Evolutionary Optimization of Multistage Interconnection Networks Performance

Jiri Jaros
Brno University of Technology
Bozetechova 2
612 66 Brno, Czech Republic
+420 54114-1207

jarosjir@fit.vutbr.cz

ABSTRACT
The paper deals with optimization of collective communications on multistage interconnection networks (MINs). In the experimental work, unidirectional MINs like Omega, Butterfly and Clos are investigated. The study is completed by bidirectional binary, fat and full binary tree. To avoid link contentions and associated delays, collective communications are processed in synchronized steps. Minimum number of steps is sought for the given network topology, wormhole switching, minimum routing and given sets of sender and/or receiver nodes. Evolutionary algorithm proposed in this paper is able to design optimal schedules for broadcast and scatter collective communications. Acquired optimum schedules can simplify the consecutive writing high-performance communication routines for application-specific networks on chip, or for development of communication libraries in case of general-purpose multistage interconnection networks.

Categories and Subject Descriptors
I.2.8 [**Artificial intelligence**]: Problem Solving, Control Methods and Search – *heuristic methods, scheduling*.

General Terms
Algorithms, Performance, Design.

Keywords
Collective communications, communication scheduling, evolutionary design, multistage interconnection networks.

1. INTRODUCTION
On-chip networks play a critical role in the performance of computing systems including high-speed network routers, embedded devices and chip multiprocessors (CMPs) [1]. Moving forward, as we integrate progressively more functionality on a single die, the communication infrastructure that binds them will play a central role in overall chip performance.

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

When the number of communicating nodes is small enough, a single switch is sufficient to interconnect them within a switched-media network. However, the number of switch ports is limited by existing VLSI technology, cost considerations, power consumption, and so on. When the number of required network ports exceeds the number of ports supported by a single switch, a fabric of interconnected switches is needed. All the connections to the network fabric and between switches within the fabric use point-to-point links as opposed to shared links. To save chip area wormhole switching [21] is usually implemented to reduce necessary buffer size. A common way of addressing the crossbar scaling problem consists of splitting the large crossbar switch into several stages of smaller switches interconnected in such a way that a single pass through the switch fabric allows any destination to be reached from any source. Topologies arranged in this way are usually referred to as multistage interconnection networks (MIN) or multistage switch fabrics [12].

The reduction in switch cost of MINs comes at the price of performance: contention is more likely to occur on network links, which degrades its performance. Contention in the form of packets blocking in the network arises due to simultaneously sharing one or more links by different message transfers.

In this paper, we want to boost the performance of MINs by designing of such communication schedules that prevent any possible link contention. Optimized communication schedules can be uploaded into switch routing tables and make profit in many parallel algorithms. For this reason, four common collective communications CC engaging all nodes in a topology based on broadcast and scatter services will be analyzed.

The optimization part of the algorithm is derived form evolutionary techniques. These techniques applied already to CC scheduling problem on hypercubes of medium size (tens of nodes) [3] were able to find optimum solutions obtained analytically. However, for some networks studied in this paper no analytic methods for scheduling exist, thus the results can be compared only with theoretical lower bound only.

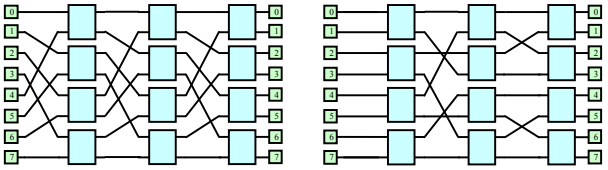
The paper is structured as follows. In Section 2 the investigated multistage interconnection networks are defined. Section 3 specifies the CC scheduling problem and presents time complexity of optimal schedules. An improved evolutionary algorithm solving this problem is proposed in Section 4. The results of CC scheduling in various network topologies are summarized and discussed in Section 5. Results obtained by evolutionary approach are discussed in Conclusion and possible future improvements are suggested.

2. MULTISTAGE INTERCONNECTION NETWORKS

A simple definition of unidirectional MINs can be found in [13], where MIN is a network generally used for the interconnection of a set of N input terminals to M output terminals (processing nodes) using sets of fixed-size switches arranged in stages. If N=M we say that the MIN is of size N. The degree of the MIN is defined as the size of crossbars used to build MIN [14].

More formally, MIN is a succession of stages of switching elements (SEs) and interconnection wires connecting N processing nodes. SEs in the most general architecture are themselves interconnection networks of small sizes. The most used SEs are hyperbars [15] and more specifically crossbars. If N is the MIN's degree and k is the SE's degree (the number of input/output ports), the minimum number of switches in a stage must be N/k.

