Finding Your Niche: An Evolutionary Approach to HPC Topologies

Stephen J. Young, Joshua Suetterlein, Jesun Firoz, Joseph Manzano, Kevin Barker
Pacific Northwest National Laboratory, USA
{stephen.young,joshua.suetterlein,jesun.firoz,joseph.manzano,kevin.barker}@pnnl.gov

***Abstract*—Traditional interconnection network design approaches focus on building *general* network topologies by optimizing the bisection bandwidth or minimizing the network's diameter to reduce the maximum distance between any two nodes, thus amortizing the overall execution time of the HPC workloads. While such network topologies may accommodate a wide variety of applications in general, this may result in sub-optimal performance for many frequently-executed or dynamic workloads. In this paper, instead of focusing on designing an all-encompassing, general-purpose network topology, we develop a *methodology* to design *customized* network interconnects, evolved by "finding" the optimal topologies for a particular target workload given by its communication and contention profiles. To this end, we implement a Genetic Algorithm (GA)-based approach for network topology design tailored to improve the overall execution time of a particular workload of interest. We conducted extensive experiments with well-known motifs in physics-based workloads (Sweep3D and FFT), as well as with a representative graph application (MiniVite), using the well-known Structural Simulation Toolkit (SST) Macroscale Element Library (`SST/macro`) simulator for network interconnect evaluation. We demonstrate that our genetic algorithm-based approach is robust enough to find the underlying optimal topology of a particular workload.**

I. Introduction

Historically, interconnect design was driven by a deep understanding of the limitations of the extant (or near-term) hardware and characteristics of desired workloads, typically physics-based simulation and dense linear algebra. However, with the increasing diversity of HPC workloads and their communication patterns, interconnects have moved toward designs that minimize the latency/diameter of the network, such as DragonFly [24], SlimFly [8], and Fat tree [27]. Despite the success of these topologies, modern scientific workloads still significantly stress the underlying communication fabric. The pressure from communication is driven by the complexity of modern scientific workloads, which often include steps guided by surrogate artificial intelligence (AI) models coupled with exploratory data analysis [26], [2], [25]. Each stage demonstrates a distinct execution profile with unique communication characteristics [28]. Further, HPC system's characteristics, such as non-uniform access patterns across different memory hierarchies, offloading of computation to heterogeneous accelerators, and I/O operations with networked filesystems, compound the difficulty in engineering the performance of the network for these composite workflows.

While distributed physics-based simulation and dense linear algebra computation – where regular communication pattern may be known *apriori* – are still prevalent, distributed sparse data-driven computation with irregular communication profile is becoming an integral part of the scientific and machine learning workloads. For the later, communication over the network takes a significant fraction of the total execution time and is generally the main bottleneck. With the deployment of Exascale machines, the gap between the bytes transferred over the different communication fabrics and flops will only be exacerbated [19]. However, the static network topologies used in practice, such as Fat tree, DragonFly, Torus, and recently proposed expander graph based topologies (JellyFish [31], Xpander [36], SpectralFly [38], etc.) may leave crucial performance on the table due to their inability to adapt to the dynamic communication profile of data-driven workloads.

A solution to the "too general/static" challenge of current HPC systems come in the form of co-designing mission critical workloads with their computing substrates and software stacks [7]. Under this paradigm, the underlying computational components (i.e., accelerators, network switches, etc) can be tailored in accordance to the their underlying workload needs. In response to this trend, reconfigurable computing has seen a resurgence in interest with new proposals for Coarse-Grained Reconfigurable Architectures (CGRA) [33], optical switch networks [34], [35], and customized network-on-chips (NoC) [21]. However, methodologies for designing customized off-chip network topologies for specific workload have been relatively unexplored.

In an effort to circumvent the problems associated with static network topologies for dynamic workloads with runtime-driven communication patterns, and to further complement co-design efforts, we propose a *data-driven methodology* for leveraging modern fine-grained simulation software to design customized network topologies for target workloads. Specifically, by modeling network topologies as graphs, where the switches in a network and the connections among the switches are considered as the vertices and the edges of the graph respectively, we employ a Genetic Algorithm (GA) based meta-heuristic [32] to evolve from an initial topology to a topology tailored for a target workload. The objective is to "find" the workload-specific optimal connections among the switches so as to improve the overall performance of the target workload when running on the final evolved topology. The "searching" process may involve capturing the underlying *communication structure* of the workload while considering the impact of *congestion* simultaneously.

To demonstrate the effectiveness of the data-driven approach to customized topology design, we consider two broad classes of workloads. First, we consider two structured communication motifs, Sweep3D and FFT, from the Ember communication patterns library [1]. These motifs capture the regular, structured communication patterns frequently encountered in physics-based scientific workloads. Our chosen second workload category is unstructured communication, represented by an ECP ExaGraph project proxy application, MiniVite [18]. We demonstrate that for structured topologies the GA approach reduces the run time by as much as 26.8% over the the initial (unstructured) JellyFish topologies with a well-known adaptive routing scheme, by evolving a structure which aligns with the underlying communication structure. For unstructured communication workloads, the initial JellyFish topology is extremely well-suited, thus the improvement is more modest with a maximum improvement at around 1.7% at most when compared with the best adaptive routing results.

The paper makes the following contributions:

- A novel GA based methodology to design and develop customized network topologies tailored for specific workloads running on the well-regarded network topology simulator, Structural Simulation Toolkit (SST) Macroscale Element Library (`SST/macro`) [3].
- An extensive network evaluation with two distinctive categories of workloads, characterized by their communication patterns: workloads with structured communication patterns (such as Sweep3D and FFT) and input-dependent workloads with unstructured communication patterns (MiniVite).
- A performance comparison between our custom, workload specific topologies against a representative commercial topology, i.e., Jellyfish, with adaptive routing, i.e., UGAL.

II. Methodology

A. Overall Workflow

We provide a high-level overview of our experimental methodology in Figure 1. The methodology is divided into three main phases: the preprocessing step, the "searching loop", and postprocessing and analysis phase. During the preprocessing step, we utilize the PMPI-based [4] VISUS tool to collect the number of bytes per message, record the source-destination information for each message and compute the cumulative bytes transferred between each source-destination pair for each of the workloads(1b). For running with the `SST/macro` simulator, we prepare the workload by either skeletonizing them to retain only the communication motifs (for Sweep3D and FFT) or executing the whole workload (i.e., capturing both communication and computation, for MiniVite) (1a).

