

Article

# Edge-Based Heuristics for Optimizing Shortcut-Augmented Topologies for HPC Interconnects

Kazi Ahmed Asif Fuad <sup>†</sup> , Kai Zeng <sup>†</sup> and Lizhong Chen <sup>\*†</sup> 

Kelley Engineering Center, School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, OR 97331, USA

\* Correspondence: chenliz@oregonstate.edu; Tel.: +1-541-737-3317

† These authors contributed equally to this work.

**Abstract:** Interconnection network topology is critical for the overall performance of HPC systems. While many regular and irregular topologies have been proposed in the past, recent work has shown the promise of shortcut-augmented topologies that offer multi-fold reduction in network diameter and hop count over conventional topologies. However, the large number of possible shortcuts creates an enormous design space for this new type of topology, and existing approaches are extremely slow and do not find shortcuts that are globally optimal. In this paper, we propose an efficient heuristic approach, called *EdgeCut*, which generates high-quality shortcut-augmented topologies. EdgeCut can identify more globally useful shortcuts by making its considerations from the perspective of edges instead of vertices. An additional implementation is proposed that approximates the costly all-pair shortest paths calculation, thereby further speeding up the scheme. Quantitative comparisons over prior work show that the proposed approach achieves a 1982× reduction in search time while generating better or equivalent topologies in 94.9% of the evaluated cases.

**Keywords:** high-performance computing system; interconnection network; topology; shortcut; design space exploration; heuristic search; shortest path; hop count



**Citation:** Fuad, K.A.A.; Zeng, K.; Chen, L. Edge-Based Heuristics for Optimizing Shortcut-Augmented Topologies for HPC Interconnects. *Electronics* **2022**, *11*, 2778. <https://doi.org/10.3390/electronics11172778>

Academic Editor: José L. Abellán

Received: 7 July 2022

Accepted: 31 August 2022

Published: 3 September 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

High-performance computing (HPC) systems are essential in order to run a variety of large-scale applications in multiple domains such as bio-informatics, astronomical analysis, nuclear simulations, financial services, etc., as well as large machine learning models applied to numerous use cases. As the backbone of HPC systems, interconnection networks are responsible for connecting up to hundreds or thousands of compute nodes (a compute node may, in turn, consist of hundreds to thousands of processing cores) [1] by providing fast and low-cost communication in the order of microseconds [2]. A primary factor in dictating the performance of interconnection networks is topology, which specifies the structure that is used to connect compute nodes. To achieve high interconnected performance, it is critical to design network topologies that have small diameters and low hop counts.

Prior works have proposed a number of regular and irregular topologies including rings, meshes, tori, hypercubes, fat trees, Clos, Butterfly, and Dragonfly. Many of them have been deployed in practical HPC systems. Interestingly, despite the seemingly mature development of topologies, recent work has demonstrated the large potential of a very different class of topologies, which we refer to as *shortcut-augmented topologies*. Such a topology starts with a base topology (e.g., a ring) and adds a series of shortcuts on top of that. Surprisingly, with careful selection, shortcut-augmented topologies are able to reduce both network diameter and average shortest hop count by multiple folds, compared with existing widely used regular and irregular topologies [3].

While this is promising, a major roadblock of further improving this class of topologies is the enormous design space that is formed by the combinations of possible shortcuts.

As an example, for a 64-node network, there are around  $2 \times 10^3$  possible edges. If 128 edges are selected, there are over  $4 \times 10^{205}$  combinations! To make things worse, this design space grows super-exponentially as the network size increases, thus greatly exceeding the capabilities of exhaustive methods. Conventional methods of design space searching, such as simulated annealing (SA), would be extremely slow. This calls for novel, efficient heuristic approaches that exploit unique characteristics of the shortcut selection problem.

A straightforward heuristic is to add shortcuts randomly (subject to available free ports in the switches). With a large number of repetitions, a good shortcut-augmented topology may be found for a small network size. However, given the rapid increase in design space, this method quickly becomes insufficient for larger networks. Several papers have pointed out that adding random shortcuts can enhance the performance of their proposed network topologies [4–7], but these works do not directly propose approaches that can generate shortcuts more effectively. The state-of-the-art heuristic [3] is to consider the usefulness of shortcuts when adding shortcuts for a given vertex. Specifically, for each vertex  $v$ , the method examines a set of random nodes that are connected to  $v$ , and selects the top  $y$  (say 3) nodes that have the longest shortest paths from  $v$ . It then adds  $y$  shortcuts from  $v$ —one to each of the  $y$  nodes. We refer to this method as *vertex-based random shortcut* (VRS) approach. While this improves the quality of the generated topologies, our analysis reveals that VRS often adds shortcuts that are locally useful at a vertex but are not much use globally. This is because the  $y$  longest shortest paths at a vertex may not necessarily be considered as long paths in the entire network. Consequently, VRS requires more shortcuts to be added, which leads to additional costs of links and higher-degree switches.

