Seeds For A Heterogeneous Interconnect

Adam Hackett*, Deepak Ajwani§, Shoukat Ali†, Steve Kirkland* and John P. Morrison‡

*Hamilton Institute
National University of Ireland Maynooth, Ireland
Email: adam.hackett/stephen.kirkland@nuim.ie

†Exascale Systems Group
IBM Research - Ireland
Email: shoukat.ali@ie.ibm.com

‡ The Centre for Unified Computing
University College Cork, Ireland
Email: j.morrison@cs.ucc.ie

§ Industrial Mathematics and Operations Research Group
Emerging Computing Technologies
Bell Laboratories, Ireland
Email: deepak.ajwani@alcatel-lucent.com

***Abstract*—Traditionally, a parallel application is partitioned, mapped and then routed on a network of compute nodes where the topology of the interconnection network is known beforehand and is homogeneous. However, such homogeneity in interconnects is rarely required or needed for several important classes of applications. Nevertheless such interconnects are designed this way, i.e., with redundant links, to accommodate the communication patterns of a wide range of applications. However, with recent advances in technology for optical circuit switches, it is now possible to construct a network with much fewer links, and to make the link endpoints configurable to suit the communication pattern of a given application. While this is economical (saving both links and the power to run them), it raises the difficult problem of how to configure the network and how to reconfigure it quickly when the application's communication pattern changes. Since the space of all configurable topologies is large and determining the quality of a topology is a time-consuming process, it is not feasible to explore the entire space.**

One way of dealing with this limitation is to start the search from a "good" initial topology and then conduct a restricted search around it. The success of such a strategy crucially depends on the choice of the initial or *seed* topology. In the past, such an initial topology was computed by mimicking the communication requirements of the application. In this paper, we propose a different approach by showing that interconnect topologies such as chordal rings (circulant graphs) chosen based on metrics such as bisection width and average shortest path length can provide a better starting point. The topology obtained by searching around such an initial topology provides almost as good a performance as an application-specific initial topology, and the search time is *significantly reduced.*

I. Introduction

One of the fundamental problems in parallel and distributed computing is to partition an application into parallel components, map the components onto a set of compute nodes, and then route the communications between the components on the network that interconnects the compute nodes. In a traditional setting, the topology of the interconnection network is known beforehand and is homogeneous,i.e., all nodes have the same degree, e.g., as in $2d$ torus and fat tree interconnects. However, several studies (e.g., [1]) suggest that the homogeneity (elsewhere known as regularity) of such interconnects is rarely required or needed for several important classes of applications. However such interconnects are designed this way, i.e., with redundant links, to accommodate the communication patterns of a wide range of applications. But for a given application, several links might just be redundant, and could have been removed to save both cost and the power. With recent advances in technology for optical circuit switches, it is now possible to construct a network at run-time, i.e., the link endpoints are configurable and are configured at run-time to suit the communication pattern of a given application. While this move from familiar homogeneity to heterogeneity

is economical (saving both links and the power to run them), it raises the difficult problem of how to configure the network and how to reconfigure it quickly when the application's communication pattern changes.

At the same time, the freedom to establish a heterogeneous degree distribution on the nodes of the network to suit the requirements of an application is a potentially rich vein of theoretical investigation. Whereas the traditional parallel computing task has been to assign computational clusters and a communication pattern to a static interconnect in order to elicit, for example, the best possible throughput, one can now explore the possibility of improving performance even further by selecting a particular heterogeneous degree distribution on the nodes of the network such that the resulting topology has characteristics that are advantageous to a given task graph. In view of this emerging reciprocity between software and hardware design (possibly termed co-design elsewhere), it is natural that one design an iterative approach that takes turns in optimizing the application partitioning and routing for a given interconnect topology, and optimizing the interconnect topology to alleviate the bottlenecks identified in the partitioning and routing phase. While the first optimization is well known and well studied in the literature (e.g., [2], [**?**], [3]), the latter optimization of the interconnect topology has not been well studied. This is mostly because configurable optical switches have only recently become feasible in terms of price and switching time, and have therefore recently afforded configuration of the interconnect topology on an application-by-application basis. Accordingly, there has been considerable recent research (e.g., [4], [5], [**?**]) in tailoring the interconnect to suit the requirements of a given application.

Since the space of all configurable topologies is large and determining the quality of a topology is a time-consuming process, it is not feasible to explore the entire space.[1] One way of dealing with this limitation is to start the search from a "good" initial topology and then conduct a restricted search around it. This restricted search can be conducted using the iterative co-optimization framework described above.