The "searching loop" phase involves simulating a set of custom topologies for a target workload in `SST/macro` (❷), collecting the overall runtime, applying the genetic algorithm to evaluate the current quality of the topologies and preparing the next set of topologies for simulation (❸ and ❹). Initially, a set of seed topologies are created based on given constraints, e.g., switches with a specific radix. This loop continues until a predetermined number of steps have been executed or until GA

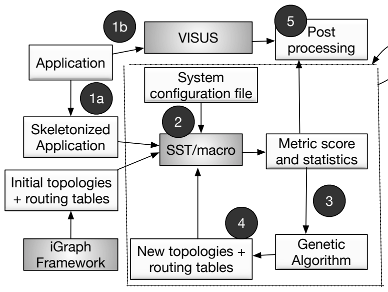

1b
VISUS
5
Post processing
Application
1a
System configuration file
Skeletonized Application
2
SST/macro
Metric score and statistics
Initial topologies + routing tables
3
4
iGraph Framework
New topologies + routing tables
Genetic Algorithm

Fig. 1: An overview of our methodology.

converges. This process is described more in detail in Section II-B.

Once the customized topologies are designed, in the post-processing phase, a set of Python scripts takes the different results for each searching loop iteration per topology (❺). This information includes, but is not limited to, sorted runtimes, communication patterns, queue lengths, and network characteristics such as hops. Finally, this phase generates visual representations of the candidate network's behavior/structure to evaluate the quality of the generated solution.

B. Genetic Algorithm for Evolving Topologies

To derive a better topology tailored for a workload, the searching phase of our methodology employs a well-known meta-heuristic based evolutionary algorithm, namely *Genetic Algorithm* (GA), inspired by evolutionary biology. In informal terms, the genetic algorithm is built upon the notion that, within a population, only the "survival of the fittest" is observed over the course of evolution.

To apply GA, the *genotype or genome* of a network topology (*an individual*) is encoded as a n-vertex graph where each vertex and each edge in the graph represent a network switch and a communication link respectively. The graph is k-regular – each vertex has a degree of k, thus representing a switch with radix k. In each generation, the *population* of the GA algorithm consists of a set of candidate topologies, encoded as graphs. The next generation of candidate topologies are generated by considering the topologies with better *fitness scores*. In our case, the better the runtime of a workload with a evolved topology, the higher the score. Once the candidate set is *selected* for evolution, *child* topologies are generated from the *parent* topologies (*breeding*), applying *mutation*.

Initial Population. In order to form our initial population we generate 800 instances of the radix-k Jellyfish topology [31] on n vertices using `igraph` [11], which uses an acceptance/rejection approach to randomly choose a k-regular graph from the configurational model [1]. We note that, in addition to representing the "zero knowledge" baseline, this initial population is likely to have good communication properties for almost all communication patterns [6]. Indeed,

[1]In this model a random perfect matching is formed on n groups of k "mini-vertices" corresponding to vertices of the graph. This process is repeated until a simple graph is generated.

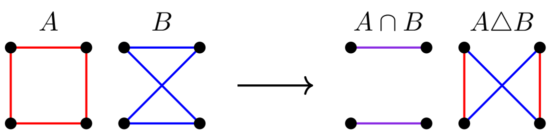


Fig. 2: Splitting between common edges (purple) and symmetric difference of edges (red/blue).

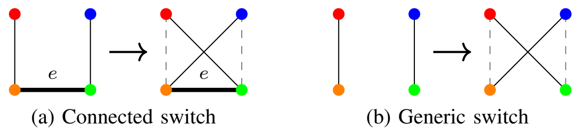


Fig. 3: Two different switch operations used in the generation of non-initial populations

if $k \geq 3$, then with high-probability the generated graph is a nearly-optimal spectral expander [13], [14] having low-diameter and high bisection bandwidth [10]. Extrapolating from other expansion-based topologies, they are likely to be highly performant independent of the specific application.

Population Evaluation. To evaluate the "fitness" of a particular topology for an application, we use a simulation platform to evaluate the overall run time of the application using a direct topology of concentration one (i.e., there is one switch per vertex in the topology and one compute node per switch) using minimal routing. While many options are available to perform the application simulation, we use `SST/macro` [3] in order to leverage the existing library of mini-applications designed to evaluate topologies for communication patterns. In order to eliminate any variability resulting from randomness, a given MPI rank is assigned to the corresponding switch number. Thus, the simulation time provided by `SST/macro` is deterministic and can be used as a measure of fitness.

Non-Initial Generations. For every generation beyond the initial one, our "gene pool" consists of the 40 graphs with the greatest fitness (i.e., fastest run time) among *all* previous generations. In order to form subsequent generations we take two randomly chosen "parents," $A = (V, E_A)$ and $B = (V, E_B)$, from the gene pool and consider their intersection ($A \cap B$) and symmetric difference ($A \triangle B$), see Figure 2. The edges in $A \cap B$ are immediately passed on to the child while the edges in $A \triangle B$ are subject to mutation before a randomly chosen half (subject to the regularity constraint) are passed on to the child network. This process is repeated 800 times to form the current generation.

The random mutation takes the form of the "connected switch operation" (see Figure 3a) which locally perturbs the edge set while maintaining the overall connectivity structure of $A \triangle B$. As this process may result in disconnected children, these graphs are post-processed by using a generic switch operation (see Figure 3b) to minimally merge disconnected components. It is worth noting that the (connected) switch operation is a fundamental primitive in Markov Chain Monte Carlo (MCMC) methods to uniformly sample (connected) graphs with a prescribed degree distribution ([22], [12]).

C. Earthmover's Distance

While the genetic algorithm framework is explicitly optimizing the overall run time, this has the effect of implicitly optimizing the nature of the interactions between the structure of the network and communication profile. In this work we will focus on the effect of the genetic algorithm optimization on two primary interactions between the network and communication profile; latency (measured by hops-per-byte) and contention (measured by the packet queue length of a switch observed upon packet insertion). While uniformly decreasing either the latency or contention across the board should decrease the overall run time, in practice, it is often necessary to make trade-offs between reducing some aspects while increasing others; for instance, it may be the case that rewiring a switch would decrease the number of hops for certain bytes, but increase the number of hops for others or increase the queue size on packet insertion at other places/times in the application. Thus, it is necessary to be able to compare both latency and contention between different topologies in a statistical manner.