To address this issue, this paper proposes *EdgeCut*, an effective heuristic approach for generating high-quality shortcut-augmented topologies. The main novelty is to identify more globally useful shortcuts by thinking from the perspective of edges, rather than from the perspective of vertices in prior work. We propose two variations of EdgeCut. EdgeCut-Full considers the performance impact on all node pairs after a shortcut is added. It produces the best topologies but needs to update all pairs' shortest paths after each addition, thus resulting in longer search time. EdgeCut-Lite approximates this function, leading to  $11 \times$  reduction over EdgeCut-Full in search time while still generating satisfying topologies. Evaluation results show that, compared with simulated annealing, the proposed EdgeCut reduces the search time by  $1982 \times$  and generating better or equivalent diameter in 94.9% of the test network topologies and sizes. Compared with VRS, EdgeCut reduces the network diameter by 55.1% while being slightly faster. These results highlight the effectiveness of the proposed approach.

The rest of the paper is organized as follows. Section 2 provides more background on HPC interconnection networks and shortcut-augmented topologies. Section 3 describes details of the proposed EdgeCut approach for identifying high-quality shortcuts. Section 4 presents evaluation methodology and results, and Section 5 includes further discussions. Finally, Section 6 concludes the paper.

## 2. Related Work

### 2.1. HPC Topologies

The topology of HPC interconnection networks is a very active research direction. While many topologies have been proposed in the past, new topologies are continually being proposed due to new challenges in latency, costs, scalability, reliability, etc., that are associated with ever-growing HPC systems. Only limited research has been conducted on shortcut-augmented topologies, leaving many opportunities for further improvement.

In *direct* topologies such as tori, meshes, and hypercubes, every switch is connected to a compute node, whereas in *indirect* topologies such as fat trees, Clos, and Butterfly, only the input and output switches at the edge of a network are associated with computer nodes, and packets sent from computer nodes are forwarded indirectly through middle-stage switches before reaching their destination [8]. Both direct and indirect topologies have been used in practice, e.g., 3D and 5D torus networks are used in Cray Gemini [9] and IBM

BlueGene/Q, respectively [10], and Dragonfly networks [11] with virtual routers are used in Cray Cascades [12]. In a sense, various direct and indirect topologies differ in how they trade-off among degree, diameter, and hop count [3].

Variations of regular topologies that result in irregular or ad hoc designs have also been proposed. For example, the Jellyfish topology [6] utilizes random graphs to develop high-capacity networks that allow incremental expansion on a daily basis to large-scale data centers, at a higher cost of cabling. Slim Fly [4,5] approximates the optimal diameter, which results in lower latency, cost, and energy consumption while sustaining high bisection bandwidth. However, it is not suitable for gradual size expansion due to limited flexibility in the small design space. Distributed Loop Networks (DLN) add chordal edges or shortcuts to a simple ring topology to reduce diameter while maintaining low degree distribution. By adding shortcuts in a less regular manner than being evenly spaced, DLN can achieve more efficient designs, e.g., the diameter of a 36-vertex ring can be reduced from 18 to 9 by adding only five shortcuts [13,14].

Another related line of topology research stems from the famous *small-world phenomenon*, first proposed by [15], that demonstrates that people living in a country constitute societies of a network with only short path lengths. Later, Watts and Strogatz characterized the small-world phenomenon into the Watts–Strogatz (WS) model [16], which allows networks to be generated with short average distance and large clustering coefficient [17,18]. The model uses a few additional long edges to reduce the diameter in random graphs for social networks and Internet topologies [3,18]. Since then, researchers have been exploring the small-world phenomenon in computer networks [19–21]. In particular, to exploit the small-world effect in HPC, [3] proposes several methods that add random shortcuts to a base topology, the best of which is the vertex-based random shortcut (VRS) method that is mentioned in Section 1. Although shortcuts are selected optimally at each local vertex, they are not necessarily the most useful shortcuts to add globally. This deficiency is addressed by our proposed approach.

## 2.2. Design Space Exploration

The design space of shortcut-augmented topologies is enormous. This is not only because of the large number of possible shortcut candidates (especially for networks with high-radix switches), but also because of the huge number of combinations of the shortcut candidates. There are three typical ways of exploring design space. The first one is exhaustive search, which is impractical in this problem. The second one is general-purpose search algorithms, such as ant colony algorithm and simulated annealing [22]. These algorithms may be able to find the optimal or near-optimal solutions but usually require extremely long search time for large design space. The third one is heuristic approaches that leverage problem-specific characteristics to enable approximate but fast searches. In this paper, we aim to demonstrate that, for the problem of shortcut-augmented topologies, it is possible to design heuristic approaches that can find comparable solutions of general-purpose search algorithms but take only a tiny fraction of their time.

