mapping challenges for NoC design from several perspectives, namely, minimizing communication energy, enhancing performance, and reducing computation time. Small-scale IP mapping issues can be formulated as integer programming problems [15,16] and resolved using the branch and bound approach [17,18]. However, neither approach can effectively address extensive IP mapping issues due to their lengthy computation times. The most prominent approach for large-scale IP mapping issues is heuristic search, which may be loosely split into two kinds: transformative heuristics and constructive heuristics [13]. Typically, heuristic approaches map cores onto routers using pre-established criteria. NMAP, an efficient NoC application mapping approach [19], maps the application tasks to the cores in three phases: initialization, minimum path computation, and pair-wise swapping until the optimum mapping solution is obtained. BMAP is a greedy binomial mapping method [20] in which a candidate solution is first proposed, followed by iterative improvement until the final mapping is achieved. To achieve more solutions, CastNet generates various alternative solutions for each core depending on the availability of adjacent free neighbors by using multiple tiles to work as initial tiles, leveraging the symmetric features of the mesh [21]. In CHMAP (constructive heuristic mapping approach) [22], the root is chosen as the core with the highest average communication traffic, then a maximum spanning tree is constructed from the communicative graph. CHMAP calculates the mapping order based on the degree and distance of the cores and then assigns each core to the most suitable router based on mapping criteria. CastNet and CHMAP do not employ further iterative-based improvements upon the initial solution to reach the best possible solution. Transformative algorithms often use evolutionary approaches to look for approximations. Examples of common evolutionary algorithms [6] for optimization encompass genetic algorithms (GAs), discrete particle swarm optimization (DPSO), and ant colony algorithms. A DPSO-based technique that employs a multi-stage PSO and certain deterministic particles for the initial population rather than random ones was reported in [14]. Optimized Mapping solutions for both 2D and 3D NoC were presented. A novel optimization technique grounded on the DPSO framework was introduced in [23], where the velocity update process included a perturbation particle and an elite particle introduced to facilitate the exploration of local

optima. The algorithm allows particles to switch between elite and conventional pools, and uses a simplified local search of elite particles to find potential solutions. Multi-application mapping based on a reconfigurable NoC architecture was reported in [24], where a Mesh of Tree (MoT) topology was implemented. A two-step efficient Particle Swarm Optimization (PSO) approach was used to reduce the communication cost for a reconfigurable architecture. A contention-aware genetic algorithm-based application mapping solution was reported in [25], where both the spatial and temporal attributes of communication were considered in order to optimize performance by avoiding contention. To optimize the NoC in terms of communication cost, average latency, and energy, an efficient mapping technique was proposed [26] using the cuckoo search technique with Levy flight. First, a greedy algorithm was used to place the most communicative tasks together for an initial quality solution. Levy flying meta-heuristic cuckoo search was then used to optimize task placements for optimum mapping. In [27], the authors put forward the IHPSA optimization method, an enhanced hybrid PSO, and the simulated annealing technique. Enhanced Particle Swarm Optimization with SA was combined in the proposed IHPSA for application mapping. The K-means clustering machine learning method arranges tasks based on their communication bandwidth. The K-means clustering algorithm employed the elbow method to intelligently predict the number of clusters in extensive applications. Eventually. Heuristics were applied to achieve the optimal cost for real benchmark applications and synthetic instances. The authors of [28] presented RAMAN, a method inspired by Reinforcement Learning (RL), for 2D NoC-based application mapping. RAMAN is an enhanced Q-learning algorithm influenced by RL that aims to achieve mapping solutions with the lowest possible and optimized communication cost. The results of this method show tremendous potential in terms of lower complexity and cost while tackling application mapping problems. In [29], linear programming was employed to mathematically model the mapping problem in NoC. Various constraints related to communication capacity and power budget were incorporated. Finally, simmulated annealing integrated with GA was implemented to take into account and consider the constraints while finding the optimum solution. The run-time application mapping technique presented in [30] aimed to balance the load on the overall network when implementing NoC for a Cube–Tree Hybrid (CTH) topology. The authors exploited the low network diameter of this topology with its high scalability, resulting in reduced mapping overhead. A significant reduction in execution time was observed during experimentation. A hybrid task mapping HyDra was proposed in [31] that incorporates design time mapping and runtime reconfiguration with the aim of reducing communication energy. At design time, multiple mapping solutions are produced to reduce latency and energy. As the applications arrive at runtime, the design time mappings are either used according to the applications or reconfigured based on the requirements. While many such algorithms have been developed, they may suffer from suboptimal performance along with inefficient convergence and complexity. To tackle these challenges, the authors of [32] used Grey Wolf optimization (GWO), which starts with cluster-based initial mapping followed by a modified GWO heuristic algorithm with polynomial regression. The modified algorithm enhances runtime efficiency and optimizes the overall mapping quality. In [33], the authors discussed various real-time application mapping techniques for complex multicore platforms. They categorized these techniques by emphasizing their optimization goals, such as communication cost and energy consumption in NoC-based systems. Future challenges, trends, and simulation tools in this area were presented as well. Application mapping in both 2D and 3D remains a complex challenge that requires further development of efficient algorithms. To address the outstanding challenges, a neural mapping model with a reinforcement learning (RL) approach (NeurMap3D) was presented in [34] to develop application-specific 3D NoCs. In addition, a neural congestion-aware mechanism was presented to address application mapping issues via placement and mapping and to incorporate TSV placement and load balancing in NoC.

3. Mathematical Formulation/Performance Metrics

This section discusses NoC platform assessment criteria such as communication cost, latency, energy, and power.

3.1. Communication Cost Model

Communication Cost in NoC-based formulations is presented in Equation (2):

$$Cost = \sum_{i,j} [B_{t_i,t_j} \times N_{i,j}], \tag{2}$$

where B_{t_i,t_j} is the bandwidth between tile t_i and t_j and $N_{i,j}$ is the Manhattan distance. The NoC architecture's Manhattan distance between the source nodes (x_i, y_i) and destination nodes (x_j, y_j) is provided by

$$N_{i,j} = |x_i - x_j| + |y_i - y_j|. \tag{3}$$

3.2. Power Model

Power and energy are calculated based on the execution of a given traffic pattern. $Pw_{act,j}$ and $Pw_{inact,j}$ are component j's active and passive power at 1.0 V and 1.0 GHz, respectively. Let $alpha_{i,j}$ be the active reading of component j in router i; then, the average power Pw_{av} [35] is expressed by

$$Pw_{av} = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N_i} [\alpha_{i,j} \cdot Pw_{act,j} + (1 - \alpha_{i,j}) \cdot Pw_{inact,j}] \tag{4}$$

where $Pw_{act,j}$ is j's component power when active, $Pw_{inact,j}$ signifies component j's power while inactive, and $\alpha_{i,j}$ reflects active measurement of component j in the router i.

3.3. Latency Model

The average latency of the NoC is shown in Equation (5):

$$Lt_{av} = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{N_i} \sum_{j=1}^{N_i} Lt_{i,j} \tag{5}$$

where $Lt_{i,j}$ is the latency of packet j from one tile to another, N represents the number of processors in the mesh, and N_i reflects the quantity of packets received by a processor after the warm-up period.

3.4. Energy Model

The energy model estimates the network router energy consumption, represented as follows:

$$E_B = E_s B + E_L B \tag{6}$$

where E_B consists of the switch energy $E_s B$ and link energy $E_L B$ of the NoC. It is this energy that is consumed to transfer one bit data between the source and the destination. The following equation computes the average network energy consumption, denoted as $EB_{(p_i,p_j)}$, used for the transfer of one bit of data between source (p_i) and destination (p_j):

$$E_{B(p_i,p_j)} = H_{count} E_{SB} + (H_{count} - 1) E_{LB} \tag{7}$$

where H_count represents the Manhattan distance between the source nodes (a_i, a_j) and destination nodes (b_i, b_j).

$$H_{count} = |a_i - b_i| + |a_j - b_j| \tag{8}$$

Consequently, the network's total energy consumption (ET) is determined by the average network energy and the link bandwidth $BW_{(p_i,p_j)}$ from p_i to p_j.

$$E_T = \sum_{i,j}^{H_{count}} (E_{B(p_i,p_j)} \times BW_{(p_i,p_j))} \tag{9}$$

Hence, the final equation takes the form

$$E_T = \sum_{i,j} \Big[(H_{count} \times E_{SB} + (H_{count} - 1) \times E_{LB}) \times BW_{(p_i,p_j))} \Big]. \tag{10}$$

For cost computation (CC), the following equation can be used:

$$CC = \sum_{i,j} (H_{count} \times BW_{(p_i,p_j)}). \tag{11}$$

Various mappings lead to distinct energy and communication cost values. The primary aim is to derive a mapping function for the NoC with minimal cost. In this research, we use communication cost as the main performance metric for distinct applications.

3.5. Mathematical Formulation Model

Application mapping for 2D NoC design, including preliminary knowledge and a mathematical model, is presented in this subsection. The following elements constitute the inputs for formulating the application mapping problem.

Definition 1. *A Communication Trace Graph (CTG) is defined as an undirected communication trace graph $G = (V, E, M)$ that is weighted and consists of a set of vertices or cores, with V, E representing the directed edge set and M referring to the volume of data and the connectivity between nodes in MS/s.*

$$V = \{c_1, c_2, c_3, \ldots, c_n\} \tag{12}$$

where $|V| = finite(n)$

$$E = \{e_{i,j} = (c_i, c_j) \in C \times C | (c_i, c_j) \in C, i \neq j\} \tag{13}$$

and

$$M : E \rightarrow M(c_i, c_j) = M_{ij} \tag{14}$$

where $|V| = finite(n)$.

Definition 2. *In the Topology Graph (TG) for a network, TG = (T, L, M) constitutes the tile collection T placed in the network topology, the collection of links for tiled pairs in T is referred to as L, and M specifies the data volume and link between the tiles in megabytes per second.*

$$T = \{t_1, t_2, t_3, \ldots, t_n\} \tag{15}$$

$$L = \{l_{i,j} = (t_i, t_j) \in TxC | (t_i, t_j) \in T, i \neq j\} \tag{16}$$

$$M : L \rightarrow M(t_i, t_j) = M_{ij} \tag{17}$$

Definition 3. $NAG = (R, C)$ *: The NoC Architecture Graph corresponds to the routing path* $(c_{i,j}) \in C)$ *between any router pairs* (r_i, r_j) *in the network. The intermediate links traversing the path* (r_i, r_j) *are termed hop counts (Hops) from the router* r_i *to* r_j*. Data are transferred or received to and from the cores through the connected routers, and the channel* $c_{i,j}$ *acts as the physical link for data transfer between the cores. The routing channel has limited bandwidth* $(B_{i,j})$ *between the nodes.*

4. Proposed Framework of IWO Algorithm

This section presents a new metaheuristic IWOA for a 2D NoC-based multicore platform with strong local and global search capabilities. On the whole, the algorithm attains several mapping solutions (explores the promising search space) by emulating whale hunting behavior, eventually converging to the best possible solution.