The interconnection pattern or patterns between MIN's stages can be represented mathematically by a set of functions. Examples of such topologies, examined in this paper, cover Omega and Butterfly network. Omega network [16] implements the perfect-shuffle permutation as its interconnection pattern for each stage; see Fig. 1a. The Butterfly network [6], see Fig. 1b, is an isomorphic variation of Omega network. In contrast of perfect-shuffle exchange implemented in Omega, Butterfly is based on butterfly permutations corresponding to the computation of a one-dimensional FFT. In both cases, eight input-output ports are interconnected with three stages of 2 x 2 switches. It is easy to see that a single pass through the three stages allows any input port to reach any output port.



(a) Omega network (b) Butterfly network

Figure 1. Illustration of 8-node Omega and Butterfly network.

The main disadvantage of permutation based MINs is their zero fault-tolerance and high blocking probability. To alleviate the bottleneck consisting in only single path between an input-output pair, the multipath Clos network has been proposed [17]. Here, each network input-output pair can be connected by a path via an arbitrary middle stage. The basic version of a Clos network consists of three SE stages, as shown in Fig. 2. Clos networks of more than three stages emerge by substituting again the middle stage SEs by Clos network.

Clos networks are defined by three integers n, m, and r; n represents the number of sources which feed into each of r input stage crossbar switches. Each input stage crossbar switch has m outlets, and there are m centre stage crossbar switches. There is exactly one connection between each input stage switch and each middle stage switch. There are r output stage switches, each with m inputs and n outputs. Each middle stage switch is connected exactly once to each output stage switch.

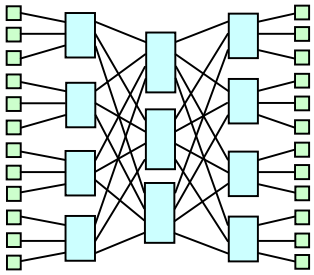


Figure 2. General form of Clos network, n=3, m=3, r=4.

The MINs described so far have unidirectional network links, but bidirectional forms are easily derived as two MINs back-to-back, folded on one another, see. Fig. 3. The overlapping unidirectional links run in different directions, thus forming bidirectional links, and the overlapping switches merge into a single switch with twice the ports (i.e., 4 x 4 switch). A representative of the class is a Fat-tree [18] topology originates in folded Butterfly network. Unlike traditional trees in computer science, fat trees resemble real trees, because they get thicker near the root.

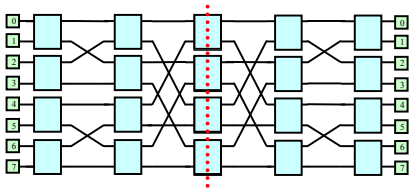


Figure 3. Unfolded version of fat-tree created by two Butterfly networks.

3. CC SCHEDULING PROBLEM

Many interactions in practical parallel programs occur in well-defined patterns involving groups of processors. Collective communications (CC) [4] involve communications among all processors connected by an interconnection network. Each CC can be seen as a set of point-to-point communications. The CC scheduling problem can be simply described as partitioning this set into as few subsets as possible that follow one another in a sequence of synchronized steps; all communications in one subset proceed in parallel. The main goal is to avoid any conflicts in shared resources – links (channels). Several messages between source-destination pairs can proceed concurrently and can be combined into a single subset if their paths are link-disjoint. All message transfers originate in processing nodes where the transported messages are created and their destinations are determined. To reach the destinations, messages are transported via intermediate stages of the MIN where the routing and switching mechanism are implemented.

Collective communications can be categorized on the number of transmitting and receiving nodes, and implemented communication service. If only one node distributes/collects message/messages to/from all other nodes, we talk about one-to-all or all-to-one communication pattern. We talk about all-to-all pattern if all nodes perform the same communication service. These communication patterns can implement two distinct services, broadcast and scatter. Broadcast service distributes the same message to all partners, whereas scatter service delivers a

private message to each partners (each node obtains a different message). Four basic types of CC will by analyzed in the paper: one-to-all broadcast (OAB), all-to-all broadcast (AAB), one-to-all scatter (OAS), and all-to-all scatter (AAS). Some other CCs, like all-to-one gather (AOG), have the same complexity but reverse structure as the basic four types.

Regardless the MIN's graph topology, there are known theoretical lower bounds on the number of communication steps. The broadcast communication (OAB) in a wormhole-switched network cannot be done in less than s steps, where $s = \lceil \log_2 N \rceil$ is given by the number of nodes informed in each step, that is initially 1, 1+1 after the first step, $2 + 2 = (2)^2$ after the second step, etc.,…, and $2^s \geq N$ nodes after step s.