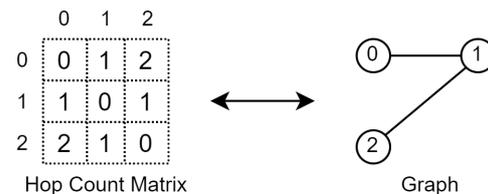
## 3. Proposed Approach

In this section, we describe the details of the proposed EdgeCut approach (implementation and instructions on running the proposed approach are available at <https://github.com/OSU-STARLAB/EdgeCut> (accessed on 1 September 2022)). We start by introducing some notations and definitions, and then present two versions of EdgeCut, namely EdgeCut-Full and EdgeCut-Lite, that offer different trade-offs between efficiency and effectiveness.

### 3.1. Definitions, Notations, and Assumptions

An interconnection network topology for  $N$  compute nodes can be abstracted as a graph with  $N$  vertices. The hop count between two nodes is the length of the path between the corresponding two vertices in the graph. Without additional information, a typical way

to estimate the hop count is to use the shortest path length (SPL) between two vertices. From this, we can define an  $N \times N$  hop count matrix, where each entry  $(i, j)$  corresponds to the SPL of two nodes  $i$  and  $j$ . As illustrated in the example in Figure 1, the shortest path between node 0 and node 1 is *one* hop, and the shortest path between node 0 and node 2 is *two* hops. As a result, entry  $(0,1)$  in the hop count matrix has a value of one and entry  $(0,2)$  has a value of two. Entries in the hop-count matrix are initialized to  $+\infty$ . Every switch has a degree cap of  $d$ . We assume that  $d$  is large enough to enable at least one valid topology that fully connects the  $N$  nodes.



**Figure 1.** Example of mapping between graph and hop count matrix.

### 3.2. EdgeCut-Full (ECF)

The problem with vertex-based random shortcut (VRS) is that it strictly requires every vertex to add a fixed number of shortcuts. Consequently, even if the shortcuts are optimal locally at a vertex, they may not be optimal globally. This also indicates that VRS uses more shortcuts to achieve a better topology than is necessary. To address this issue fundamentally, the proposed EdgeCut considers the pool of all possible shortcut candidates directly, and allows flexibility at a specific node as long as the degree cap is not violated.

The first instantiation, EdgeCut-Full (ECF), evaluates the impact of adding a potential shortcut that is more comprehensive by calculating the SPL between all node pairs if that shortcut is added. The rationale behind all-pair shortest path calculation is that adding a single shortcut may potentially affect all nodes. Algorithm 1 shows the pseudo code of ECF. Similar to other shortcut-augmented topologies, ECF starts with a base topology (such as a ring, mesh, torus, or any existing topology) and gradually adds random shortcuts based on their usefulness on reducing the network for low latency. From the base topology node connections, the hop count matrix and degree of each nodes are initialized, and the current all-pair average shortest path length (ASPL) is calculated. Then, the algorithm generates  $t$  random edges from the pool of shortcut candidates. For each edge, it re-calculates ASPL assuming that the edge is added, and updates the current best edge if ASPL is reduced.

---

#### Algorithm 1 EdgeCut-Full (ECF) for adding shortcuts

---

```

1: Initialize hop_count, degree, aspl from base network topology
2: while (added_edge_count < cap_added_edge_count) do
3:   Randomly generate  $t$  edges (under degree constraint)
4:   min_aspl = aspl
5:   for each edge( $x, y$ ) in  $t$  edges do
6:     Calculate aspl_updated
7:     if (aspl_updated < min_aspl) then
8:       min_aspl = aspl_updated
9:       best_edge = ( $x, y$ )
10:    end if
11:  end for
12:  Use best_edge to update hop_count, degree[ $x$ ], and degree[ $y$ ]
13:  added_edge_count += 1
14: end while

```

---

Once all the  $t$  edges are considered, the best edge with the lowest ASPL is added to the network. This shortcut addition is continued until the cap of allowed shortcuts is reached.

In terms of computation complexity, the *while* loop runs for  $c$  iterations assuming that up to  $c$  shortcuts can be added. To add an edge, the algorithm searches for  $t$  random edges, and each requires the recalculation of ASPL (Line 6). Since only one edge is added, the update can be done by enumerating any two points  $u$  and  $v$  with the update  $distance[u, v] = \min(distance[u, v], distance[u, x] + 1 + distance[y, v])$ , which takes  $O(N^2)$ . Similarly, updating *hop\_count* on Line 12 also takes  $O(N^2)$ . Putting things together, the time complexity of ECF is  $O(c(tN^2 + N^2))$  which is  $O(ctN^2)$ . Note that the hop count matrix of the base topology is known beforehand, so Line 1 includes only the time to copy the hop count matrix locally, not recalculating all-pair shortest paths for the base topology.