4.1. Inspiration for NoC Application Mapping

The metaheuristic method described in [36] simulates hunting behavior using a random agent or the best search agent to track prey. Researchers have found that Humpback Whales pursue prey by creating distinctive bubbles in a circular pattern, and are the only species known to engage in bubble-net feeding [37,38]. The hunting mechanism of whales is modeled in the proposed work to carry out NoC application mapping optimization. The whales' hunting mechanism uses search agents to find the optimal position. As the search agent approaches the prey, the other agents update their locations to discover the best solution. Each search agent corresponds to the task mapping solution, which is assessed on the basis of communication cost. Based on the reference mapping solution, other mapping solutions update their locations (tasks to core mapping). The basic WOA algorithm employs random initial mapping solutions, which are improved over further WOA iteration. However, in this work we modify the WOA algorithm to generate an initial mapping solution using a set of criteria, providing the optimization algorithm with a head start. The algorithm then progressively improves the initial mapping through further iteration.

Modified WOA with Targeted Initial Mapping Generation Heuristic searches for IP mapping are frequently employed; however, they often become trapped in local optima. This shortcoming is primarily attributable to the first population/solution generation stage of heuristic searches. Most transformational heuristics generate a random starting population, and the search procedure can then be seen as a search of the entire search space. In some heuristic procedures, the initial population/solution is formed using predefined criteria, expert information, and/or system attributes. These criteria may not work for large or complex problems. If the initial population generation is not effective, incremental improvement towards finding an optimal solution will be limited. In swarm-based intelligence optimization techniques, the initial population quality has a significant impact on the algorithm's speed and accuracy [39,40]. In the proposed work, the WOA algorithm is modified in terms of initial mapping generation, and is then employed to enhance overall application mapping to NoC Cores. The initial population is produced using the good point set approach instead of the random initial population generation in the standard WOA. This method is an efficient approach that can aid in minimizing the number of attempts and reaching optimal solutions in fewer iterations. The solutions produced using the good point set sequence exhibit a better distribution of solutions than the sequences chosen by the general random method [41]. This approach is used to generate initial mapping solutions for mesh-based 2D NoC architectures. To generate a mapping sequence using the good point set method, a sequence of m points in s-dimensional space is generated and represented as discussed below: Generate points such that

$$r = r_1, r_2, r_3, \tag{18}$$

$$r_i = 2 \times Cos(2i/p), \tag{19}$$

where i can be set to any value for generating various mapping points, and let p be a prime which satisfies $p \geq 2s + 3$. Now, using the several mapping points generated, a mapping sequence is generated that is stored in $T_{(s-map)}$:

$$T_{(s-map)} = (r_1 \times k), (r_1 \times k), (r_1 \times k), ...(r_m \times k) \tag{20}$$

where $r_1 - r_m$ are the total mapping points generated and k = 1, 2, 3, …, n, where n is designated as the total number of cores for which mapping is generated. For NoCs having

certain application tasks to be mapped onto s × s sized mesh architectures, we can take m as the number of generated tasks or points. As the population size can be set to any number, various individual $T_{(s-map)}$ mapping solutions are generated as an initial generation, e.g., for 16 tasks to be mapped onto a 4 × 4 Mesh 2D NoC platform, we can set m to a certain number and set *popSize* to another number for which diverse task mapping solutions are to be created. The algorithm of the good point set method is provided in Algorithm 1. The set of initial mappings generated using the good point set method can be considered as one of the pools of the initial mapping $initMapping_{gps}$. The good point set method does not use any key characteristics such as communication between tasks and the number of connected neighbors. Thus, application task graph expert knowledge can be utilized to generate initial solutions. This initial mapping can be placed in a second pool based on communication between the cores, termed $initMapping_{comm}$. The final initial mappings are selected from the top-ranked mappings of both pools, which participate in the optimization algorithm. The second pool of initial mappings maps application tasks to the NoC platform using the communication weights between core edges. A core is first selected from the core graph and that specific core's total communication bandwidth and average communication are noted.

Algorithm 1 Initial Mapping Algorithm

Initialization:
Input: $popSize, m$
stopCriteria: $popSize$
while $stopCriteria$ **do**
  for $i \leftarrow 1$ to m **do**
    Calculate r_i using Equation (19) to generate mapping points considering value of P
    Create Tasks to core mappings $T_{(s-map)}$ using Equation (20)
  end for
  $initMapping_{gps} = T_{(s-map)}$
end while

The total weight TW of the selected core is provided by

$$TW_i = \sum_{eije/E} tw_{ij}. \tag{21}$$

The average communication is expressed as follows:

$$TW_i = \sum_{eije/E} tw_{ij} \times (1/N_{c_i}) \tag{22}$$

where TW_i is the total communication weight between the cores and $N_{(c_i)}$ is the number of available neighbor cores of core c_i. In this way, the total communication and average communication of each specific core are calculated and the neighbors of every core are noted along with communication-related details. The first task to be selected is the one with the highest total communication bandwidth, which is placed onto the NoC platform at any location, and its neighbors are mapped next. If multiple neighbors are available, then the neighbor with a higher communication weight with the mapped task is placed at the closest location to the NoC. Then, the other neighbors of the first mapped task are placed based on their communication weights. When all the neighbors of the mapped task have been placed, the neighbors of already-mapped (recently mapped) tasks are placed. The choice is based on the mapped task with the maximum weight; this core is selected and its neighbors mapped using the same communication weight criteria as above. All of the cores are mapped in this manner. This initial mapping procedure assumes that any job mapped to a core must not be mapped to any other NoC location or considered for any other placement. To serve this statement well in the implementation of this algorithm, a list

of tasks that have been mapped is maintained in the list T_{mapped}, while the unallocated cores are maintained in $T_{un\text{-}allocated}$. While mapping neighbors of already mapped cores, these two lists are referenced to reach the appropriate decision. As the algorithm runs, various mappings are achieved based on placement of cores on the NoC platform in different orders. The communication cost for both pools is computed and the top-ranked elite mappings are taken from both initial mapping pools to generate an enhanced initial mapping solution. The reason for retaining and finally gathering mappings from both pools is to improve the overall diversity of the solution.

$$enhancedInitMapping = initMapping_{gps} + initMapping_{comm} \tag{23}$$

The best mapping solution is selected as the current best candidate solution. Optimization improves initial mapping solutions across numerous generations to provide final mappings for varied applications.

4.2. Mathematical Model of IWOA

This section provides the mathematical models of various hunting capabilities of the IWOA in terms of the NoC application mapping problem. Later, the IWOA algorithm using the specified features and characteristics is presented.

4.2.1. Encircling/Navigating the Hunt: Unveiling Optimal Application Mapping Strategies

The whales have a special feature of trying to identify and encircle the prey's location. Because the ideal location within the search domain is not known in advance, the algorithm implies that the current leading/best candidate mapping solution is near the optimal design. The algorithm runs for a specific number of generations and iterations; consequently, the other solutions endeavor to modify their positions with the optimal mapping solution. These equations serve to represent this:

$$D = |\vec{C}.\vec{X}^*(t) - \vec{X}(t)|, \tag{24}$$

$$\vec{X}(t+1) = \vec{X}^*(t) - \vec{A}\vec{D}, \tag{25}$$

where t represents the current iteration, $\vec{A}$ and $\vec{C}$ are quantities that represent coefficient vectors, $\vec{X}^*$ is supposed to be the most effective mapping solution found in the present generation, and $\vec{X}$ represents position vector. If a superior mapping solution exists, then $X\vec{X}^*$ should be updated after each generation or iteration. $\vec{A}$ and $\vec{C}$ are computed using the following equations:

$$\vec{A} = 2\vec{a}\vec{r} - \vec{a}, \tag{26}$$

$$\vec{C} = 2.\vec{r} \tag{27}$$

In both the exploration and exploitation phases, a diminishes linearly from 2 to 0 throughout the iterations, while r represents a random number between [0, 1]. Figure 1 demonstrates the information behind Equation (25) for the 2D NoC problem and shows how mapping solution 1 (with a certain communication cost, commCost1) is updated to a new mapping solution 2 with a better communication cost (commCost2), where commCost2 < commCost1. Equation (25) allows the algorithm to update the mapping solution within the vicinity of the current best solution, leading to an optimum solution over a certain number of iterations and population sizes. The same principle can be applied to an n-dimensional search space in 2D NoC, with the mapping search agents traversing in hypercubes around the current most promising mapping solution.

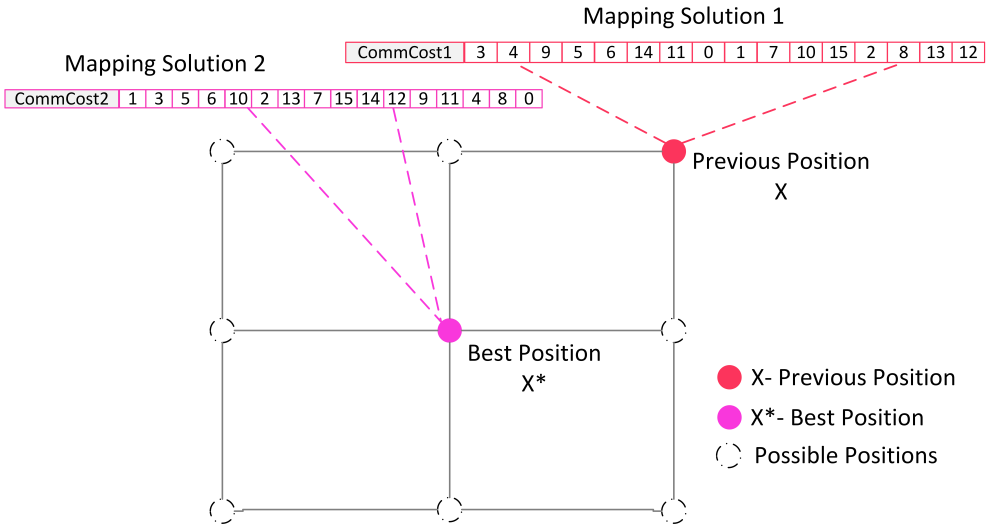

Figure 1. Task allocation and fitness updating process in IWOA for NoC platform.

As noted in the preceding section, the IWOA employs the bubble-net hunting tactic of the WOA integrated into NoC-based application mapping. This strengthens the algorithm's ability to exploit and explore, leading to better mapping solutions. The mathematical formulation of this procedure is as follows.

4.2.2. Exploitation Phase

To efficiently solve the application mapping problem, the IWOA uses the bubble net behavior of whales to achieve an optimum mapping solution within a faster convergence time.

1—Shrinking Encircling Mechanism: Exploitation for Mapping. In a 2D space, Figure 2 depicts the potential positions starting from (X) towards (X^*) that can be obtained by $0 \leq A \leq$. The shrinking encircling mechanism is modeled in a 2D NoC architecture such that X represents a mapping solution with a certain communication cost. In contrast, X^* is a better solution with a better communication cost than the earlier one. The mapping denoted as X tries to achieve a better solution, represented by X*, with a better communication cost. This mechanism runs for a certain number of iterations until final mapping is achieved. This mechanism to achieve optimal mapping uses Equation (26) by altering a value. In the equation, A represents a random value with a range of $\check{}a$ to a, where a gradually diminishes from 2 to 0 over a specified number of iterations.