In case of OAS communication, because each node can inject not more then one message at a time, the lower bound is N-1 steps.

A similar bound is applied to AAB communication, since each node has to accept N–1 distinct messages, the lower bound should be N–1 steps. Unfortunately, it would be possible only in the case, that any two communications from different sources targeted to different destinations can be realized in the same step without conflict (blocking) [19]. Considering this limitation, the lower bound cannot be reached for some of proposed networks.

For AAS communication pattern each of N processor sends an individual message to each of N-1 partners. A lower bound for AAS can be obtained considering that one half of messages from each processor cross the bisection and the other half do not. There will be altogether 2 (N/2) (N/2) of such messages in both ways and up to B_C messages in one step, where B_C is the network bisection width [4]. Considering the same limitation as in case of AAB, the reachable lower bound will be slightly higher.

4. CC SCHEDULING ALGORITHM

The selection of Evolutionary Algorithms (EA) for the scheduling problem has been justified already in [3]. Although a new methodology of designing near-optimal CC schedules is independent of the particular evolutionary algorithm, we restricted ourselves only to a simple EDA evolutionary algorithm without gene dependencies (UMDA) in this work.

Univariate Marginal Distribution Algorithm (UMDA) [7] is a very simple EDA [10] (Estimation of Distribution Algorithm) which does not reflect any interaction between genes (variables/solution parameters). The main advantages of this algorithm are better mixing of genetic material than is possible in standard GA [11], very simple implementation and much faster execution than more complex EDAs like BOA (Bayesian Optimization Algorithm [10]) algorithm. Of course, any other EA can be employed. Basic comparison of a success rate and execution time of other types of EA applied to CC scheduling problem can be found in [8], [9].

The following subsections detail the evolutionary approach. Section 4.1 shows the global data structure and a preprocessing phase. Section 4.2 describes how the dataset is encoded, Section 4.3 presents the evaluation function used in EA and Section 4.4 briefly describes acceleration and restoration heuristics used to increase a success rate and reduce execution time required to reach a sufficient result. Parameters of used EA (UMDA) are outlined in Section 4.5.

4.1 Preprocessing Phase

An input data structure maintains a MIN's topology description, a definition of CC and sets of senders, receivers and intermediate switches. The topology description is saved in the form of a processing nodes' and switching elements' neighbors list, where the nodes/switches are considered to be neighbors only if they are connected by a simple direct link.

After an input file is loaded, the data have to be preprocessed. The preprocessor takes the topology description and finds all paths (shortest ones in the case of minimal routing) between all source-destination node pairs and stores them into a special data structure. This task is performed by a modified well known Dijkstra's algorithm [20].

4.2 Encoding

As broadcast and scatter CCs are completely different communication services, candidate solutions are encoded in separate ways.

An optimal OAS schedule designed for 8-node Omega is shown in Fig. 4. This schedule reaches the lower bound of 7 steps. The initiator, node no. 0, informs one other node in each step by means of some of the shortest paths found during preprocessing.

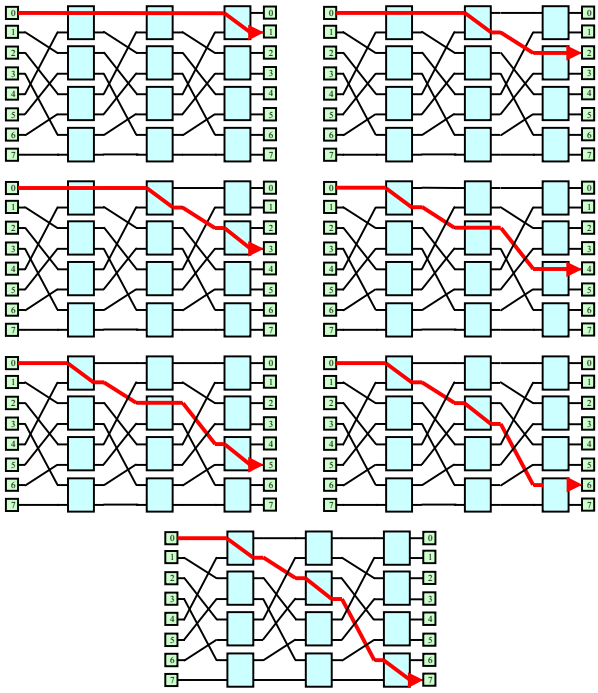


Figure 4. An OAS schedule reaching the lower bound on number of communication steps.

A direct encoding has been designed for OAS/AAS chromosome; i.e. a chromosome contains an exact description of a schedule. The chromosome contains N genes; each one represents a particular point-to-point communication between the initiator and a destination node. A gene consists of two items: a utilized path (the first component) and the used time step (the second component).

