MOOS: A Multi-Objective Design Space Exploration and Optimization Framework for NoC Enabled Manycore Systems

ARYAN DESHWAL, NITTHILAN KANAPPAN JAYAKODI, BIRESH KUMAR JOARDAR, JANARDHAN RAO DOPPA, and PARTHA PRATIM PANDE, School of Electrical Engineering and Computer Science, Washington State University

The growing needs of emerging applications has posed significant challenges for the design of optimized manycore systems. Network-on-Chip (NoC) enables the integration of a large number of processing elements (PEs) in a single die. To design optimized manycore systems, we need to establish suitable trade-offs among multiple objectives including power, performance, and thermal. Therefore, we consider multi-objective design space exploration (MO-DSE) problems arising in the design of NoC-enabled manycore systems: placement of PEs and communication links to optimize two or more objectives (e.g., latency, energy, and throughput). Existing algorithms to solve MO-DSE problems suffer from scalability and accuracy challenges as size of the design space and the number of objectives grow. In this paper, we propose a novel framework referred as *Multi-Objective Optimistic Search (MOOS)* that performs adaptive design space exploration using a data-driven model to improve the speed and accuracy of multi-objective design optimization process. We apply MOOS to design both 3D heterogeneous and homogeneous manycore systems using Rodinia, PARSEC, and SPLASH2 benchmark suites. We demonstrate that MOOS improves the speed of finding solutions compared to state-of-the-art methods by up to 13X while uncovering designs that are up to 20% better in terms of NoC. The optimized 3D manycore systems improve the EDP up to 38% when compared to 3D mesh-based designs optimized for the placement of PEs.

CCS Concepts: • **Computing methodologies** → **Machine learning**; • **Hardware** → **Network on chip**;

Additional Key Words and Phrases: Network-on-chip, manycore systems, design optimization, machine learning

ACM Reference format:

Aryan Deshwal, Nitthilan Kanappan Jayakodi, Biresk Kumar Joardar, Janardhan Rao Doppa, and Partha Pratim Pande. 2019. MOOS: A Multi-Objective Design Space Exploration and Optimization Framework for NoC Enabled Manycore Systems. *ACM Trans. Embed. Comput. Syst.* 18, 5s, Article 77 (October 2019), 23 pages. <https://doi.org/10.1145/3358206>

This article appears as part of the ESWEEK-TECS special issue and was presented in the International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2019.

Authors' addresses: A. Deshwal, N. K. Jayakodi, B. K. Joardar, J. R. Doppa, and P. P. Pande, School of Electrical Engineering and Computer Science, Washington State University, P.O. Box 642752, Pullman, Washington, 99164-2752; emails: {aryan.deshwal, n.kannappanjayakodi, biresh.joardar, jana.doppa, pande}@wsu.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Association for Computing Machinery.

1539-9087/2019/10-ART77 \$15.00

<https://doi.org/10.1145/3358206>

1 INTRODUCTION

We are experiencing phenomenal growth in exciting, yet demanding, application areas including deep learning, graph analytics, and scientific computing. To handle the computational needs of these emerging applications, we need to design appropriate manycore systems by understanding their requirements. For example, we can exploit the data parallelism in deep learning applications by using a combination of CPUs and GPUs. Network-on-Chip (NoC) is the communication backbone that enables the design of efficient manycore systems. Therefore, the design and optimization of NoC-enabled manycore systems for given application workloads is a fundamental electronic design automation problem.

To design optimized manycore systems, we need to trade-off multiple objectives including power, performance, and thermal. Therefore, we consider multi-objective design space exploration (MO-DSE) problems arising in the design of NoC-enabled manycore systems, where we need to optimize two or more objectives such as latency, energy, throughput, and temperature. For example, consider the design of heterogeneous manycore systems as shown in Figure 1, where we need to integrate processing elements (PEs) of different types including CPUs, GPUs, and various accelerators using an appropriate NoC to facilitate efficient communication between PEs [18]. The design space is combinatorial in nature, where each candidate design corresponds to a particular placement of the given PEs and communication links. In general, the objectives are conflicting in nature and all of them cannot be optimized simultaneously (e.g., energy and performance). Therefore, we need to find the *Pareto optimal* set of designs. A design is called Pareto optimal if it cannot be improved in any of the objectives without compromising some other objective. Figure 2 shows a high-level illustration of Pareto front for two objectives. MO-DSE problems in NoC design pose a number of challenges: 1) Large design space that grows in size and complexity with increasing system size and diversity in PEs as shown in Figure 1; 2) Hardness and complexity of MO-DSE problem grows with the number of objectives; and 3) Pareto front of designs is non-convex and complex for large number of objectives. Common algorithms to solve MO-DSE problems include AMOSA [2] and NSGA-II [12]. Typically, these algorithms execute many unguided and independent searches, from different starting points, to increase the chance of reaching global optima. However, as the number of objectives in the MO-DSE problem increases, the complexity of the optimization problem grows which exponentially increases the time the algorithm takes to find an acceptable solution. Additionally, these methods do not allow to leverage the training data from simulations to evaluate the quality of designs.

In this paper, motivated by the need for improving scalability and accuracy to solve MO-DSE problems, we propose a novel data-driven algorithm referred as *multi-objective optimistic search* (MOOS). MOOS performs adaptive design space exploration based on the principle of *optimism in the face of uncertainty* [1] to improve the speed and accuracy of design optimization process. The principle of optimism suggests exploring the most favorable region of the design space based on the experience gained from past exploration. MOOS framework allows to synergistically combine the prior knowledge from hardware designers and training data from simulations of designs to improve the accuracy and efficiency of design optimization process. MOOS is an iterative two-stage optimization algorithm. In each iteration, we employ a scalarized objective to select the starting solution for a local search procedure. The parameters of scalarization are chosen adaptively based on a data-driven tree-based model. We perform local search from the selected starting solution and update the tree-based model using the quality of the resulting Pareto set. The data-driven tree-based model not only guides the search towards better solutions, but also reduces the runtime of MOOS to find acceptable solutions. This is a significant departure from the standard MO-DSE algorithms [2, 12] that *do not* perform data-driven design space exploration using an explicit model.

Fig. 1. An example 3D heterogeneous manycore system.

Contributions. The main contributions of this paper are:

- We propose a novel data-driven algorithm called MOOS to improve the speed and accuracy of solving MO-DSE problems arising in NoC-enabled manycore systems design and optimization.
- We apply MOOS to design NoC-enabled 3D manycore systems with both heterogeneous and homogeneous PEs noting that proposed design methodology is applicable for any other NoC. We evaluate performance and thermal-aware MO-DSE scenarios using diverse application benchmark suits including RODINIA, PARSEC, and SPLASH-2.
- We demonstrate that MOOS improves the speed of finding solutions similar to state-of-the-art methods (MOO-STAGE and AMOSA) by upto 13X and uncovers solutions that are upto 20% better in terms of NoC. The optimized NoC-enabled 3D manycore systems improve the EDP up to 38% when compared to a 3D mesh-based design optimized for the placement of PEs.

2 RELATED WORK

Algorithms including AMOSA and NSGA-II do not leverage the knowledge gained from past design space exploration in an explicit manner. In [2], the authors demonstrate that AMOSA can outperform NSGA-II [12] in the number of distinct solutions, time needed, and overall performance. Machine learning methods have been employed to design power-aware NoCs [13]. The basic idea is to predict when to power-gate the channels/routers and when to increase the bandwidth of channels to save power. Similarly, reinforcement learning techniques are employed to develop proactive fault-tolerant mechanisms to optimize energy efficiency and performance of NoCs [37]. These works are orthogonal to the NoC design space exploration problems considered in this paper.