2—Spiral Strategies: Elevating NOC Application Mapping. This approach first approximates the deviation between the value of the objective function for the current mapping solution X and the optimal value that needs to be achieved. The spiral update position to achieve the optimal mapping is derived using the following equation:

$$\vec{X}(t+1) = \vec{D}^{'}(t) \cdot e^{bl} \cdot cos(2\pi l) + \vec{X}^*(t) \tag{28}$$

where $\vec{D}^{'}(t)$ is provided by

$$\vec{D}^{'}(t) = |\vec{X}^*(t) - \vec{X}(t)|. \tag{29}$$

The equation shows the value of the ith mapping from the best mapping solution obtained thus far, where b determines the logarithmic spiral's shape while l is a random number. As the mapping is achieved using the shrinking circle and the spiral-shaped mechanism simultaneously, it is presumed that there is an equal 50% probability of selecting either the shrinking circular mechanism or the spiral model to update the mapping methods. The mathematical model takes the following form:

$$D_{it} = \begin{cases} \vec{X}^*(t) - \vec{X}(t) & \text{if } p < 0.5 \\ \vec{D}'(t) \cdot e^{bl} \cdot cos(2\pi l) + \vec{X}^*(t) & \text{if } p \geq 0.5 \end{cases} \quad (30)$$

where p represents a random value between 0 and 1. Figure 3 shows how various mapping solutions are created from X until reaching the final solution X*.

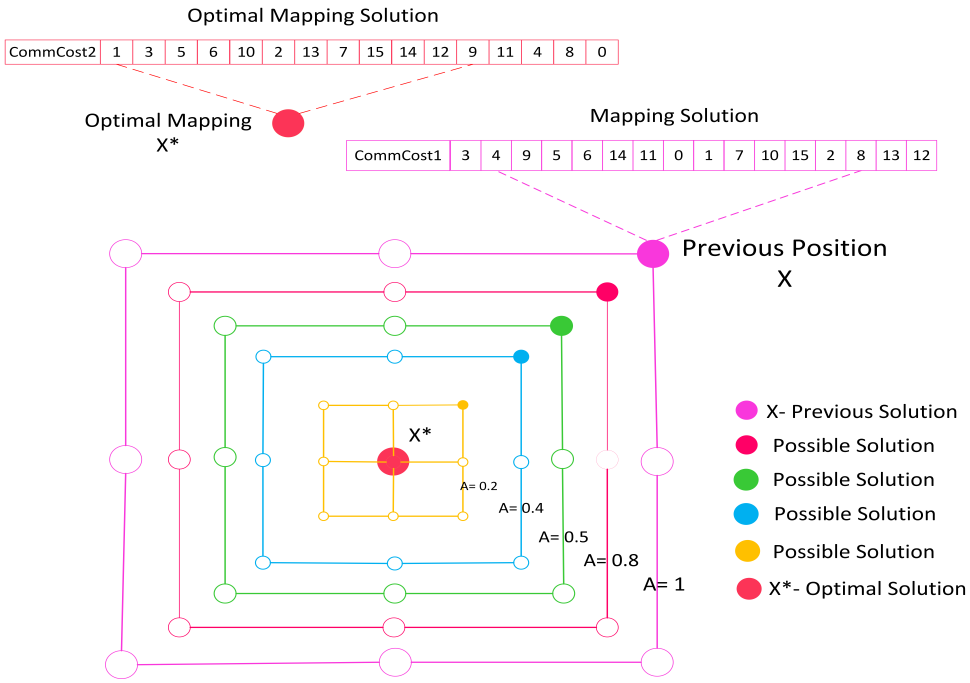

Figure 2. Shrinking encircling mechanism (X^* is the current best mapping solution).

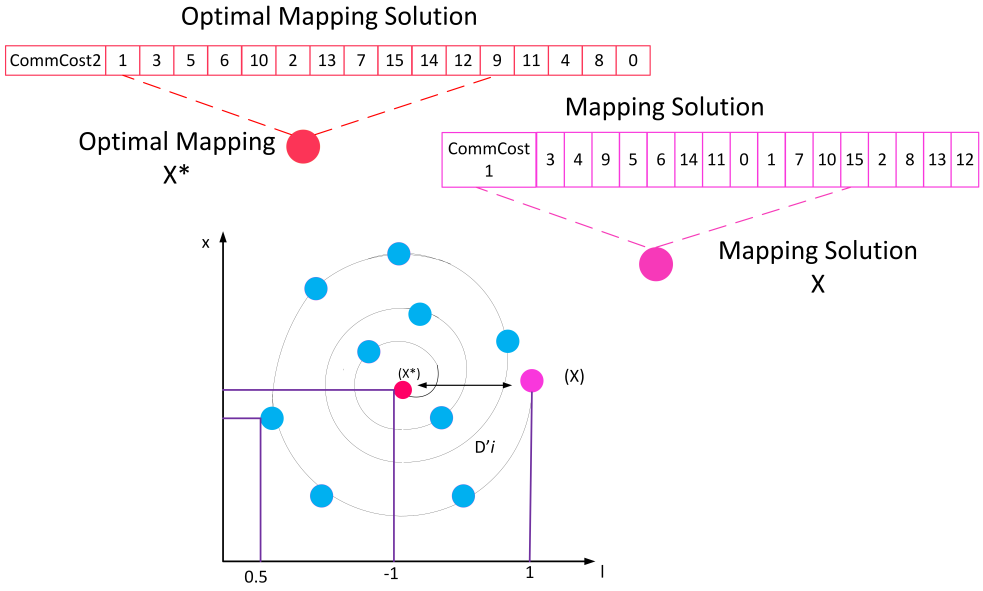

Figure 3. Spiral update position implemented in WOA (X^* is the current best solution).

4.2.3. Search for the Optimal Mapping Solution (Exploration Phase)

Thanks to its faster operation, the IWOA can use the strong exploration capabilities of the whale optimization algorithm to thoroughly explore in order to converge to the optimal solution. A variation pattern of vector A helps to find the best potential mapping with the lowest communication cost. As the mapping solutions are created by taking inspiration from the neighboring mapping solutions, we utilize A with certain randomized values

between 1 and −1 in order to compel the search agents to distance themselves from the reference mapping solution and explore more of the overall search space. Compared to the exploitation phase, adjusting the placement of a search agent (mapping solution) does not take place during exploration based on the best solution. Instead, we choose a search agent (a random mapping solution) at random and perform updates based on it. This method, along with $\mid A \mid > 1$, emphasizes exploration and enables the IWOA to perform a global search. Its mathematical model is presented as follows:

$$\vec{D} = |\vec{C} \cdot \vec{X_{rand}} - \vec{X}|, \tag{31}$$

$$X(t+1) = \vec{X_{rand}} - \vec{A} - \vec{D}, \tag{32}$$

where X rand is a mapping solution picked at random from the current population. Several mapping solutions surrounding a specific solution with $A > 1$ are created.

4.3. Proposed IWOA

In this section, a new and improved metaheuristic IWOA is presented that has powerful local and global search functionality. Each search agent that constitutes the mapping solution is initialized using a certain approach between the minimum and maximum limit in the range. In the proposed technique, the fitness function is often a primary goal to be reduced. The proposed algorithm commences with a collection of solutions achieved using the enhanced initial mapping mechanism instead of the conventional random mappings. At each iteration, search agents that represent individual mapping solutions adjust their positions concerning either a randomly chosen search agent or the current optimal solution with the minimum communication cost. The number of iterations is predefined, and varies depending on the application. Each mapping solution S is in the solution space SS, where $S \epsilon SS$ represents its components such that $n > 0$ components, i.e., $S = S_1, S_2, \ldots..S_n$, where $i = 1, 2, 3, \ldots.., n$; here, n represents the scope of the optimization problem to be solved. To promote better exploration and exploitation, the value of parameter a is varied from 2 to 0. Random mapping is chosen in the proposed mapping technique when $\mid A \mid > 1$. In contrast, the selection of the best candidate mapping is carried out when $\mid A \mid < 1$ for modifying the mapping solutions of other search agents within the search space. This adaptive variation-based feature of the vector $\vec{A}$ permits the IWOA to transition seamlessly between exploration and exploitation while achieving better mapping results. After performing in-depth analysis and comparison, the appropriate values for the input parameters of the algorithm are determined. It should be noted that a parameter combination that works well for one situation may not work the same way for another. Algorithm 2 provides the pseudocode of the IWOA. In each iteration, the algorithm efficiently obtains a better mapping solution as compared to the previous iterations, depending upon the fitness function. The termination criteria is the achievement of the required fitness value, that is, when the mapping solution with the lowest communication cost has been obtained. The IWOA's method for solving the optimization issue is a global optimizer which includes exploration and exploitation capabilities. As the algorithm runs, the initial mappings are fine-tuned to adjust the placement of tasks to the cores, ensuring that better solutions are achieved in each iteration. The flowchat of IWOA is presented in Figure 4.

Algorithm 2 Improved Whale Optimization Mapping Algorithm

Initialization:
Input: $enhancedInitMapping, taskGraph, itrWOA, itrIGA, a, C, l, A, p$
stopCriteria: $itrTotal, OptimalCommCost$
// Find pop, v, fitness
$initPopFitness(enhancedInitMapping, pop_size, meshSize)$
// Find gbestpop, gbestfitness
$getinitbest(fitness, pop)$ $//X^* = BestSearchAgent$
while $stopCriteria$ **do**
  for $i \leftarrow 1$ to $itrWOA$ **do**
    for $searchAgentj \leftarrow 1$ to pop_size **do**
      Update parameters A, C, a, l, p
      $\vec{A} = 2\vec{a}.\vec{r} - \vec{a}$
      $\vec{C} = 2.\vec{r}$
      if $p < 0.5$ **then**
        if $|A| < 1$ **then**
          Update the current mapping solution using Equation (24), finally updating $pop[j]$
        end if
        if $|A| \geq 1$ **then**
          Select a random mapping solution-Xrand
          Update the position of the current mapping solution by the Equation (32), finally updating $pop[j]$
        end if
      end if
      if $p \geq 0.5$ **then**
        Update the position of the current mapping solution by the Equation (28) updating $pop[j]$
      end if
    end for
    // Check If a search agent-based solution exceeds the search space, modify it.
    for $particlej \leftarrow 1$ to pop_size **do**
      $calCommCost(meshSize, pop[j], tashGraph)$
    end for
    $gbestfitness = mim(pop[j])$
    $gbestpop = min(pop)$
  end for
  if $gbestfitness == optimalCommCost$ **then**
    $finalmapping \leftarrow gbestpop$
    $optimalCommCost \leftarrow gbestfitness$
  end if
  $itrIWOA = itrIWOA + 1$
  $update(gbestfitness, finalMapping$
end while

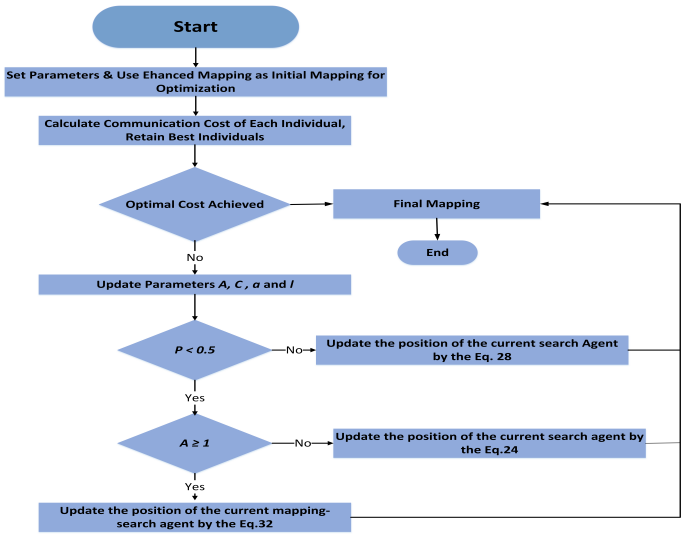

Figure 4. IWOA Flowchart.