To this end, we introduce the use of Earthmover's distance (EMD) [29] as a means of comparing discrete distributions associated with communication performance. At a high-level, the EMD (and it's continuous analogue, the Wasserstein metric [23]) can be thought of as the amount of work[2] needed to transform from one probability distribution to another distribution. For arbitrary discrete distributions, the EMD can be calculated as the solution to a minimum cost flow where the edge weights are given by the distances between elements. If both probability distributions take values in $1, \dots, k$ (as is the case for latency and contention), the EMD can be calculated via a single pass through the probability distributions.

III. Experimental Results

Simulation platform. For our experiments, we use the simulated time from `SST/macro` to evaluate network interconnects. To perform a simulation, we first generated a topology (i.e., graph) with the `igraph` library. To simulate the `igraph` generated topologies, we perform an all-pairs shortest path calculation to obtain the routing table and transform the graph and table into JSON. The configuration of the topology, routing, and network parameters is specified in a *parameter* file. Table I provides a summary of the key simulation parameters. To avoid deadlocks, the total number of virtual channels was set to the graph's diameter plus one.

Routing Configurations. We use minimal routing for packet transfer for all topologies evaluated with the GA, including the initial population of JellyFish topologies. To compare the benefits of topology modification with the adaptive routing protocols, we consider both the initial population and the best-observed topology using the local information implementation of Universally Global Adaptive Load-balanced

[2]If two probability distributions can be thought of as discrete piles of earth, the EMD is the physical work (`mass` × `distance`) needed to move one to the other, hence the name.

Parameter	Value
Link bandwidth	10 GB/s
Switch latency	100 ns
Switch buffer	32 MB
MTU	4 KB
Network model	SNAPPR
Virtual channels	Topology diameter + 1

TABLE I: Relevant `SST/macro` simulation parameters

routing (UGAL) [30]. The best and worst runtimes for the initial population using UGAL are reported with the notation `UGAL min` and `UGAL max`. These two lines bracket the JellyFish topology's baseline behavior using adaptive routing. Further, we report the runtime of the best-evolved topology with adaptive routing (`UGAL final`).

The objective of choosing these baselines is to demonstrate that even with applying adaptive routing, e.g., UGAL, there is significant room for improvement for a given topology. In general, as we will see later in this section, these baselines help to establish that our GA-evolved topologies may not only improve the performance of a workload by searching for the structure of the communication pattern but also is capable of incorporating the congestion information (for both structured and data-driven communication patterns), potentially improving performance over adaptive routing.

Quantifying the "searching" process. One of the key scientific questions in analyzing the results of a GA is understanding what the process is finding through the course of the evolution. To this end, to interrogate the evolution process, we capture for selected topologies in the GA the number of hops-per-byte (latency) and the queue size on packet insertion (contention). We view these statistics as a probability distribution (i.e., how many hops does a randomly chosen byte move and what is the distribution of queue sizes for a randomly chosen packet) and use EMD to measure the difference between distribution pairs. In order to measure the "direction" of overall evolution for these statistics, we consider the EMD to a fixed idealized situation (every byte moves exactly one hop, and every queue is empty on packet insertion) and report the results in Figure 5 and 9. To understand the overall variation of the evolution, we use a linear programming formulation to determine the radius minimum ball (in EMD) that contains all of the best survivors throughout the GA. These results are reported in Figure 6b and 8d.

A. Structured Communication Patterns

To evaluate the performance of real-world structured traffic patterns with different topologies, we consider the communication motifs from the Ember Communication Pattern Library [1]. In particular, as examples of the wavefront and Sub-communicator all-to-all communication patterns, Sweep3D and FFT are chosen, respectively.

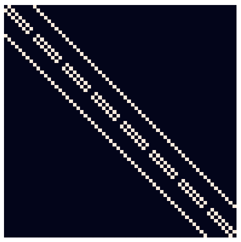
Sweep3D. We first look at the Sweep3D benchmark, part of the ember suite, representing the highly-structured communication of a 3D discrete ordinates neutron transportation code. While it is well established that this benchmark performs well on topologies such as tori and meshes [20], we leverage this regular benchmark to demonstrate the ability of a GA to achieve similar performance. Indeed, the runtime of the evolved topology from the GA for `Sweep3D` on 64 ranks is within 4% of the runtime of the built-in 8×8 torus topology simulated with `SST/macro`.

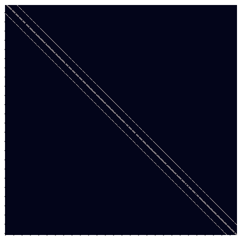
Figure 4a represents the communication pattern across 64 nodes. At each iteration of the application, an MPI rank receives two messages from distinct ranks and sends a message to two other ranks using blocking sends and receives. Figure 4e presents the performance of evolved topologies across 1000 generations conforming to the GA description in Section II-B. The fanout of each switch is limited to four to match the known communication pattern. In total, we have run 800,000 simulations. We present the min, max, and average simulated time achieved per generation to show the overall performance of the GA. We find the average time per generation to depict the average performance and plot the closest simulation. The red line represents the evolution of the fastest topologies across generations (i.e., survivors).

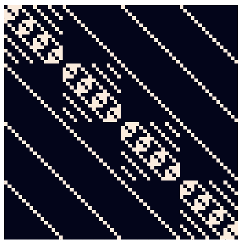
As observed in Figure 4e, the survivors of the evolution improve the execution time over `UGAL max`, `UGAL min` and `UGAL final` by 42.3%, 26.8%, and 5.5%. In addition, we observed a difference of 26.4% across the survivors ending at generation 656. In Figure 4i, we summarize the structure of the survivors in terms of network hops per byte and the average hops per byte. In this figure, the number of hops experienced per byte decreases as more direct connections emerge between communicating ranks. Figures 6a and 5 build on the distribution of hops presented in Figure 4i. Figure 6a shows the evolution of Sweep3d based on the normalized EMD from the ideal (lower is better). Here we see the structural distance based on the hops per byte steadily decreasing across the survivors. In addition, the observed contention (based on the distributions of queue lengths observed per packet per switch) decreases more modestly across survivors.