There is a body of literature focusing on solving constrained combinatorial problems (CCPs), especially for the system synthesis [32]. SAT-decoding [26] is the dominant state-of-the-art approach in tackling CCPs. The key idea behind this approach is to use a SAT solver to generate valid solutions in the design space from the genotypes searched over by an evolutionary algorithm. The valid solutions are represented by a set of Pseudo-Boolean (PB) constraints, formulated using binary variables. This approach works very-well in multiple scenarios. One of the key constraints incorporated in our design space is *small-world (SW) network* constraint, which are shown to be very effective to design high-performance and energy-efficient NoCs [30, 36]. To design SW-enabled 3D NoC, we follow the power-law-based connectivity pattern, where the probability of connecting two nodes varies exponentially with distance between them: $p(r) \propto r^{-\alpha}$, where $p(r)$

is the probability of connecting two nodes separated by link length r , and α is the connectivity parameter that dictates the amount of “small-worldness” introduced in the network. Clearly, power law is based on a non-linear function making the small world constraint non-linear. Such non-linear constraints cannot be represented by the PB functions formulation used by the SAT-decoding procedure.

Wavefront-MCTS [15] is a variant of Monte-Carlo Tree Search to solve NoC design optimization problems with multiple objectives. It defines a search space in the form of a Markov Decision Process (MDP) using graph rewriting rules and employs a variant of weighted reward function (referred as Max reward) to evaluate the terminal states. This method has several drawbacks: (a) The branching factor of MDP can be very large to closely approximate the complete design space. This can critically impact the accuracy and speed of optimization; and (b) The scalarized weights employed to design the reward function and addition parameters used to define Max reward can significantly impact the accuracy of Pareto front especially for non-convex Pareto fronts and large number of objectives. In summary, all these methods suffer from scalability and accuracy challenges as the size of the design space and the number of objectives grow.

Towards the goal of addressing some of the above challenges, machine learning (ML) techniques can be used as part of the design optimization framework [19, 20]. Specifically, work in [9–11] adapted a online machine learning (ML) technique called STAGE to optimize the design of NoC for homogeneous manycore systems, and showed that it is much more efficient than simulated annealing (SA) and genetic algorithm (GA) for a single objective function. Recent work [18] developed MOO-STAGE algorithm, a generalization of STAGE to solve multi-objective optimization problems. It learns an evaluation function to intelligently select starting designs towards the goal of improving the accuracy of local search. MOO-STAGE was used to optimize the design of NoC for heterogeneous manycore systems and was shown to be much more efficient than AMOSA. The proposed MOOS algorithm improves over MOO-STAGE in terms of both speed and accuracy of the optimized designs. The key insight is that MOOS performs efficient search guided by a data-driven tree-based model over scalarization parameters (small and simple search space) when compared to MOO-STAGE that performs data-driven search guided by learned evaluation function over input design space (large and complex search space).

Our MOOS framework differs from the existing work on algorithms to optimize multiple *expensive* objective functions including Bayesian Optimization (BO) [14, 28, 31, 34] in the following ways: (a) These methods are only applicable for continuous spaces (i.e., design variables are continuous), whereas our designs are discrete combinatorial structures; and (b) They assume expensive objective function evaluations, whereas we leverage prior knowledge from designers to define analytical objective functions that are significantly cheaper to evaluate. MOOS can be potentially extended to optimize expensive objective functions as well. For example, we can build a surrogate statistical model (e.g., Gaussian Process) for each expensive objective function. We can treat each statistical model as another analytical objective within the MOOS framework. Over iterations of MOOS, the statistical models of these expensive objective functions will become accurate.

3 PROBLEM SETUP AND APPROACH

In this section, we describe our problem setup formally and provide a high-level overview of our algorithmic approach.

3.1 Problem Setup

We consider the multi-objective design space exploration (MO-DSE) problems arising in NoC-enabled manycore systems. Each MO-DSE problem takes the following inputs. We will use the heterogeneous manycore system shown in Figure 1 for illustrating these inputs.

Fig. 2. Illustration of Pareto front obtained by optimizing two objective functions O_1 and O_2 . The set of red points denote the Pareto front (i.e., non-dominating designs). Design B is dominated by design A. Hence, B is not part of the Pareto front.

1. Design Space $\mathcal{D}$. For a fixed system size, we are provided with resources in the form of different types of processing elements (PEs) and communication links. For example, different types of PEs in Figure 1 include CPUs, GPUs, and specialized accelerators. For N PEs and M communication links, we get a fully-specified design space in the form of all possible manycore design configurations. For example, each NoC-enabled manycore design $d \in \mathcal{D}$ corresponds to a specific placement of PEs and communication links. Figure 3 shows an illustration of a candidate design for four PEs and four communication links. The design space grows exponentially with increasing system size (N). Suppose we have T types of PEs with $n_1, n_2, \dots, n_T$ number of PEs for each type such that $N = \sum_{i=1}^T n_i$. The size of the search space for candidate core placements is $\frac{N!}{n_1! \cdot n_2! \cdot \dots \cdot n_T!}$. Each of these candidate core placements can pick a link placement from the space of all possible link placements, whose size is N choose M . Therefore, the total size of the design space is a product of the sizes of both (core placement and link placement) search spaces.

2. Objective Set $\mathcal{O}$. To evaluate each design $d \in \mathcal{D}$, we are given a set of $k > 1$ objectives $\mathcal{O} = \{O_1, O_2, \dots, O_k\}$. Some example objectives include latency, energy, performance, and thermal. We employ analytical objective functions defined using the information obtained by profiling application workloads. Without loss of generality, we assume that the goal is to minimize all the objectives for the sake of technical exposition.

3. Physical Design Constraints $\mathcal{C}$. To come up with practical, feasible designs, we are given a set of physical constraints $\mathcal{C}$. Some examples constraints are as follows: (a) The overall communication network should have a path between any two tiles; and (b) Restricting a router's maximum number of inter-router ports so that no router becomes unrealistically large.

Our goal is to find the Pareto set (i.e., non-dominated set of designs) $\mathcal{D}^*$ from $\mathcal{D}_C \subseteq \mathcal{D}$, where $\mathcal{D}_C$ is the set of designs that satisfy all the physical constraints $\mathcal{C}$. A design d_2 is dominated by design d_1 if $\forall i, O_i(d_1) \leq O_i(d_2)$; and $\exists j, O_j(d_1) < O_j(d_2)$. Figure 2 illustrates the concept of Pareto front for the case of two objective functions. Once the Pareto set $\mathcal{D}^*$ is computed, the designer employs some decision-making criteria (e.g., energy-delay-product from simulations) to select the best design d^* from the Pareto set $\mathcal{D}^*$. We perform the entire optimization process at the *design-time*.

3.2 Overall Approach

Recall that we profile the application workloads to extract the amount of communication between different PEs, which is used to define some of the objective functions (e.g., latency and throughput)

Fig. 3. High-level illustration of MOOS based NoC design optimization for a simple design space with four PEs and four communication links. The core vector and link matrix corresponds to the set of discrete variables representing the location of PEs and communication links.

to evaluate candidate NoC designs. One key challenge is that the computation and communication patterns of the applications can vary over time and should be considered during design optimization process. Our approach takes into account the run-time variations of application workloads as follows. Suppose $WL(t)$ denote the temporal application workload, where t indicate the dependency on time at an appropriate level of granularity. We invoke MOOS algorithm (described in the next section) to solve the MO-DSE problem for each t to obtain the Pareto set $\mathcal{P}_t$. We compute the aggregate set of designs $\cup_t \mathcal{P}_t$ as the resulting solution $\mathcal{P}_{aggregate}$. We pick a NoC design $d^* \in \mathcal{P}_{aggregate}$ that has the best average performance (EDP) across different time steps. Algorithm 1 shows the pseudo-code of our overall NoC design optimization process.

