MARCO: A High-performance Task Mapping and Routing Co-optimization Framework for Point-to-Point NoC-based Heterogeneous Computing Systems

HUI CHEN, Nanyang Technological University, Singapore
ZIHAO ZHANG, Nanyang Technological University, Singapore
PENG CHEN, Nanyang Technological University and National University of Singapore, Singapore
XIANGZHONG LUO, Nanyang Technological University, Singapore
SHIQING LI, Nanyang Technological University, Singapore
WEICHEN LIU, Nanyang Technological University, Singapore

Heterogeneous computing systems (HCSs) , which consist of various processing elements (PEs) that vary in their processing ability, are usually facilitated by the network-on-chip (NoC) to interconnect its components. The emerging point-to-point NoCs which support single-cycle-multi-hop transmission, reduce or eliminate the latency dependence on distance, addressing the scalability concern raised by high latency for long-distance transmission and enlarging the design space of the routing algorithm to search the non-shortest paths. For such point-to-point NoC-based HCSs, resource management strategies which are managed by compilers, scheduler, or controllers, e.g., mapping and routing, are complicated for the following reasons: (i) Due to the heterogeneity, mapping and routing need to optimize computation and communication concurrently (for homogeneous computing systems, only communication). (ii) Conducting mapping and routing consecutively cannot minimize the schedule length in most cases since the PEs with high processing ability may locate in the crowded area and suffer from high resource contention overhead. (iii) Since changing the mapping selection of one task will reconstruct the whole routing design space, the exploration of mapping and routing design space is challenging. Therefore, in this work, we propose MARCO, the mapping and routing co-optimization framework, to decrease the schedule length of applications on point-to-point NoC-based HCSs. Specifically, we revise the tabu search to explore the design space and evaluate the quality of mapping and routing. The advanced reinforcement learning (RL)algorithm, i.e., advantage actor-critic, is adopted to efficiently compute paths. We perform extensive experiments on various real applications, which demonstrates that the MARCO achieves a remarkable performance improvement in terms of schedule length (+44.94% ∼ +50.18%) when compared with the state-of-the-art mapping and routing co-optimization algorithm for homogeneous computing systems. We also compare MARCO with different combinations of state-of-the-art mapping and routing approaches.

CCS Concepts: • **Hardware → Network on chip**.

Additional Key Words and Phrases: Mapping, Routing, NoC, Heterogeneous Computing Systems

1 INTRODUCTION

Nowadays, more than hundreds of processing elements (PEs) can be integrated into one system. The processing rate among different PEs varies since various application-specific integrated circuits (ASICs) are developed to assist or accelerate specific tasks. Therefore, the task mapping among PEs is of great significance due to the heterogeneous property of such heterogeneous computing systems (HCSs). Also, as the number of processing elements (PEs) integrated into one system increases, data transmission among PEs becomes the bottleneck for performance improvement, letting network-on-chip (NoC) become a promising solution for interconnecting hundreds or thousands of PEs. Moreover, the emerging point-to-point NoCs which support single-cycle-multi-hop transmission, reduce or eliminate the latency dependence on distance, addressing the scalability concern raised by high latency for long-distance transmission and enlarging the design space of the routing algorithm to search the non-shortest paths [7] without resource contentions. Thus, the schedule length of one application runs on point-to-point NoC-based HCSs, i.e., the total time along its critical path,

depends on resources management strategies, e.g., mapping and routing, which are managed by compilers, scheduler, or controllers.

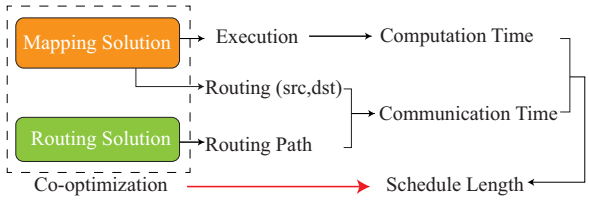

Fig. 1. Relationship between task mapping and routing for point-to-point NoC-based HCSs.

Mapping and routing are inter-dependent, as shown in Figure 1, so that the co-optimization of mapping and routing is commonly proposed. In homogeneous computing systems, since PEs are with the same processing ability, the mapping would rarely influence the task execution time, targeting the communication time minimization only. However, HCSs, which consist of various PEs that vary in their processing ability, involve the additional design objective for mapping algorithm, i.e., computation time minimization. Thus, the co-optimization algorithm [35] proposed for point-to-point NoC-based homogeneous computing systems cannot be directly applied for point-to-point NoC-based HCSs. To illustrate this, in Figure 2, we generate some random task mapping and routing solutions for application H.264, which is a widely-used video compression standard, on an emerging point-to-point NoC-based HCS with the mesh size of 8×8 and heterogeneity coefficient $V_{HCS} = 0.25$ (we will explain it in Section 3.1). We illustrate the result of the co-optimization algorithm for homogeneous computing systems [35] using the purple pentagram. Compared with the best solution among the random mapping and routing denoted by the red pentagram, the method proposed in [35] decreases the transmission latency but ignores the execution time optimization, resulting in longer schedule length.

Moreover, conducting mapping and routing consecutively may not achieve minimum schedule length since the PEs with high processing ability may locate in the crowded area and suffer from high resource contention overhead. In Figure 2, we test combination of two mapping results, i.e., communication-aware and computation-aware, and the state-of-the-art (SOTA) contention minimization routing algorithm [7] for emerging point-to-point NoCs. Specifically, the **communication-aware mapping** is the mapping result of [35] and the **computation-aware mapping** is the mapping with the least task execution time (we will explain this in Section 5). Since the routing used in the co-optimization algorithm [35] is XY routing, using the SOTA routing algorithm further reduces the transmission time. As shown in this example, the best solution is neither the one with the minimum execution time nor the one with the minimum transmission time. Although the previous motivation example adapts point-to-point NoCs as its communication backbone, the computation and communication trade-off through mapping and routing exists in other NoC-based HCSs. Inspired by the dilemma that execution time minimization and transmission time reduction cannot be achieved at the same time, searching the optimal solution on emerging NoC-based HCSs, i.e., the minimum schedule length of applications, needs to explore the entire design space of all possible mapping and routing choices. Note that, due to the complex dependency of task mapping and routing for NoC-based HCSs, co-optimization is also meaningful for other interesting objectives, e.g., energy consumption reduction.

However, the exploration of mapping and routing design space on NoC-based HCSs is challenging. Since once we change the mapping of a single task, the entire routing solution should be modified due to the following reasons: (i) At least the source or destination of one message is changed,

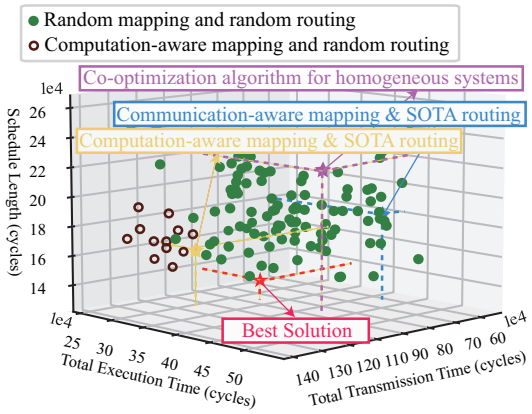

Fig. 2. Illustration of design space exploration for task mapping and routing.

influencing transmission of other messages. (ii) The execution time of that task is changed due to the different capacities of PEs, resulting in that the link usage state changes correspondingly. Also, the parallelism that exists in both task execution and data transmission lets the computation timeline and communication timeline overlap sometimes. Such parallelism makes the relationship of schedule length, transmission time, and execution time complex, thereby complicating the co-optimization of task mapping and routing.

In conclusion, the motivation for mapping and routing co-optimization is observed from the fact that: (i) The mapping should consider task execution time (since the PEs' processing rate varies) and data communication time (since it decides the source and destination of the transmission) concurrently. (ii) The data communication time is unknown for task mapping due to the different performance of various routing candidates. (iii) For one specific mapping, the least data transmission time can be achieved is decided since the source and destination may locate in a crowded region and no NoC resource is available to transmit data. Thus, the co-optimization of task mapping and routing would benefit users and engineers who deploy or design applications on emerging NoC-based HCSs. In this article, the task mapping and routing are optimized simultaneously for application schedule length reduction. To the best of our knowledge, this is the first study on the co-optimization of task mapping and routing for emerging point-to-point NoC-based HCSs. The main contributions of our article are as follows:

1) We analyze the design space of task mapping and routing for emerging NoC-based HCSs. We identify that algorithms that unilaterally explore task mapping or routing cannot get the optimal solution since task mapping and routing are strongly related.
2) We propose MARCO, a task mapping and routing co-optimization framework for emerging NoC-based HCSs, to decrease the schedule length of applications. Specifically, we revise the tabu search to explore the design space and evaluate the quality of task mapping and routing. The advanced reinforcement learning algorithm, i.e., advantage actor-critic, is adopted to compute paths efficiently.
3) We perform extensive experiments on various real applications, which demonstrates that the MARCO achieves a remarkable performance improvement in terms of schedule length

(+44.94% ∼ +50.18%) when compared with the state-of-the-art mapping and routing co-optimization algorithm for homogeneous computing systems. We also compare MARCO with different combinations of state-of-the-art independent mapping and routing approaches.