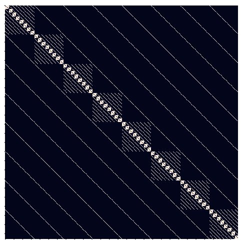
Figure 5 depicts the change of the EMD to the ideal distribution between the first/last survivors, represented by the dark and light color bars. The table below the figure highlights the normalized changes in structure and contention between the first/last survivors for each benchmark. A positive value means the evolution became closer to the ideal (i.e., a decrease in hops-per-byte or contention). A negative value indicates the topology evolved away from the ideal (i.e., more hops-per-byte or observed contention). The last rows show the change in time and the percent difference in performance. The color of each cell highlights the magnitude of the change across experiments (i.e., forming a heatmap per row). For example, Sweep3D on 64 nodes (s1) shows the most significant improvement in reducing the hops-per-byte and observed contention yielding a performance improvement of 26.4%.

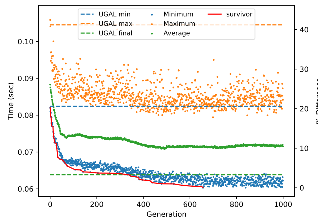
Figure 6b compares the distances of the survivors for a benchmark's evolution. The x-axis shows the structural distance explored (variations in hops per byte distributions), and the y-axis shows the contention observed across the evolution (variations in the insertion queue lengths). The distance from


(a) `S 64,4 Comm.`

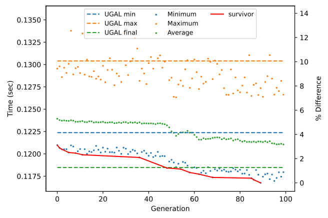

(b) `S 529,4 Comm.`


(c) `F 64,9 Comm.`

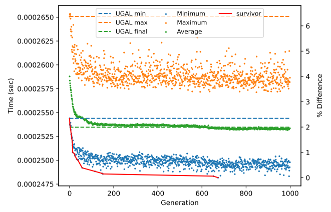

(d) `F 512,21 Comm.`



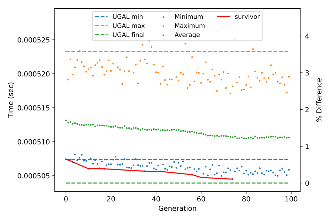
(e) `S 64,4: Performance`



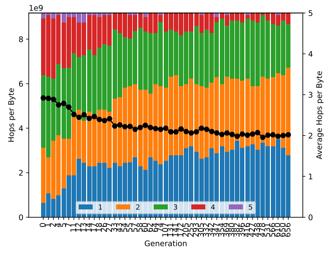
(f) `S 529,4: Performance`



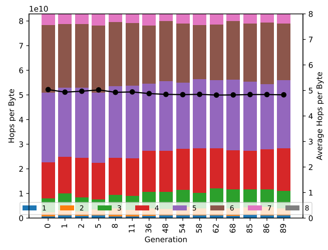
(g) `F 64,9: Performance`



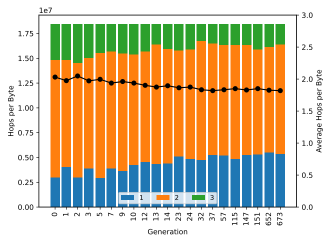
(h) `F 512,21: Performance`



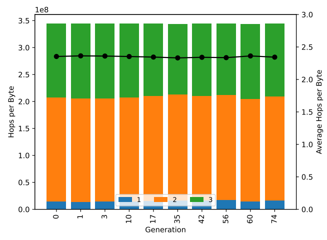
(i) `S 64,4: Distribution`



(j) `S 529,4: Distribution`

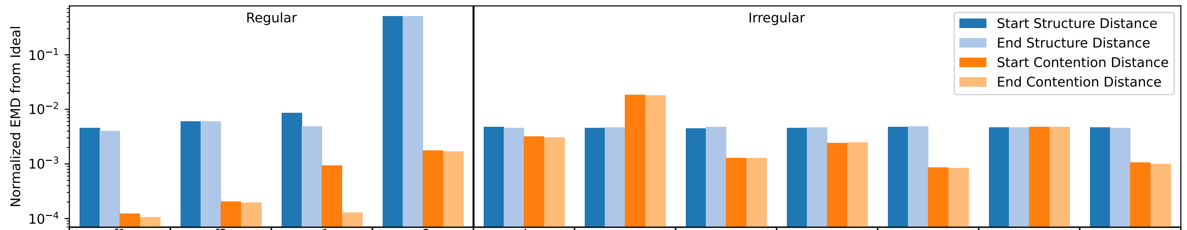


(k) `F 64,9: Distribution`



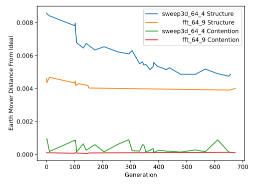
(l) `F512,21: Distribution`

Fig. 4: Sweep3D (S) and FFT (F) characterization and experimental results: X, Y denotes nodes and fanout respectively

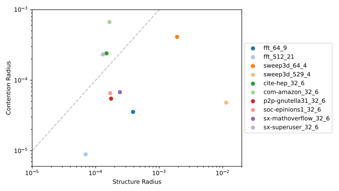


	f1	f2	s1	s2	ch	ca	pg	se	sm	ss	wr
Δ Structure	6.05E-04	-3.65E-05	0.004	0.004	1.69E-04	-8.87E-05	-3.51E-04	-5.29E-05	-8.68E-05	-4.63E-05	1.04E-04
Δ Contention	1.60E-05	9.68E-06	8.07E-04	7.31E-05	1.14E-04	3.76E-04	6.34E-07	-5.07E-05	2.59E-05	-1.26E-05	7.43E-05
Δ Time (sec)	6.22E-06	2.93E-06	0.022	0.004	1.98E-06	3.60E-06	6.47E-07	1.23E-06	1.50E-06	3.11E-06	1.16E-06
Δ Percent	2.445	0.577	26.445	3.195	1.016	0.846	0.66	1.025	1.632	1.714	1.205

Fig. 5: Effect of GA on the distance from ideal distribution distribution for hops-per-byte (i.e., structure) and queue length on packet insertion (i.e., contention) for the first and last survivors per evolution



(a) s1 and f1 Evolution



(b) Structure vs Contention

Fig. 6: EMD Comparison

the origin helps to show the size of the space explored during evolution. The dotted line refers to where the structural and contention distances are equal. The figure shows that the 64-node Sweep3D explores a larger structural design space than the contention space.