### 3.3. EdgeCut-Lite (ECL)

EdgeCut-Full estimates the impact of adding a candidate shortcut accurately but is also computationally expensive. The dominating component comes from updating ASPL in the inner loop, which takes  $O(N^2)$  each time. To alleviate the complexity issue of ECF, we propose EdgeCut-Lite (ECL) that simplifies this estimation. To select a shortcut to add, we still generate and evaluate among  $t$  random edges. For each random edge  $(x, y)$ , we focus on how much hop count can be reduced between node  $x$  and node  $y$  if the edge is added. Since the edge directly connects  $x$  and  $y$  to have a hop count of one, the reduced hop count is equivalent to the previous hop count between  $x$  and  $y$  (minus one). This hop count value can be retrieved from the hop count matrix with just  $O(1)$  time. The random edge with the greatest hop count reduction is selected. The pseudo code for ECL is given in Algorithm 2. Lines 4–10 correspond to the above process. As an example, consider three candidate random edges for node pairs  $(2, 6)$ ,  $(3, 8)$ , and  $(5, 10)$  where the corresponding nodes have an SPL of 4, 8, and 6 respectively. In this case, ECL chooses the edge for  $(3, 8)$  as it has the largest SPL; an additional edge between node 3 and 8 would reduce their shortest distance to 1, resulting in a saving of 7 hops. At the end of the *for* loop, the hop count matrix is updated with the selected edge.

---

#### Algorithm 2 EdgeCut-Light (ECL) for adding shortcuts

---

```

1: Initialize hop_count, degree, aspl from base network topology
2: while (added_edge_count < cap_added_edge_count) do
3:   Randomly generate  $t$  edges (under degree constraint)
4:   max_reduced_hops = 0
5:   for each edge  $(x, y)$  in  $t$  edges do
6:     if (hop_count[ $x$ ][ $y$ ] > max_reduced_hops) then
7:       max_reduced_hops = hop_count[ $x$ ][ $y$ ]
8:       best_edge =  $(x, y)$ 
9:     end if
10:  end for
11:  Use best_edge to update hop_count, degree[ $x$ ], and degree[ $y$ ]
12:  added_edge_count += 1
13: end while

```

---

Following a similar complexity analysis to ECF, the complexity of each *while* loop in ECL consists of the *for* loop which is now simply  $O(t)$  and the hop count update which is  $O(N^2)$ . Overall, the time complexity of ECL is  $O(c(t + N^2))$ .

## 4. Result and Analysis

### 4.1. Methodology

The proposed EdgeCut-Full (ECF) and EdgeCut-Lite (ECL) are evaluated quantitatively against simulated annealing (SA) and vertex-based random shortcut (VRS) algorithms. Multiple topologies including ring, mesh, and torus are used as the base topology to assess the efficacy of the algorithms in adding shortcuts. All the algorithms are implemented in Python 3 and Numpy and executed on the same CPU to ensure fair comparison. In our ex-

periments, in addition to  $N$  that denotes the number of nodes in the network, we sometimes also use  $n = \sqrt{N}$  to better reveal scalability trends. Different network sizes are evaluated including  $N = 16, 25, 36, \dots, 256$  nodes, which corresponds to  $n$  of 4, 5, 6, ..., 16. We use a switch degree cap of 10 and the limit of  $2n$  allowable shortcuts for the main evaluation in this section. More sensitivity studies on different degree cap values and shortcut limits are presented in the Discussion section.

#### 4.2. Computation Time

Computation time (i.e., search time) required for SA, VRS, ECF, and ECL to reach a stable solution is measured by the running time on an Intel(R) Core(TM) i9 9900 CPU @3.1 GHz. Table 1 compares the search time for different network sizes under the four schemes. For clarity, only the results for using torus as the base topology are shown, as the ring/mesh have very similar numbers. As can be seen, SA takes the longest time for each size, and the search time increases rapidly as the network size increases. This is not scalable and can be costly to repeat for design space exploration if any of the network parameters change. VRS is much faster due to its simplicity in heuristic but has poor performance, as shown shortly. In comparison, the proposed ECF and ECL are  $154\times$  and  $1982\times$  faster than SA, respectively (and with better generated topologies as shown later). ECL is  $11\times$  faster than ECF and is also slighter faster VRS on average.

**Table 1.** Computation time (in seconds) required for different sizes of torus networks.

Network Size ( $\sqrt{N}$ )	SA	VRS	ECF	ECL
4	2.927	0.005	0.048	0.004
5	10.21	0.015	0.151	0.014
6	29.21	0.035	0.381	0.034
7	69.94	0.074	0.817	0.074
8	151.83	0.144	1.577	0.149
9	292.0	0.255	2.829	0.259
10	543.3	0.457	4.962	0.436
11	952.8	0.697	7.871	0.710
12	1603.2	1.095	12.13	1.084
13	2531.9	1.637	17.75	1.644
14	3978.8	2.363	26.53	2.386
15	6003.7	3.353	36.38	3.358
16	8931.5	4.586	50.91	4.505

#### 4.3. Reduction in Diameter

Diameter measures the longest shortest path in a network and is an important metric that relates to a number of considerations such as latency, cost, reliability [23]. Table 2 compares the impact on reducing diameter when shortcuts are added for the four schemes.