The success of such a strategy crucially depends on the choice of the initial or *seed* topology. While the starting point for application partitioning is clear, it is not so clear where one would start looking when searching for an optimal topology from amongst the space of topologies allowed by a given configurable optical switch. In the sparse existing work on configurable interconnect design (e.g., [6], [5]) the choice of the initial topology is driven by the communication requirements of the application rather than traditional a priori classifiers on topology such as diameter, bisection width and average shortest path length. In this paper, we shall argue that choosing the topology from among an interesting class of graphs, based on some traditional topology metrics, remains a useful approach. In particular we show that chordal rings (circulant graphs) chosen based on bisection width and average shortest path length provide better starting points. The topology obtained by searching around such initial topologies provides almost as good a performance as application-specific initial topologies, but *the search time is significantly reduced.*

The rest of this paper is organized as follows. We introduce some preliminaries in Section II, and motivate our choice for an initial topology in Section III. We present out experiments and results in Section IV. We conclude in Section V.

II. Preliminaries

In this section we provide an outline of the problem to be addressed.

A. Space of Configurable Topologies

We consider an architecture with configurable interconnect topology (e.g., using an optical circuit switch). This allows one to configure any topology $H(V_H, E_H)$ satisfying the constraint that the (i) maximum degree of a node in H is at most $d_{\max}$ (referred to as the *maximum degree constraint*) and the (ii) total number of links in H is at most $E_{\max}$ (referred to as the *maximum edge constraint*). These constraints derive respectively from the fact that each compute node has a limited number of ports to connect to the configurable switch, and that the switch itself has a limit on the number of links that it can maintain simultaneously.

We assume that each compute node in H has the same processing speed, denoted $\underline{S_{\mathrm{C}}}$. We also assume that the bandwidth on each link in H is identical, denoted $\underline{S_{\mathrm{L}}}$.

B. Stream-computing Application Task Graphs

Stream computing is an important computing model that captures applications like real-time analysis of financial and medical data [7], audio/video systems, continuous database queries, intelligent transportation systems [8] and analysis of dynamic social networks [9], [10]. These applications are termed as stream computing applications, in that they consist of processing continuous, high-volume data-streams in real time. Such applications

[1] Different reconfigurable switches may place different constraints on the topology configuration, e.g., the maximum degree allowed at any network node or the maximum number of edges allowed in the entire network. The space of configurable topologies consist of all topologies satisfying these constraints.

are particularly suited for utilizing a configurable parallel architecture, like the one enabled by a configurable optical switch ([6], [1], [11], [12]), because these applications have long duration flows (in seconds) that can easily amortize the cost of configuring the switch (usually in a few 10s of millisecs).

We therefore consider stream computing applications in this particular investigation, and represent their computational task graph as an undirected graph $G(V_G, E_G)$. A vertex $u \in V_G$ denotes a computational kernel (e.g., a split, join or a filter) and its weight $w_V(u)$ quantifies the average amount of computation that must be performed in the kernel to produce an element of output. An edge $\{u, v\} \in E_G$ represents a data-stream (capturing the data dependency) between the kernels u and v and its weight $w_E(\{u, v\})$ represents the average amount of data transfer between the two kernels.

C. Suitability of a Topology for a Given Application

To evaluate the suitability of a topology H for a stream-computing application task graph G, we simulate a run of G on H. In this run, each vertex $v_i \in V_G$ is mapped to a compute node $u_i \in V_H$ and each edge $\{v_i, v_j\} \in E_G$ is routed along a sequence of links (or *path*) in H. Let $\mu(v_i)$ determine the node to which a particular vertex v_i is mapped, and let $\rho(v_i, v_j)$ determine the sequence of links that are used to route an edge $\{v_i, v_j\} \in E_G$. Given such mapping and routing schemes, the computational load on each node $u_i \in V_H$ is

$$w_V(u_i) = \sum_{\substack{u_i = \mu(v_i) \\ v_i \in V_G}} w_V(v_i), \tag{1}$$

and the communication load over a link $\{u_i, u_j\} \in E_H$ is

$$w_E(\{u_i, u_j\}) = \sum_{\substack{\{u_i, u_j\} \in \rho(v_i, v_j) \\ \{v_i, v_j\} \in E_G}} w_E(\{v_i, v_j\}). \tag{2}$$

From these equations we define the throughput of a compute node u_i to be $S_C / w_V(u_i)$ and the throughput of a link $\{u_i, u_j\}$ to be $S_L / w_E(u_i, u_j)$. The *computation throughput* of the system is then the minimum throughput over all compute nodes, and the *communication throughput* of the system is the minimum throughput over all links. The *system throughput* is the smaller of the computation and communication throughputs. This throughput is used as a performance metric to determine the suitability of a topology for the given application. Note that this definition of throughput arises out of stream computing applications, where we view the compute nodes and communication links as processing units running concurrently so that the overall throughput is equal to the throughput of the slowest processing unit.

D. Iterative Co-Optimization Framework

Optimizing system throughput is contingent on a good algorithm for mapping and routing an application to the topology. Since most variants of topology-aware partitioning, mapping and routing algorithms are NP-hard, one often uses heuristics to obtain good solutions. Even with heuristics, computing good solutions to these problems remains a time-consuming operation. Thus, we can only explore a very small portion of the space of configurable topologies to find a good topology.