The remaining of this article is organized as follows: The background of emerging NoC-based HCSs and existing mapping and routing optimization methodologies as well as the motivation example are discussed in Section 2. Section 3 shows the definition of our problem. Section 4 presents our co-optimization framework, MARCO. In Section 5, to show the efficiency of our proposed algorithm, experimental results, which are compared with the state-of-the-art mapping and routing algorithms, are presented. The conclusion is proposed in Section 6.

2 BACKGROUND AND MOTIVATION

2.1 Background

• **Heterogeneous Computing Systems.** In the internet-of-things (IoT) era, edge computing encourages the development of dedicated energy-efficient processing elements which handle specific tasks. HCSs, which consist of various PEs that accelerate specific tasks is an emerging paradigm to process tasks rapidly with low energy consumption. Since HCSs consist of various PEs that vary in their capabilities, the task mapping be an important factor to determine the application performance. To improve performance of HCSs, many works [13, 17, 22, 25, 28, 29, 35] have been proposed in order to maximize the utility of PEs, thereby greatly reducing the total execution time.
• **Networks-on-chip.** NoCs are generally deployed to transfer data among the PEs. In the past, the end-to-end transmission latency greatly depends on the distance (number of hops) between the source and destination nodes due to the per-hop arbitration and buffering. However, with the development of NoC architecture, SMART NoCs [5] support single-cycle-multi-hop transmission, addressing the scalability concern raised by high latency for long-distance transmission and enlarging the design space of the routing algorithm to search the non-shortest paths. Based on such technology, many designs [3, 6, 10] featuring low power consumption or high performance are proposed. In this article, we denote the NoCs based on single-cycle-long-distance transmission design as "point-to-point NoCs". In point-to-point NoCs, if no contention occurs, messages bypass the intermediate arbitration and buffering which removes or decrease the dependency of distance. However, the end-to-end latency of NoC may be seriously degraded when communication contention increases.
• **Resource Management for NoC-based HCSs.** An NoC-based HCS mainly consists of PEs, routers, and links. For the HCS, its resources, PEs, are mainly managed through task mapping. As a sub-system, resources of NoC, routers, and links, are mainly managed both explicitly and implicitly through task mapping and routing. Specifically, when tasks are mapped to PEs, from the computation perspective, the task execution is determined. From the communication perspective, when the source and destination of a message are mapped to PEs, the minimum number of routers links this message should use is fixed. The routing algorithm further decides the specific links and routers to be assigned to this message. For these resource management strategies, there are many outstanding works tried to optimize in one specific aspect. Considering the new feature provided by point-to-point NoC, the corresponding co-optimization algorithm for the homogeneous system is proposed in [35]. In this work, contentions caused by limited NoC resources become the main optimization objective according to the architecture change.
• **Mapping Algorithms.** The objectives of mapping diverse from minimizing task execution time to minimizing data transmission time. Chou et al. [13] summarized the basic dynamic task mapping algorithms whose objective targets energy consumption or contentions. Also, Mandelli et al. [22] proposed an energy-aware dynamic task mapping algorithm that allows multiple tasks allocated

per PE. In this work, the communication energy is reduced as well as the parallelism of applications is increased. These two works, no matter for single or multi-task, target homogeneous computing systems. For HCSs, in [17], authors proposed a run-time spatial mapping algorithm so that the quality of service (QoS) requirements are met. Quan et al. [25] presented their hybrid task mapping algorithm in static mapping exploration and dynamic mapping optimization to improve system performance. Singh et al. [28, 29] showed their communication-aware task mapping algorithms in which the tasks of an application are mapped in close proximity to minimize the communication overhead. Also, the communication and computation-aware mapping proposed in [19] reduces communication overhead by mapping highly communicating tasks on the same PE. However, this mapping method needs to know communication time in advance, which is hard due to the complex NoC resource management.

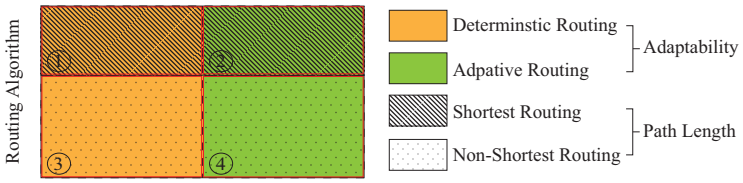

Fig. 3. Routing algorithms classification.

• **Routing Algorithms.** Talking about adaptability, there are two types of routing algorithms discussed in NoC: deterministic and adaptive routing. The deterministic routing decides the path of a message by its source and destination in advance. XY routing is the well-known deterministic routing algorithm where the data should be transmitted in the X direction at first and then the Y direction. While for the adaptive routing the path for a message is computed considering the network states. Odd-Even [12] and DyAD [18] are two successful algorithms. Talking about the path length, the routing algorithms can be classified into shortest routing and non-shortest routing. According to their names, shortest routing goes through the shortest path, otherwise, routing algorithms are called non-shortest routing. Previously, end-to-end latency in traditional NoCs highly depends on the path length so that the shortest routing algorithms are commonly adapted. However, in the point-to-point NoCs, non-shortest paths can be used to transmit data with the same latency as the shortest path, so that routing algorithms set contention minimization as the main objective. The SOTA routing algorithm for emerging point-to-point NoCs is proposed in [7].

These two kinds of classification are not exclusive and their relationship is shown in Figure 3. One routing algorithm can belong to one class based on its adaptability and belong to another class according to its path length. For example, XY routing is a deterministic and shortest routing located in area ①; Odd-Even and DyAD are adaptive and shortest routing algorithms located in area ②. In the context of application-specific design, the communication information, including the source, destination, and message volume, is relatively static. The routing algorithm designed for a known communication graph is discussed in many works [8, 23]. Together with point-to-point NoCs, we want to design routing algorithms that set contention minimization as the main objective according to the network state without considering the path length.

2.2 Motivational Example

Previously, we show the performance comparison of co-optimization for homogeneous computing systems and combinations of independent mapping and routing algorithms in Figure 2, motivating us to design the co-optimization framework for HCSs. In this section, we use a detailed example to show how mapping defines the routing design space and then decides the minimum communication

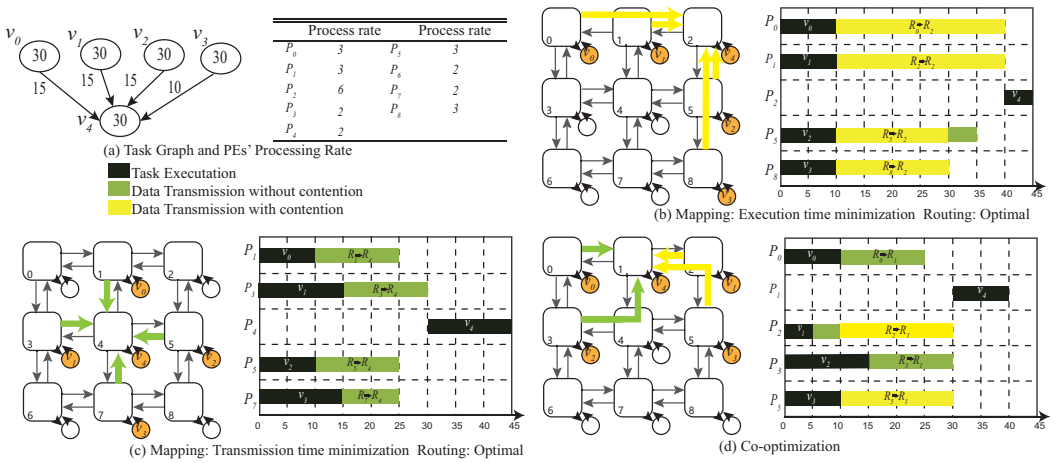

Fig. 4. Motivation examples. (a). DAG modeled application and processing rate of different PEs; (b). Computation-aware mapping and SOTA routing; (c). Communication-aware mapping and SOTA routing; (d). Co-optimized mapping and routing.

time that can be achieved. Also, we present how communication time and computation time jointly determine the schedule length.