ALGORITHM 1: Temporal Workload-Aware NoC Design Optimization

Input: $WL(t)$ = NoC workload at time t ; $\mathcal{D}$ = design space; $\mathcal{O}$ = set of optimization objectives $\mathcal{C}$ = physical constraints; MAX = maximum number of iterations

```

1: for all  $t=1$  to  $T$  do
2:   Define objectives in  $\mathcal{O}$  that depend on  $WL(t)$ 
3:    $\mathcal{P}_t \leftarrow MOOS(\mathcal{D}, \mathcal{O}, \mathcal{C}, MAX)$ 
4: end for
5:  $\mathcal{P}_{aggregate} \leftarrow \cup_t \mathcal{P}_t$ 
6:  $d^* \leftarrow \arg \min_{d \in \mathcal{P}_{aggregate}} EDP(d)$ 
7: return the best NoC design  $d^*$ 

```

Extracting Temporal Workload. We capture the temporal variations in workload behavior $WL(t)$ directly from the detailed simulations using Gem5, the simulator employed in our experiments. Gem5 allows the creation of checkpoints at any point of the code after specific intervals. *Checkpoints* in Gem5 are break points in the simulation that do not stop the actual execution of the application, but are useful to obtain statistics for a small section/time. Moreover, by modifying the code inside Garnet of Gem5, it is possible to obtain detailed information about individual messages, e.g., source, destination, cycle at which it was generated, etc. Therefore, by utilizing both

these features in Gem5, we obtain the temporal workload information $WL(t)$ which is then used during the optimization process.

4 MOOS FRAMEWORK

In this section, we propose a novel algorithm referred as *Multi-Objective Optimistic Search (MOOS)* to efficiently solve large and complex MO-DSE problem. We first provide an overview of the MOOS algorithm followed by the details of its key elements.

4.1 Overview of MOOS Algorithm

MOOS is an iterative algorithm that performs data-driven adaptive design space exploration based on the principle of *optimism in the face of uncertainty*. The adaptivity in MOOS is enabled by a tree-based model $\mathcal{M}$ over the space of scalarization parameters λ . Algorithm 2 provides the high-level steps of MOOS and Figure 4 shows key algorithmic steps performed in each iteration of MOOS. We initialize the global Pareto set $\mathcal{P}_{global}$ and tree-based model $\mathcal{M}$ before running MOOS. In each iteration, we perform the following sequence of steps. We select the most promising design d_{start} from $\mathcal{P}_{global}$ as the starting state to execute a local search procedure (e.g., greedy search) that has the maximum potential to improve the global Pareto set. We employ the Pareto Hyper-Volume (PHV) [39] to evaluate the quality of a candidate Pareto set. PHV is defined as the size of the objective space that is dominated by the solutions in Pareto set.

A scalarized objective SO parameterized by λ is used to perform this selection (line 8) and λ is chosen by the tree-based model $\mathcal{M}$ (lines 4-6). Subsequently, we perform local search from the selected starting design d_{start} (line 9) and update the tree-based model $\mathcal{M}$ using the new evaluation data (line 12): λ and quality of the updated global Pareto set after local search. Alternatively, when applicable, we can use the decision-making criteria of the designer (e.g., EDP from simulations) to provide a stronger training signal for the evaluation of λ . For example, we can select the design from local Pareto set $\mathcal{P}$ that minimizes the scalarized objective and use the corresponding EDP as the evaluation of λ . The data-driven tree-based model $\mathcal{M}$ not only guides the search towards better solutions, but also reduces the runtime of MOOS to find acceptable solutions. This is a significant

ALGORITHM 2: MOOS Algorithm

Input: $\mathcal{D}$ = design space; $\mathcal{O}$ = set of optimization objectives; $\mathcal{C}$ = physical constraints; MAX = maximum number of iterations

- 1: Initialize $\mathcal{P}_{global}$ with some initial design
 - 2: Initialize the tree-based model $\mathcal{M}$ over the space of λ values
 - 3: **while** convergence or MAX iterations **do**
 - 4: Select the best leaf node n^* from $\mathcal{M}$ for expansion
 - 5: Create three child nodes n_{left} , n_{center} , and n_{right} corresponding to equal partitions of the hyper-rectangle represented by node n^* along the longest dimension
 - 6: Suppose λ_{left} and λ_{right} be the centers of hyper-rectangles represented by n_{left} and n_{right}
 - 7: **for all** $\lambda \in \{\lambda_{left}, \lambda_{right}\}$ **do**
 - 8: $d_{start} \leftarrow \text{SELECT-START-DESIGN}(\mathcal{O}, \lambda, \mathcal{P}_{global})$
 - 9: $\mathcal{P}_{local} \leftarrow \text{LOCAL-SEARCH}(\mathcal{O}, d_{start}, \mathcal{C}, MAX)$
 - 10: $\mathcal{P}_{global} \leftarrow$ non-dominating designs from $\mathcal{P}_{global} \cup \mathcal{P}_{local}$
 - 11: $PHV_{global} \leftarrow PHV(\mathcal{O}, \mathcal{P}_{global})$
 - 12: Update tree-based model $\mathcal{M}$ using the new evaluation data (λ, PHV_{global})
 - 13: **end for**
 - 14: **end while**
 - 15: **return** global Pareto set $\mathcal{P}_{global}$
-

Fig. 4. Overview of MOOS algorithm.

Fig. 5. Illustration of optimistic search via tree based model. Each cell corresponds to a partition of the space of λ parameters. After initial partition, the leaf cell with the best PHV denoted in green (higher PHV is better when minimizing all the $k=2$ objectives) is partitioned into $W = 3$ children cells for evaluation. This process is repeated till convergence.

departure from the standard MO-DSE algorithms that *do not* perform data-driven design space exploration using an explicit model.

4.2 Optimistic Search via Tree-based Model

In this section, we provide the details of tree-based model and the principle behind optimistic search [1] to select scalarization parameter values λ in each iteration of MOOS algorithm. The principle of optimism suggests exploring the most favorable region of the design space based on the experience gained from past exploration.

As an *illustrative example*, let us consider two-objective DSE problem with two scalarization parameters λ_1 and λ_2 . Figure 5 shows a graphical illustration of tree-based model and optimistic search over scalarization parameters. The domain $X_\lambda = [0, 1] \times [0, 1]$. Let us consider a hierarchical partition of this domain where the cells are of the form $\{\lambda \in X_\lambda : b_{1,1} \leq \lambda_1 \leq b_{1,2}, b_{2,1} \leq \lambda_2 \leq b_{2,2}\}$. Such a cell will be denoted by the notation $[[b_{1,1}, b_{1,2}], [b_{2,1}, b_{2,2}]]$. Then a hierarchical partition with $W=3$ starts with root node $M_{0,1} = [[0, 1], [0, 1]]$. This can be further sub-divided into three children cells at $h=1$ given by $M_{1,1} = [[0, 1/3], [0, 1]]$, $M_{1,2} = [[1/3, 2/3], [0, 1]]$

and $\mathcal{M}_{1,3} = [[2/3, 1], [0, 1]]$. $\mathcal{M}_{1,2}$ can be further partitioned into $\mathcal{M}_{2,1} = [[1/3, 2/3], [0, 1/3]]$, $\mathcal{M}_{2,2} = [[1/3, 2/3], [1/3, 2/3]]$ and $\mathcal{M}_{2,3} = [[1/3, 2/3], [2/3, 1]]$ and so on. The center of the cell $[[b_{1,1}, b_{1,2}], [b_{2,1}, b_{2,2}]]$ is the point $((b_{1,1} + b_{1,2})/2, (b_{2,1} + b_{2,2})/2)$.

Selection of λ for Evaluation. Given a tree-based model $\mathcal{M} = \{\mathcal{M}_h\}$, we use the principle of *optimism in the face of uncertainty* [1] to select the cell (node) and the corresponding center λ for evaluation. In this context, the term optimism refers to the fact that we employ a strategy to expand at each round tree cells that may contain optimum λ values. Specifically, for each depth i , we expand a leaf node n^* from the set of leaves (nodes whose children are not in $\mathcal{M} = \{\mathcal{M}_h\}$) at depth i if the evaluation $\mathcal{E}(n^*)$ is less than the value of all leaf nodes at any depth j less than i . This is the key part of the MOOS algorithm, where we take an optimistic approach. We split the selected leaf node n^* into three ($K=3$) child nodes n_{left} , n_{center} , and n_{right} corresponding to equal partitions of the hyper-rectangle represented by node n^* along the longest dimension. We select the centers of n_{left} and n_{right} , namely, λ_{left} and λ_{right} , for next evaluation. Recall that center of n_{center} is same as the center of the parent node n^* and therefore, we already have its evaluation.