One way to limit this search is to start with a good initial topology and iteratively adapt it to the application [5]. Each iteration of the adaptation process:

- computes a partitioning, mapping and routing solution for the given topology
- identifies the bottleneck (either a link or a compute node)
- modifies the topology to alleviate the identified bottleneck

The modification step in each iteration removes at most two links from the topology and adds two neighboring links to replace them. Thus, the modified topology is in a certain neighborhood of the original topology in the space of configurable topologies.

Since each iteration involves the time-consuming operation of computing the mapping and routing solution, we can only afford to do a limited number of these iterations. We perform these iterations until there is no improvement in the system throughput for a pre-specified number of consecutive iterations. Therefore, the iterative procedure amounts to a restricted search in the topology space around the initial topology.

The subspace of topologies that gets explored by the iterative procedure crucially depends on the initial topology. Thus, the choice of initial topology is vitally important to obtaining a final topology that provides a good system throughput for the given application. Also, the time to convergence of the iterative framework depends on the choice of initial topology.

III. Choosing the Initial Topology

The framework in [5] computes the initial topology by mimicking the communication pattern of the application. It computes a partitioning of the application graph (using any partitioning library like METIS [13] or SCOTCH [14]) into n clusters and then contracts each cluster to a single node forming a *condensed graph* (also referred to as quotient graph in the literature).

The weight of an edge in a condensed graph is the sum of weights of edges between the two partitions in the application graph. It then removes some low-weight edges to satisfy the maximum degree and maximum edge constraints and treats the resultant graph (ignoring its weights) as the initial topology. Since this topology mimics the communication pattern of the application, it is argued that the application should map well to this topology, yielding good performance (high throughput for a streaming application).

There are many shortcomings of the above process. Firstly, the initial topology crucially relies on the computed partitioning. A bad partitioning can lead to a condensed graph that is very different in structure than the application. Secondly, even if the condensed graph is similar in structure, the maximum allowed number of edges or the degree may be significantly less than the condensed graph and this may make the resultant topology very different in structure from the condensed graph. Specifically, re-routing the traffic that originally existed on the removed links through new paths in the topology can cause congestion in some links leading to poor performance. There is no purpose built capacity in a condensed graph to handle such re-routing. And most importantly, even if the topology mimics the communication pattern of the application well, the process of routing and re-routing can still lead to poor communication throughput, particularly if the resultant topology has poor connectivity.

To overcome these shortcomings, we propose an alternative framework. Rather than starting with a condensed graph, we initialize our topology search with a topology with certain desirable characteristics. We focus our attention on chordal rings. A chordal ring is a k-regular circulant graph formed by the addition of edges between pairs of vertices at specified distances along the ring. More precisely, a ring with vertices labelled $i = 0, \ldots, n-1$ and an associated set of chord lengths, $Q \subseteq \{q_1, q_2, \ldots, q_{\lfloor n/2 \rfloor}\}$, forms the chordal ring $C(n; q_1, \ldots, q_{|Q|})$, if for each $q_j \in Q$ there is an edge from vertex i to vertex $(i + q_j) \mod n$. The degree of the graph is determined by the choice of Q. In general $k = 2|Q| + 2$ (see Fig. 1). For even values of n, chords of length $n/2$ yield $k = 2|Q| + 1$. Recall that k is the vertex degree in this k-regular graph.

This class of graphs has been used extensively in theoretical studies of interconnect design (e.g., [15], [16], [17]), and it is known to possess a number of structural properties that have traditionally been seen as vital for robustness against node or link failures [18], [19]. Given that the maximum edge constraint enforced by the configurable optical switch may result in removal of some edges from our chosen initial topology (even if it is a condensed graph), it makes sense to start with a topology that is known to be robust against such link failures (*intentional link removal* is a type of link failure).

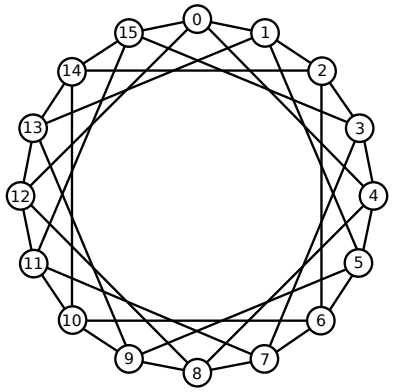

Fig. 1: The chordal ring $C(16; 4)$.

Furthermore, as mentioned in Sec. II-C, our optimization criterion is to maximize throughput of a streaming application mapped to a (configured) network topology. To that end, application edges are mapped to the network topology edges so as to minimize the maximum congestion (or load) on any network topology edge. The same characteristics of chordal rings that make them attractive for robustness against failed links also make them attractive for robustness against congested links. In both cases, the structure of the chordal rings makes it likely that many communication paths will remain available even in the face of congestion of certain links. Note that this is not a design feature of the condensed graphs.