Figure 4b presents 100 generations of Sweep3D running on 529 nodes with a fanout of four. In Figure 4f, we observe a decrease in time across the survivors, albeit more modest. The improved execution time over `UGAL max`, `UGAL min`, and `UGAL final` are 11%, 4.5%, and 1.4%. Figure 4j summarizes the structure's evolution via its hops per byte distributions. We immediately notice an increase in hops relative to the 64-node case. The modest performance improvement per generation (in both time and hops per byte) can be attributed to the exponential increase in topological exploration space as the GA searches for the correct structure. Figure 6b (x-axis) highlights the large search space. From Figure 5, we see Sweep3D on 529 nodes (s2) is the furthest from the ideal topology and traverses a design space similar in size to the evolution of Sweep3d on 64 nodes resulting in a 3.2% improvement in performance. From Figures 5 and 6b, we see the GA is evolving more structurally than in terms of contention. We would anticipate this to change as we increase the number of generations.

FFT. The second benchmark in our evaluation is the Fast Fourier Transform (FFT). Figure 4c shows the computation pattern of this highly structured application. Figure 4g shows the performance of FFT with our generated topologies after 1000 generations, running on 64 node topologies, each with a fanout of nine (chosen to match the communication of FFT). The improved execution time over `UGAL max`, `UGAL min`, and `UGAL final` are 6.3%, 2.4%, and 2.1%, respectively. The survivors' simulated time decreases by 2.4%, with a minimum at generation 673. From Figure 4g, we see distributions with fewer variations than Sweep3D on 64 nodes resulting in a smaller structural EMD in Figures 6a, 5 and 6b. For FFT on 64 nodes (column f1 in Figure 5), Figure 6a shows the normalized EMD from the ideal across the evolution. We see modest changes in both structure and contention. Figure 5 highlights the topologies being structurally further from the ideal while exhibiting low contention. Despite running for 1000 generations, the space explored by the GA remains modest compared to our other experiments. This would indicate that the GA found a local minima requiring additional techniques to improve.

In Figure 4d, we present the simulated times for 100 generations of FFT on 512 nodes with a fanout of 21. Figure 4h presents the performance of the evolved topologies. The improved execution time over `UGAL max` and `UGAL min` are 3.5% and .6%, respectively. The `UGAL final` time outperforms our final survivor by .1%. Even though we have increased the number of nodes, the maximum number of hops remains 3 in Figure 4l. Moreover, from Figures 5 (column f2) and 6b, we observe that the GA traverses little space (structurally and in contention), ultimately opting to reduce contention at the expense of structure, demonstrating little improvement in time. This behavior indicates that the GA is not finding enough of an evolutionary signal to guide its evolution.

B. Unstructured Communication Patterns

Graph algorithms generally involve irregular memory accesses and low computation to communication ratio. The communication pattern is dictated by the input graph properties and revealed over the course of the runtime of an application. The dynamic unfolding of message exchanges results in an unstructured communication pattern distinguishes these family of algorithms from physics-based applications.

Louvain Algorithm for Community Detection. To evaluate the effectiveness of the GA to "find" the underlying topology of a graph application with an unstructured communication pattern, we consider the miniVite ([18], [5]) proxy application from ECP. MiniVite implements the first phase of the well-known Louvain algorithm for graph community detection in MPI and OpenMP. The main network operations of this workflow are global collective operations emphasizing its all-to-all nature. To evaluate our GA-based evolved topology for MiniVite, we consider a diverse set of graph inputs from different domains (Table II). While the communication pattern is fixed within the benchmark (i.e., all-to-all), the bytes transferred will vary with different inputs.

Network Type	Graph	# Vertices	# edges
Social	soc-Epinions1 (**se**)	76K	508K
	wiki-RFA (**wr**)	11K	16K
Citation	cit-HepPh (**ch**)	34.5K	421.5K
Product	com-Amazon (**ca**)	335K	926K
Peer-to-peer	p2p-Gnutella31 (**pg**)	62.5K	148K

TABLE II: Characteristics of the input graphs used for evaluating different topologies with Minivite.

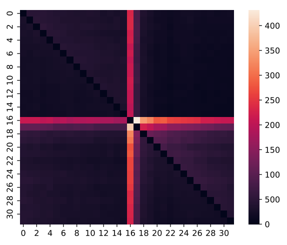
We report the experimental results for MiniVite in Figure 7. For each of these experiments, we perform 200 generations on 32 nodes with a fanout of six. For all MiniVite inputs, we observe that there is little to no performance difference between the min and UGAL routing. This trend is because contention communication is spread across nodes throughout their application execution, making the Valiant path less likely.

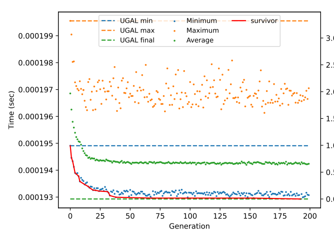
Figure 7a shows the communication pattern for MiniVite running the cite-HepPH dataset. Figure 7b depicts the evolution of the survivors resulting in a 1% performance improvement. In Figure 5, the final topology evolves toward the ideal in structure and contention. Figure 6b shows that the GA observes a more significant difference in contention than structure as the topologies evolve. This behavior is likely due to the node 16 traffic observed in Figure 7a.

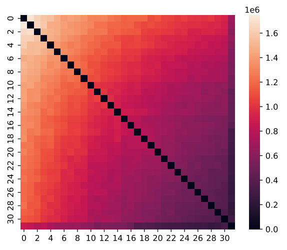
Figure 7c shows the communication pattern for MiniVite with our largest input, com-amazon. Figure 7d shows a modest performance evolution. Finally, figure 5 shows that the topology evolves away from the ideal in favor of reducing contention. Further, we see in Figure 6b, that the GA experiences the most considerable contention distance.

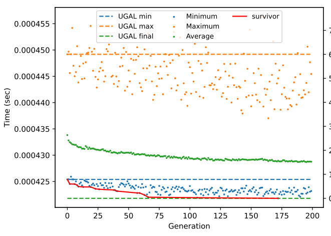
In Figure 7e, we present the communication pattern for the p2p-Gnutella dataset. The heavy communication along the diagonal is reminiscent of the structured experiments in Section III-A. While the communication structure is similar, collective versus point-to-point communication causes a marked difference in the performance profile seen in Figure 7f. Similarly to com-Amazon, the final topology moves away from the ideal in favor of reducing contention in Figure 5. Figure 6b shows GA explores a greater structural space.