Given the task graph in Figure 4(a), for each node $v \in \mathcal{V}$, its task workload is indicated using the number inside the node, and for each edge $e_{u,v} \in \mathcal{E}$ from task u to task v, its message size is represented by the number beside the edge. The processing rate of different PEs is listed in the table of Figure 4(a). Since the communication time depends on NoC performance, which is the same in the homogeneous and heterogeneous systems, the communication-aware mapping which performs well in the homogeneous system also performs excellently in the heterogeneous system in terms of communication time. As mentioned before, the **communication-aware mapping** is the mapping result of [35] and the **computation-aware mapping** is the mapping with the least task execution time.

Figure 4(b), (c) and (d) show three instances for task mapping and routing. The computation-aware mapping in Figure 4(b) is the mapping algorithm that minimizes total task execution time. Based on this mapping, if the optimal routing strategy which minimizes the transmission time is applied, the total schedule length is 45 due to the contention. Besides, in Figure 4(c), we apply the mapping with minimum transmission time, i.e., communication-aware mapping, then compute the optimal routing without contention. However, the schedule length is also 45 for this case due to time-consuming task execution. However, if we co-optimize the task mapping and routing as indicated in Figure 4(d), without minimizing task execution time or communication time, we can get the best schedule length, 40 in this example, on the contrary.

3 PROBLEM FORMULATION AND PRELIMINARIES

3.1 Problem Formulation

In Table 1, we summarize the notions used across this work. Furthermore, we model the problem regarding several critical considerations like application and architecture as below, followed by our optimization objective.

• **Application.** An application is represented by a directed acyclic graph (DAG), i.e., $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of tasks v and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of data transmission e between tasks.

Table 1. Notations Used in This Article

Notation	Description	Notation	Description
$\mathcal{V}$, $\mathcal{E}$ $\mathcal{M}$	The set of tasks nodes, edges, and messages.	$\|.\|$	The number of elements.
$\mathcal{P}$, $\mathcal{R}$, $\mathcal{L}$	The set of PEs, routers, and links.	p, r_p	The PE and its router.
$\tau_{v,p}$	The execution time of v on p.	$\mathcal{F}, \mathcal{G}$	The mapping and routing algorithm.
γ_m	The route to transmit data of m.	L	The latency related to data transmission.
S	The schedule length of an application.	T	The matrix of expected time to execute.
$C(\mathcal{F}, \mathcal{G})$	The cost function of mapping $\mathcal{F}$ and routing $\mathcal{G}$.	U	The usage status of each link in NoC.

Besides, for each node $v \in \mathcal{V}$, the task workload is represented by $|v|$. For each edge $e_{u,v} \in \mathcal{E}$ from u to v, its message is represented by $m_{e_{u,v}}$. The size message $m_{e_{u,v}}$ is denoted using $|m_{e_{u,v}}|$. The set of all messages is notated using $\mathcal{M}$.

• **Architecture.** Since the mesh is the most popular NoC topology for many core systems [24], as it is scalable, is easy to layout, and offers path diversity, we apply point-to-point NoC-based HCSs with the 2D mesh topology across this work, which consists of $N \times N$ PEs and routers, respectively. $\mathcal{P}$ is the set of PEs p. For every PE, there is a router r_p connect to it through the network interface (NI). The set of routers is notated by $\mathcal{R}$ and these routers are connected through a set of links $\mathcal{L} \subseteq \mathcal{R} \times \mathcal{R}$. Each link $l_{r_p,r_q} \in \mathcal{L}$ can be used to send data from r_p to r_q.

• **Mapping and Routing Algorithm.** Application mapping $\mathcal{F}$ is a function from tasks $\mathcal{V}$ to processors $\mathcal{P}$. $\mathcal{F}(v) = p$ represents the mapping of task v onto processor p. The routing algorithm is a function $\mathcal{G}(m) = \gamma$ that represents the message m is transmitted through the route γ. The route γ is a set of links that forward this message from the source to the destination.

• **HCS and Task Execution Time.** HCSs incorporate specialized processing capabilities to handle particular tasks. Without loss of generality, we model the execution time of tasks run on HCS system. Specifically, we use a coefficient-of-variation-based (CVB) generation method [1] to get the execution time matrix T. In T, each element $\tau_{v,p}$ refers to the expected execution time of task v on PE p and $\tau_{v,p}$ is defined using Eq. 1.

$$\tau_{v,p} = \Gamma(\alpha, \beta_v) \tag{1}$$

where $\alpha = 1/V_{HCS}^2$, $\beta_v = \mu_v/\alpha$. μ_v is the expected execution time of task v on all PEs in the HCS; V_{HCS} is defined by user to describe the heterogeneity degree of the HCS. We should note that, when a specific application runs on given HCS that our algorithm would be applied to, the matrix τ can be generated through profiling. In T, the variation along the row is referred to the PE's heterogeneity, i.e., the degree of PE execution times vary for a given task v. The reason for choosing the gamma distribution Γ distribution is the non-negative constraint and variable coefficient. Also, it can approximate to Erlang-k distribution (when α is fixed to be an integer) and Gaussian distribution (when α is large enough). Based on [1], the heterogeneity degree of the HCS is controlled by a hyper-parameter heterogeneity coefficient V_{HCS}. The large the heterogeneity coefficient V_{HCS} demonstrates the high heterogeneity degree.

• **NoC and Latency.** Generally, in NoCs, flits are forwarded hop by hop. Thus, the latency of a message m with the specific size is dependent on the length of route $|\gamma_m|$ and contention time L_c. As we mentioned before, for point-to-point NoCs, the dependency of $|\gamma_m|$ is removed or decreased. Thus the routing algorithm focuses on contention minimization only. One of the point-to-point NoCs which supports routing algorithms without constraints named ArSMART is proposed in [6]. We choose ArSMART NoCs as the communication backbone due to the following reasons: (i) ArSMART NoCs show a better performance in terms of latency reduction. Compared with the other point-to-point NoCs, i.e., SMART NoCs, ArSMART NoCs show an average reduction of 40.7% in application schedule length; (ii) The single-cycle-long-distance transmission supported by

the ArSMART NoCs decreases the latency dependency on the path length, which encourages the routing algorithm to search more paths, including non-shortest paths, with less resource contention to reduce latency; (iii) ArSMART NoCs support arbitrary routing algorithms and do not involve any constraint on the path selection, e.g., direction or number of turns, which further extends the design space of routing. In ArSMART NoCs, the latency of a message is shown in Equation 2.

$$L_m^{ar} = L_{cs} + L_{conf} + L_{tr} \tag{2}$$

ArSMART NoCs eliminate the contention at intermediate routers by blocking the low priority message at the source using time, L_{cs}. In ArSMART NoCs, the route is configured by the controller directly with configuration time, L_{conf}. Then, the data can be transmitted from the source to the destination costing L_{tr}. Generally, the L_{conf} is small and L_{tr} is proportional to the message size, letting the contention become the optimization objective to decrease latency.

• **Optimization Objective.** The objective of our co-optimization framework is to minimize the schedule length of an application. The schedule length of an application equals to its final task finish time t_e minus its first task start time t_s. Since the schedule length is determined by mapping and routing, the objective of co-optimization framework is to find a mapping function $\mathcal{F}^*$ and corresponding routing function $\mathcal{G}^*$ to minimize the S as follows:

$$\mathcal{F}^*, \mathcal{G}^* = \arg\min_{\mathcal{F},\mathcal{G}} S \tag{3}$$

3.2 Preliminaries

• **Reinforcement Learning.** In literature, reinforcement learning (RL) mainly focuses on maximizing the cumulative reward within a series of actions. Over the past few years, RL has achieved remarkable success in a wide range of real-world embedded applications, such as hardware-aware neural architecture search [32], computation offloading in edge systems [11], and real-time job scheduling [9]. For task mapping, Life-Guard [26] uses RL to leverage the different impact of applications on ading and improve the system health. To summarize, the RL-based method should include the following components:

- A set of environment and agent states $S = \{s_t\}$ where s_0 denotes the initial state;
- A set of actions $A = \{a_t\}$ of the agent;
- A reward function $R(s_t)$ to quantify the reward in terms of current state s_t;
- A transition function $P_{a_t}(s_t, s_{t+1})$ to attain the probability of transition from state s_t to state s_{t+1} under the action a_t;

Next, we demonstrate how the agent interacts with the environment and vice versa. Assume the agent observes the environment with state s_t at the time step t. Based on the defined transition function $P_{a_t}(s_t, s_{t+1})$, the agent chooses an action $a_t \in A$, thereby mapping the environment from state s_t to state s_{t+1}. Given the reward function $R(s_t)$, the agent receives an immediate reward denoted as r_{t+1} regarding the new environment state s_{t+1}. There exist two categories of learning strategies, i.e., value-based and policy-based. The former aims to learn a value function, e.g., action-utility function, to quantitatively evaluate the action while the latter aims to learn an optimal mapping between the environment state and the action from the agent, i.e., $s_t \rightarrow a_t$. Regardless of the value-based and policy-based strategies, the agent finally learns a set of optimal policies $\pi^* : A \times S \rightarrow [0, 1]$ to maximize the expected cumulative reward, which can be formulated as follows:

$$\pi^*(s_t) = \arg\max_{a_t \in A} \sum_{s_{t+1}} P_{a_t}(s_t, s_{s+1}) R'(s_{t+1}) \tag{4}$$

where $R'(\cdot)$ denote the discounted reward and can be formulated as follows:

$$R'(s_t) = \sum_t \eta^t r_t \tag{5}$$

where $\eta \in [0, 1]$ denotes the discounted factor. The η is included due to the empirical principle, i.e., older states have less and less influence on the later states.