5. IWOA-IGA—Enhanced WOA Algorithm Featuring Improved Genetic Mechanism

In this paper, a modified and improved WOA (IWOA) is presented to solve the application mapping problem in NoC. The incorporated IWOA algorithm has better global and local search ability; however, for certain scenarios, such as large-scale complex problems, it may undergo lower convergence accuracy and can fall into local optimum. To avoid such problems and scenarios, an improved GA is executed along with the IWOA to achieve optimal solutions. The GA uses a biological evolutionary mechanism, and, as a global-based search optimizer, it searches for the optimum solution in the search space for complex problems. GAs have proven to be robust and effective in exploring complex spaces. Hence, they can effectively solve a wide range of pattern recognition, artificial intelligence, resource allocation, and similar complex challenges. To achieve better mapping solutions, the proposed modified IWOA algorithm is integrated with an improved genetic algorithm incorporating improved crossover and mutation abilities. This further enhances the ability to find the optimum solution within the search space with better convergence towards the final mapping solution. In addition, according to the original whale optimization algorithm [30], in order to accurately mimic whale behavior and obtain improved optimization results, evolution-based characteristics must be added to the WOA algorithm. Instead of using conventional GA characteristics such as random selection of parents and choosing random points for crossover [42], the proposed technique uses modified crossover and mutation. This can help to avoid local optima and promotes faster convergence with better searchability and higher population diversity. Hence, to achieve optimal mapping solutions for the 2D NOC platform, the modified GA is directly integrated with the IWOA, influencing the overall algorithm's ability to generate high-quality individuals with reduced energy, latency, and power requirements.

5.1. Important Aspects of Genetic Algorithm

The genetic flowchart depicted in Figure 5 resembles biological evolution [9]. A traditional GA begins with an initial random population comprised of randomly chosen chromosomes that produce offspring via crossover and mutation. It continues to work iteratively until a predetermined count of iterations is completed or a termination criterion is satisfied. A pre-set fitness function determines the fitness of the chromosomes and the communication cost on the NoC platform.

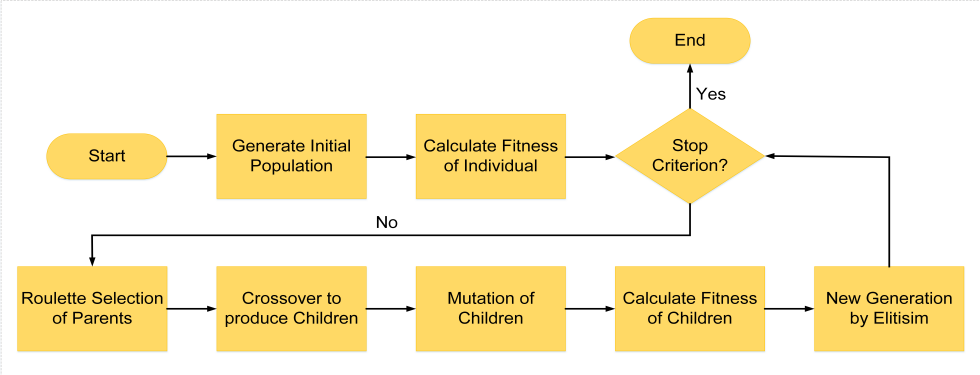

Figure 5. Genetic algorithm flowchart.

5.2. IGA Framework

This article focuses solely on a single objective, namely, use of an improved GA integrated with an improved WOA for solving application mapping through better search ability and population diversity. The population obtained from the IWOA is fed into the modified GA, which uses direction-based crossover features [43] and mutation ability to generate high-quality mapping solutions. Each chromosome in the GA signifies a mapping solution of the 2D NoC mapping problem. The subtasks imply the genes in the chromosomes. The various individual mapping solutions undergo through the genetic-based process for some number of iterations and generations to produce high-quality mapping solutions.

5.2.1. Improved Genetic Algorithm Formulation

The genetic operators exert different levels of impact on the algorithm, with selection identifying the most promising chromosomes for crossover to enhance the solutions. Crossover combines genetic data to improve population characteristics, while mutation introduces new genes to address the weaknesses of crossover.

5.2.2. Expert Initial Curation for NoC Application Mapping Excellence

The selection operator selects the individuals that participate in crossover and mutation; hence, their selection substantially impacts the entire GA process [44]. The proposed modified algorithm uses expert criteria-based selection to choose the initial mapping solutions to undergo crossover and mutation. This is contrary to the random selection used in the conventional GA. Initially, the mapping solutions achieved by the IWOA are ordered according to the objective function $F(X)$ (communication cost) and treated as an initial population for the IGA, represented as

$$X = \{X_1, X_2, \ldots\ldots X_n, \} \tag{33}$$

while the sorted population is provided by

$$X^s = \{X_1^s, X_2^s, \ldots..X_n^s\} \tag{34}$$

which satisfies $F(X_1^s) \geq F(X_2^s) \geq, \ldots\ldots F(X_n^s)$. Each element in X and X^s shows individual mapping solutions. The population obtained by the IWOA is sorted and divided into four groups (X^1, X^2, X^3, X^4), which are paired with each other $((X^1, X^2), (X^1, X^3), (X^1, X^4), (X^2, X^3), (X^2, X^4), (X^3, X^4))$ to form $X^{(map_a)}$ and $X^{(map_b)}$. Crossover is performed using the elements of $X^{(map_a)}$ and $X^{(map_b)}$.

5.2.3. NoC Application Mapping: Directional Optimization Incorporating Crossover and Mutation

The GA's central process significantly affects the algorithm's ability to seek improved and optimal solutions [45]. The more optimal the objective function is for any individual

solution, the closer the mapping combination will be to the optimal region. Hence, the proposed IWOA-IGA uses a direction-based crossover operator. The modified crossover operation has good ability for searching the overall search space, and can produce mapping solutions with a larger probability of enhancing the objective function, thereby speeding up the algorithm's convergence. As in the initial step of the GA, the population of mapping solutions is divided into two groups, $X^{(map_a)}$ and $X^{(map_b)}$. The communication cost of the individual solutions in $X^{(map_a)}$ is superior to those of the solutions in $X^{(map_b)}$; thus, $X^{(map_a)}$ is the leader in the direction of crossovers for producing high quality solutions. The mathematical equation that generates the mapping solutions with directional-based crossover is provided by

$$\begin{cases} X_i^* = X_i^{map_a} + r_{ij} \cdot \vec{D}_{ij} \\ \vec{D}_{ij} = X_i^{map_a} - X_i^{map_b} \end{cases} \quad i = 1, ...3n/2, j = 1, ..., m' \tag{35}$$

where $\vec{D}_{ij}$ represents the directional vector. The parameter r_{ij} is a uniformly distributed random number ranging from -1 to 1. Thus, in directional crossover based on grouping-wise solutions, each mapping pair develops into one mapping solution, eventually generating new individuals. Finally, it sorts the high-quality mapping solutions with the best communication cost, developing n individuals as offspring of the crossover.

To see the effect of the direction-based crossover operation on two of the paired mapping solutions, paired individuals are shown to occupy a certain space in the appropriate region. As r_{ij} takes on boundaries between 1 and -1, the paired mapping solutions will take various directions to reach the optimal communication cost value. The various directions for solution X_1 can be X_1 with D11, D12, D11, D12, as shown in Figure 6. The solution in the region is shown by the rectangle with its center at X_1 and X_2 as one vertex. Having now obtained various paired mapping solutions, they can be used to generate many such rectangular patterns; along with the selection process, direction-based crossover assists the algorithm in generating high-quality solutions in the direction of the optimal solution. Parents X_1 and X_2 can produce offspring that will tend to have better communication costs. The crossover operation recombines the parents' chromosomes, and the best offspring undergo mutation and continue on to produce superior results.

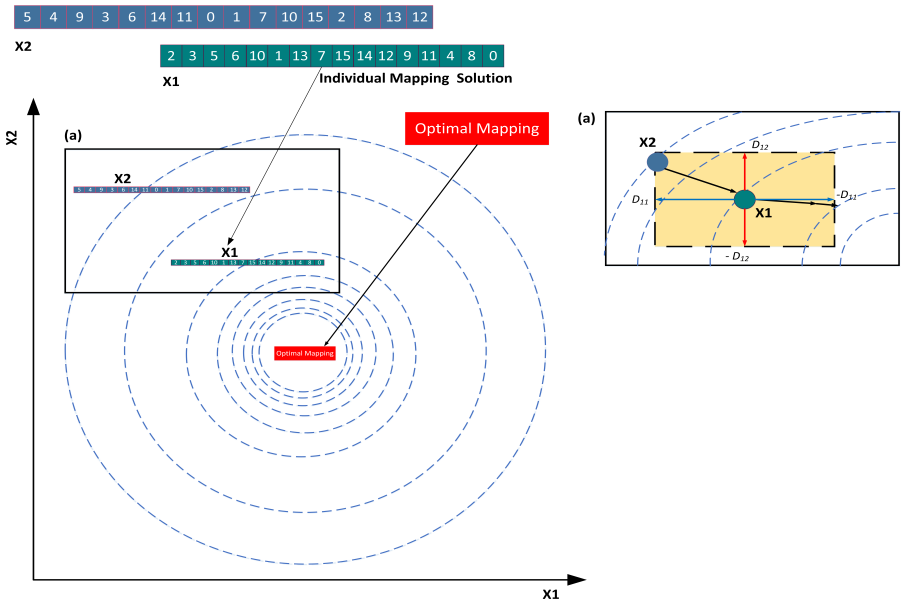

Figure 6. Direction-based crossover.

The mutation operation is applied to offspring to achieve more optimal results. Mutation allows for altering the placement of one or more mapped tasks of individuals in a population with a specified mutation probability, which can help to promote population

diversity and prevent premature events. The communication cost of all the resulting mappings are computed after mutation, and the mutation is accepted if the communication cost of the mutated mapping is less than that of the original ones. To increase population diversity and prevent the GA version from falling into local optima, the mechanism of replacement operation is incorporated [43]. The basic goal of the replacement operation is to set aside the most elite K individuals from the population for every K generation. In traditional approaches, certain elite individual solutions are retained in parents. Nevertheless, the high-quality solutions created as a result of crossover may be affected by the mutation operation, and their quality may degrade to some extent. Thus, in this work we adopt a better approach to retain the most elite mapping solutions produced by the genetic operation. In this way, the number of elite mapping solutions can be maximized to enhance the population attributes.