4.3 Selecting Starting Designs via Scalarization

Given a global Pareto set $\mathcal{P}_{global}$, our goal is to select the most promising design from $\mathcal{P}_{global}$ as the starting design for local search procedure. We employ a scalarized objective SO to perform this selection using the scalarization parameters $\lambda = (\lambda_1, \lambda_2, \dots, \lambda_k)$ picked by our tree-based model $\mathcal{M}$. We select the design from $\mathcal{P}_{global}$ that minimizes the scalarized objective SO (Algorithm 3, line 1). Linear scalarization is the most simplest approach: $SO(\lambda, d) = \sum_{i=1}^k \lambda_i \cdot O_i(d)$. However, linear scalarization is known to perform poorly (confirmed in our experiments) due to its inability to explore non-convex regions of the Pareto front [8]. Therefore, we propose to explore *Chebychev scalarization* in this work noting that *any* other scalarization method can be used to similar effect.

ALGORITHM 3: Selection of Starting Design for Local Search

Input: $\mathcal{O}$ = set of optimization objectives; λ = scalarization parameters; $\mathcal{P}_{global}$ = global Pareto set

1: $d_{start} = \arg \min_{d \in \mathcal{P}_{global}} \text{SCALARIZED-OBJECTIVE}(\mathcal{O}, \lambda, d)$

2: **return** starting design d_{start}

Chebychev Scalarization. The chebychev scalarization objective [29] with parameters λ is defined as $SO(\lambda, d) = \max_{i=1, \dots, k} \lambda_i \cdot (O_i(d) - z_i^*)$ where z_i^* is a reference point usually defined as the minimum possible value of the objective O_i . Without loss of generality, we assume $z_i^* = 0$ for our purpose. Intuitively, chebychev scalarization picks the objective which is worse off in the current solution and tries to optimize that objective. This non-linear operation in chebychev scalarization helps in uncovering all designs even in the presence of a non-convex Pareto set.

4.4 Local Search Procedure

The goal of a local search algorithm (e.g., greedy search) is to traverse through a sequence of neighboring states (designs) starting from a given design d_{start} to find a solution that minimizes the set of objectives $\mathcal{O}$. To accommodate multiple objectives, we employ the Pareto Hyper-Volume (PHV) [39] heuristic to evaluate the quality of a set of solutions. In short, the PHV is the size of the objective space that is dominated by a set of solutions. To compute the PHV, we employ a fast and scalable PHV algorithm called Hypervolume by Slicing Objectives [39]. It employs the divide-and-conquer principle to achieve efficiency: it repeatedly divides the PHV computation into simpler

problems with fewer objectives and aggregates the solutions of simpler problems to compute the total Hypervolume.

In this work, we employ a simple greedy search with the objective of minimizing PHV with respect to the set of objectives O as the local search procedure (Algorithm 4). However, it should be noted that *any* other local search method can be used to similar effect. Starting from the initial state d_{start} , we find the best neighboring state from $N(d_{curr})$ that improves the PHV heuristic at each greedy search step (Algorithm 4, line 3). We prune the neighboring states that violate at least one physical design constraint from C . In our running example of designing 3D heterogeneous systems, a neighboring state is where exactly one planar link is repositioned, or two tiles are swapped (both irrespective of layers). If this best neighboring state improves the PHV value, we add this state to the Pareto set $\mathcal{P}_{local}$ while ensuring that all designs in $\mathcal{P}_{local}$ are non-dominated (Algorithm 4, lines 4-5). This is repeated until the best neighboring state does not improve the PHV value, at which point, we return the local optima set $\mathcal{P}_{local}$.

ALGORITHM 4: Local Search Procedure

Input: O = set of optimization objectives; d_{start} = starting design; C = physical constraints; MAX = maximum iterations

```

1:  $\mathcal{P}_{local} \leftarrow \{d_{start}\}$ ; and  $d_{curr} \leftarrow d_{start}$ 
2: while convergence or  $MAX$  iterations do
3:    $d_{next} = \operatorname{argmax}_{d \in N(d_{curr}) \cup \{d_{curr}\}} PHV(O, \mathcal{P}_{local} \cup \{d\})$ 
4:   if  $PHV(O, \mathcal{P}_{local} \cup \{d_{next}\}) > PHV(O, \mathcal{P}_{local})$  then
5:      $\mathcal{P}_{local} \leftarrow$  non-dominating designs from  $\mathcal{P}_{local} \cup \{d_{next}\}$ 
6:   end if
7:    $d_{curr} \leftarrow d_{next}$ 
8: end while
9: return local Pareto set  $\mathcal{P}_{local}$ 
  
```

Incorporating Design Constraints. One example of a constraint in our design space is the consideration of small-world NoC design. For a given value of small-world parameter α , we know the distribution of links of different lengths. Recall that design space exploration mainly occurs in the local search part of our MOOS algorithm. We check whether a design violates a constraint or not during each search step of the local search procedure. At each search step within the local search procedure, the set of neighbourhood designs of a current design is defined as the set of all possible single core swapping and set of all possible single link replacement for the same link length (strictly enforces the small-world design constraint). We continually perturb the current design (giving equal probability to core swapping and link replacement) over this neighbourhood to find the best design in terms of scalarized objective which also satisfies the small-world constraints. We can enforce other design constraints (e.g., maximum and average size of the router) in a similar manner.

5 EXPERIMENTS AND RESULTS

We discuss our experiments and results for the design of both NoC-enabled heterogeneous and homogeneous manycore platforms using the proposed MOOS algorithm.

5.1 Heterogeneous Manycore Systems Design

5.1.1 Experimental Setup.

Design Space of 3D Heterogeneous Manycore Systems. We consider NoC-enabled 3D Heterogeneous manycore systems [16–18] as a first instance of MO-DSE problem to evaluate the performance of the proposed MOOS algorithm. We employ a 64-tile system comprised of 8 CPUs,

16 Last-Level-Caches (LLCs), and 40 GPUs. The tiles have been distributed in a $4 \times 4 \times 4$ 3D system with the heat sink at the bottom. The tiles are interconnected via a custom NoC consisting of both vertical and planar links. It should be noted that the proposed optimization methodology can be applied to any composition of tile types and layer/stack size. Vertical links connecting the planar layers are implemented using TSVs. Each candidate design in the design space corresponds to a specific placement of PEs (CPUs, GPUs, and LLCs) and input planar links.

Physical Design Constraints. The list of physical constraints employed during design optimization includes: 1) Candidate designs must have at least one path between any two PEs for communication; and 2) The number of links provided as input are same as that of a standard 3D Mesh to account for area constraints and to make sure that the optimized design has same overhead as 3D Mesh.

Benchmark Applications. We employ eight different applications from the Rodinia benchmark suite [5]: Back Propagation (BP), Breadth-First search (BFS), Gaussian Elimination (GAU), Hot Spot (HS), k -Nearest Neighbor (k -NN), LU Decomposition (LUD), Needleman-Wunsch (NW), and Path Finder (PF).

Simulation Setup. We employ Gem5-GPU [33], a heterogeneous full-system simulator to obtain network and processor level information. We modify the Garnet network model within Gem5-GPU to implement candidate NoC-enabled 3D heterogeneous systems for performance evaluation. The CPUs and GPUs are based on standard x86 and NVIDIA Maxwell architectures respectively. Each CPU and GPU have a private 32Kb L1 cache. A 4MB L2 Cache is distributed among the LLCs equally. The operating clock frequencies for CPUs and GPUs are 2.5GHz and 0.7GHz respectively. The candidate designs employ the deadlock-free ALASH [27] routing. We employ GPUWattch [22], McPAT [23] and 3D-ICE [35] tools to obtain the power and thermal profiles. All analyses have been performed on an Intel Xeon CPU E5-2620 @ 2GHz machine with 16 GB RAM running CentOS 6.