4 MAPPING AND ROUTING CO-OPTIMIZATION

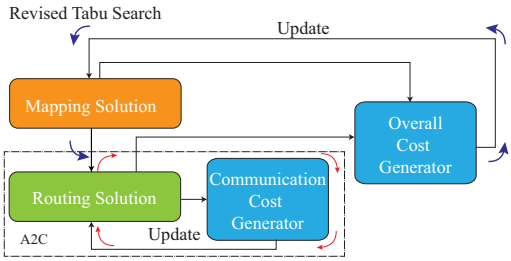

Fig. 5. The overview of co-optimization framework.

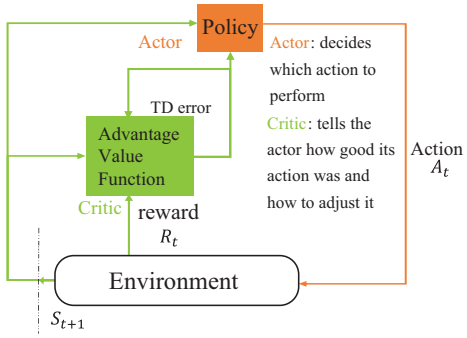

Fig. 6. Illustration of A2C algorithm.

Our co-optimization framework is based on robust tabu search. This section explains why we choose robust tabu search and how we apply it to explore the design space of task mapping and routing.

4.1 Overview

The mapping problem can be modeled as a quadratic assignment problem (QAP) [16]. Existing studies[2, 27], by comparing kinds of algorithms e.g., genetic algorithms, simulated annealing, and tabu search, have shown that, in most cases, the most efficient method to find optimal or near optimal solutions for the QAP is tabu search and its variants, which take both computational time and the quality of solution into consideration.

Based on these studies, we are inspired to explore the design space of task mapping and routing through robust tabu search [31]. We revise the quality of the solution so that it reflects the quality of both mapping and routing. The overview of our co-optimization framework based on revised tabu search is shown in Figure 5. For a task mapping solution, we propose a routing solution by advantage actor-critic (A2C). The performance for a specific mapping and routing solution is evaluated by the overall performance evaluator.

4.2 Robust Tabu Search

In order to show our algorithm in a more specific way, we provide both a flowchart and pseudo-code. Figure 7 shows how robust tabu search works to get the result. To avoid being overly complicated and easy to read, we split the flowchart into two parts. The main flow is described in Figure 7 and the "expand neighborhood" function is presented in Figure 8. The pseudo-code is at the end of this section.

To make our algorithm converge fast, we initialize mapping function $\mathcal{F}_0$ that maps tasks to PEs with highest processing ability. It is possible for multiple tasks to be mapped to the same PE. This

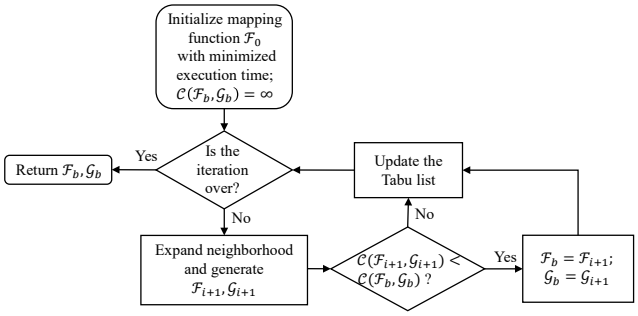

Fig. 7. The main process of tabu search.

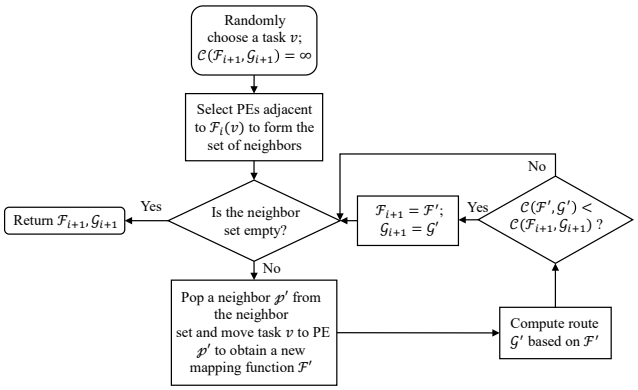

Fig. 8. Expand neighborhood flow chart at iteration step i.

can decrease data transmission latency but may not be advocated due to its negative impact on program parallelism.

After the initial operation, it comes to the iterative process. In tabu search, the set of neighbors of a certain state is defined as a set which is all possible states that can be obtained after one operation of the state $\mathcal{F}$. Based on this definition, the operation in our algorithm is, first select a task v which is mapped to $\mathcal{F}(v)$, then move this task to PEs adjacent to $\mathcal{F}(v)$. In 2D-mesh NoC, we set the neighbors for the PE p as the set of PEs whose distance to p less than 2. Thus, each PE has up to eight adjacent PEs (e.g., the PEs adjacent PE 4 in Figure 4 are PE 0, 1, 2, 3, 5, 6, 7, 8) and at least three adjacent PEs (e.g., the PEs adjacent PE 8 in Figure 4 are PE 4, 5, 7). Therefore, based on this setting, each set of neighbors has three to eight possible states to explore.

After the process of expanding neighborhood, the algorithm should choose the best state from the set of neighbors. As a co-optimization method, our algorithm integrates mapping and routing to select the best solution. Specifically, we compute the routing for each mapping result by our routing algorithm mentioned in section 4.4, then evaluate the performance for this mapping and routing.

4.3 Overall Performance Evaluator

To measure the quality of a solution, we define a cost function $C(\mathcal{F}, \mathcal{G})$ which corresponds to task mapping and routing. According to the optimization objective proposed in 3.1, C should be the schedule length. However, in point-to-point NoC-based HCSs, the overall schedule length, i.e.,

the number of cycles the entire task graph runs, is hard to compute due to the parallelism. Such parallelism includes two aspects. Generally, the parallelism of the application is influenced by its in and out degree of its task graph. Moreover, the tasks are executed and data is transmitted in parallel on point-to-point NoC-based HCSs. Moreover, the efficiency or parallelism of executing tasks and transferring data is different in point-to-point NoC-based HCSs. A simple example is that a PE has only one processor to execute tasks, but a router has four links to transmit data. Due to the influence of parallelism of task execution and data transmission, linear accumulation of the execution time and data transmission time does not equal the schedule length.

Therefore, if we want our algorithm to perform well in different types of task graphs on various point-to-point NoC-based HCSs, we need to compute or estimate the schedule length efficiently. Given the task mapping algorithm F and routing algorithm G, we can get the total schedule length using the simulator. However, this method is time-consuming so it cannot be integrated into our co-optimization framework. Thus, we propose a simple method to estimate the schedule length: According to the communication performance evaluation algorithm proposed in Section 4.4.3, the estimated time interval for the message, $[t_{1_m}, t_{2_m}]$, is recorded. For every message $m_{e_{u,v}}$, we add the execution time $\tau_{v,\mathcal{F}(v)}$ of its destination task v to its end time and notate this value as $D_m = t_{2_m} + \tau_{v,\mathcal{F}(v)}$. The maximum $D^* = Max(D_m)$ is the schedule length estimation we used across the work. This estimation may not be the same as the actual schedule length since the time interval is not accurate.

The smaller C is, the better the quality of this solution. At the mapping level, this means finding a PE with strong computing ability; at the routing level, this means that there is less contention in the result we find. When we find the best solution among the set of neighbors in the current state, we set it to the state of the next iteration, and then add it to the tabu list.

4.4 Routing algorithm

4.4.1 Overview. Our routing algorithm is based on advantage actor-critic(A2C) which is a RL-based algorithm. At first, we discuss why we choose A2C to compute the route for each message.

In our co-optimization framework, we need to calculate routes for messages frequently, so our routing algorithm should be efficient and not time-consuming. For the SOTA contention-aware routing algorithm [7] proposed for point-to-point NoCs, its time complexity of is $O(|\mathcal{E}|^2 \cdot |\mathcal{R}|^2)$, where $|\mathcal{E}|$ is the number of edges in the task graph and $|\mathcal{R}|$ is the number of routers. Although this algorithm shows its brilliant performance, it consumes a lot of time and therefore cannot be used in our co-optimization framework. Other routing strategies, e.g., XY, DyAD, and odd-even, although very efficient in terms of time complexity, as we mentioned, only explore shortest paths and cannot find the optimal solution, especially in point-to-point NoCs. To adapt to the development of emerging NoC architectures, we propose a new algorithm.