5.3. IWOA-IGA: Modified IWO Algorithm Featuring IGA Characteristics

This section proposes a new enhanced hybrid IWOA-IGA application mapping technique for 2D NoCs. This algorithm combines a modified GA with improved whale features to create a powerful local and global search capability with faster convergence. The proposed hybrid approach effectively combines the exploratory advantages of the IWOA with improved search capability with the accelerated convergence and local optima avoidance features of the IGA. Their combination can maintain a high level of population diversity to achieve the global optimal application mapping solution. The initial mapping solution is generated using the enhanced initial mapping to give a head start to the optimization algorithm, enhancing its performance in the search space. To effectively integrate the IGA and IWOA, The IWOA incorporates the IGA into each of its K iterations until the ideal solution is found. The algorithm runs for a certain number of iterations and simulations. Each K iteration is predetermined and changes depending on the application. The fitness function holds significant importance within the realm of optimization mapping problems, as it represents the objective we seek to minimize in order to reach the optimal solution. If the best global solution remains unchanged for the last K number of iterations, this suggests that the algorithm might have become stuck at an optimal local value. The IWOA optimizes the mapping solution in certain generations, and the adaptive strategy drives a well balanced search operation. To enhance the rate of convergence and prevent the algorithm from becoming stuck in local optima, additional improved genetic features such as groupwise selection, direction-based crossover, and mutation characteristics are incorporated. Direction-based crossover allows the diverse mapping solutions in each generation to efficiently converge towards the optimum solution. The input parameters for the technique are chosen after a thorough assessment. Certain parameters are allowed to adapt according to the environment and the solutions acquired in each K iteration, allowing the algorithm to achieve better convergence and search features. The IWOA-IGA pseudocode and framework are presented in Algorithm 3 and Figure 7, respectively.

Parameter Settings for the Proposed Hybrid Algorithm

The proposed hybrid method necessitates the specification of a few fundamental parameters to determine the efficacy of group searching. Numerous simulations and satisfying results were used to pick the parameter values for the proposed algorithm in order to produce an effective solution. Well balanced exploration and exploitation ability is derived using adaptive parameter adjustment, which results in updating of the vector $\vec{A}$. The algorithm uses a few other parameters, such as r (a random vector between [0,1]), and a linear decline from 2 to 0 is performed throughout iteration during both the exploration and exploitation phases. For the shrinking encircling mechanism, A takes a random value between $-a$ and a, where a gradually diminishes from 2 to 0 throughout iteration. Configuring A within $[-1, 1]$ can result in a new position for the mapping solution between the original position and the current best position. For the spiral mapping position, the

parameter b is kept at a constant value to maintain the shape of the spiral, while l is varied between $[-1, 1]$ to help the algorithm achieve better mapping solutions.

Algorithm 3 IWOA Integrated with IGA

Input: $enhancedInitMapping, taskGraph, itrWOA, itrIGA, a, C, l, A, p$
stopCriteria: $itrTotal, OptimalCommCost$
// Find pop, v, fitness and gbestpop, gbestfitness
$initPopFitness(enhancedInitMapping, pop_size, meshSize)$
$getinitbest(fitness, pop)$ $//X^* = BestSearchAgent$
while $stopCriteria$ **do**
  for $i \leftarrow 1$ to $itrWOA$ **do**
    for $searchAgentj \leftarrow 1$ to pop_size **do**
      Update parameters A, C, a, l, p
      $\vec{A} = 2\vec{a}.\vec{r} - \vec{a}$, $\vec{C} = 2.\vec{r}$
      if $p < 0.5$ **then**
        if $|A| < 1$ **then**
          Update current solution using Equation (24), finally updating $pop[j]$
        end if
        if $|A| \geq 1$ **then**
          Choose random mapping solution-Xrand
          Update current mapping solution by the Equation (32), updating $pop[j]$
        end if
      end if
      if $p \geq 0.5$ **then**
        Update Current mapping solution by Equation (28), updating $pop[j]$
      end if
    end for
    for $particlej \leftarrow 1$ to pop_size **do**
      $calCommCost(meshSize, pop[j], tashGraph)$
    end for
    $gbestfitness = mim(pop[j])$
    $gbestpop = min(pop)$
  end for
  if $(gbestfitness == optimalCommCost)$ **then**
    $finalmapping \leftarrow gbestpop$
    $optimalCommCost \leftarrow gbestfitness$
  end if
  //Apply IGA to global best solution
  $itrIGA \leftarrow 0$
  //Take Pop as initial population for IGA to Optimize
  //Find gbestpop, gbestfitness
  $getinitbest(fitness, pop)$
  $currentbest \leftarrow gbestpop//X* = best_search_agent$
  $currentbestpop \leftarrow gbestfitness$
  $SetParametersr$
  while itrIGA **do**
    //Execute Selection Process
    X^{map_a} & X^{map_b}
    //Apply Direction Based CrossOver
    Update X_i^* and $\vec{D_{ij}}$ using Equation (35)
    $itrIWOA = itrIWOA + 1$
    $update(gbestfitness, finalMapping$
    Keep Elite Solutions & then perform Mutation forming Offsprings
    if $commCost_{Offsprings} > currentBestCost$ **then**
      $currentBest \leftarrow gbest_offspring$
      $currentbestCost \leftarrow gbestfitness$
    end if
    if $gbestfitness == optimalCommCost$ **then**
      $finalmapping \leftarrow gbestpop$
      $optimalCommCost \leftarrow gbestfitness$
    end if
    $itrIGA = itrIGA + 1$
  end while
end while

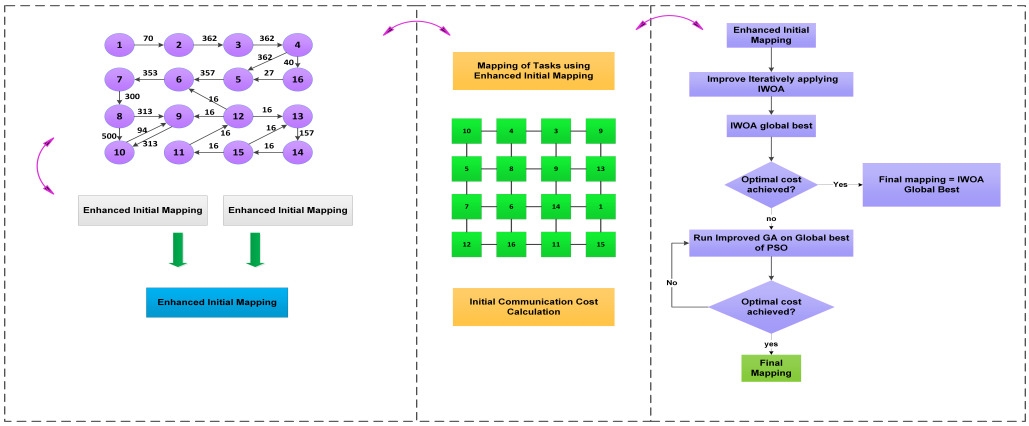

Figure 7. IWOA-IGA framework.

The direction-based vector D allows the algorithm to converge more quickly towards the optimum. It entails a significant probability of producing high quality mapping solutions in the direction of the optimal communication cost. The parameter r_{ij} in the direction-based crossover mechanism is randomly generated. The parameter's population size and generations are set to 150 and 100, respectively. The algorithm first generates an initial mapping, which is iteratively improved for the population size and number of generations, eventually leading to an optimal mapping solution for a set of benchmarks.

6. Simulation Results

The simulation results of the IWOA and IWOA-IGA are examined in this part and compared with those of existing mapping methods.

6.1. Simulation Setup and Scenario

Experiments were conducted on available real-time benchmarks and synthetic task graphs to assess the algorithms. All benchmarks for real-world applications are limited to small-scale tasks, often demanding fewer than 32 cores; consequently, custom benchmarks were produced using task graph tools to evaluate the performance of our method on more challenging issues. TGFF instances can have a problem scale anywhere from 16 to 196, with small-scale (core count < 35), medium-scale (36 < core count < 70), and extensive-scale (core count > 70) problems all taken into account. The TGFF tool takes the heterogeneous communication characteristics of the cores into account and builds task graphs at random depending on this behaviour. The set of real-world benchmark instances we used were video object plan decoder (VOPD), 263decoder (263DEC), MPEG-4 decoder, 263encoder (263ENC), Mp3encoder (Mp3ENC), and MWD. All real-time benchmark applications we examined used the standard network size of 4 × 4. This network size is the same as in prior state-of-the-art architectures, allowing for a fair comparison. The VOPD application was divided into 16 subtasks, each of which could be allocated to a 4 × 4 mesh. Larger mesh sizes were used for the synthetic task graphs. Details of the real-time benchmark applications are listed in Table 1.

Modifications were implemented in the NoCTweak simulator to conduct a comparative analysis of different application mapping techniques [38]. ENoCTweak was used by applying the algorithm on various real-time and synthetic task graphs. NoCTweak is a SystemC-based open-source NoC simulator that offers a variety of performance characteristics, including energy, latency, communication cost, and throughput. In order to apply the proposed algorithm on both the synthetic and real-time applications, it was run on a computer system equipped with an Intel Core i3 platform, 8 GB of RAM, and a clock frequency of 1.6 Ghz. The simulation environment used to execute the benchmark applications on a 2D NoC architecture is detailed in Table 2. The proposed application mapping approach was implemented in Python to provide the optimal task mapping for the NoC. The resulting optimized mapping was executed on the NoC using the unified

simulation framework known as ENoCTweak, which was used to simulate and analyze critical NoC system characteristics such as latency, throughput, energy, and power.

Table 1. Standard NoC benchmarks with 2D mesh sizes.

Benchmarks	Nodes	Edges	Mesh Size
PIP	8	8	3 × 3
MPEG-4	12	26	4 × 4
MWD	12	13	4 × 4
263encMP3dec	12	12	4 × 4
263decMP3dec	14	15	4 × 4
Mp3EncMp3Dec	13	14	4 × 4
VOPD	16	21	4 × 4
CAVLC	16	22	4 × 4
MMS	25	38	5 × 5

Table 2. Simulation setup details.

Configuration	Detail
Network Type	Mesh
Type of Platform	Embedded
Applications	VOPD, , MP3encMp3dec, MPEG4, MWD, 263encMp3dec, 263decMp3dec
Mapping Algorithm	IWOA, IWO-IGA, SA, PSO
PacketDeliveryMode	Without ACK
SendingACKPolicy	Send ACK Optimally
Packet Distribution	Exponential
Fixed Packet Length	10 (flits) moment
FlitinInjectionRate	0.1 (flits/cycle/node)
Type of Router	Wormhole-Pipeline
Routing Algorithm	XY DIMENSION-ORDER
OutputChannelSelection	XY-ORDER
BufferSize	1 (flit)
Pipeline Type	8
StagesOfPipeline	4
InputVoltage	1 (V)
Operating Clock Frequency	100,000 (MHz)
Warm-Uptime	20,000 cycles

6.2. Performance-Based Comparative Analysis

As the main aim of this study was to enhance the communication cost, which represents the overall cost associated with running a particular application on NoC (depicted by Equation (1)), the communication cost for both the IWOA and IWOA-IGA were first evaluated based on real-world and TGFF-based graphs.