Optimization Objectives. NoC design objectives often vary widely depending on application. For instance, power and performance are important for mobile platforms, while security-critical applications can focus on more reliability at the cost of power. Here, we evaluate MOOS to design an efficient NoC-enabled 3D heterogeneous manycore system employing few common objectives: latency, throughput, energy, and thermal. The analytical expressions of these objectives are taken from [6, 18] and are described below for the sake of completeness.

Cycle-accurate simulations provide more accurate evaluation of designs. However, they tend to be extremely slow and can take several hours to a few days to finish their execution. Therefore, we have used generic models for evaluating each of these objectives. Moreover, as discussed by the authors in [18, 25], these models can accurately compare designs in terms of their relative values, which is *sufficient* for design optimization. Hence, we can still determine which designs are better than others relatively without performing detailed simulations. Finally, we perform detailed simulations to get the absolute objective values for the subset of designs (very small) in the Pareto Front resulting from the optimization process.

1. Latency Objective. Latency is the primary concern for CPUs as higher latencies cause CPUs to stall, leading to poor performance. For C CPUs and M LLCs, we model the average CPU-LLC latency as shown below [10]:

$$Latency = \frac{1}{C * M} \sum_{i=1}^C \sum_{j=1}^M (r \cdot h_{ij} + d_{ij}) \cdot f_{ij},$$

where r is the number of router stages, h_{ij} is the number of hops from CPU i to LLC j , d_{ij} is the total link delay, and f_{ij} is the frequency of communication between core i and core j .

2. Throughput Objective. Unlike CPUs, GPUs rely on high-throughput memory accesses to enable high data parallelism. Load balancing, considering *Mean* and *Standard deviation (STD)* of link utilization as objectives, helps improve throughput [18]. The expected utilization U_k of link k is obtained from below:

$$U_k = \sum_{i=1}^R \sum_{j=1}^R f_{ij} \cdot p_{ijk},$$

where R is the total number of tiles and p_{ijk} indicates whether a planar/vertical link k is used to communicate between core i and core j respectively. The mean (μ) and standard deviation (σ) of the link utilization can be computed as shown below:

$$\mu = \frac{1}{L} \sum_{k=1}^L U_k, \quad \sigma = \sqrt{\frac{1}{L} \sum_{k=1}^L (U_k - \mu)^2}$$

3. Energy Objective. Designers often have a fixed energy budget for optimum performance. Therefore, it is necessary to design high-performance systems within given energy budget. The total network energy is the sum of router and link energy

$$E = \sum_{i=1}^N \sum_{j=1}^N f_{ij} \cdot \left(\sum_{k=1}^L p_{ijk} \cdot d_k \cdot E_{link} + \sum_{k=1}^R r_{ijk} \cdot E_r \cdot P_k \right),$$

where N is the number of cores, R and L denote the total number of routers and links respectively, d_k represents the physical length of link k , E_{link} and E_r denote the average link energy per unit length and router logic energy per port respectively and P_k denotes the number of ports at router k . Both p_{ijk} and r_{ijk} are binary variables that indicate whether a link/router k is used to communicate between core i and core j respectively. f_{ij} represents the frequency of communication between core i and core j respectively.

4. Thermal Objective. To accurately estimate the temperature of a core, we use the fast approximation model presented in [35]. A 3D manycore system can be divided into N single-tile stacks, each with K layers, where N is the number of tiles on a single layer and K is the total number of layers. The temperature of a core due to the vertical heat flow within a single-tile stack n located at layer k from the sink ($T_{n,k}$) is given by:

$$T_{n,k} = \sum_{i=1}^k \left(P_{n,i} \sum_{j=1}^i R_j \right) + R_b \sum_{i=1}^k P_{n,i}$$

Here, $P_{n,i}$ is the power consumption of the core i layers away from the sink in single-tile stack n , R_j is the vertical thermal resistance, and R_b is the thermal resistance of the base layer on which the dies are placed. The values of R_j and R_b are obtained using 3D-ICE [7]. The horizontal heat flow is represented through the maximum temperature difference in the same layer k ($\Delta T(k)$) [7]:

$$\Delta T(k) = \max_n T_{n,k} - \min_n T_{n,k}$$

The overall thermal prediction model includes both vertical and horizontal heat flow equations. Following [7], we use T as our comparative temperature metric for any given NoC-enabled 3D manycore system design:

$$T = \left(\max_{n,k} T_{n,k} \right) \left(\max_k \Delta T(k) \right)$$

Baseline Algorithm. We consider AMOSA [2] and MOO-STAGE [18] as the baseline approaches to solve the MO-DSE problems undertaken here. AMOSA is a simulated annealing based algorithm in which the optimization process is guided with the help of an archive of solutions. However, AMOSA relies on a hill-climbing process and needs to be annealed sufficiently slowly to obtain near-optimum solution set [2]. MOO-STAGE is another class of data-driven algorithms to solve MO-DSE problems. It repeatedly performs a hill-climbing search by learning an evaluation function to select good starting states based on the training data from past searches.

MO-DSE Problem Scenarios. To compare the baselines with the proposed MOOS algorithm, we consider four MO-DSE scenarios as follows. **2-OBJ**: Mean and STD of link utilization; **3-OBJ**: Latency objective in addition to Mean and STD of link utilizations; **4-OBJ (Perf)**: Energy objective in addition to three objectives in 3 OBJ case; and **4-OBJ (Therm)**: Thermal objective in addition to three objectives in 3 OBJ case.

5.1.2 Performance-Aware Optimization. We compare the performance of MOOS algorithm with both AMOSA and MOO-STAGE for optimizing NoC-enabled 3D heterogeneous architectures for all the eight applications in performance-aware setting. We present the results averaged over five runs.

Metrics. We compare MOOS and the baseline algorithms in terms of two metrics.

1. Speedup factor of MOOS over baseline is defined as $\frac{T_{Baseline}}{T_{MOOS}}$, where $T_{Baseline}$ is the convergence time of the baseline algorithm and T_{MOOS} is the time taken by MOOS to reach the best solution uncovered by baseline.

2. Percentage quality gain of MOOS over baseline when comparing the solutions uncovered by MOOS and baseline at the end of maximum time-bound. For performance-aware setting, we measure the quality of a candidate Pareto set using the NoC energy-delay-product (EDP) of the best design in the Pareto set—smaller EDP means better design.

Anytime performance of MOOS and baselines in terms of EDP. Figure 6 shows the normalized EDP of the best design uncovered by MOOS, AMOSA, and MOO-STAGE as a function of time (in hours) for the Back Propagation application as a representative example noting that results are similar for other applications. EDP combines two metrics: network latency and energy. Here, latency is a measure of performance. EDP captures the trade-off between performance and power for any design. An NoC can have low latency (i.e. high performance) by burning more power which is undesirable. Considering either of these parameters by itself will not capture the trade-off involved. Therefore, EDP is a suitable metric for comparison for this set of experiments.

Here, normalization is done w.r.t the final design from MOOS, which has the lowest EDP. We can see that MOOS uncovers designs with lower EDP values significantly faster than AMOSA and MOO-STAGE. Additionally, MOOS finds designs with better EDP than both AMOSA and MOO-STAGE after the convergence of all algorithms (see also % quality gain shown in Table 2). Table 1 shows the quality of Pareto fronts in terms of PHV metric obtained by AMOSA and MOO-STAGE normalized with respect to MOOS (i.e., normalized PHV of MOOS is 1). We show the average PHV of all benchmarks for different number of objectives. These results show the advantage of data-driven adaptive design space exploration of MOOS guided by the tree-based model.

Speedup factor and percentage quality gain of MOOS. Tables 2 and 3 show the percentage quality gain and speed-up factor of MOOS over both the baselines. As the number of objectives increase, the speedup factor and percentage quality gain of MOOS increases showing the benefits of MOOS in improving the scalability and accuracy of solving large and complex MO-DSE problems.

(a) 2-OBJ

(b) 3-OBJ

(c) 4-OBJ (Perf)

Fig. 6. Anytime performance of best designs from AMOSA, MOOSTAGE and MOOS for Back propagation traffic: Normalized EDP vs. time in hours.