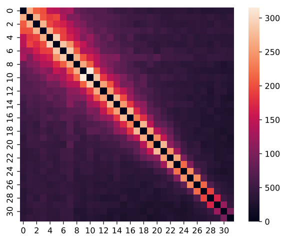
The soc-Epinions dataset presents an interesting structural pattern in Figure 7g. Figure 7h shows only eight minimum survivors across 200 generations. Combined with a greater spread between the minimums per generation, this suggests the communication structure is more difficult for the GA to find. This insight is supported by a smaller structural exploration observed from Figure 6b. Further, from Figure 5, the final

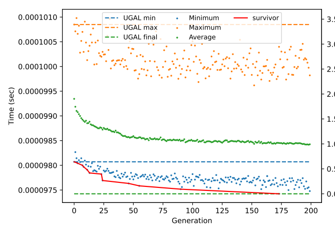

(a) `32,6: ch, Comm.`

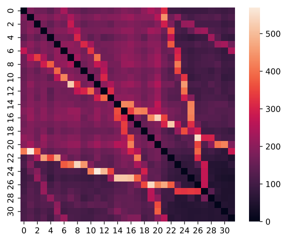

(b) `32,6: ch, Evolution`


(c) `32,6: ca, Comm.`


(d) `32,6: ca, Evolution`


(e) `32,6: pg, Comm.`


(f) `32,6: pg, Evolution`


(g) `32,6: se, Comm.`

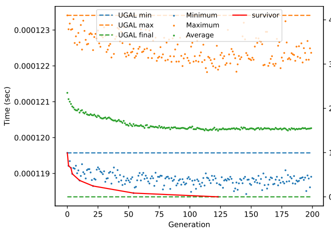

(h) `32,6: se, Evolution`

Fig. 7: MiniVite characterization and experimental results with different datasets: X, Y denotes nodes and fanout respectively

topology has evolved away from the ideal.

C. *Case Study: MiniVite with wiki-RFA dataset*

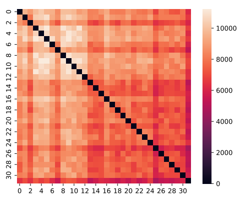
The following presents a deep dive into the factors involved in evolving a topology and the characteristics of the resulting execution profile. We perform this analysis on MiniVite using the wiki-RFA input. Figure 8a shows the communication profile, which exhibits a subtle structure underlying an all-to-all communication pattern. To better understand how topologies evolve, we have performed three evolutions for a 32-node network with fanouts of 2, 4, 8, and 16. The first evolution performed uses the simulated time to evaluate the fitness of the topology identical to our previous experiments. The additional two use the structural and contention EMD from the ideal topology as their evolution metric. In Figure 8, we present the results of each evolution with blue, orange, and green gradients representing the evolutions based on time, structure, and contention, respectively. We present a moving average of five data points for Figures 8b and 8c to highlight the trends.

Figure 8b shows the survivors' normalized evolution metric (*higher is better*). **The structure-based evolution with a fanout of 8 stands out with the most significant development**, followed by the contention-based evolutions and the structure-based evolution with a fanout of 4. The time-based evolutions show the least change. Figure 8c shows the resulting time for the survivors (*lower is better*). The evolutions with a fanout of 2 perform the worst as the GA explores the layout of ranks in a ring topology (since the only connected topology with a fanout of two is a ring). **The structure-based evolution with a fanout of 8 strongly correlates with its evolution metric; however, its performance is the worst of the remaining experiments.** We see little change in time across the surviving generations for the remaining evolutions.

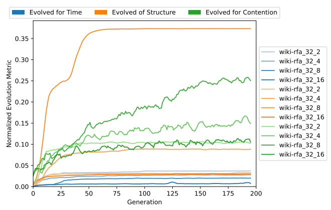
Figures 8d and 9 depict *how* each topology evolves. Figure 8d shows the maximum EMD of the survivor topologies explored. The distance from the origin gives the size of the explored design space in terms of structure (x-axis) and observed contention (y-axis). Of the three evolution sets, fanouts of 2 and 16 show little structural exploration (i.e., x-axis) as the design space is limited. There is less contention for the fanouts of 16 due to the topologies' connectivity resulting in a reduced contention space exploration. **In summary, the evolution of the topology sets is restricted by the fanouts. Fanouts of 2 are too low to test different configurations. Fanouts of 16 are too high reducing the impact of congestion.**

For the rest of the evolution sets, we see a greater exploration in structure and contention for fanouts of 8 over fanouts of 4. **For the structure-based evolution with a fanout of 8, we observed a large exploration space in both contention and structure, as the fanout 8 point moves away from the origin sharply in both axes.** This behavior manifests itself in the observed trend in time. The contention-based evolution with a fanout of 8 also explored a sizeable structural design space while modestly exploring the contention space.

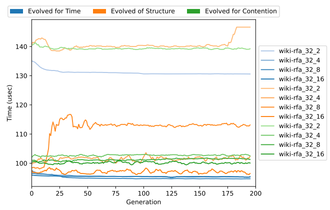
Figure 9 presents the EMD from the ideal of the first and final survivor. The table below the figure shows the change in distance from the ideal in structure and contention per evolution. In addition, we present the change in simulation time, the percent difference relative to the start time for the evolution, and the overall percent difference relative to the starting time of the time-based evolutions. The colors in a row highlight the intensity of the change across all experiments (i.e., forming a heatmap per row). Each column corresponds to the experiment above it (e.g., the structure-based evolution with a fanout of 8 exhibits the most significant structural change with a difference of 0.229). Overall, the structural distance from the ideal is much larger than the contention.



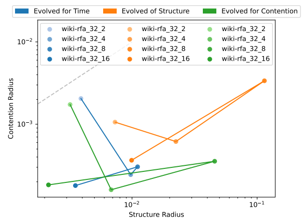
(a) Comm. Pattern



(b) Normalized Evo. Metric

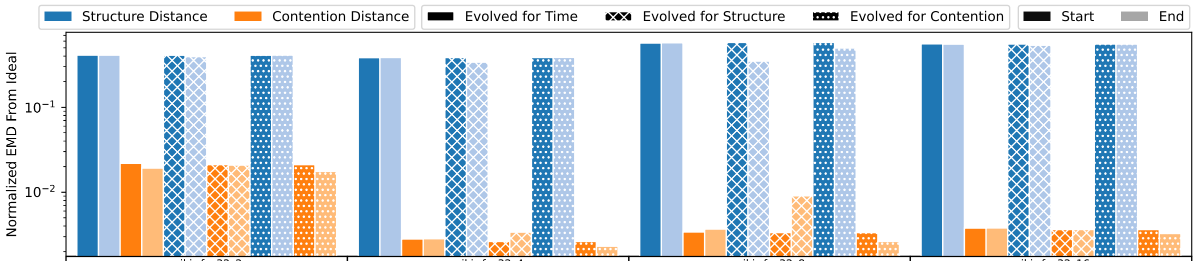


(c) Time (usec)