6.2.1. Performance Analysis Based on Standard Benchmark Instances

Both of the developed techniques were tested using real-world applications as well as TGFF-based graphs (presented in the next section). A comparison of the IWOA and IWOA-IGA with other advanced heuristic algorithms and with the exact mathematical solution was carried out. Table 3 depicts the performance comparison in terms of communication cost for various standard benchmark applications. When it comes to estimation of communication cost, ILP (Integer Linear Programming) is regarded as the optimal solution [3]. The results demonstrate that the proposed modified IWOA and its genetically integrated version obtained optimal results derived from real-world instances.

Table 3. Communication cost (bw × nh) comparison with real-world benchmark applications.

Algorithm	VOPD	CALVC	MMS	MPEG4	MWD	Mp3Enc	263enc	263dec	PIP
ILP	4119	-	-	3567	1120	17.021	230.407	19.823	-
ONMAP	4119	-	663,379	3567	-	-	-	-	640
GA	4141	-	-	3567	1321	17.133	230.69	19.911	-
SA	4125	-	-	3567	1451	-	-	-	-
BEMAP	4119	6701	664,636	3567	-	-	-	-	640
ACO	-	-	-	3670	-	17.231	-	-	-
PSO	4119	-	-	3567	1120	17.021	230.45	19.823	-
BA	4119	-	-	3567	1120	17.834	231.45	19.936	-
HDPSO	4119	-	-	3567	-	-	-	-	-
CSO	4119	6721	652,637	3567	-	-	-	-	640
SCSO	4119	-	-	3567	1122	17.021	230.407	19.823	-
DPSO	4119	-	688,297	3567	1120	17.021	-	19.823	640
RAMAN	4135	-	-	3774	1184	17.87	234.4	19.87	640
SFOA	4119	-	-	3567	1120	17.021	230.407	19.823	-
ACA	4119	6721	652,637	3567	1120	17.021	-	-	-
iHPSA	4119	-	-	3567	1120	17.021	230.407	19.823	-
IWOA	4119	6701	663,379	3567	1122	17.021	230.69	19.823	-
IWOA-IGA	4119	6701	663,379	3567	1122	17.021	230.69	19.82	-

6.2.2. Performance Analysis on TGFF Random Instances

This section presents the details of the generated TGFF random instances and their use in evaluating the algorithms with varying mesh sizes. For the sake of experimentation with large-scale applications, random synthetic task graphs were generated using the TGFF tool, which generates task graphs with 32, 64, and 128 cores. TGFF, known as Task Graphs For Free, is meticulously crafted to offer a standardized approach for generating arbitrary task graphs for experimentation purposes in research work. In our evaluation, we used mesh sizes of 6×6 for up to 32 core placements, 8×8 to accommodate 64 core graphs, and 12×12 for 128 core task graphs. The task graphs were configured to use a communication bandwidth of 50–600 MB/s between the interconnected tasks. As the tasks using the TGFF tools are randomly generated, there is no specific optimal value for the communication cost; however, there must be some criteria to evaluate the performance of the generated large-scale tasks. A few important considerations could include the number of generations for which the algorithms can execute and the communication cost value be achieved at that instance. Another consideration is the number of iterations for which the algorithm runs and the targeted communication cost to be achieved after a specified number of iterations. We observed that when the number of iterations was configured to a value between 180–200 and the IWAO-IGA was executed for the 200 iterations, the IWAO-IGA achieved the desired communication cost in fewer iterations than the IWOA for larger applications.

This is due to the integration of directional crossover and mutation features into the IWAO mechanism, allowing the algorithm to perform better when searching for optimal solutions in the search space and to attain the desired cost in a lesser number of iterations than to the simple IWOA. Moreover, the proposed algorithm attains the desired cost within a reduced number of iterations in comparison with other advanced and optimization algorithms, e.g., SA, GA, and PSO. A cost savings-based comparative analysis is presented in Tables 4 and 5 and in Figure 8. Across the tested benchmarks, the proposed IWOA reliably lowers the communication cost by an average of 41.31%, 46.61%, and 44.00% compared to PSO, SA, and GA, respectively, for 32 cores. In the case with 64 cores, the average improvement in cost is 38.70%, 31.94%, and 39.78%, respectively. The proposed algorithm consistently achieves a significant average reduction in communication cost when applied to scenarios involving 128 cores, with respective improvements of 17.37%, 20.99%, and 16.12% over PSO, SA, and GA. These outcomes highlight that the proposed algorithm outperforms applications with a moderate number of cores. However, as the nubmer of cores rises to surpass a certain threshold, the algorithm's performance becomes average compared to its efficiency with a lower number of cores. As the algorithm was further modified by adding an improved GA algorithm to ensure better results when larger appli-

cations are executed on it, the modified IWOA-IGA was tested again on larger TGFF-based applications implemented on 32, 64, and 128 cores. The IWOA-IGA minimizes communication costs effectively, with an average improvement of 45.66%, 50.56%, and 48.15%, respectively, for 32 cores. For the configuration with 64 cores, the average improvement in communication cost was 40.29%, 33.70%, and 41.33%, respectively. With 128 cores, the proposed technique produced respective average reductions in communication cost of 30.60%, 33.31%, and 26.67%. Table 5 shows the results.

Table 4. Percent communication cost savings with IWOA compared to other algorithms for extensive task graphs.

Task Graph	Over PSO	Over SA	Over GA
32 Cores	41.31%	46.661%	44.00%
64 Cores	38.70%	31.94%	39.78%
128 Cores	17.37%	20.99%	16.12%

Table 5. Percent communication cost savings with IWOA-IGA compared to other algorithms for extensive task graphs.

Task Graph	Over PSO	Over SA	Over GA
32 Cores	45.66%	50.56%	48.15%
64 Cores	40.29%	33.70%	41.33%
128 Cores	30.60%	33.31%	26.67%

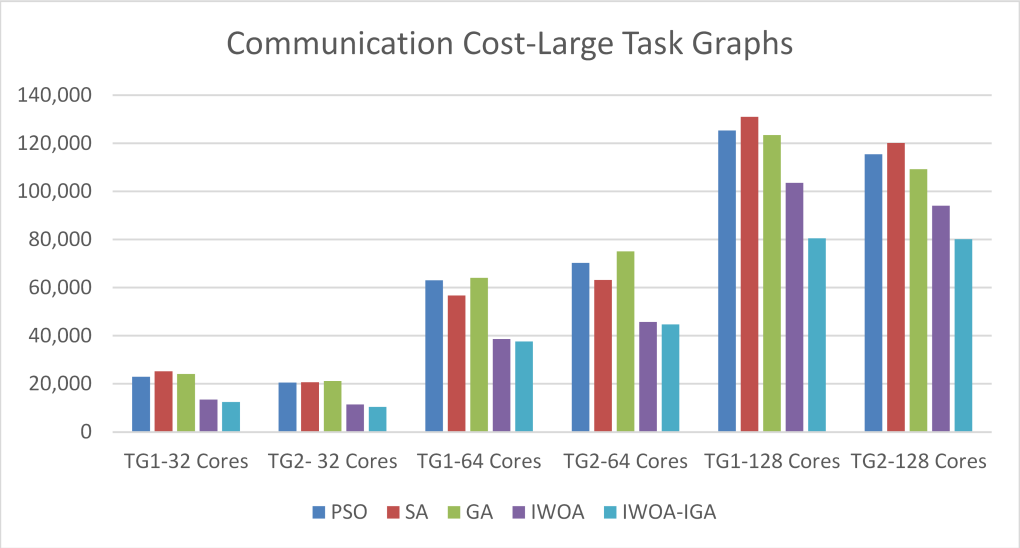

Figure 8. Comparison of communication costs for large synthetic task graphs.

6.2.3. Performance Comparison of IWOA and IWOA-IGA: Optimizing Power and Energy in 2D NoC Benchmark Applications.

To assess the proposed algorithm's power efficiency, we opted for the Orion Model [40,46], which analyzes the power and energy of the network and can be integrated into a simulation environment to compute the network's overall energy. The NoCtweak simulator was used for power estimation, which was conducted using a standard cell library and the ORION-2 model, allowing for accurate evaluation of power consumption at different CMOS nodes. This simulation tool derives both power and energy consumption. By utilizing post-layout power details sourced from 2D NoC components, these estimations are based on component behavior when executing traffic patterns or real-time programs with a given mapping. Regarding the power metrics, the proposed IWOA improves power reduction by an average of 21.93%, 30.20%, 22.07%, 26.92%, 15.84%, 8.70%, 7.78%, 1.9%, 7.66%, and 5.48% over the PSO, SA, GA, ACO, BA, SCSO, ILP, iHPSO, CSO, and SFO algorithms, respectively.

We observed that IWOA integrated with GA further improves the power reduction by an average of 22.09%, 30.34%, 22.23%, 27.07%, 16.24%, 8.89%, 7.97%, 2.07%, 7.84%, and 5.63% over the PSO, SA, GA, ACO, BA, SCSO, ILP, iHPSO, CSO, and SFO algorithms, respectively. It is notable that there was an average power difference of 3–4% when the simulation was carried out using the IWOA-IGA instead of the IWOA with real-time benchmarks. This is because both algorithms have different features and abilities to achieve optimal mapping solutions. Overall, these results demonstrate that the proposed algorithms consume less power than existing bio-inspired algorithms. The comparison of projected power values through the proposed methods is visually depicted in Figure 9.

Figure 10 shows the energy consumption for the various methods. The mapping technique implemented in the proposed work outshines competing state-of-the-art techniques such as PSO, SA, GA, and iHPSA in terms of energy consumption. In comparison to PSO, SA, GA, and iHPSA, IWOA reduces energy consumption by 1.7%, 13.89%, 17.47%, and 0.2%. The improved version yields improvements of over 3.22% compared to PSO, 15.38% compared to SA, 27.58% compared to GA, and 1.64% compared to iHPSA.

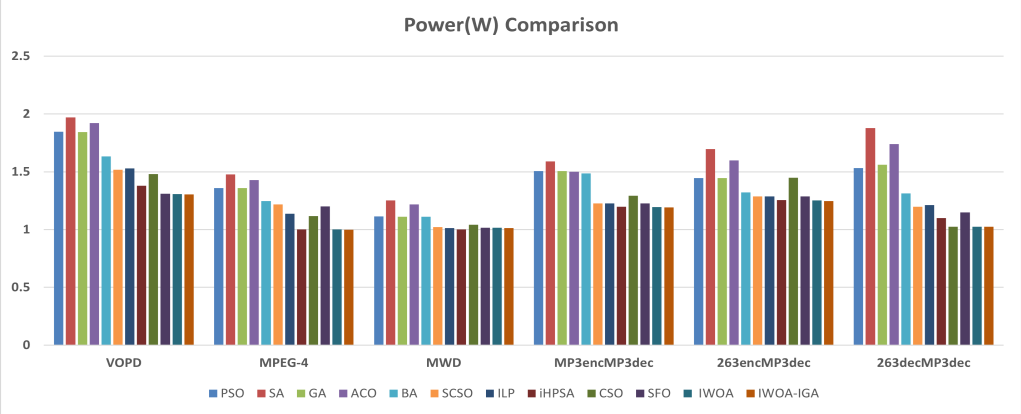

Figure 9. Power estimation for standard benchmarks.