Table 1. Average PHV of Pareto Fronts Generated by AMOSA and MOO-STAGE Normalized w.r.t MOOS Across All Benchmarks

Number of objectives	AMOSA	MOO-STAGE
2	0.968	0.982
3	0.942	0.961
4	0.916	0.940

Table 2. % Quality Gain Achieved by MOOS Compared to AMOSA and MOO-STAGE at Convergence Time for Performance Setting

Application	AMOSA			MOO-STAGE		
	2-OBJ	3-OBJ	4-OBJ	2-OBJ	3-OBJ	4-OBJ
BP	4.0	7.6	20.2	3.0	3.7	13.3
BFS	3.1	8.4	18.3	2.0	5.1	11.8
GAU	4.4	7.7	19.1	3.8	4.2	15.9
HS	3.2	7.4	17.8	2.9	6.8	15.0
k-NN	3.8	9.8	16.6	3.8	3.8	14.4
LUD	2.9	10.9	13.9	2.0	2.7	12.0
NW	3.7	7.9	20.5	2.2	2.6	15.1
PF	3.8	7.5	17.5	3.1	4.5	13.4
Average	3.61	8.40	17.98	2.85	4.17	13.86

Table 3. Speedup Achieved by MOOS Compared to AMOSA and MOO-STAGE in Performance Setting

Application	AMOSA			MOO-STAGE		
	2-OBJ	3-OBJ	4-OBJ	2-OBJ	3-OBJ	4-OBJ
BP	2	3.3	10.0	1.0	1.2	3.5
BFS	1.4	2.9	10.6	1.2	1.7	3.7
GAU	1.7	3.8	11.6	1.5	1.9	3.1
HS	1.4	2.6	10.9	1.1	2.4	3.8
k-NN	1.5	3.4	11.1	1.0	1.4	2.9
LUD	1.2	3.2	12.3	1.0	1.8	3.7
NW	1.2	3.6	13.1	1.0	1.3	3.9
PF	1.9	3.6	12.0	1.0	1.9	3.3
Average	1.53	3.3	11.45	1.10	1.70	3.48

Optimized NoC-enabled 3D Heterogeneous Design vs. 3D-Mesh. Figure 8(a) shows the normalized EDP of NoC-enabled 3D heterogeneous design optimized by MOOS for all eight applications. We can see that the optimized NoC designs are significantly better in terms of EDP when compared to MESH: upto 37.7% EDP improvement and 31.7% improvement on an average.

Analysis of optimized designs from MOOS and AMOSA. We compare the best designs (Opt-Perf) obtained by MOOS and AMOSA in terms of the following physical properties of designs to understand their qualitative differences: (a) Average hop count normalized w.r.t the optimized design from MOOS; (b) Distribution of links in planar layers; and (c) Distribution of load (traffic) among communication links in terms of mean and standard deviation of their utilization. Table 4 shows the results for (a) and (b), and Figure 7 shows the results for (c).

We make the following observations: 1) Opt-Perf from MOOS has lower hop count, which means messages have to travel shorter distance leading to better performance; 2) Opt-Perf from MOOS has high-density of links in the middle two layers, which also corresponds to the placement of LLCs (as shown in Figure 9). This results in better performance under many-to-few traffic due to improved diversity in communication paths enabled by the high number of links close to LLCs; 3) Opt-Perf from MOOS has the best mean (μ) and standard deviation (σ) of link utilizations. Note that lower utilization does not mean change in the number of packets that are communicated.

Table 4. Comparison of Optimized Designs from MOOS and AMOSA: Normalized Average Hopcount (Avg hc) and Number of Links in Each Planar Layer l1, l2, l3, l4

Metric	Design	2-OBJ	3-OBJ	4-OBJ
Avg hc	Opt-Perf AMOSA	1.09	1.11	1.14
	Opt-Perf MOOS	1	1	1
Links	Opt-Perf AMOSA	14, 33, 37, 12	17, 26, 34, 19	21, 21, 35, 19
	Opt-Perf MOOS	19, 41, 33, 3	20, 34, 30, 12	11, 37, 36, 12

Fig. 7. Distribution of load (traffic) among links in 3D Mesh vs. Opt-Perf w/ MOOS and AMOSA for Back Propagation application.

Instead, it reduces the number of heavily congested links by redistributing traffic flows. This reduces the amount of contention for heavily utilized links. As a result, there is less network congestion and throughput is improved.

5.1.3 Thermal-Aware Design Optimization. As high power density and temperature hotspots are detrimental to the performance of 3D manycore systems, we consider the thermal-aware optimization setting, namely, 4-OBJ(Thermal), with MOOS algorithm. In the case of thermal-performance joint optimization, it is difficult to capture the trade-off in a single metric. Custom metrics, e.g., Temperature-Delay Product (the equivalent of EDP that may captures the trade-off between temperature and performance) are not standard and are not commonly seen in prior works as well. Therefore, in this case, we define best design as the one that exhibits lowest EDP within a specific temperature threshold. The temperature threshold is decided on a per application basis and is set at 5% higher than the best temperature observed for any design d in the design space. Any design that has an on-chip temperature below this threshold is a *valid solution*. The best design is then picked from the set of valid designs based on performance.

Performance vs. Thermal Optimization. Figure 8 compares the optimized designs from performance-aware and thermal-aware settings, i.e., 4-OBJ(Perf) and 4-OBJ(Thermal). Figure 8(a) and (b) shows the normalized EDP (w.r.t 3D Mesh) and the peak temperature of optimized designs for all eight applications respectively. There is clear trade-off between normalized EDP and temperature for the resulting designs: Opt-Perf produces designs with better EDP and higher temperature; and Opt-Thermal produces designs with lower temperature and higher EDP. The EDP of designs from Opt-Perf are 11% better on an average when compared to the ones from Opt-Thermal. Temperature of designs from Opt-Thermal is 21°C lower on an average when compared to the designs from Opt-Perf.

(a) Normalized EDP results

(b) Temperature results

Fig. 8. Normalized EDP w.r.t 3D Mesh and peak temperature results for Opt-Perf and Opt-Thermal optimized with MOOS.

Dist. of tiles for Opt-Perf and Opt-Thermal. Figure 9 shows the tile placement of optimized NoC-enabled 3D heterogeneous designs, namely, Opt-Perf and Opt-Thermal, in performance-aware and thermal-aware settings respectively. We also show the peak temperature of each optimized design. We make the following observations: 1) In Opt-Perf, most of the LLCs are in the middle two layers, as a result they can access vertical links in both directions for efficient communication under many-to-few traffic. This results in on an average 11% improvement in EDP over designs from Opt-Thermal as shown in Figure 8(a); 2) In Opt-Thermal, the most power-consuming cores (GPUs) are placed closer to the sink to improve temperature. As a result, LLCs are pushed to the top two layers mostly, which results in on an average 21°C improvement in temperature over designs from Opt-Perf as shown in Figure 8(b).

5.2 Homogeneous Manycore Systems Design

5.2.1 Experimental Setup.

Design Space of 3D Homogeneous Manycore Systems. To further evaluate MOOS, we consider NoC-enabled 3D homogeneous manycore systems as another instance of MO-DSE problem.

Fig. 9. Distribution of tiles in Opt-Perf and Opt-Thermal for Back Propagation application. Layer 1 is farthest from sink. We also show the peak temperature of each optimized design.

We employ a TSV-based 3D manycore system consisting of 64 cores and 64 NoC routers equally partitioned in four layers. In each die, 16 cores are placed in a regular grid. We employ the design space of small-world NoCs that is shown to be beneficial in terms of energy-efficiency and reliability [10].

Simulation Setup. We employ Gem5 [4], a homogeneous manycore system simulator to obtain detailed processor and network information. CPUs are based on standard x86 architecture. The memory system is comprised of private 64 KB L1 instruction and data caches, one shared 16 MB L2 cache (256 KB distributed L2 per core). The rest of the setup is similar to the description in heterogeneous section.