Moreover, RL, as an efficient design space exploration method, has shown its efficiency in solving many problems [14, 15, 21]. For our routing algorithm, although the transmission of messages in NoC is a continuous process, we can regard the transmission along with links as step-by-step when we are calculating the route. The agent only needs to determine which direction the message should be sent in the current step.

Thus, we adopt A2C [30], a revised RL-based method, to compute the route for every message. Other efficient methods for path computation, e.g., imitation learning-based method [20], will be discussed and compared in our future works. A detailed introduction about our routing algorithm is be placed in Section 4.4.2 and Section 4.4.3.

4.4.2 Path Computation. Based on actor-critic algorithm [4, 34], A2C is proposed to coverage quickly. Figure 6 shows a schematic diagram of A2C algorithm. It consists of two networks, actor

Algorithm 1: Robust Tabu Search

Input: $\mathcal{V}, \mathcal{E}$, The set of tasks nodes and edges.
$\mathcal{P}$, The set of processing elements.
Output: Best mapping result $\mathcal{F}_b$ and its corresponding routing result $\mathcal{G}_b$

Initialize Initialize mapping function $\mathcal{F}_0$ with minimum execution time
$C(\mathcal{F}_b, \mathcal{G}_b) = \infty$
Tabu list, TL=$\emptyset$
for $i \in [0, max\_iteration]$ **do**
  $v = RandomSelection(\mathcal{V})$ //randomly choose a task from $\mathcal{V}$
  $C(\mathcal{F}^{'}, \mathcal{G}^{'}) = \infty, j^{'} = 0$
  for $j \in Neig(\mathcal{F}_i(v))\ AND\ (\mathcal{F}_i(v), j) \notin TL$ **do**
    $\mathcal{F}_i(v) = j$
    $\mathcal{G}_i = routeComputation(\mathcal{F}_i)$ //In Section 4.4
    if $C(\mathcal{F}_i, \mathcal{G}_i) < C(\mathcal{F}', \mathcal{G}')$ **then**
      $\mathcal{F}^{'} = \mathcal{F}_i, j^{'} = j$
    restore $\mathcal{F}_i$
  $\mathcal{F}_{i+1} = \mathcal{F}^{'}$
  if TL is full **then**
    delete $TL[0]$
  add $(\mathcal{F}_i(v), j^{'})$ to TL's tail
  if $C(\mathcal{F}_{i+1}, \mathcal{G}_{i+1}) < C(\mathcal{F}_b, \mathcal{G}_b)$ **then**
    $\mathcal{F}_b = \mathcal{F}_{i+1}$
    $\mathcal{G}_b = \mathcal{G}_{i+1}$

and critic. Actor, which is a typical policy-based function, accepts the state s_t of the agent in step t as input, and then outputs the action a_t that the agent takes in this state. Critic, which is a typical value-based function evaluate this action a_t made by actor. The value function of A2C in critic is replaced with advantage value $A_w(s, a)$ defined by E.q. 6.

$$A_w(s, a) = Q_w(s, a) - V_w(s) \tag{6}$$

In this formula, $V_w(s)$ is the state value and Q_w is the state-action value corresponding to the action a in state s. Therefore, the advantage value $A_w(s, a)$, is the advantage of current action a over the average action in state s, which can be used to compare the current action and the average action. Using the TD estimation, advantage value $A_w(s, a)$ is estimated using E.q. 7, where θ is the discount-rate parameter.

$$A_w(s, a) = r + \theta V(s') - V(s) \tag{7}$$

In A2C, our environment is links' usage status. Specifically, links can be occupied by messages in a certain period of time. Our agent is the message $m \in \mathcal{M}$ that needs to calculate the route. In mesh typologies, the agent has at most four actions to choose, accelerating the computation. The state is directions the agent has gone through in NoC (it refers to the route of this message). The reward of a certain state is the opposite number of the output of the communication performance evaluator during transmission through the route given by this state.

Specifically, for one message, in state s_t, if it reaches a router but not reached the destination, it will wait for the actor to tell which direction it should transmit next. After the agent executes the action, i.e., the message moves to the next router, the state changes correspondingly. Then, the

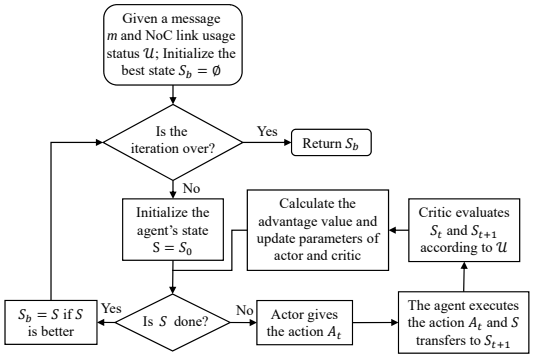

Fig. 9. Illustration of using A2C algorithm to compute route.

critic evaluates advantage values based on reward r and updates the parameters of actor and critic. Figure 9 shows the detailed process of calculating the route. When the agent advances a step in the environment, the communication performance evaluator mainly estimates the contention time of the route the agent has traveled, and then we use its opposite number as the current reward r_t for computation. A detailed description of the communication performance evaluator is discussed in Section 4.4.3.

At the beginning of the entire algorithm, we add the outgoing messages of task 0 to the queue, i.e., the task that can be executed immediately, and sort them according to their message size to achieve high throughput by following the principle of "Shortest-Job-Next". After one message is dequeued and the route of the this message is calculated, we check whether the destination task of this message can start running (if it has received all required messages), if so, we add all messages of the destination task to the queue. In addition, we update links occupied by the route of this message to U.

4.4.3 Communication Performance Evaluator. Given a route γ_m to transmit data of m and current usage status U of each link in NoC, the communication performance evaluator mainly estimates contention time during the transmission. To reduce the waste of communication resources, the path length $|\gamma_m|$ is added to the evaluation result. We should note that, generally, if contention occurs, t_c is much larger than $|\gamma_m|$, thus adding $|\gamma_m|$ to the evaluation do not limit our path selection to shortest ones, only making our algorithm prefer shorter path among paths with same contention estimation.

For each task v, its start time is the maximum end time of all input message. For the message m, we denote the end time of the source task as t_1, which is also the earliest start time of this message. In the ideal situation, if links used in the route γ_m do not have any contention, then this message can start sending at t_1 and end at $t_1 + Max(L_{w/o})$, where $Max(L_{w/o})$ is the upper of data transmission time without contention. According to [6], $Max(L_{w/o})$ can be computed by $|m|$ and $|\gamma_m|$. We denote $t_1 + Max(L_{w/o})$ as t_2.

However, in reality, the contention may be unavoidable. At time point t_1, we query current link usage status U and check if links that γ_m needs are occupied by other messages. If not. we need to wait for time t_c until all links in γ_m are available. In this situation, the start time of m becomes $t_1 + t_c$, and the end time becomes $t_2 + t_c$. This t_c is the estimated contention time.

To calculate t_c, we need to query U to know which links in γ_m have been occupied (or partially occupied) by other messages in the time interval of $[t_1, t_2]$. Among these other messages, we find the latest message that ends, record its end time $t_2^{'}$, and then use $t_2^{'} - t_1$ as the initial value of t_c.

Then we try whether the new interval $[t_1 + t_c, t_2 + t_c]$ is available on all links in γ_m. If it is available, then this t_c is the final result of our calculation; if not, we continue to find $t_2^{'}$ and increase the value of t_c until all links in γ_m are available during the time interval $[t_1 + t_c, t_2 + t_c]$. When t_c is determined, we need to update U and reserve the time interval $[t_1 + t_c, t_2 + t_c]$ on links used by γ_m.

5 EXPERIMENTS

In this section, we conduct a series of experiments to verify the performance of our algorithm. We explore the performance of our algorithm on various task graphs, different mesh sizes and different degrees of heterogeneity.