We note that other graphs like de Bruijn graphs [20] or Kautz graphs [21] also provide such robustness. However, the chordal rings provide much more design flexibility in terms of choosing the required number of vertices while also satisfying the maximum degree constraint. The maximum edge constraint can then be satisfied by removing a random subset of edges (while preserving the connectivity).

In this paper, we assume that $d_{\max}$ is 4. To satisfy this particular maximum degree constraint, we focus our discussion on simple degree four chordal rings $C(n; q)$ [22], i.e., a ring in which all chords have the same length, q. There is a number of interesting properties unique to this particular subclass. Iwasaki *et al.* [23] have shown that for every graph in $C(n; q)$ there are four independent spanning trees rooted at each vertex. In

other words, between every pair of (source, destination) vertices there are four edge-disjoint paths, the maximum possible number for a 4-regular graph.

Another interesting property of $C(n;q)$ is their bisection width. Bisection width is important for our particular problem because a higher bisection width will translate into a larger number of paths between any two bisections, thereby permitting a better congestion alleviation between the two given bisections.

For large values of n, the bisection width of chordal rings is usually better than that of 2-D torus. For instance, the bisection width of $C(2^k;4)$ is known to be 2^{k-1} [24], suggesting a linear growth as a function of n, while that of 2-D torus is $2n^{1/2}$, a sub-linear function. This suggests that such a chordal ring may be a better candidate for a seed topology than a 2-D torus.

Finding the bisection width of a general chordal ring is a hard problem. Here we present a lower bound on bisection width of $C(n;q)$. If G is a graph on n vertices and a is its algebraic connectivity (i.e. smallest strictly positive eigenvalue of its Laplacian matrix) then the bisection width for G is bounded below by $\lceil \frac{na}{4} \rceil$ [25]. For $C(n;q)$, the algebraic connectivity is given by $a = \min_{k=1,\ldots,n-1}\{4 - 2\cos\left(\frac{2\pi k}{n}\right) - 2\cos\left(\frac{2\pi kq}{n}\right)\}$. Substituting this value, one gets the lower bounds of 3, 6, 8, 6, 8 and 5 for $C(16;2)$, $C(16;3)$, $C(16;4)$, $C(16;5)$, $C(16;6)$, and $C(16;7)$, respectively.

To compute the exact bisection width of these chordal rings, we conducted an exhaustive search by considering the cut value of all possible equal size bipartitionings and taking the minimum. From this, we concluded that the bisection width of $C(16;2)$, $C(16;3)$, $C(16;4)$, $C(16;5)$, $C(16;6)$, and $C(16;7)$ is exactly 6, 8, 8, 8, 10 and 8, respectively. As a matter of comparison, note the bisection width of 2-D torus with 16 vertices is 8.

IV. Experiments and Results

We performed our experiments for stream computing applications. To generate a random application graph emulating the communication pattern of a stream computing application, we use a graph generator from [26].

We characterize our experiments with the following parameters: T_{init}, N_{nod}, N_{ver}, $d_{\max}$, $E_{\max}$, S_{C}, and S_{L}, where T_{init} is the initial topology, N_{nod} is the number of compute nodes in the topology and N_{ver} is the number of vertices in the application graph. Let $\mathcal{T}$ be the set of seed topologies being compared in an experiment. Let a trial be defined as one execution of the sequence "generate an application graph of N_{ver} vertices, fork $|\mathcal{T}|$ ways, use $\mathcal{T}_i$ as T_{init} in the i-th fork as an initial topology, satisfy constraints implied by $d_{\max}$ and $E_{\max}$, and search the space around this initial topology using the iterative procedure in [5]." Two trials differ from each other only in the random numbers used to seed the application graph generator and the graph partitioner. That is, for each new trial we create a new random application graph, and also seed our own algorithm with a new seed.

We focus on 16-node topologies with $d_{\max} = 4$ and $22 \leq E_{\max} \leq 32$. Note that even for this restricted setting, the number of graphs satisfying the two constraints is quite large and exploring the entire search space (every time we re-configure the switch) is unfeasible.

We report here results for several experiments. By varying the ratio between S_{C} and S_{L}, we can alter the bottleneck from computation to communication. A high ratio implies that computation is less likely to be the bottleneck as the compute nodes can process the computation load faster than the links can move the data around.

To ensure that our reported results are statistically rigorous, we consider an experiment completed only when enough trials have been performed to give us a certain level of confidence in the results. Let the *imprecision*, of a set of values of x be defined as the half-width of the 95% confidence interval divided by the mean of x. An experiment is completed when enough trials have been performed to give a precision of at least 5% for the throughput obtained. This reduces the dependence of our results on a "lucky" selection of application graphs or the parameters for the search algorithms.

The initial topology of a condensed graph is used as our baseline in this paper. All other topologies are compared to this baseline. Output of a given trial gives the following information for each seed topology T_i: δ_i^{thru}, the ratio of improvement in throughput achieved by T_i over the condensed graph to the throughput achieved by condensed graph, and δ_i^{time}, the ratio of improvement in time to convergence achieved over the condensed graph to time to convergence for topology T_i. At the end of an experiment, we compute the following additional metrics for each seed topology T_i: Δ_i^{thru}, the average value of δ_i^{thru} over all trials and Δ_i^{time}, the average value of δ_i^{time} over all trials.