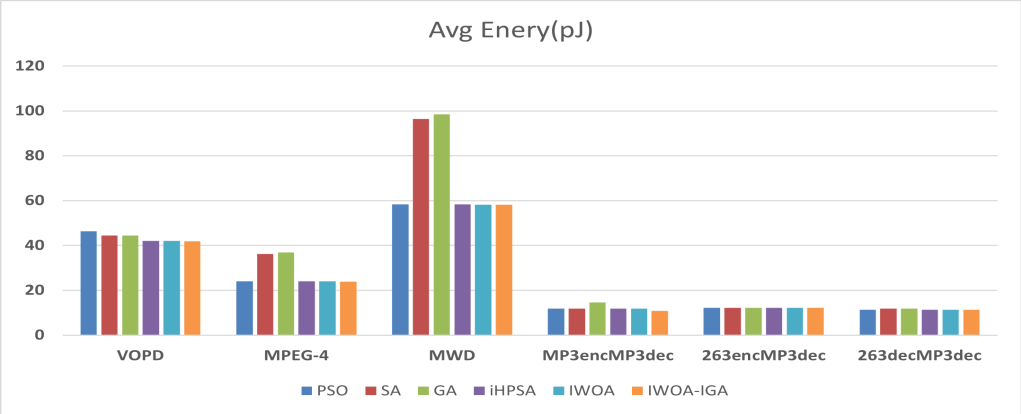

Figure 10. Energy estimation for standard benchmarks (normalized).

6.2.4. Performance Comparison of IWOA and IWOA-IGA: Optimizing Latency for 2D NoC Benchmark Applications

Network communication latency is a crucial factor to consider when evaluating mapping solutions. This subsection discusses the average latency derived with various approaches. Utilizing the NoC simulator [39], the proposed work was evaluated on a 4×4 mesh topology with an X–Y routing algorithm and wormhole-based routers. The evaluation showed that the IWO algorithm yielded latency improvements of 1.91% over PSO, 5.14% over SA, 5.7% over GA, and 1% over iHPSA. The extended IWOA-IGA version showed improvements of 2.87% over PSO, 6.08% over SA, 6.61% over GA, and 2% over iHPSA.

The comparison is illustrated in the Figure 11.

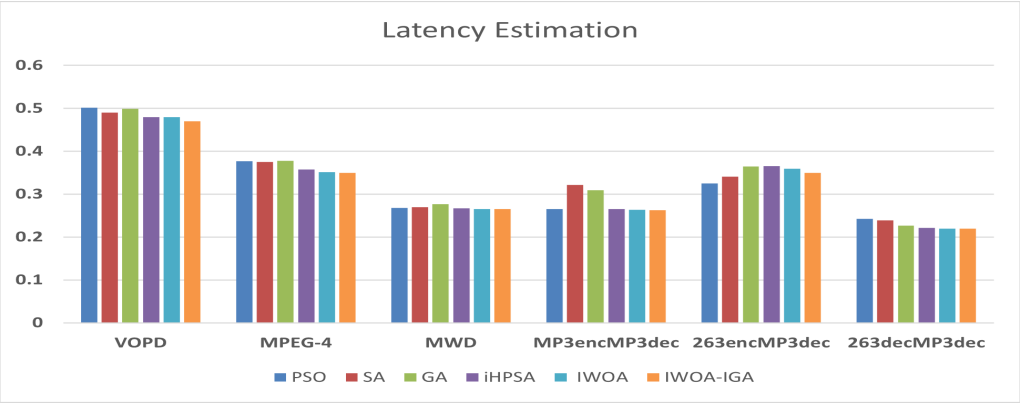

Figure 11. Average latency for standard benchmarks.

6.2.5. Convergence Evaluation Based on Convergence Factor

Convergence of any algorithm is one of the most important factors of an application mapping technique intended to acquire optimal mapping in a lesser number of iterations. Different algorithms running different applications may utilize different numbers of iterations and communication costs for the calculation of optimal mapping. Mathematically exact methods are able to solve small-scale problems (less than 20 cores) in around 7–8 h. While heuristic techniques are much faster, they may not offer optimal communication cost. They normally achieve near-optimal cost after the algorithm has run for a specific number of iterations. An algorithm that requires fewer iterations to achieve the lowest communication cost is considered to converge well, and is rated as highly suitable for the application mapping problem domain. Thus, one of the performance metrics considered in this paper is the convergence factor, which evaluates the algorithm's ability to converge effectively. Performance based on algorithm convergence ability is measured by calculating the average number of times the algorithm successfully converges, taking into account the number of hits achieved by the algorithm out of the total number of runs. To evaluate performance based on the convergence factor, the algorithms were tested 30 times while measuring the convergence factor for a range of real-time and synthetic task graphs of varying sizes. The proposed method was compared with PSO, SA, GA, and iHPSA based on the number of times the algorithms achieved the lowest communication cost over a certain number of runs. The equation for calculating the convergence factor is provided by

$$Perf_{cf} = Nav_c / NR_{total}, \tag{36}$$

where $Perf_{cf}$ is the performance based on the convergence factor, Nav_c is the average number of times the algorithm converged, and NR_{total} is the total number of times the algorithm was run. The comparison results for the real-world and synthetic applications are shown in Table 6, which shows that the algorithm converged several times when being run 30 times. The table shows Nav_c, the average number of times the algorithm converged for the specific application, which is calculated here based on half of the algorithm's executions R1–R15 (average halved) instead of 30, which is NR_{total}, the total number of times the algorithm was run for a specific application.

Table 6. Performance evaluation based on convergence.

Applications	Nav_c	NR_{total}	$Perf_{cf}$
PSO	10.71	15	7.14
SA	10.28	15	6.85
GA	11.14	15	7.42
IWOA	11.42	15	7.61
IWOA-IGA	12.42	15	8.2

The average value of convergence Nav_C calculated for PSO, SA, GA, IWOA, and IWOA-IGA against real and synthetic benchmarks was found to be 10.71, 10.28, 11.14, 11.42, and 12.42, respectively. As Nav_C is directly proportional to $Perf_{CF}$, a higher Nav_C value for an algorithm indicates better performance in terms of average convergence for a specified number of runs. The calculated $Perf_{CF}$ values for PSO, SA, GA, IWOA, and IWOA-IGA are 0.71, 0.68, 0.74, 0.76, and 0.82 respectively, which shows that the proposed algorithms converged several more times compared to the others when applied to the real and synthetic task graphs.

Figure 12 shows the graphical representation of the convergence factor results as derived using Nav_C. The graph shows the better values of Nav_C and performance based on the convergence factor achieved by the proposed algorithms in comparison to other competitive algorithms.

A statistical analysis of the results reveals that the IWOA-IGA hybrid algorithm significantly reduces communication costs across different core configurations. For 32 cores, the hybrid algorithm achieves an average reduction of 48.12% compared to the other optimization algorithms. Similarly, for 64 cores the reduction is 38.11%, whie for 128 cores it is 30.53%. Regarding energy efficiency, the algorithm outperforms PSO by 3.22%, SA by 15.38%, GA by 27.58%, and iHPSA by 1.64%. Additionally, the IWOA-IGA exhibits superior convergence, with a $Perf_{CF}$ value of 0.82, indicating its effectiveness on both real and synthetic task graphs. The IWOA-IGA shows improved latency as well, achieving improvements of 2.87% over PSO, 6.08% over SA, 6.61% over GA, and 2% over iHPSA. In conclusion, the improved whale optimization algorithm integrated with specialized modified genetic algorithm properties presents superior performance in solving mapping problems within Network-on-Chip (NoC) architectures. Its effectiveness surpasses that of other state-of-the-art algorithms, establishing it as a viable and advantageous solution for addressing complex NoC challenges.

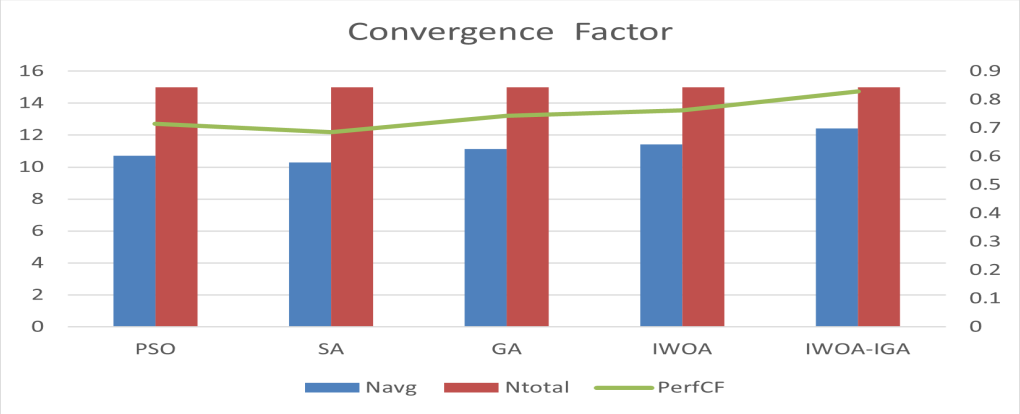

Figure 12. Performance based on convergence factor.

7. Conclusions

In this paper, we have presented a hybrid model incorporating an improved whale optimization algorithm with a modified enhanced version of genetic algorithm for the allocation of real-world applications onto 2D NoCs. The proposed algorithm incorporates enhanced initial mapping instead of random initial mapping to provide a head start to the optimization algorithm in achieving the global optimum solution. In the first step, the IWOA algorithm is introduced to address the application mapping challenge. To further improve the efficiency of the proposed algorithm, an enhanced and modified genetic algorithm which uses expert-based selection, direction-based cross-over, and mutation abilities is integrated with the IWOA to produce high-quality mapping solutions. This tweaked GA features helps the IWOA to achieve optimal mapping with faster convergence, allowing it to avoid local optima through its enhanced search capability. Extensive experimentation

and analysis were performed with both real-time benchmarks and synthetic large-scale task graphs. The proposed IWOA-IGA shows significant improvements regarding communication cost, average power, energy, and latency over other competitive algorithms, demonstrating its high potential. In future work, the proposed algorithm can be employed to map real-time applications onto alternative NoC topologies.

Author Contributions: Conceptualization, S.S., F.H. and N.K.B.; methodology, S.S.; implementation, S.S.; validation, S.H., F.H. and N.K.B.; formal analysis, S.S.; investigation, S.S., F.H. and N.K.B.; data curation, S.S.; writing—original draft preparation, S.S.; writing—review and editing, F.H. and N.K.B.; supervision, F.H. and N.K.B.; project administration, N.K.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work is not funded by any organization.