Benchmark Applications. We employ four different applications from SPLASH-2 benchmark suite (FFT, RADIX, LU, and WATER) [40] and four applications from PARSEC [3]: VIPS, DEDUP, FLUIDANIMATE (FLUID), and CANNEAL (CAN). These benchmarks vary in characteristics from computation intensive to communication intensive, thus, are of particular interest for our evaluation.

Optimization Objectives. To design efficient NoC-enabled 3D homogeneous manycore systems, we employ the following objectives: latency, energy, and thermal. These objectives are same as the ones defined in the heterogeneous section.

Physical Design Constraints. We employ the same constraints as listed in the heterogeneous manycore systems design setup.

5.2.2 Performance-Aware Optimization. In this setting, we employ latency and energy objectives in our NoC design optimization process. Figure 10 shows the normalized EDP of NoC-enabled 3D homogeneous designs optimized by MOOS for all eight applications. The EDP values are normalized with respect to an optimized 3D mesh based design. Similar to the heterogeneous case, the designs optimized by MOOS show significant improvement in EDP when compared to 3D mesh: upto 28% EDP improvement and 20% improvement on an average across all the benchmarks.

5.2.3 Thermal-Aware Optimization. For thermal-aware optimization setting, we consider latency and temperature objectives. Figure 11 shows the temperatures of NoC-enabled 3D

Fig. 10. Normalized EDP w.r.t 3D Mesh for performance and thermal-aware optimized homogeneous designs.

Fig. 11. Temperature (in °C) of optimized homogeneous designs generated by MOOS for performance and thermal-aware settings.

homogeneous designs optimized with MOOS in both performance-aware and thermal-aware settings. As expected, temperatures of Opt-Thermal is 10°C lower on an average when compared to Opt-Perf. The EDP of designs from Opt-Perf are 10% better on an average when compared to the ones from Opt-Thermal.

5.3 Scalability Analysis

In this section, we present the scalability results comparing MOOS, MOO-STAGE, and AMOSA for designing homogeneous manycore systems by increasing the number of cores.

Experimental Setup. The Gem5/Gem5-GPU simulator employed in our experiments does not support system size beyond 64 cores currently. Therefore, it is not possible to run the conventional SPLASH, PARSEC or Rodinia benchmarks on this simulator to get detailed statistics for system sizes higher than 64 (e.g., 128 and 256). Instead, we use a cycle-accurate NoC simulator from [38] considering 128 and 256 cores. In this experimental study, we employ seven MCSL benchmarks [24] with widely varying traffic patterns: FFT, SPEC fpp (FPP), H.264 decoder, Robot control (ROBOT),

Table 5. Percentage Quality Gain Achieved by MOOS Compared to AMOSA and MOO-STAGE for Systems with 64, 128, and 256 Cores

System size	AMOSA			MOO-STAGE		
	64	128	256	64	128	256
FFT	5.42	6.91	8.57	1.10	1.71	3.96
FPP	4.25	4.88	7.11	1.21	1.33	3.12
H264	7.11	7.32	7.83	1.17	1.90	3.87
RS_ENC	7.13	6.72	7.13	1.11	1.23	2.91
RS_DEC	5.20	5.48	6.85	1.04	1.41	2.58
ROBOT	5.17	5.98	8.59	1.08	1.67	3.77
SPARSE	5.89	6.90	7.30	1.14	2.10	3.88
Average	5.73	6.31	7.60	1.115	1.62	3.44

Table 6. Speed up Factors Achieved by MOOS Compared to AMOSA and MOO-STAGE for Systems with 64, 128, and 256 Cores

System size	AMOSA			MOO-STAGE		
	64	128	256	64	128	256
FFT	2.20	3.10	4.20	1.00	1.10	2.20
FPP	2.10	2.40	3.80	1.00	1.00	2.00
H264	1.80	2.70	4.10	1.00	1.10	2.00
RS_ENC	1.40	2.10	3.50	1.00	1.20	2.10
RS_DEC	1.70	2.20	4.00	1.00	1.10	2.10
ROBOT	1.90	2.90	3.70	1.00	1.20	2.00
SPARSE	2.40	3.50	4.40	1.00	1.10	2.00
Average	1.92	2.70	3.95	1.00	1.10	2.05

Reed-Solomon encoder and decoder (RSenc and RSdec), and sparse matrix solver (SPARSE). These benchmarks enable us to explore NoC design with higher system sizes.

Results and Discussion. The results for all seven benchmarks are shown in Table 5 and Table 6 for homogeneous systems with varying sizes (64, 128, and 256). We can see that the effectiveness of MOOS (percentage quality gain and speedup factor) grows as a function of system size when compared with AMOSA and MOO-STAGE. The main reason for this behavior is that the design space for system with 128 and 256 cores is significantly larger than the system with 64 cores. Hence, MOO algorithms such as AMOSA that perform unguided design space exploration take significantly large amount of time to converge and still produce lower-quality solutions than MOOS. MOO-STAGE performs better than AMOSA, but still produces lower-quality solutions than MOOS. On the other hand, MOOS quickly identifies most promising regions of the design space using the learned tree-based model over scalarization parameters. These results indicates that MOOS is more scalable as system size grows.

Interestingly, we note that the gains with these new benchmarks are relatively lower in some cases, than the conventional SPLASH, PARSEC benchmark suites. The main reason for this behavior is that in several of the MCSL benchmarks, e.g. ROBOT, RSenc etc., a large number of cores do not communicate with each other. Due to the extreme sparsity in traffic patterns, the important NoC features often show marginal to no variation at all for different designs. In other words,

such designs show identical characteristics and are virtually indistinguishable to the learner. As a result, the learner encounters difficulties in learning a good tree-based model over scalarization parameters in short amount of time due to lack of diversity in the training data. These challenges have been encountered and analyzed in prior work, e.g., [21] as well.

6 CONCLUSIONS

The design of network-on-chip enabled manycore systems involve searching a large combinatorial design space to optimize multiple objectives. Prior methods including conventional heuristic-based approaches scale poorly as the design space and number of objectives increases. We proposed a multi-objective design space exploration and optimization framework called MOOS to improve the scalability and accuracy of optimization process when compared to state-of-the-art methods. MOOS can adaptively identify and explore better areas of the design space in a data-driven manner. We demonstrated that MOOS improves the speed of finding solutions similar to state-of-the-art methods (MOO-STAGE and AMOSA) by upto 13X and uncovers solutions that are upto 20% better in terms of NoC. The optimized NoC-enabled 3D manycore systems improve the EDP up to 38% when compared to a 3D mesh-based design optimized for the placement of PEs.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewer#2 for the detailed feedback that helped in improving the paper. This work was supported in part by the National Science Foundation under Grant OAC-1910213 and Grant IIS-1845922, and in part by the U.S. Army Research Office under Grant W911NF-17-1-0485 and Grant W911NF-19-1-0162.

REFERENCES

- [1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* 47, 2–3 (2002), 235–256.
- [2] Sanghamitra Bandyopadhyay, Sriparna Saha, Ujjwal Maulik, and Kalyanmoy Deb. 2008. A simulated annealing-based multiobjective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation (TEC)* 12, 3 (2008), 269–283.
- [3] Christian Bienia. 2011. *Benchmarking Modern Multiprocessors*. Ph.D. Dissertation. Princeton University.
- [4] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, et al. 2011. The Gem5 simulator. *ACM SIGARCH Computer Architecture News* 39, 2 (2011), 1–7.
- [5] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W. Sheaffer, Sang-Ha Lee, and Kevin Skadron. 2009. Rodinia: A benchmark suite for heterogeneous computing. In *IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 44–54.
- [6] Wonje Choi, Karthi Duraisamy, Ryan Gary Kim, Janardhan Rao Doppa, Partha Pratim Pande, Diana Marculescu, and Radu Marculescu. 2018. On-chip communication network for efficient training of deep convolutional networks on heterogeneous manycore systems. *IEEE Transactions on Computers (TC)* 67, 5 (2018), 672–686.
- [7] Jason Cong, Jie Wei, and Yan Zhang. 2004. A thermal-driven floorplanning algorithm for 3D ICs. In *Proceedings of the 2004 IEEE/ACM International Conference on Computer-aided Design*. IEEE Computer Society, 306–313.
- [8] Indraneel Das and John E. Dennis. 1997. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization* 14, 1 (1997), 63–69.
- [9] Sourav Das, Janardhan Rao Doppa, Daehyun Kim, Partha Pratim Pande, and Krishnendu Chakrabarty. 2015. Optimizing 3D NoC design for energy efficiency: A machine learning approach. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 705–712.
- [10] Sourav Das, Janardhan Rao Doppa, Partha Pratim Pande, and Krishnendu Chakrabarty. 2017. Design-space exploration and optimization of an energy-efficient and reliable 3-D small-world network-on-chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 36, 5 (2017), 719–732.
- [11] Sourav Das, Janardhan Rao Doppa, Partha Pratim Pande, and Krishnendu Chakrabarty. 2017. Monolithic 3D-enabled high performance and energy efficient network-on-chip. In *Proceedings of IEEE International Conference on Computer Design (ICCD)*. 233–240.