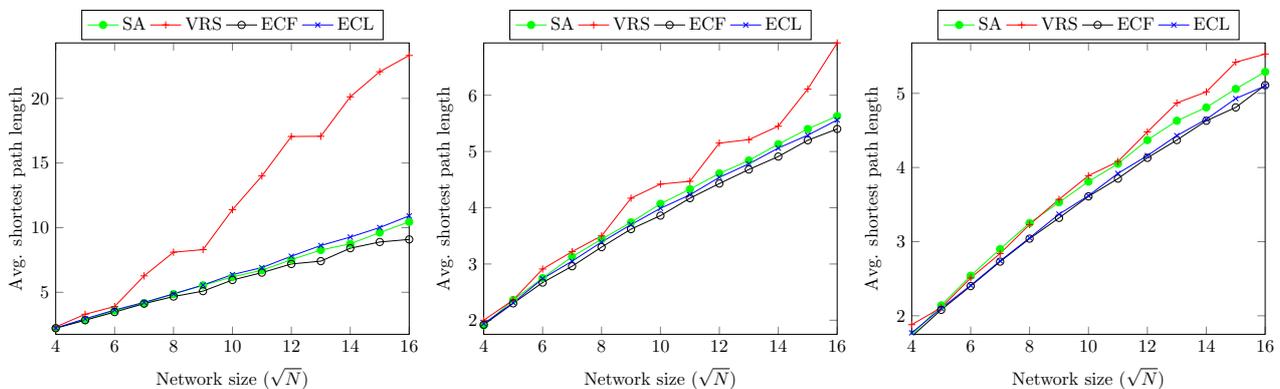
As can be seen from the table, the proposed ECL, while being the fastest among the four, also achieves very low diameter. Specifically, ECL produces better or equal diameter in 37 out of 39 test sizes on three base topologies (94.9%) compared with SA. This is significant given that SA is designed to achieve near-optimal diameter at the cost of long search time. The proposed ECF optimizes the network diameter further than SA (and with much faster search than SA). For example, with a ring base topology of  $16 \times 16 = 256$  nodes, the diameter of 19 in ECF is better than the 22 in ECL, and is also lower than the 24 in SA. In contrast, the existing vertex-based heuristic VRS has the largest diameter, e.g., 49 in the above example, reducing only about 61% from the base diameter, whereas ECF reduces 85% from the base. Comparing between VRS and ECL, ECL reduces the diameter by 55.1% relatively. Across the three base topologies, improvement of different schemes depends on the ratio of the number of added shortcuts to the number of base edges. For the given  $2n$  shortcuts, ring with  $N = n^2$  base edges benefits more than mesh with  $2n(n - 1)$ , and torus with  $2n^2$  base edges benefits the least.

**Table 2.** Diameter for simulated annealing (SA), vertex-based random shortcut (VRS), EdgeCut-Full (ECF), and EdgeCut-Lite (ECL). “S” denotes shortcut limit, and “Base” denotes base diameter.

$\sqrt{N}$	S	Ring					Mesh					Torus				
		Base	SA	VRS	ECF	ECL	Base	SA	VRS	ECF	ECL	Base	SA	VRS	ECF	ECL
4	8	8	4	5	4	4	6	3	4	3	3	4	3	4	3	3
5	10	12	5	6	5	6	8	4	4	4	4	4	4	4	3	4
6	12	18	7	8	6	7	14	5	7	5	6	6	5	5	4	4
7	14	24	9	13	8	8	12	6	6	5	6	6	5	6	5	5
8	16	32	9	16	10	9	14	6	6	6	7	8	6	7	5	5
9	18	40	11	19	10	11	14	8	9	7	7	8	5	7	6	6
10	20	50	13	29	12	13	18	8	9	8	7	10	7	7	6	6
11	22	60	14	34	14	15	20	9	8	8	8	10	7	8	7	7
12	24	72	15	34	14	15	22	9	10	8	8	12	8	9	7	8
13	26	84	18	21	15	17	24	9	10	9	9	12	9	9	8	8
14	28	98	18	58	19	17	26	10	11	9	10	14	9	9	8	8
15	30	112	22	52	19	20	28	11	11	10	11	14	9	11	8	9
16	32	128	24	49	19	22	30	11	13	11	11	16	10	10	9	9

4.4. Average Shortest Path Length

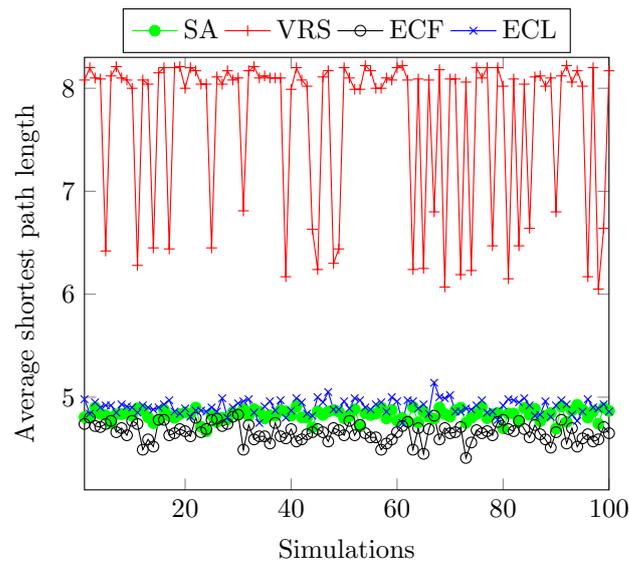
While diameter captures boundary cases, the average shortest path length (ASPL) provides insights on the average cases for network latency. Figure 2 plots the ASPL for different sizes of ring, mesh, and torus networks when shortcuts are added. The proposed EdgeCut-Full has the lowest ASPL in all the evaluated sizes and topologies. The proposed EdgeCut-Lite has slightly higher ASPL than EdgeCut-Full, but is still very close to SA. Meanwhile, VRS performs significantly worse, with a wider gap for larger network sizes, indicating that the heuristic in VRS does not explore the full potential of shortcuts.