Figure 2 shows results for a subset of experiments in which only the seed topology, T_{init}, maximum number of allowed edges, $E_{\max}$, and the speed of the network links, S_{L}, were changed while other parameters were kept at N_{nod} = 16, N_{ver} = 300, and $d_{\max}$ = 4. These results show that, in general, throughput is better if the seed topology is a condensed graph. However, using a suitable chordal ring as a seed topology gave, for most of these experiments, an average throughput that was no

worse than 95% of that obtained using the condensed graph, *but* did so in a much shorter time.

There are also some specific insights highlighted in Fig. 2, which shows results for 6 experiments. For easier navigation, the subplots have been arranged so that the experimental set-up becomes more and more challenging communication wise if one moves either to the right or the top. That is, Fig. 2(c) has the most communication wise challenging (hereafter referred to as just 'challenging') set-up. This is because (i) the link speed is the lowest, at 100, of three speeds (100, 200 and 500) shown in this figure, and (ii) the number of edges simultaneously allowed in the network is the smaller, at 22, of the two values shown here (22 and 28). Figure 3 has been set up similarly to Fig. 2 except that it shows the effect on convergence time of the choice of initial topology. Notice that any move from a less challenging to a more challenging set-up shows that the rings with higher chord lengths generally do better. Specifically, one can see the following.

1) A rightward movement from (a) to (b) to (c) in Fig. 2 shows that the number of chordal rings that achieve a throughput that is within 8% of the baseline drops from 4 to 3 to 2.[2] While $C(16;6)$ and $C(16;7)$ reach 92% of the baseline throughput for all three scenarios, $C(16;5)$ drops out of the "well-performing" set of rings when S_{L} is decreased from 500 to 200. And both $C(16;4)$ and $C(16;5)$ drop off this set when S_{L} is further decreased from 200 to 100. That is, for subplot (c), the chord lengths of 2, 3, 4, and 5 are all in a set that does not give throughput competitive to that of the condensed graph. A similar change in performance is seen when one moves 'up' in Fig. 2, from (d) to (a), from (e) to (b), or from (f) to (c). For example, when we move from (d) to (a) in Fig. 2 (i.e., decreasing $E_{\max}$ to 22 while keeping S_{L} constant at 500), the rings with the smaller chord lengths, $C(16;2)$ and $C(16;3)$, start performing quite poorly. As a specific example, the throughput improvement worsens by about 18% for $C(16;2)$.
2) A similar rightward movement in Fig. 3 shows that the improvement in time taken to converge has in general gotten better. Note that the savings in the convergence time are quite significant. For example, for Fig. 3(a), the iterative algorithm converged sooner, as compared to the baseline, by about 19% to 31%, depending on the choice of chord length. The corresponding value for 2-D torus was 20%. When we move one step right to Fig. 3(b), the convergence time improvement for the chordal rings gets in the range of 28% to 31%; for 2-D torus, 31%. Note that the apparent much higher improvements, 116% (not shown to reduce congestion on y-axis) and 54%, in convergence times for $C(16;2)$, and $C(16;3)$, respectively, are meaningless here because their performance was worse by more than 8% of the baseline. After another step to the right, in Fig. 3(c), we see time improvements of 51% and 68% for the only two rings, $C(16;6)$ and $C(16;7)$, respectively, that have survived falling off the so-called "well-performing" set. The 2-D torus saves 55% in time in this case.
3) The 2-D torus as an initial topology works just well as an appropriate chordal ring as far as the throughput improvement is concerned. However, for savings in convergence time, an appropriate chordal ring may be better (see Fig. 3(a), (e), and (f)).

One explanation of the above effect of larger chord length chordal rings "holding out" longer in the face of increasingly adverse communication environments is offered by the average shortest path length, L_{asp}, of chordal rings (Fig. 4). The number of links in the topology path that are loaded to route a given application edge is smaller for a chordal ring with a smaller L_{asp}. This reduces the maximum load on a given topology link, and subsequently increases the communication throughput. Also, as the links are removed from the topology to satisfy the constraint on the total number of edges, the load on the removed links has to be re-routed through another path. For a topology with a smaller L_{asp} value, fewer links get loaded in the process of re-routing and this results in less congestion overall.

As Fig. 4 shows, the L_{asp} value decreases from chord length 2 to 6, except for chord length of 5. This pattern is consistent with the throughput increase pattern shown in Fig. 2. Note that this consistency between the throughput pattern and average shortest path is more prominent for $E_{\max} = 22$ case as compared to $E_{\max} = 28$ case because more links are removed to satisfy the $E_{\max} = 22$ constraint and the flow over the removed links get re-routed through shorter paths.