5.1 Experiment Settings

In the experiments, we chose the co-optimization algorithm for homogeneous computing systems as well as the combination of SOTA mapping and routing algorithms to compare with our MARCO framework. For the heterogeneous system, PEs' energy consumption and processing ability are not directly related, e.g., GPUs can compute high dimensional convolution with less energy but higher speed than CPUs. The energy consumption of the heterogeneous system is hard to reasonably model or simulate based on our task execution model due to the various kinds of PEs integrated, e.g., GPU or ASICs, thus we do not include energy consumption in the optimization objective and experiments. The co-optimization algorithm for homogeneous computing systems using emerging point-to-point NoCs is proposed in [35]. Since the mapping part of this co-optimization method considering the communication performance only, we treat its mapping result as the **communication-aware mapping**. Moreover, the **computation-aware mapping** is, blocking out the influence of communication in our tabu search and searching for the mapping result considering execution time only. For the routing, we choose the widely applicable XY routing and the routing proposed in [8]. The latter one is the state-of-the-art contention minimization routing proposed for emerging point-to-point NoCs and we notate it as **"SOTA"**. Their combinations are listed in Table 2. Specifically, case 1 and case 4 apply the same routing algorithm to show the difference between communication-aware mapping and computation-aware mapping. Besides, by comparing case 1, case 2, and case 3, we show the efficiency of our routing algorithm using RL. To prove the necessity of co-optimization, we set up case 5.

Since MARCO is dependent on the task graph generated by prior static profiling of application, the randomness of applications caused by the unpredictable operations (e.g., of conditional branches) influences the efficiency of our proposed algorithm. To reduce the influence of such randomness, we conduct experiments on streaming applications with few branches from StreamIt benchmark [33]. The task graphs executed in the experiment are Autocor, Audiobeam, FMRadio, and H264. The parameters of these task graphs are list in Table 3. As mentioned before, the NoC infrastructure applied in this work is ArSMART NoC. We implement ArSMART NoC in gem5 and its default configuration is shown in Table 4. We try to decrease the influence randomness during the execution in future works.

Currently, the support for the heterogeneous system is limited in gem5. Thus, to simulate the HCS execution model we proposed in Section 3.1, we revise the task execution as following: Given a CVB-based execution time matrix T of system and a task graph G, for the task v mapped to PE p, during the simulation, after PE p receiving all data it needs based on task graph G, it pends for $T_{v,p}$ cycles. Using the CVB-based execution time matrix, although we use the 64 in-order SPARC processor model provided by the Gem5, the execution time for the same task runs on different PEs varies, simulating the task execution for heterogeneous systems. All configurations about our heterogeneous system are listed in Table 3. However, using this method, the power and energy consumption modeling provided by original gem5 is no longer applicable since the PEs are not

Table 2. Different cases in experiments

Notation	Mapping	Routing
Case 1[1]	Communication-aware mapping	XY routing
Case 2	Communication-aware mapping	SOTA routing algorithm.
Case 3	Communication-aware mapping	RL-based routing algorithm.
Case 4	Computation-aware mapping	XY routing algorithm.
Case 5	Computation-aware mapping	RL-based routing algorithm.
case 6[2]	**Communication&Computation-aware mapping.**	**RL-based routing algorithm.**

[1] Case 1 is the co-optimization algorithm proposed for homogeneous system.
[2] Case 6 is our co-optimization algorithm, MARCO, proposed for heterogeneous system.

really heterogeneous. We omit the energy comparison in this article. In future work, we will revise the gem5 design and let it support more heterogeneous cores.

For experiments in this section, the default V_{HCS} is 0.25. Additionally, Our experiments are carried out on Ubuntu 16.04 running on an Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz host machine with 32 GB RAM.

Table 3. Benchmark Descriptions

Benchmark	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	In/Out[1]	Introduction
Audiobeam	22	35	15/15	Audio beamformer
Autocor	12	18	8/8	Auto-correlation series producer
Fmradio	31	37	4/4	FM radio with equalizer
H264	51	71	11/12	Video compression standard

[1] In/Out represents the maximum in/out degree of the task graph.

Table 4. Default Configurations

Parameter	Specification	Parameter	Specification
		Process	
Tile Size	1 mm × 1 mm	Frequency	1 GHz
Technology	45 nm		
		NoC	
Topology	2D Mesh	NoC Size	8 × 8
HPC_{max}	8	Flit Width	128-bit
Package Size	4 flits	Buffer Size	4 flits
VCs	2 VCs/port	Router Size	5 ports
		System[1]	
Processors	64 in-order SPARC	Cache Coherence	MOESI distributed directory
L1 Cache	Private,32KB	L2 Cache	Shared,1 MB per core

[1] Heterogeneity is modeled using CVB-based execution time matrix T.

Table 5. Time Consumption of Routing Algorithms

Size		Audiobeam	Autocor	Fmradio	H264
4×4	[8]	11.624 s	3.768 s	20.140 s	29.214 s
	Ours	10.959 s	5.471 s	11.891 s	23.563 s
	Saving	-0.665 s	-1.703 s	8.249 s	5.651 s
8×8	[8]	143.164 s	63.673 s	278.702 s	439.043 s
	Ours	30.098 s	16.623 s	31.428 s	71.809 s
	Saving	113.066 s	47.05 s	247.274 s	367.234 s
16×16	[8]	161.175 s	98.154 s	346.207 s	834.181 s
	Ours	77.391 s	50.203 s	99.447 s	139.118 s
	Saving	83.784 s	47.951 s	246.760 s	695.063 s

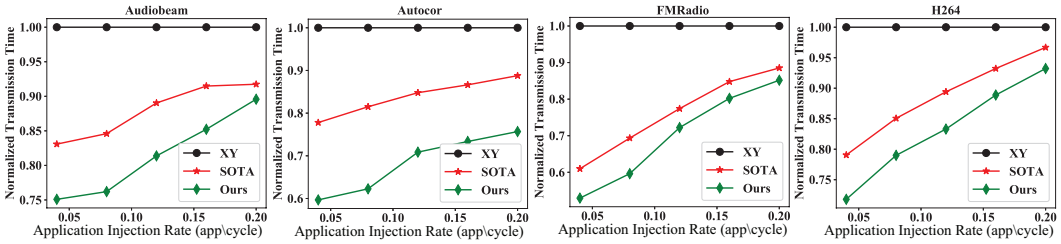

Fig. 10. Transmission time comparison in terms of application injection rate.

5.2 Routing Algorithm Performance

Previously, we introduced our RL-based routing algorithm whose optimization objective is contention minimization. Although RL, as an efficient design space exploration method, has shown its efficiency in solving many problems [14, 15, 21], we conduct some experiments to show that using RL to solve our contention minimization problem is efficient.

5.2.1 Time Consumption. In our revised tabu search algorithm, for each mapping solution, we should compute a routing solution. Thus, the routing algorithm should be quick enough. In Table 5 we compare the time consumption of the SOTA routing algorithm and our proposed routing algorithm. Although our routing algorithm spends similar time with the SOTA algorithm on small mesh size, it consumes much less time on medium and large mesh size compared with the SOTA approach. Also, the time consumption of both routing algorithms is proportional to the complexity of the task graph which is related to $|\mathcal{E}|$. In general, due to the efficiency of our routing algorithm, the time saving of our algorithm is more obvious than the SOTA method on complex task graphs.

5.2.2 Influence of Application Injection Rate/Traffic Levels. Since MACRO optimizes performance for a given application with its task graph provided, application injection rate (AIR) is used to represent the different traffic levels. On 16×16 mesh size, the influence of AIR is shown in Figure 10. Applying the same mapping strategy (communication-aware mapping), our RL-based routing algorithm performs better than traditional XY routing and SOTA routing. Again, both XY and SOTA algorithms handle simple and serial applications effectively. As the figure shows, with the AIR increases, the advantage of SOTA routing and our proposed routing vanishes, demonstrating the limited performance improvement can be achieved through proper routing on high traffic loads.

5.3 Overall Performance

We choose schedule length, the number of cycles the entire task graph runs, as our main measurement index. In addition, total execution time, transmission time, and contention time are selected for comparison, to show the performance breakdown of our algorithm. Note that, the definition of case 1-6 are listed in Table 2 and detailed information is presented in Section 5.1.

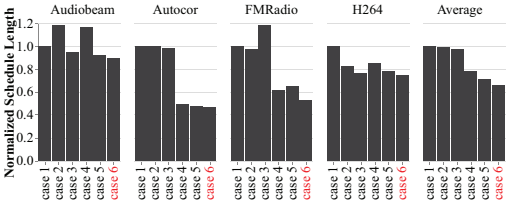

Fig. 11. Normalized Schedule length in 4×4 2D-mesh point-to-point NoC-based HCS.

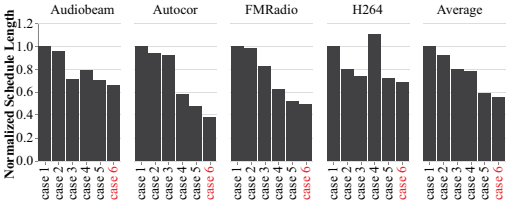

Fig. 12. Normalized Schedule length in 8×8 2D-mesh point-to-point NoC-based HCS.