**Figure 2.** Comparison of average shortest path length (ASPL) for different networks and schemes: Ring as base topology (left), Mesh as base topology (middle), and Torus as base topology (right). As the networks scale up, proposed EdgeCut networks have lower latency compared to VRS and SA.

4.5. Randomness

All the compared algorithms exhibit randomness, as part of the algorithms involves randomly generated edges. To examine this aspect in more detail, we plot the ASPL over 100 independent runs of the algorithms for adding shortcuts to a 64-node ring network. Figure 3 demonstrates that SA, ECF, and ECL not only generate lower ASPL than VRS but also produce stable results constantly. In comparison, VRS generates good results only 25% of the time.

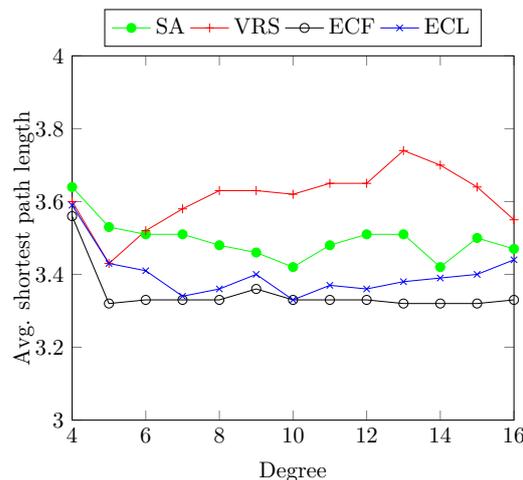


**Figure 3.** Average shortest path length (ASPL) of over 100 runs of SA, VRS, ECF, and ECL. The proposed ECF can generate topology with the lowest ASPL even with randomness.

**5. Discussion**

*5.1. Effect on Degree Limit*

A per-switch degree cap of 10 is used in the above experiments. To gain more insight on how the four schemes may perform under different degree limits, we have evaluated a 64-node mesh network when the degree limit is varied. The results are plotted in Figure 4. Although the specific improvement varies, it is evident that the proposed ECF and ECL have lower ASPL than SA and VRS for all degree limits. ECF is slightly better than ECL, as expected, due to the more comprehensive consideration of the impact of adding shortcuts.

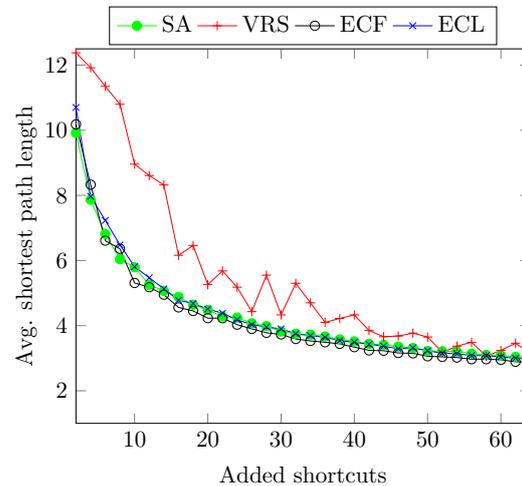


**Figure 4.** Average hop count for different degree limit values per switch. The proposed ECL and ECF consistently achieve lower average hop count across various limits.

*5.2. Effect of Shortcuts Limit*

Adding shortcuts reduces latency but increases the cost of cables. Moreover, switches have finite ports to connect to compute nodes and other switches, which also limits the number of shortcuts that can be added. The evaluation in Section 4 assumes a shortcut limit of  $2n$  where  $n = \sqrt{N}$ . Here, we conduct additional experiment that varies the shortcut limit for a 64-node network with ring as the base topology. The results are presented in Figure 5. Note that the previous  $2n$  corresponds to 16 here, and the number of added

shortcuts ranges from 2 to 64. As expected, the ASPL reduces in all four schemes as more shortcuts are added. The proposed ECF and ECL consistently perform better than the other two schemes. Although the figure shows a trend of diminishing return with small differences when a large number of shortcuts are added, this is prohibitively expensive. Therefore, for practical systems, the proposed EdgeCut has substantial advantage.



**Figure 5.** Impact of adding different number of shortcuts on optimizing a 64-node ring network. ECF and ECL achieve similar or lower ASPL than SA with a fraction of the search time.