The relatively large performance difference between $C(16;2)$ and $C(16;6)$ may also be attributed to a large difference in their bisection widths. For $i \in \{2, ..., 7\}$, bisection width of $C(16;i)$ is always 8 except for $C(16;2)$ and $C(16;6)$ where it is 6 and 10, respectively (cf. Sec. III). A larger bisection width means that the minimum number of links that have to be removed to bisect the ring is larger. The task of optimizing for

[2]The number 8% is chosen here arbitrarily to make our point. It just needs to be a small number.

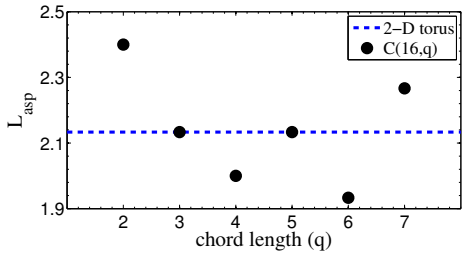

Fig. 4: L_{asp} against chord length

throughput is in part a task of finding a larger set of paths that go from one bisection of the graph to another. These are the paths that will be used to balance a large streaming load, to, in turn, increase its throughput.

Another trend that emerges from Fig. 3 is that when S_{L} is decreased for a constant $E_{\max}$, the convergence advantage of the chordal rings becomes greater. This convergence advantage of chordal rings can also be seen by stressing the system in a different way, specifically, by increasing the size of the application graph. A larger sized application graph makes it more critical for the iterative algorithm to co-optimize routing more carefully. Table I shows the effect on throughput improvement and convergence when application graphs of size 1000 are used. All other parameters are the same as in Fig. 2 except that $E_{\max}$ has been set to 26 in this case. It can be seen that moving to the larger application graph makes $C(16;2)$ (i.e., a 16-node chordal ring with chord length of 2) perform even worse with respect to throughput. For the other chordal rings, however, the throughput improvement over the baseline case of condensed graphs does not change much. But the convergence time does improve significantly in most cases.

TABLE I: The effect on throughput improvement and convergence time of using larger application graphs.

	S_{L}	1000 vertices		300 vertices	
		Δ_i^{thru}	Δ_i^{time}	Δ_i^{thru}	Δ_i^{time}
$C(16;2)$	200	-0.27	1.43	-0.16	0.68
$C(16;3)$	200	-0.04	0.43	-0.04	0.27
$C(16;4)$	200	-0.02	0.51	-0.03	0.35
$C(16;5)$	200	-0.07	0.37	-0.06	0.36
$C(16;6)$	200	-0.02	0.47	-0.01	0.22
$C(16;7)$	200	-0.03	0.36	-0.02	0.24
2-D Torus	200	-0.02	0.31	-0.02	0.30

V. Conclusions and Future Work

We present in this paper one method for seeding a pre-existing technique for determining the degrees of nodes in a heterogeneous interconnection network. In a traditional setting, the topology of the interconnection network is known beforehand and is homogeneous, i.e., all nodes have the same degree, even if a given application does not need such homogeneity. With recent advances in technology for optical circuit switches, it is now possible to construct a network at run-time, i.e., the link endpoints are configurable and are configured at run-time to suit the communication pattern of a given application. This raises the interesting question of how to find a topology well-suited to a given application. A natural way to solve this problem and the interdependent problem of mapping the application to the topology is to iteratively optimize the two problems in turns: compute a partitioning, mapping and routing scheme for a topology and then modify the topology in response to the bottleneck identified in the mapping and routing process. A fundamental question in this framework is the choice of seed topology to start this iterative co-optimization.

In this paper, we propose to initialize the iterative co-optimization with chordal rings and torus topologies. These topologies are chosen for their structural properties such as robustness to link failures, good bisection width and small average shortest paths. We show that seeding the iterative framework with these topologies gives almost as good a performance as seeding with topologies that mimic the communication patterns of the application. In addition, such seeding was found to *reduce the convergence time of the iterative framework significantly*. A faster decision can allow the system to make better use of the reconfigurable switch in a dynamic setting. This paper, therefore, is one of the first steps towards developing fast network configuration algorithms for reconfigurable heterogeneous networks.

References

[1] J. Shalf, S. A. Kamil, L. Oliker, and D. Skinner, "Analyzing ultra-scale application communication requirements for a reconfigurable hybrid interconnect," in *ACM/IEEE Conf. on Supercomputing (SC '05)*, 2005.

[2] T. Agarwal, A. Sharma, and L. V. Kalé, "Topology-aware task mapping for reducing communication contention on large parallel machines," in *20th IEEE Int'l Conf. on Parallel and Distributed Processing (IPDPS '06)*, 2006.

[3] A. Bhatele and L. V. Kalé, "Heuristic-based techniques for mapping irregular communication graphs to mesh topologies," in *13th IEEE Int'l Conf. on High Performance Computing & Communication (HPCC '11)*, 2011.