The OAS chromosome corresponding to Fig. 4 is displayed in Fig. 5. The gene no. 0 does not include any value since it is not necessary to transmit the message to itself through the network. The node no. 0 is the initiator. The allele of the gene no. 1 implies employing of the path 1 (there is only a single path from 0 to 1) and finishing during the first communication step. Analogously, the allele of gene no. 7 implies utilizing of the single path between the initiator and node no. 7. This communication will be executed during the seventh step.

An AAS chromosome is created by extending the vector to a matrix, each row of which corresponds to one of OAS communication.

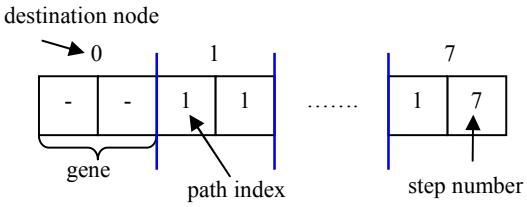


Figure 5. The structure of OAS chromosome for 8-node Omega network.

An optimal OAB schedule designed for 8-node Omega network is shown in Fig. 6. This schedule reaches the lower bound of 3 steps. The initiator, node no. 0 informs node no. 2 during the first communication step. Since the distributed messages are the same for all nodes, these two nodes can become initiators for the second step, such node no. 3 receives the message from the node no. 0, and node no. 7 from the node no. 2. Finally, the message is distrusted by nodes 0, 2, 3 and 7 to nodes 6, 5, 4 and 1.

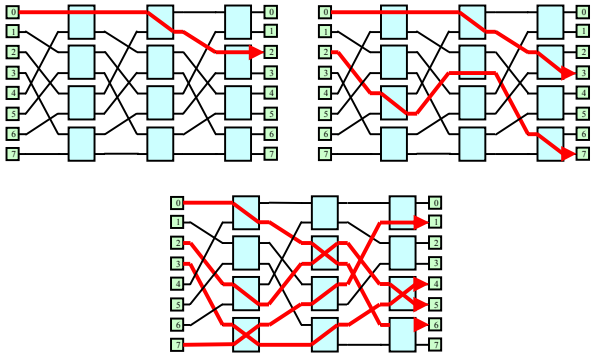


Figure 6. An OAB schedule reaching the lower bound on number of communication steps.

A direct encoding has been designed for OAB. Each chromosome consists of *N* genes, one for each destination node. Individual genes are composed of three items: a source node index, an index of the used path, and a step number. Fig. 7 shows an encoding corresponding to the optimal schedule displayed in Fig. 6. As we can see from the encoding, the node no. 1 receives the broadcasted message through the node no. 7 in the third communication step, whereas the node no. 7 receives the message from the node no. 2 during the second communication step.

The main disadvantage of this encoding is possible formation of some inadmissible solutions during the process of genetic manipulation. We say that a solution is inadmissible if it cannot lead to a correct broadcast schedule. E.g. the situation when in a certain step a node should receive a message from a node that has not received it yet (e.g. node 2 from node 1 in the first step). That is why admissibility has to be verified for each chromosome before evaluating fitness and if it is necessary the chromosome is restored, see section 4.4. The AAB chromosome is then a collection of *N* OAB chromosomes, a kind of a matrix chromosome.

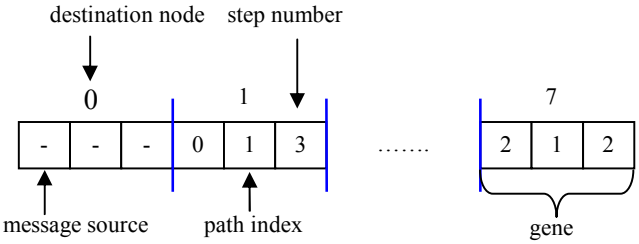


Figure 7. The structure of OAB chromosome for 8-node Omega network.

4.3 The Conflict Counting Fitness Function

The main idea of fitness function is based on testing a conflict-free (non blocking) condition. We say two communications are in conflict if and only if they share the same link in the same communication step (see Fig. 8). The fitness function is based on counting conflicts between all point-to-point communications realized in the same steps. The valid communication schedule for a given number of communication steps must be conflict-free. Valid schedules are either optimal (the number of steps equals the lower bound) or suboptimal. Evolution of a valid schedule for the given number of steps is finished up as soon as fitness (number of conflicts) drops to zero. If it does not do so in a reasonable time, the prescribed number of steps must be increased.