- [12] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation (TEC)* 6, 2 (2002), 182–197.
- [13] Dominic DiTomaso, Ashif Sikder, Avinash Kodi, and Ahmed Louri. 2017. Machine learning enabled power-aware network-on-chip design. In *Proceedings of the IEEE/ACM Conference on Design, Automation & Test in Europe (DATE)*. 1354–1359.
- [14] Daniel Hernández-Lobato, Jose Hernandez-Lobato, Amar Shah, and Ryan Adams. 2016. Predictive entropy search for multi-objective Bayesian optimization. In *Proceedings of International Conference on Machine Learning (ICML)*. 1492–1501.
- [15] Yong Hu, Daniel Mueller-Gritschneider, and Ulf Schlichtmann. 2018. Wavefront-MCTS: Multi-objective design space exploration of NoC architectures based on Monte Carlo tree search. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. ACM, 102:1, 102:8.
- [16] Biresh Kumar Joardar, Janardhan Rao Doppa, Partha Pratim Pande, Diana Marculescu, and Radu Marculescu. 2018. Hybrid on-chip communication architectures for heterogeneous manycore systems. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*. ACM, 62.
- [17] Biresh Kumar Joardar, Ryan Gary Kim, Janardhan Rao Doppa, and Partha Pratim Pande. 2019. Design and optimization of heterogeneous manycore systems enabled by emerging interconnect technologies: Promises and challenges. In *Proceedings of IEEE/ACM International Conference on Design, Automation & Test in Europe Conference & Exhibition, (DATE)*. 138–143.
- [18] Biresh Kumar Joardar, Ryan Gary Kim, Janardhan Rao Doppa, Partha Pratim Pande, Diana Marculescu, and Radu Marculescu. 2018. Learning-based application-agnostic 3D NoC design for heterogeneous manycore systems. *IEEE Trans. Comput.* 68, 6 (2018), 852–866.
- [19] Ryan Gary Kim, Janardhan Rao Doppa, and Partha Pratim Pande. 2018. Machine learning for design space exploration and optimization of manycore systems. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*. IEEE, 48.
- [20] Ryan Gary Kim, Janardhan Rao Doppa, Partha Pratim Pande, Diana Marculescu, and Radu Marculescu. 2018. Machine learning and manycore systems design: A serendipitous symbiosis. *IEEE Computer* 51, 7 (2018), 66–77.
- [21] Dongjin Lee, Sourav Das, Dae Hyun Kim, Janardhan Rao Doppa, and Partha Pratim Pande. 2018. Design space exploration of 3D network-on-chip: A sensitivity-based optimization approach. *JETC* 14, 3 (2018), 32:1–32:26.
- [22] Jingwen Leng, Tayler Hetherington, Ahmed ElTantawy, Syed Gilani, Nam Sung Kim, Tor M. Aamodt, and Vijay Janapa Reddi. 2013. GPUWatch: Enabling energy optimizations in GPGPUs. In *ACM SIGARCH Computer Architecture News*, Vol. 41. ACM, 487–498.
- [23] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. 2009. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. ACM, 469–480.
- [24] Weichen Liu, Jiang Xu, Xiaowen Wu, Yaoyao Ye, Xuan Wang, Wei Zhang, Mahdi Nikdast, and Zhehui Wang. 2011. A NoC traffic suite based on real applications. In *2011 IEEE Computer Society Annual Symposium on VLSI*. IEEE, 66–71.
- [25] Xiaoxiao Liu, Wei Wen, Xuehai Qian, Hai Li, and Yiran Chen. 2018. Neu-NoC: A high-efficient interconnection network for accelerated neuromorphic systems. In *ASP-DAC*. IEEE, 141–146.
- [26] Martin Lukasiewicz, Michael Glaß, Christian Haubelt, and Jurgen Teich. 2007. Sat-decoding in evolutionary algorithms for discrete constrained optimization problems. In *2007 IEEE Congress on Evolutionary Computation*. IEEE, 935–942.
- [27] Olav Lysne, Tor Skoe, S.-A. Reinemo, and Ingebjørg Theiss. 2006. Layered routing in irregular networks. *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 17, 1 (2006).
- [28] Giovanni Mariani, Gianluca Palermo, Vittorio Zaccaria, and Cristina Silvano. 2012. OSCAR: An optimization methodology exploiting spatial correlation in multicore design spaces. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)* 31, 5 (2012), 740–753.
- [29] Hirotaka Nakayama, Yeboon Yun, and Min Yoon. 2009. *Sequential Approximate Multiobjective Optimization using Computational Intelligence*. Springer Science & Business Media.
- [30] Umit Y. Ogras and Radu Marculescu. 2006. “It’s a small world after all”: NoC performance optimization via long-range link insertion. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 14, 7 (2006), 693–706.
- [31] Berkin Ozisikyilmaz, Gokhan Memik, and Alok Choudhary. 2008. Efficient system design space exploration using machine learning techniques. In *Proceedings of the 45th Annual Design Automation Conference (DAC)*. ACM, 966–969.
- [32] Jacopo Panerati, Donatella Sciuto, and Giovanni Beltrame. 2017. Optimization strategies in design space exploration. In *Handbook of Hardware/Software Codesign*. 189–216.
- [33] Jason Power, Joel Hestness, Marc S. Orr, Mark D. Hill, and David A. Wood. 2014. Gem5-GPU: A heterogeneous CPU-GPU simulator. *IEEE Computer Architecture Letters* 14, 1 (2014), 34–36.

- [34] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando De Freitas. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proc. IEEE* 104, 1 (2015), 148–175.
- [35] Arvind Sridhar, Alessandro Vincenzi, Martino Ruggiero, Thomas Brunschweiler, and David Atienza. 2010. 3D-ICE: Fast compact transient thermal modeling for 3D ICs with inter-tier liquid cooling. In *Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 463–470.
- [36] Christof Teuscher. 2007. Nature-inspired interconnects for self-assembled large-scale network-on-chip designs. *Chaos: An Interdisciplinary Journal of Nonlinear Science* 17, 2 (2007), 026106.
- [37] Ke Wang, Ahmed Louri, Avinash Karanth, and Razvan Bunescu. 2019. High-performance, energy-efficient, fault-tolerant network-on-chip design using reinforcement learnin. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1166–1171.
- [38] Paul Wettin, Ryan Kim, Jacob Murray, Xinmin Yu, Partha P. Pande, Amlan Ganguly, and Deukhyoun Heoamlan. 2014. Design space exploration for wireless NoCs incorporating irregular network routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 33, 11 (2014), 1732–1745.
- [39] Lyndon While, Philip Hingston, Luigi Barone, and Simon Huband. 2006. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation (TEC)* 10, 1 (2006), 29–38.
- [40] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. 1995. The SPLASH-2 programs: Characterization and methodological considerations. *ACM SIGARCH Computer Architecture News* 23, 2 (1995), 24–36.

Received May 2019; revised June 2019; accepted July 2019