[4] D. Lugones, K. Katrinis, and M. Collier, "A reconfigurable optical/electrical interconnect architecture for large-scale clusters and datacenters," in *9th ACM Conf. on Computing Frontiers*, 2012.

[5] D. Ajwani, S. Ali, and J. P. Morrison, "Graph partitioning for reconfigurable topology," in *26th IEEE Int'l Symposium on Parallel and Distributed Processing (IPDPS '12)*, 2012.

[6] S. Kamil, A. Pinar, D. Gunter, M. Lijewski, L. Oliker, and J. Shalf, "Reconfigurable hybrid interconnection for static and dynamic scientific applications," in *4th Int'l Conf. on Computing Frontiers (CF '07)*, 2007.

[7] B. Gedik, H. Andrade, K.-L. Wu, P. S. Yu, and M. Doo, "Spade: the System S declarative stream processing engine," in *2008 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD '08)*, 2008.

[8] A. Arasu, M. Cherniack, E. Galvez, D. Maier, A. S. Maskey, E. Ryvkina, M. Stonebraker, and R. Tibbetts, "Linear road: a stream data management benchmark," in *30th Int'l Conf. on Very Large Data Bases (VLDB '04)*, 2004.

[9] J. Riedy, H. Meyerhenke, D. A. Bader, D. Ediger, and T. G. Mattson, "Analysis of streaming social networks and graphs on multicore architectures," in *37th IEEE Int'l Conf. on Acoustics, Speech, and Signal Processing (ICASSP '12)*, 2012.

[10] D. Ediger, E. J. Riedy, D. A. Bader, and H. Meyerhenke, "Tracking structure of streaming social networks," in *5th Workshop on Multithreaded Architectures and Applications (MTAAP '11)*.

[11] K. J. Barker, A. F. Benner, R. R. Hoare, A. Hoisie, A. K. Jones, D. J. Kerbyson, D. Li, R. G. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. B. Stunkel, and P. Walker, "On the feasibility of optical circuit switching for high performance computing systems," in *ACM/IEEE Conf. on High Performance Networking and Computing (SC '05)*, 2005.

[12] L. Schares, X. J. Zhang, R. Wagle, D. Rajan, P. Selo, S. P. Chang, J. R. Giles, K. Hildrum, D. M. Kuchta, J. L. Wolf, and E. Schenfeld, "A reconfigurable interconnect fabric with optical circuit switch and software optimizer for stream computing systems," in *Optical Fiber Communication Conf. (OFC '09)*.

[13] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, 1998.

[14] F. Pellegrini, "Contributions au partitionnement de graphes parallle multi-niveaux / contributions to parallel multilevel graph partitioning," LaBRI, Université Bordeaux, Habilitation, 2009.

[15] S. Bujnowski, B. Dubalski, and A. Zabludowski, "Analysis of chordal rings," in *Mathematical Techniques and Problems in Telecommunications (MTPT '03)*. Centro Int'l de Mathematica, Tomar, 2003, pp. 257–279.

[16] L. Narayanan, J. Opatrny, and D. Sotteau, "All-to-all optical routing in optimal chordal rings of degree four," in *10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '99)*, 1999.

[17] J. M. Pedersen, T. M. Riaz, B. Dubalski, and O. B. Madsen, "A comparison of network planning strategies," in *10th Int'l Conf. on Advanced Communication Technology (ICACT '08)*, 2008.

[18] B. Parhami, "A class of odd-radix chordal ring networks," *CSI Journal on Computer Science and Engineering*, vol. 4, no. 2&4, pp. 1–9, 2006.

[19] ——, "Periodically regular chordal rings are preferable to double-ring networks," *J. of Interconnection Networks*, vol. 9, no. 1, pp. 99–126, 2008.

[20] N. G. de Bruijn, "A combinatorial problem," *Koninklijke Nederlandse Akademie v. Wetenschappen*, vol. 49, pp. 758–764, 1946.

[21] D. Li, X. Lu, and J. Su, "Graph-theoretic analysis of Kautz topology and DHT schemes," in *Network and Parallel Computing: IFIP Int'l Conf.*, 2004.

[22] R. F. Browne and R. M. Hodgson, "Symmetric degree-four chordal ring networks," *Computers and Digital Techniques, IEE Proc. E*, vol. 137, no. 4, 1990.

[23] Y. Iwasaki, Y. Kajiwara, K. Obokata, and Y. Igarashi, "Independent spanning trees of chordal rings," *Information Processing Letters*, vol. 69, pp. 155–160, 1999.

[24] X. Yang, D. J. Evans, and G. M. Megson, "Maximum induced subgraph of a recursive circulant," *Information Processing Letters*, vol. 95, no. 1, pp. 293–298, 2005.

[25] R. B. Boppana, "Eigenvalues and graph bisection: An average-case analysis," in *28th Annual Symposium on Foundations of Computer Science*, 1987.