### 5.3. Routing and Multi-Level Networks

Irregular networks including shortcut-augmented topologies usually requires topology-agnostic adaptive routing [3]. Research on this aspect is well established, and routing scheme much as [24] with up\*/down\* routing as the escape path is able to remove routing from the performance bottleneck, thus reflecting the benefits of the underlying topology.

Shortcut-augmented topologies, such as the ones that are generated from our proposed EdgeCut, can be employed in various networks. Large networks are often constructed in multiple levels, e.g., an intra-group network that connects compute nodes within a rack, and an inter-cabinet network that connects racks. EdgeCut can be used to generate topologies at each of those levels.

### 5.4. EdgeCut over Deterministic Shortcuts

In addition to comparing with VRS and SA that generates shortcuts randomly, we have also compared the proposed EdgeCut with two topology designs that add shortcuts deterministically.

The first topology is hierarchical rings that connect a set of rings hierarchically to achieve better scalability than regular ring networks [25]. Our evaluation shows that, for a 64-node network, hierarchical ring topology has an ASPL of 5.84 and a diameter of 12. In comparison, the proposed ECL with  $2n$  shortcuts has an ASPL of 4.88 and a diameter of 10, and ECF has an ASPL of 4.67 and a diameter of 9. Therefore, EdgeCut is considerably better than hierarchical rings in both metrics.

The second topology is flattened butterfly (FB) that adds shortcuts between nodes on the same row/column in a mesh network [26]. For an  $8 \times 8$  network, we follow the original paper to construct a flattened butter from four  $4 \times 4$  sub-networks to keep the bisectional bandwidth reasonable. Evaluation shows that FB has an ASPL of 3.39 and a diameter of 7; whereas ECL and ECF have ASPL of 3.39 and 3.3, respectively, and diameter of 7 and 6, respectively, so again better than FB in both metrics. Furthermore, ECL and ECF use only 128 total edges in contrast to 204 edges required in FB, thus making EdgeCut a more cost-effective approach.

### 5.5. Additional Considerations

The evaluation in this paper focuses on average hop count, diameter, and number of edges, all of which are well-established metrics to assess topologies. For a given system, however, the choice of topologies also depends on several other considerations, such as traffic patterns, queuing effects, cost of links (e.g., electrical vs. optical), cost of switches, etc. Additional evaluation is needed to take these factors into account, such as simulating on a cycle-accurate interconnection network simulator.

## 6. Conclusions

Shortcut-augmented topologies have the potential to surpass conventional regular and irregular topologies but have been hindered by the challenge in searching their enormous design space. In this paper, we address this important issue by proposing an efficient and effective heuristic approach that aims to generate more globally useful shortcuts. The proposed EdgeCut-Full considers the performance impact of shortcut candidates more comprehensively but also incurs higher computation, whereas EdgeCut-Lite simplifies the search process while retaining the ability to find good shortcuts. Evaluation results show that the proposed approach is able to generate comparable high-quality topologies as simulated annealing and achieve faster search time than vertex-based method, essentially reaping the benefits of both worlds and offering a better trade-off.

**Author Contributions:** Conceptualization, K.A.A.F., K.Z. and L.C.; formal analysis, K.A.A.F. and L.C.; methodology, K.A.A.F. and K.Z.; software, K.A.A.F. and K.Z.; supervision, L.C.; validation, L.C.; writing—original draft, K.A.A.F., K.Z. and L.C.; writing—review & editing, L.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research (including APC) was funded, in part, by the National Science Foundation grant number 1750047.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Top 500 Supercomputer Sites. Available online: <https://www.top500.org/lists/top500/2022/06/> (accessed on 30 June 2022).
2. Tomkins, J. Interconnects: A buyers point of view. In Proceedings of the ACS Workshop, Baltimore, MD, USA, 13–14 June 2007.
3. Koibuchi, M.; Matsutani, H.; Amano, H.; Hsu, D.F.; Casanova, H. A case for random shortcut topologies for HPC interconnects. In Proceedings of the 2012 39th Annual International Symposium on Computer Architecture (ISCA), IEEE, Portland, OR, USA, 9–13 Jun 2012; pp. 177–188.
4. Besta, M.; Hoefler, T. Slim Fly: A Cost Effective Low-Diameter Network Topology. In Proceedings of the SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, New Orleans, LA, USA, 16–21 November 2014; pp. 348–359. [[CrossRef](#)]
5. Besta, M.; Hoefler, T. Slim Fly: A Cost Effective Low-Diameter Network Topology. *arXiv* **2019**, arXiv:1912.08968v2. [[CrossRef](#)].
6. Singla, A.; Hong, C.Y.; Popa, L.; Godfrey, P.B. Jellyfish: Networking data centers randomly. In Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12), San Jose, CA, USA, 25–27 April 2012; pp. 225–238.
7. Fujiwara, I.; Koibuchi, M.; Matsutani, H.; Casanova, H. Skywalk: A Topology for HPC Networks with Low-Delay Switches. In Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium, Phoenix, AZ, USA, 19–23 May 2014; pp. 263–272. [[CrossRef](#)]
8. Baboli, M.; Husin, N.S.; Marsono, M.N. A comprehensive evaluation of direct and indirect network-on-chip topologies. In Proceedings of the 2014 International Conference on Industrial Engineering and Operations Management, Bali, Indonesia, 7–9 January 2014; pp. 2081–2090.
9. Alverson, R.; Roweth, D.; Kaplan, L. The Gemini System Interconnect. In Proceedings of the Proceedings of the 2010 18th IEEE Symposium on High Performance Interconnects, IEEE Computer Society, Mountain View, CA, USA, 8–20 August 2010; pp. 83–87. [[CrossRef](#)]