(d) Contention vs structural radius

Fig. 8: Impact of varying fanouts and fitness scoring techniques on MiniVite runtime with the wiki-RFA on 32 nodes



	wiki-rfa_32_2			wiki-rfa_32_4			wiki-rfa_32_8			wiki-rfa_32_16		
Δ Structure	5.57E-04	0.015	-0.002	-0.001	0.045	-9.67E-04	-0.004	0.229	0.08	0.005	0.02	9.09E-04
Δ Contention	0.003	1.50E-04	0.003	-1.80E-05	-7.56E-04	3.20E-04	-2.81E-04	-0.006	7.10E-04	-1.02E-05	-5.62E-06	3.67E-04
Δ Time (usec)	6.591	-2.031	0.766	3.745	-0.478	-0.249	4.402	-5.729	-0.322	0.859	0.305	-0.141
Δ Percent	4.796	-1.497	0.565	3.779	-0.488	-0.255	4.439	-5.93	-0.333	0.89	0.318	-0.147
Relative %	4.796	-0.173	1.862	3.779	0.621	0.852	4.439	-3.195	2.258	0.89	0.663	0.2

Fig. 9: Effect of GA on the distance to ideal distribution distribution for hops-per-byte and queue length on packet insertion

As noted, the GA finds the greatest evolutionary signal in the structured-evolved topologies of fanouts 4 and 8, resulting in more contention than observed in the initial topologies. In the contention-evolved topologies with a fanout of 4, the survivors evolve away from the ideal structurally to avoid contention. Conversely, for a fanout of 8, the topology moves closer to the ideal. The time-based evolutions with fanouts 4 and 8 evolve away from the ideal in structure and contention, resulting in a 3.8 and 4.4% increase in performance. **We postulate that these results come from a combination of a small search space (i.e., the GA is starting close to the optimal), the dependent relationship between topology and contention, and the load-balance of work in the application.**As noted, there are few structural differences for fanouts of 2 and 16. *For the topologies with a fanout of 2, the GA evolutions can be viewed as heuristics solving the rank placement problem to reduce contention.* In this case, the time and contention-based evolutions tie in reducing contention. The time-based evolution performs the best, balancing the reduction of hops per byte and observed contention resulting time improvement of 4.8%. Given that there is less space to traverse structurally, the contention-based evolution increases performance by 1.9%. The topologies with a fanout of 16 exhibit the least change in overall performance (i.e., $< 1\%$).

IV. Related Work

TOPOOPT [37] is a recently-proposed direct-connect fabric that co-optimizes the underlying topology and the degree of parallelism for distributed neural networks (DNN). TOPOOPT is different from our work in that, it only extracts the pattern of allreduce collective execution from DNN workload and optimizes the performance by making trade-offs between topology and parallelism. In contrast, we propose a generic approach in designing customized topologies for interesting workloads. In [9], the authors discussed a machine learning approach to understand network congestion. Our GA-based approach strive to improve the overall performance by evolving the underlying topology to mitigate the impact of congestion.

GAs have been adapted to solve various network optimization problems, including multistage process planning (MPP) problem, fixed charge transportation problem (fc-TP), minimum spanning tree problem, centralized network design, local area network (LAN) design, and shortest path problem [16], [17], [15].

V. Conclusion

In this paper, we presented a data-driven methodology, based on a GA metaheuristic to design network topologies, customized for specific workloads. Workloads with structured and unstructured communication patterns are considered and topologies are evolved separately for different fitness metrics (time, structure, and contention). To quantify the change in the structure of the topology and contention (relative to a communication motif) we apply EMD, noting that the timing can decrease with the increase/decrease of the number of hops-per-byte and increase/decrease with the observed queue length on packet insertion. This indicates that, despite the fine-grained nature of these distributions, there are significant impacts on communication which are not captured by these distributions.

Acknowledgment

We thank the reviewers for their helpful feedback. This research used funding and resources supported by the U.S. DOE Office of Science, Office of Advanced Scientific Computing Research, under award 66150: "CENATE - Center for Advanced Architecture Evaluation". The Pacific Northwest National Laboratory is operated by Battelle for the U.S. Department of Energy under contract DE-AC05-76RL01830.

References

[1] "Ember Communication Pattern Library," https://github.com/sstsimulator/ember, accessed: 2022-01-01.

[2] "Pegasus Workflow Execution Instances," https://github.com/wfcommons/pegasus-instances, accessed: 2022-12-30.

[3] "Structural Simulation Toolkit (SST) Macroscale Element Library," https://github.com/sstsimulator/sst-macro, accessed: 2023-01-05.

[4] (2021) Using the mpi profiling library. [Online]. Available: https://www.ibm.com/docs/en/ppt/2.3.0?topic=libraries-using-mpi-profiling-library

[5] (2022) Minivite: Mpi+openmp implementation of the first phase of louvain method for graph community detection. [Online]. Available: https://github.com/ECP-ExaGraph/miniVite

[6] S. G. Aksoy, P. Bruillard, S. J. Young, and M. Raugas, "Ramanujan graphs and the spectral gap of supercomputing topologies," *The Journal of Supercomputing*, pp. 1–37, 2020.

[7] J. Ang, A. A. Chien, S. D. Hammond, A. Hoisie, I. Karlin, S. Pakin, J. Shalf, and J. S. Vetter, "Reimagining codesign for advanced scientific computing: Report for the ascr workshop on reimagining codesign," USDOE Office of Science (SC)(United States), Tech. Rep., 2022.

[8] M. Besta and T. Hoefler, "Slim fly: A cost effective low-diameter network topology," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '14. IEEE Press, 2014, p. 348–359. [Online]. Available: https://doi.org/10.1109/SC.2014.34

[9] A. Bhatele, A. R. Titus, J. J. Thiagarajan, N. Jain, T. Gamblin, P.-T. Bremer, M. Schulz, and L. V. Kale, "Identifying the culprits behind network congestion," in *2015 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2015, pp. 113–122.