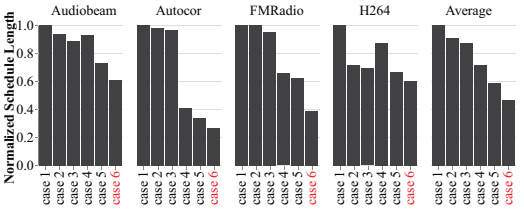

Fig. 13. Normalized Schedule length in 16×16 2D-mesh point-to-point NoC-based HCS.

5.3.1 Schedule Length. Figure 11 demonstrates the experimental results on the 4×4 2D-mesh point-to-point NoC-based HCS. Since there are only 16 PEs in such point-to-point NoC-based HCS, the probability of multiple task nodes being mapped to the same PE increases significantly. When the source and destination task nodes of an edge $e \in \mathcal{E}$ are mapped to the same PE, transmission can be completed without latency. This is one of the reasons why the transmission time has changed. Figure 12 demonstrates the normalized schedule length between our algorithm, i.e., case 6, and other algorithms in different task graphs on the 8×8 2D-mesh point-to-point NoC-based HCS. From the experimental results given in the figure, compared with case 1, case 2, and case 3, our algorithm reduces the number of cycles required to run the task graph by an average of 50.18%, 46.11%, 44.94%. Besides, in order to verify the necessity of co-optimization, we set case 5, which separates the joint optimization part of our co-optimization framework into two steps to be performed mapping and routing separately. Although it also takes into account the heterogeneity of point-to-point NoC-based HCSs, its performance is still not as good as the co-optimization method. This also emphasises the motivation of our algorithm. Figure 16 shows the experimental results on the 16×16 2D-mesh point-to-point NoC-based HCS. The size of point-to-point NoC-based HCS becomes larger, and the design space where the routing algorithm can be optimized also becomes larger. Therefore, in this case, the contention time of case 1 with XY routing is much larger than that of other algorithms.

5.3.2 Execution and Transmission Time. We sum up execution and transmission time spent by every task $v \in \mathcal{V}$ and message $min\mathcal{F}$ to show how our co-optimization framework well balances the time consumption on execution and transmission. Although we cannot present the parallelism of

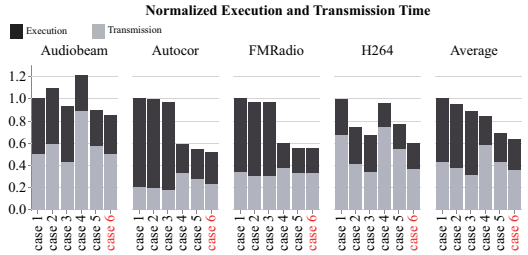

Fig. 14. Normalized execution and transmission time in 4 × 4 2D-mesh point-to-point NoC-based HCS.

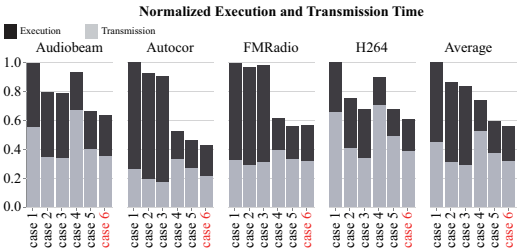

Fig. 15. Normalized execution and transmission time in 8 × 8 2D-mesh point-to-point NoC-based HCS.

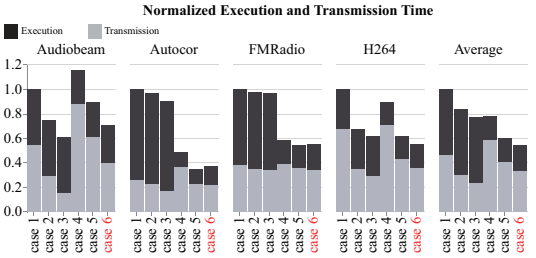

Fig. 16. Normalized execution and transmission time in 16 × 16 2D-mesh point-to-point NoC-based HCS.

execution and transmission in this experiment, the results demonstrate the design concept of our co-optimization framework. The mapping algorithm used in case 1, 2, and 3 is communication-aware. These cases do not consider the heterogeneity of the HCS, so their execution time are larger than case 4, case 5 and case 6, which is illustrated in Figure 14, Figure 15 and Figure 16. For this reason, even though they use a good routing algorithm with low contention times, their performance on schedule length is still not as good as our algorithm. In addition, by comparing case 1 and case 4, using the same routing algorithm, we show the difference between communication-aware mapping and computation-aware mapping. Neither two of these mapping algorithms can perform better on all applications due to the differentiation of task graphs. Although according to Figure 15, the average execution time of case 4 and case 5 is smaller than that of case 6. However, because they do not consider the communication performance, which makes its contention time much larger than that of case 6. Their total transmission and execution are longer than case 6.

5.3.3 Influence of Heterogeneity Coefficient. As we mentioned in Section 3.1, we use gamma distribution to model HCSs. The degree of heterogeneity of HCSs depends on the heterogeneity coefficient V_{HCS}. According to the algorithm mentioned in Section 3.1, the larger the heterogeneity coefficient, the greater the gap in the processing ability of PEs. Note that, for one specific application, the corresponding execution time matrix can be computed and our algorithm can be applied to derive the application-specific solution. The Figure 17 shows the curve of schedule length with different heterogeneity coefficient. It can be seen that as the heterogeneity of HCSs increases, the results obtained by the three computation-aware algorithms of case 4, case 5 perform better in most experimental instances. This is because PEs with high computation ability appear more in the HCS with a larger heterogeneity coefficient, and algorithms of case 4 and case 5 find these PEs and map tasks to them, thereby reducing the number of cycles required to execute the entire task graph. However, these two cases randomly define the routing design space, resulting in not

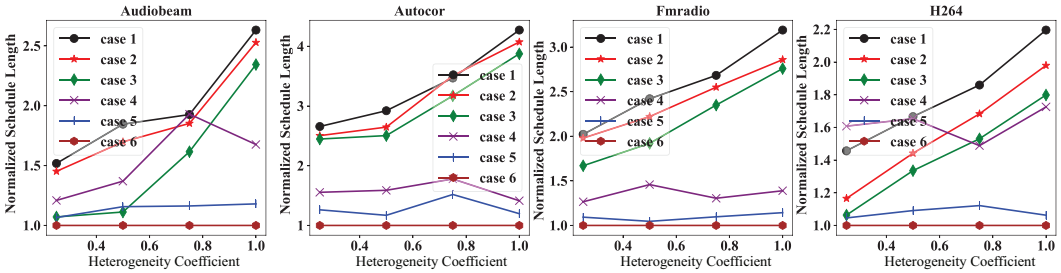

Fig. 17. Schedule length comparison in terms of different heterogeneity degree.

fully minimizing the communication time. Also, since the mapping algorithm used in case 1, case 2, and case 3 only considers communication, the large processing ability gap generated by high heterogeneity degree results in a huge performance difference. In this experiment, we show that no matter how heterogeneous the HCS is, our algorithm demonstrates good performance in terms of schedule length.

6 CONCLUSION

In this article, for the first time, we proposed a task mapping and routing co-optimization framework, MARCO, to boost the task execution and data transmission efficiency in point-to-point NoC-based HCSs. Although this work adapts point-to-point NoCs as its communication backbone to feature the advanced technologies, the computation and communication trade-off through mapping and routing exists in other NoC-based HCSs, inspiring us to extend our co-optimization to other platform in the future. Compared with the state-of-the-art mapping and routing co-optimization algorithm for homogeneous computing systems, our approach has average performance improvements ranging from 44.94% to 50.18%. We revised the tabu search to explore the design space and adopted the RL-based method to efficiently compute paths. We will compare and apply other efficient algorithms to further improve performance in future works. Since MARCO is conducted offline and depends on the task graph generated by prior static profiling of application, the randomness of applications caused by the unpredictable operations (e.g., of conditional branches) may influence its efficiency. Currently, to reduce the influence of such randomness, we conduct experiments on streaming applications with few branches. In future works, we will continue to investigate this problem to extend MARCO to the adaptive on-the-fly framework. Preliminary mapping and routing with run-time adjustments may be a feasible method to design the online algorithm. Also, currently, our MARCO framework focuses on time optimization only and can be applied to time-critical systems. In the future, we try to extend our work to focus on more metrics.

ACKNOWLEDGMENTS

This work is partially supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 (MoE2019-T2-1-071) and Tier 1 (MoE2019-T1-001-072), and Nanyang Technological University, Singapore, under its NAP (M4082282) and SUG (M4082087).