10. Chen, D.; Eisley, N.; Heidelberger, P.; Kumar, S.; Mamidala, A.; Petrini, F.; Senger, R.; Sugawara, Y.; Walkup, R.; Steinmacher-Burow, B.; et al. Looking under the Hood of the IBM Blue Gene/Q Network. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, USA, 10–16 November 2012; IEEE Computer Society Press: Washington, DC, USA, 2012.
11. Kim, J.; Dally, W.J.; Scott, S.; Abts, D. Technology-Driven, Highly-Scalable Dragonfly Topology. *SIGARCH Comput. Archit. News* **2008**, *36*, 77–88. [[CrossRef](#)]
12. Faanes, G.; Bataineh, A.; Roweth, D.; Court, T.; Froese, E.; Alverson, R.; Johnson, T.; Kopnick, J.; Higgins, M.; Reinhard, J. Cray cascade: A scalable HPC system based on a Dragonfly network. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, USA, 10–16 November 2012; IEEE/ACM: New York, NY, USA, 2012; p. 103.
13. Bollobás, B.; Chung, F.R.K. The Diameter of a Cycle Plus a Random Matching. *SIAM J. Discret. Math.* **1988**, *1*, 328–333. [[CrossRef](#)]
14. Bermond, J.C.; Comellas, F.; Hsu, D.F. Distributed loop computer-networks: A survey. *J. Parallel Distrib. Comput.* **1995**, *24*, 2–10. [[CrossRef](#)]
15. Milgram, S. The Small-World Problem. *Psychol. Today* **1967**, *1*, 61–67.
16. Watts, D.J.; Strogatz, S.H. Collective dynamics of ‘small-world’ networks. *Nature* **1998**, *393*, 440–442. [[CrossRef](#)] [[PubMed](#)]
17. Yasudo, R.; Koibuchi, M.; Nakano, K.; Matsutani, H.; Amano, H. Designing High-Performance Interconnection Networks with Host-Switch Graphs. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *30*, 315–330. [[CrossRef](#)]
18. Yasudo, R.; Nakano, K.; Koibuchi, M.; Matsutani, H.; Amano, H. Designing low-diameter interconnection networks with multi-ported host-switch graphs. In *Concurrency and Computation: Practice and Experience*; Wiley: Hoboken, NJ, USA, 2000; p. e6115. [[CrossRef](#)]
19. Kleinberg, J. The small-world phenomenon and distributed search. *SIAM News* **2004**, *37*, 1–2.
20. Bonnet, F.; Kermarrec, A.M.; Raynal, M. Small-World Networks: From Theoretical Bounds to Practical Systems. In *Principles of Distributed Systems, Proceedings of the 11th International Conference, OPODIS 2007, Guadeloupe, French West Indies, France, 17–20 December 2007*; Tovar, E., Tsigas, P., Fouchal, H., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 372–385.
21. Nguyen, V.; Martel, C. Designing Low Cost Networks with Short Routes and Low Congestion. In Proceedings of the IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications, Barcelona, Spain, 23–29 April 2006; pp. 1–12. [[CrossRef](#)]
22. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [[CrossRef](#)]
23. Parhami, B.; Yeh, C.H.; Parameters, T. Why Network Diameter is Still Important. In Proceedings of the International Conference in Communications (CIC-2000), Las Vegas, Nevada, USA, 26–29 June 2000.
24. Silla, F.; Duato, J. High-performance routing in networks of workstations with irregular topology. *IEEE Trans. Parallel Distrib. Syst.* **2000**, *11*, 699–719. [[CrossRef](#)]
25. Ausavarungnirun, R.; Fallin, C.; Yu, X.; Chang, K.K.W.; Nazario, G.; Das, R.; Loh, G.H.; Mutlu, O. Design and Evaluation of Hierarchical Rings with Deflection Routing. In Proceedings of the 2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing, Paris, France, 22–24 October 2014; pp. 230–237. [[CrossRef](#)]
26. Kim, J.; Dally, W.J.; Abts, D. Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks. In Proceedings of the 34th Annual International Symposium on Computer Architecture, San Diego, CA, USA, 9–13 June 2007; Association for Computing Machinery: New York, NY, USA, 2007; pp. 126–137. [[CrossRef](#)]