Data Availability Statement: This research used publicly available data for experimentation and analysis purposes.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Owens, J.D.; Dally, W.J.; Ho, R.; Jayasimha, D.; Keckler, S.W.; Peh, L.S. Research challenges for on-chip interconnection networks. *IEEE Micro* **2007**, *27*, 96–108.
2. Kumar, S.; Jantsch, A.; Soininen, J.P.; Forsell, M.; Millberg, M.; Oberg, J.; Tiensyrja, K.; Hemani, A. A network on chip architecture and design methodology. In Proceedings of the IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design, ISVLSI 2002, Pittsburgh, PA, USA, 25–26 April 2002; pp. 117–124.
3. Tosun, S.; Ozturk, O.; Ozen, M. An ILP formulation for application mapping onto network-on-chips. In Proceedings of the 2009 International Conference on Application of Information and Communication Technologies, Baku, Azerbaijan, 14–16 October 2009; pp. 1–5.
4. Ingle, V.V.; Gaikwad, M.A. Review of mesh topology of NoC architecture using source routing algorithms. *Int. J. Comput. Appl.* **2013**, *975*, 8887.
5. Han, J.J.; Lin, M.; Zhu, D.; Yang, L.T. Contention-aware energy management scheme for NoC-based multicore real-time systems. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *26*, 691–701.
6. Sahu, P.K.; Chattopadhyay, S. A survey on application mapping strategies for network-on-chip design. *J. Syst. Archit.* **2013**, *59*, 60–76.
7. Pop, R.; Kumar, S. *A Survey of Techniques for Mapping and Scheduling Applications to Network on Chip Systems*; Research Report; School of Engineering, Jonkoping University: Jönköping, Sweden, 2004; Volume 4.
8. Sharma, P.K.; Biswas, S.; Mitra, P. Energy efficient heuristic application mapping for 2-D mesh-based network-on-chip. *Microprocess. Microsystems* **2019**, *64*, 88–100.
9. Ogras, U.Y.; Hu, J.; Marculescu, R. Key research problems in NoC design: A holistic perspective. In Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Jersey City, NJ, USA, 19–21 September 2005; pp. 69–74.
10. Tosun, S.; Ozturk, O.; Ozkan, E.; Ozen, M. Application mapping algorithms for mesh-based network-on-chip architectures. *J. Supercomput.* **2015**, *71*, 995–1017.
11. Wang, X.; Liu, H.; Yu, Z.; Shen, K. A novel two-phase heuristic for application mapping onto mesh-based Network-on-Chip. *IEICE Electron. Express* **2016**, *13*, 20151097.
12. Brezočnik, L.; Fister Jr, I.; Podgorelec, V. Swarm intelligence algorithms for feature selection: A review. *Appl. Sci.* **2018**, *8*, 1521.
13. Amin, W.; Hussain, F.; Anjum, S.; Khan, S.; Baloch, N.K.; Nain, Z.; Kim, S.W. Performance evaluation of application mapping approaches for network-on-chip designs. *IEEE Access* **2020**, *8*, 63607–63631.
14. Sahu, P.K.; Shah, T.; Manna, K.; Chattopadhyay, S. Application mapping onto mesh-based network-on-chip using discrete particle swarm optimization. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2013**, *22*, 300–312.
15. Tosun, S. Cluster-based application mapping method for network-on-chip. *Adv. Eng. Softw.* **2011**, *42*, 868–874.
16. Khajekarimi, E.; Hashemi, M.R. Energy-aware ILP formulation for application mapping on NoC based MPSoCs. In Proceedings of the 2013 21st Iranian Conference on Electrical Engineering (ICEE), Mashhad, Iran, 14–16 May 2013; pp. 1–5.
17. Khan, S.; Anjum, S.; Gulzari, U.A.; Afzal, M.K.; Umer, T.; Ishmanov, F. An efficient algorithm for mapping real time embedded applications on NoC architecture. *IEEE Access* **2018**, *6*, 16324–16335.
18. Liu, L.; Wu, C.; Deng, C.; Yin, S.; Wu, Q.; Han, J.; Wei, S. A flexible energy-and reliability-aware application mapping for NoC-based reconfigurable architectures. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2014**, *23*, 2566–2580.

19. Murali, S.; De Micheli, G. Bandwidth-constrained mapping of cores onto NoC architectures. In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, Paris, France, 16–20 February 2004; Volume 2, pp. 896–901.
20. Shen, W.T.; Chao, C.H.; Lien, Y.K.; Wu, A.Y. A new binomial mapping and optimization algorithm for reduced-complexity mesh-based on-chip network. In Proceedings of the First International Symposium on Networks-on-Chip (NOCS'07), Princeton, NJ, USA, 7–9 May 2007; pp. 317–322.
21. Tosun, S. New heuristic algorithms for energy aware application mapping and routing on mesh-based NoCs. *J. Syst. Archit.* **2011**, *57*, 69–78.
22. Cheng, C.H.; Chen, W.M. Application mapping onto mesh-based network-on-chip using constructive heuristic algorithms. *J. Supercomput.* **2016**, *72*, 4365–4378.
23. Wang, X.; Choi, T.M.; Yue, X.; Zhang, M.; Du, W. An effective optimization algorithm for application mapping in network-on-chip designs. *IEEE Trans. Ind. Electron.* **2019**, *67*, 5798–5809.
24. Upadhyay, M.; Shah, M.; Bhanu, P.V.; Soumya, J.; Cenkeramaddi, L.R. Multi-application based network-on-chip design for mesh-of-tree topology using global mapping and reconfigurable architecture. In Proceedings of the 2019 32nd international conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID), Delhi, India, 5–9 January 2019; pp. 527–528.
25. Yan, R.; Zhou, Y.; Cai, A.; Li, C.; Yan, Y.; Yin, M. Contention-aware mapping and scheduling optimization for NoC-based MPSoCs. In Proceedings of the International Conference on Automated Planning and Scheduling, Nancy, France, 26–30 October 2020; Volume 30, pp. 305–313.
26. Mohiz, M.J.; Baloch, N.K.; Hussain, F.; Saleem, S.; Zikria, Y.B.; Yu, H. Application mapping using cuckoo search optimization with Lévy flight for NoC-based system. *IEEE Access* **2021**, *9*, 141778–141789.
27. Amin, W.; Hussain, F.; Anjum, S. iHPSA: An improved bio-inspired hybrid optimization algorithm for task mapping in Network on Chip. *Microprocess. Microsystems* **2022**, *90*, 104493.
28. Choudhary, J.; Soumya, J.; Cenkeramaddi, L.R. Raman: Reinforcement learning inspired algorithm for mapping applications onto mesh network-on-chip. In Proceedings of the 2021 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP), Munich, Germany, 4 November 2021; pp. 52–58.
29. Reza, M.F.; McCloud, Z. Heuristics-enabled high-performance application mapping in network-on-chip based multicore systems. In Proceedings of the 2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS), Berlin, Germany, 23–25 July 2023; pp. 1–6.
30. Bose, A.; Ghosal, P. The CTH Network: An NoC Platform for Scalable and Energy Efficient Application Mapping Solution. *IEEE Trans. Nanotechnol.* **2023**, *22*, 58–69.
31. Amin, W.; Hussain, F.; Anjum, S.; Saleem, S.; Ahmad, W.; Hussain, M. HyDra: Hybrid Task Mapping Application Framework for NOC-based MPSoCs. *IEEE Access* **2023**, *11*, 52309–52326.
32. Amin, W.; Hussain, F.; Anjum, S.; Saleem, S.; Baloch, N.K.; Zikria, Y.B.; Yu, H. Efficient application mapping approach based on grey wolf optimization for network on chip. *J. Netw. Comput. Appl.* **2023**, *219*, 103729.
33. Saleem, S.; Hussain, F.; Amin, W.; Ahmed, R.; Zikria, Y.B.; Ishmanov, F.; Yu, H. A Survey on Dynamic Application Mapping Approaches for Real-Time Network-on-Chip-Based Platforms. *IEEE Access* **2023**, *11*, 122694–122721.
34. Ramesh, S.; Manna, K.; Gogineni, V.C.; Chattopadhyay, S.; Mahapatra, S. Congestion-Aware Vertical Link Placement and Application Mapping Onto Three-Dimensional Network-On-Chip Architectures. *IEEE Trans.-Comput.-Aided Des. Integr. Circuits Syst.* **2024**, 1. https://doi.org/10.1109/TCAD.2024.3371255.
35. Tran, A.T.; Baas, B. *NoCTweak: A Highly Parameterizable Simulator for Early Exploration of Performance and Energy of Networks On-Chip*; Technical Report ECE-VCL-2012-2; VLSI Computation Lab, ECE Department, University of California: Davis, CA, USA, 2012.
36. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67.
37. Watkins, W.A.; Schevill, W.E. Aerial observation of feeding behavior in four baleen whales: Eubalaena glacialis, Balaenoptera borealis, Megaptera novaeangliae, and Balaenoptera physalus. *J. Mammal.* **1979**, *60*, 155–163.
38. Goldbogen, J.A.; Friedlaender, A.S.; Calambokidis, J.; Mckenna, M.F.; Simon, M.; Nowacek, D.P. Integrative approaches to the study of baleen whale diving behavior, feeding performance, and foraging ecology. *BioScience* **2013**, *63*, 90–100.
39. Chen, Q.; Huang, W.; Peng, Y.; Huang, Y. A reinforcement learning-based framework for solving the IP mapping problem. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2021**, *29*, 1638–1651.
40. Haupt, R.L.; Haupt, S.E. *Practical Genetic Algorithms*; John Wiley & Sons: Hoboken, NJ, USA, 2004.
41. Ning, G.Y.; Cao, D.Q. Improved whale optimization algorithm for solving constrained optimization problems. *Discret. Dyn. Nat. Soc.* **2021**, *2021*, 8832251.
42. Tei, Y.Z.; Marsono, M.N.; Shaikh-Husin, N.; Hau, Y.W. Network partitioning and GA heuristic crossover for NoC application mapping. In Proceedings of the 2013 IEEE International Symposium on Circuits and Systems (ISCAS), Beijing, China, 19–23 May 2013; pp. 1228–1231.
43. Song, Y.; Wang, F.; Chen, X. An improved genetic algorithm for numerical function optimization. *Appl. Intell.* **2019**, *49*, 1880–1902.
44. Ismkhan, H. Black box optimization using evolutionary algorithm with novel selection and replacement strategies based on similarity between solutions. *Appl. Soft Comput.* **2018**, *64*, 260–271.

45. Elhoseny, M.; Tharwat, A.; Hassanien, A.E. Bezier curve based path planning in a dynamic field using modified genetic algorithm. *J. Comput. Sci.* **2018**, *25*, 339–350.
46. Wang, H.S.; Zhu, X.; Peh, L.S.; Malik, S. Orion: A power-performance simulator for interconnection networks. In Proceedings of the 35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002. (MICRO-35), Istanbul, Turkey, 18–22 November 2002; pp. 294–305.