REFERENCES

[1] Shoukat Ali, Howard Jay Siegel, Muthucumaru Maheswaran, Debra Hensgen, and Sahra Ali. 2000. Task execution time modeling for heterogeneous computing systems. Proceedings of the Heterogeneous Computing Workshop, HCW (2000), 185–199. https://doi.org/10.1109/hcw.2000.843743

[2] Marvin A Arostegui Jr, Sukran N Kadipasaoglu, and Basheer M Khumawala. 2006. An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems. International Journal of

Production Economics 103, 2 (2006), 742–754.
[3] Yashar Asgarieh and Bill Lin. 2019. Smart-Hop Arbitration Request Propagation: Avoiding Quadratic Arbitration Complexity and False Negatives in SMART NoCs. ACM Trans. Des. Autom. Electron. Syst. 24, 6, Article 64 (Oct. 2019), 25 pages. https://doi.org/10.1145/3356235
[4] A. G. Barto, R. S. Sutton, and C. W. Anderson. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on Systems, Man, and Cybernetics SMC-13, 5 (1983), 834–846. https://doi.org/10.1109/TSMC.1983.6313077
[5] Chia Hsin Owen Chen, Sunghyun Park, Tushar Krishna, Suvinay Subramanian, Anantha P. Chandrakasan, and Li Shiuan Peh. 2013. SMART: A single-cycle reconfigurable NoC for SoC applications. Proceedings -Design, Automation and Test in Europe, DATE (2013), 338–343. https://doi.org/10.7873/date.2013.080
[6] Hui Chen, Peng Chen, Jun Zhou, Duong HK Luan, and Weichen Liu. 2020. ArSMART: An Improved SMART NoC Design Supporting Arbitrary-Turn Transmission. arXiv preprint arXiv:2011.09261 (2020).
[7] P. Chen, W. Liu, H. Chen, S. Li, M. Li, L. Yang, and N. Guan. 2020. Reduced Worst-Case Communication Latency Using Single-Cycle Multi-Hop Traversal Network-on-Chip. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (2020), 1–1. https://doi.org/10.1109/TCAD.2020.3015440
[8] Peng Chen, Weichen Liu, Mengquan Li, Lei Yang, and Nan Guan. 2020. Contention Minimized Bypassing in SMART NoC. Proceedings of the Asia and South Pacific Design Automation Conference, ASP-DAC 2020-January (2020), 205–210. https://doi.org/10.1109/ASP-DAC47756.2020.9045103
[9] Shengkai Chen, Shuiliang Fang, and Renzhong Tang. 2019. A reinforcement learning based approach for multi-projects scheduling in cloud manufacturing. International Journal of Production Research 57, 10 (2019), 3080–3098.
[10] Xianmin Chen and Niraj K Jha. 2016. Reducing wire and energy overheads of the SMART NoC using a setup request network. IEEE Transactions on Very Large Scale Integration (VLSI) Systems 24, 10 (2016), 3013–3026.
[11] Xianfu Chen, Honggang Zhang, Celimuge Wu, Shiwen Mao, Yusheng Ji, and Medhi Bennis. 2018. Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning. IEEE Internet of Things Journal 6, 3 (2018), 4005–4018.
[12] Ge Ming Chiu. 2000. The odd-even turn model for adaptive routing. IEEE Transactions on Parallel and Distributed Systems 11, 7 (2000), 729–738. https://doi.org/10.1109/71.877831
[13] Chen Ling Chou and Radu Marculescu. 2008. User-aware dynamic task allocation in networks-on-chip. Proceedings -Design, Automation and Test in Europe, DATE (2008), 1232–1237. https://doi.org/10.1109/DATE.2008.4484847
[14] Chaochao Feng, Zhonghai Lu, Axel Jantsch, Jinwen Li, and Minxuan Zhang. 2010. A reconfigurable fault-tolerant deflection routing algorithm based on reinforcement learning for network-on-chip. In Proceedings of the Third International Workshop on Network on Chip Architectures. 11–16.
[15] Quintin Fettes, Mark Clark, Razvan Bunescu, Avinash Karanth, and Ahmed Louri. 2018. Dynamic voltage and frequency scaling in NoCs with supervised and reinforcement learning techniques. IEEE Trans. Comput. 68, 3 (2018), 375–389.
[16] Ikki Fujiwara and Michihiro Koibuchi. 2013. Mapping Non-trivial Network Topologies Onto Chips. In 2013 IEEE 7th International Symposium on Embedded Multicore Socs. IEEE, 73–78.
[17] Philip K.F. Hölzenspies, Timon D. Ter Braak, Jan Kuper, Gerard J.M. Smit, and Johann M. Hurink. 2010. Run-time Spatial Mapping of Streaming Applications to Heterogeneous Multi-Processor Systems. International Journal of Parallel Programming 38, 1 (2010), 68–83. https://doi.org/10.1007/s10766-009-0120-y
[18] Jingcao Hu and Radu Marculescu. 2004. DyAD: smart routing for networks-on-chip. In Proceedings of the 41st annual Design Automation Conference. 260–263.
[19] Samarth Kaushik, Amit Kumar Singh, and Thambipillai Srikanthan. 2011. Computation and communication aware run-time mapping for NoC-based MPSoC platforms. In 2011 IEEE International SOC Conference. IEEE, 185–190.
[20] Anish Krishnakumar, Samet E Arda, A Alper Goksoy, Sumit K Mandal, Umit Y Ogras, Anderson L Sartor, and Radu Marculescu. 2020. Runtime task scheduling using imitation learning for heterogeneous many-core systems. arXiv preprint arXiv:2007.09361 (2020).
[21] Ting-Ru Lin, Drew Penney, Massoud Pedram, and Lizhong Chen. 2020. A Deep Reinforcement Learning Framework for Architectural Exploration: A Routerless NoC Case Study. In 2020 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 99–110.
[22] Marcelo Mandelli, Alexandre Amory, Luciano Ost, and Fernando Gehm Moraes. 2011. Multi-task dynamic mapping onto NoC-based MPSoCs. Proceedings - SBCCI2011: 24th Symposium on Integrated Circuits and Systems Design (2011), 191–196. https://doi.org/10.1145/2020876.2020920
[23] Maurizio Palesi, Rickard Holsmark, Shashi Kumar, and Vincenzo Catania. 2006. A methodology for design of application specific deadlock-free routing algorithms for NoC systems. CODES+ISSS 2006: Proceedings of the 4th International Conference on Hardware Software Codesign and System Synthesis (2006), 142–147. https://doi.org/10.1145/1176254.1176289

[24] Sunghyun Park, Tushar Krishna, Chia-Hsin Chen, Bhavya Daya, Anantha Chandrakasan, and Li-Shiuan Peh. 2012. Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45nm SOI. In Proceedings of the 49th Annual Design Automation Conference. 398–405.

[25] Wei Quan and Andy D. Pimentel. 2015. A hybrid task mapping algorithm for heterogeneous MPSoCs. ACM Transactions on Embedded Computing Systems 14, 1 (2015), 1–25. https://doi.org/10.1145/2680542

[26] Vijeta Rathore, Vivek Chaturvedi, Amit K Singh, Thambipillai Srikanthan, and Muhammad Shafique. 2019. Life guard: A reinforcement learning-based task mapping strategy for performance-centric aging management. In 2019 56th ACM/IEEE Design Automation Conference (DAC). IEEE, 1–6.

[27] Gamal Abd El-Nasser A Said, Abeer M Mahmoud, and El-Sayed M El-Horbaty. 2014. A comparative study of meta-heuristic algorithms for solving quadratic assignment problem. arXiv preprint arXiv:1407.4863 (2014).

[28] Amit Kumar Singh, Thambipillai Srikanthan, Akash Kumar, and Wu Jigang. 2010. Communication-aware heuristics for run-time task mapping on NoC-based MPSoC platforms. Journal of Systems Architecture 56, 7 (2010), 242–255. https://doi.org/10.1016/j.sysarc.2010.04.007

[29] Amit Kumar Singh, Jigang Wu, Alok Prakash, and Thambipillai Srikanthan. 2009. Efficient heuristics for minimizing communication overhead in NoC-based heterogeneous MPSoC platforms. Proceedings of the International Workshop on Rapid System Prototyping (2009), 55–60. https://doi.org/10.1109/RSP.2009.18

[30] Richard S Sutton and Andrew G Barto. 2018. Reinforcement learning: An introduction. MIT press.

[31] Éric Taillard. 1991. Robust taboo search for the quadratic assignment problem. Parallel computing 17, 4-5 (1991), 443–455.

[32] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2820–2828.

[33] William Thies and Saman Amarasinghe. 2010. An empirical characterization of stream programs and its implications for language and compiler design. In 2010 19th International Conference on Parallel Architectures and Compilation Techniques (PACT). IEEE, 365–376.

[34] Ian H Witten. 1977. An adaptive optimal controller for discrete-time Markov environments. Inf. Control. 34, 4 (1977), 286–295.

[35] Lei Yang, Weichen Liu, Peng Chen, Nan Guan, and Mengquan Li. 2017. Task Mapping on SMART NoC: Contention Matters, Not the Distance. Proceedings - Design Automation Conference Part 12828 (2017). https://doi.org/10.1145/3061639.3062323