[10] F. R. K. Chung, *Spectral graph theory*, ser. CBMS Regional Conference Series in Mathematics. Published for the Conference Board of the Mathematical Sciences, Washington, DC; by the American Mathematical Society, Providence, RI, 1997, vol. 92.

[11] G. Csardi, T. Nepusz *et al.*, "The igraph software package for complex network research," *InterJournal, complex systems*, vol. 1695, no. 5, pp. 1–9, 2006.

[12] T. Feder, A. Guetz, M. Mihail, and A. Saberi, "A local switch markov chain on given degree graphs with application in connectivity of peer-to-peer networks," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*. IEEE, 2006, pp. 69–76.

[13] J. Friedman, "A proof of Alon's second eigenvalue conjecture," in *Proc. of the thirty-fifth ACM Symposium on Theory of Computing*, 2003.

[14] ——, "A proof of Alon's second eigenvalue conjecture and related problems," *Mem. Amer. Math. Soc.*, vol. 195, no. 910, pp. viii+100, 2008. [Online]. Available: https://doi.org/10.1090/memo/0910

[15] M. Gen, R. Cheng, and L. Lin, *Network models and optimization: Multiobjective genetic algorithm approach*. Springer Science & Business Media, 2008.

[16] M. Gen, R. Cheng, and S. S. Oren, "Network design techniques using adapted genetic algorithms," *Advances in Engineering Software*, vol. 32, no. 9, pp. 731–744, 2001.

[17] M. Gen, A. Kumar, and J. R. Kim, "Recent network design techniques using evolutionary algorithms," *International Journal of Production Economics*, vol. 98, no. 2, pp. 251–261, 2005.

[18] S. Ghosh, M. Halappanavar, A. Tumeo, A. Kalyanaraman, and A. H. Gebremedhin, "Minivite: A graph analytics benchmarking tool for massively parallel systems," *2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. [Online]. Available: https://par.nsf.gov/biblio/10099628

[19] S. Heldens, P. Hijma, B. V. Werkhoven, J. Maassen, A. S. Z. Belloum, and R. V. Van Nieuwpoort, "The landscape of exascale research: A data-driven literature analysis," *ACM Comput. Surv.*, vol. 53, no. 2, mar 2020. [Online]. Available: https://doi.org/10.1145/3372390

[20] A. Hoisie, G. Johnson, D. J. Kerbyson, M. Lang, and S. Pakin, "A performance comparison through benchmarking and modeling of three leading supercomputers: Blue gene/l, red storm, and purple," in *Proceedings of the 2006 ACM/IEEE Conference on Supercomputing*, ser. SC '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 74–es. [Online]. Available: https://doi.org/10.1145/1188455.1188534

[21] N. E. Jerger, T. Krishna, and L.-S. Peh, *On-chip networks*. Springer Nature, 2022.

[22] M. Jerrum and A. Sinclair, "Fast uniform generation of regular graphs," *Theoret. Comput. Sci.*, vol. 73, no. 1, pp. 91–100, 1990. [Online]. Available: https://doi.org/10.1016/0304-3975(90)90164-D

[23] L. V. Kantorovich, "Mathematical methods of organizing and planning production," *Manage. Sci.*, vol. 6, no. 4, p. 366–422, jul 1960. [Online]. Available: https://doi.org/10.1287/mnsc.6.4.366

[24] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," ser. ISCA '08. USA: IEEE Computer Society, 2008, p. 77–88. [Online]. Available: https://doi.org/10.1109/ISCA.2008.19

[25] T. Kuhr, C. Pulvermacher, M. Ritter, T. Hauth, and N. Braun, "The belle ii core software: Belle ii framework software group," *Computing and Software for Big Science*, vol. 3, pp. 1–12, 2019.

[26] H. Lee, M. Turilli, S. Jha, D. Bhowmik, H. Ma, and A. Ramanathan, "Deepdrivemd: Deep-learning driven adaptive molecular simulations for protein folding," in *2019 IEEE/ACM Third Workshop on Deep Learning on Supercomputers (DLS)*. IEEE, 2019, pp. 12–19.

[27] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, D. Hillis, B. C. Kuszmaul, M. A. St. Pierre, D. S. Wells *et al.*, "The network architecture of the connection machine cm-5," in *Proceedings of the fourth annual ACM symposium on Parallel algorithms and architectures*, 1992, pp. 272–285.

[28] B. Li, R. Arora, S. Samsi, T. Patel, W. Arcand, D. Bestor, C. Byun, R. B. Roy, B. Bergeron, J. Holodnak *et al.*, "Ai-enabling workloads on large-scale gpu-accelerated system: Characterization, opportunities, and implications," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE, 2022, pp. 1224–1237.

[29] Y. Rubner, C. Tomasi, and L. Guibas, "A metric for distributions with applications to image databases," in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, 1998, pp. 59–66.

[30] A. Singh, "Load-balanced routing in interconnection networks," Ph.D. dissertation, Stanford University, 2005.

[31] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, 2012, pp. 225–238.

[32] E.-G. Talbi, *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.

[33] C. Tan, C. Xie, A. Li, K. J. Barker, and A. Tumeo, "Opencgra: An open-source unified framework for modeling, testing, and evaluating cgras," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*. IEEE, 2020, pp. 381–388.

[34] M. Y. Teh, Z. Wu, M. Glick, S. Rumley, M. Ghobadi, and K. Bergman, "Performance trade-offs in reconfigurable networks for hpc," *Journal of Optical Communications and Networking*, vol. 14, no. 6, pp. 454–468, 2022.

[35] M. Y. Teh, S. Zhao, P. Cao, and K. Bergman, "Enabling quasi-static reconfigurable networks with robust topology engineering," *IEEE/ACM Transactions on Networking*, 2022.

[36] A. Valadarsky, M. Dinitz, and M. Schapira, "Xpander: Unveiling the secrets of high-performance datacenters," in *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, 2015, pp. 1–7.

[37] W. Wang, M. Khazraee, Z. Zhong, M. Ghobadi, Z. Jia, D. Mudigere, Y. Zhang, and A. Kewitsch, "Topoopt: Co-optimizing network topology and parallelization strategy for distributed training jobs," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, vol. 2023.

[38] S. Young, S. Aksoy, J. Firoz, R. Gioiosa, T. Hagge, M. Kempton, J. Escobedo, and M. Raugas, "Spectralfly: Ramanujan graphs as flexible and efficient interconnection networks," in *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2022, pp. 1040–1050.