[26] D. Ajwani, S. Ali, and J. P. Morrison, "Application-agnostic generation of synthetic task graphs for stream computing applications," IBM Research, Technical Report RC25181 (D1107-003), 2011.

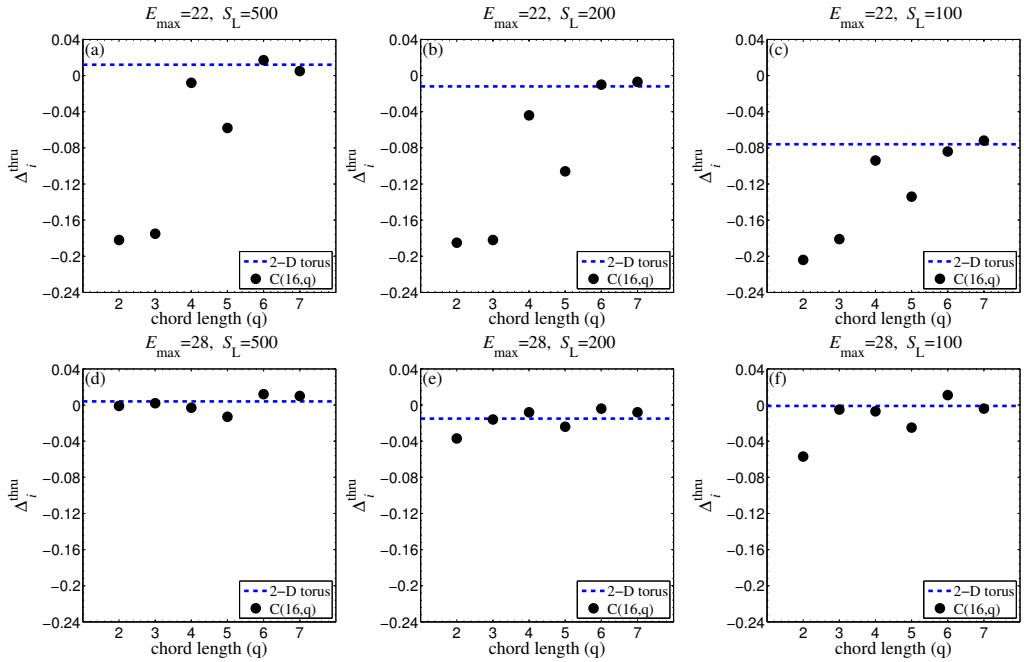

Fig. 2: The effect on throughput improvement, Δ_i^{thru}, of changes in the chord length of the 16-node chordal ring.

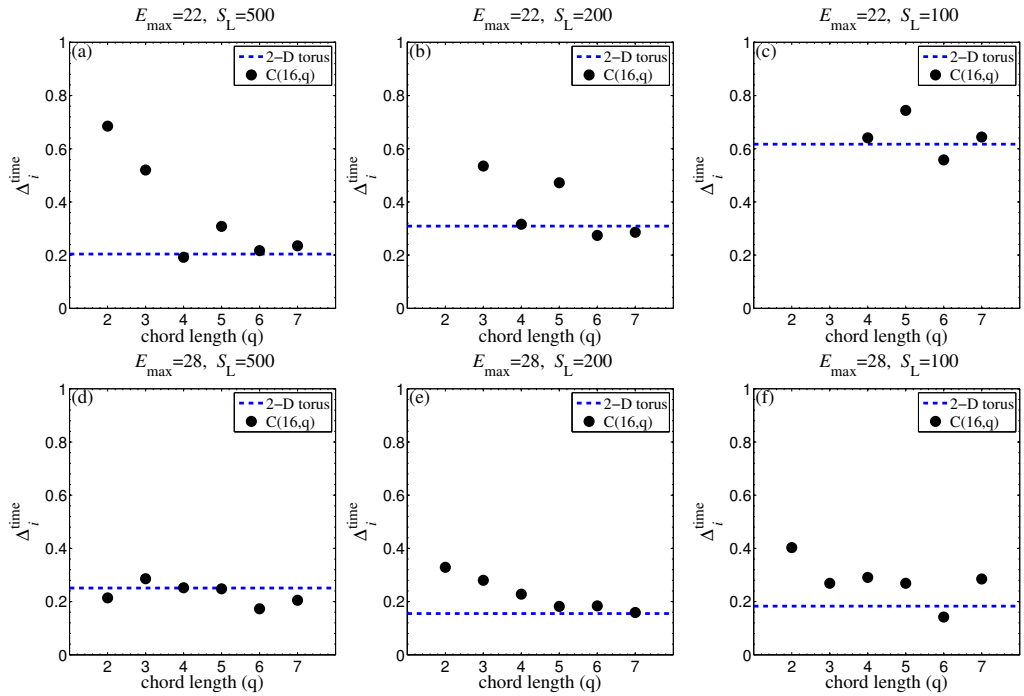

Fig. 3: The effect on convergence time, Δ_i^{time}, of changes in the chord length of the 16-node chordal ring.