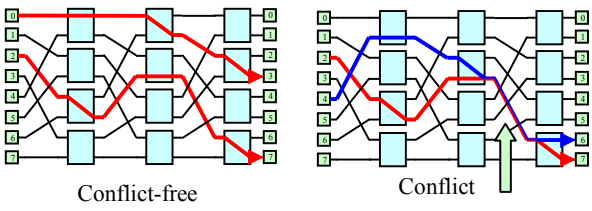


Figure 8. Two point-to-point communications.

4.4 Acceleration and Restoration Heuristics

New heuristics have been developed to improve OAS/AAS optimization speed taking into account a search space restriction due to a limited message injection capability of network nodes. Because no node can send more than one message in a communication step, an acceleration heuristic checks this condition in the whole chromosome and redesigns terminal node utilization in all communication steps before the fitness function is evaluated.

The second OAS/AAS heuristic replaces the mutation operator in an employed EA. It randomly swaps time slots of two point-to-

point communications. These simple heuristics dramatically decrease the initial conflict count and lead to the better convergence of EA.

New heuristics for OAB/AAB chromosome restoration have been also developed and employed. It proceeds in subsequent communication steps and construct a correct broadcast schedule. A check is made for every node whether the node receives the message really from the node already informed. If not so, the source node of this point-to-point communication is randomly replaced by a node that has already received the message. A change of the source node has naturally an impact on utilized links. Hence the original path is replaced by newly chosen one from a list of exploitable paths between new input-output pair.

To accelerate the convergence of the EA, an OAB/AAB specific heuristics have been developed. In the first step good building blocks are injected into the initial population. For all point-to-point communications of OAB, the time slot is set initially to the same value (step no. 0). By selecting correct time slots, the restoration heuristic produces corrected broadcast trees.

4.5 Parameters of EA

The simple UMDA evolutionary algorithm has been used for the search for near optimal communication schedules. The value of the population size was set to 60 individuals because higher values did not improve the quality of founded schedules and did not justify an increased computation time. The binary tournament selects the better half of the current population to form the parent subpopulation. The univariate marginal probabilistic model is created according to the parent subpopulation in each generation. New chromosomes are generated by the sampling of the estimated probabilistic model. Each chromosome is then mutated by a simple mutation operator with probability of 90%. This operator is responsible for testing and changing possible source-destination paths for particular point-to-point communications. The mutation rate is very high due to great number of source-destination pairs whose amount growth exponentially with the number of stages. Finally, the newly generated solutions replace the worse half of the current population.

5. RESULTS OF EVOLUTIONARY OPTIMIZATION

The evolutionary algorithm described previously has been applied to several MINs that already found the commercial application such as Omega, Butterfly and Clos networks [5]. The bidirectional MINs have been represented by binary and fat tree where terminal nodes were placed only in leaves. This study was completed by a full binary tree, where each node represents one processing node.

First, we verified the ability of EA to discover optimal communication schedules for unidirectional MINs, see Table 1. Two integers in one cell separated by a slash indicate that the lower bound (a smaller integer) has not been reached. A single integer represents both the lower and the upper identical bounds reached by EA. Obtained schedules for 8-node Omega and Butterfly met the theoretical lower bound for all classes of collective communications, and thus cannot be improved anymore. The limits of simultaneously executable transfers are reached by 12-node and 16-node topologies. For successful accomplishment of all-to-all communications, EA had to add one additional communication step to the theoretically derived value. The Clos network embodies the same problem that leads also in one step addition.

Table 1. Performance of unidirectional MINs with *N* terminal nodes (reached steps/theoretical step).

Topology	OAB	AAB	OAS	AAS
Omega 8	3	7	7	7
Omega 16	4	16/15	15	16/15
Butterfly 8	3	7	7	7
Butterfly 16	4	16/15	15	16/15
Clos 12	4	12/11	11	12/11
Clos 16	4	16/15	15	16/15

As an example of the resulting schedules, the optimal AAB schedule for 8-node Omega network is presented in Table 2. In a cell, there is shown an index of the destination node for particular communication (row) and time slot (column). The lower bound for AAB is very tight and indicates that all the links are busy in all 7 steps.

Table 2. AAB in 7 steps on 8-node Omega network.

	steps →						
src	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	4	2	1	5	3	6	7
1	5	6	0	2	4	7	3
2	0	1	3	7	6	5	4
3	6	4	5	0	7	2	1
4	2	7	6	3	5	1	0
5	1	3	7	4	2	0	6
6	7	5	4	1	0	3	2
7	3	0	2	6	1	4	5

The graphic visualization of the first of seven AAB communication steps is shown in Fig. 9. All terminal nodes send one their message to a single destination without blocking.

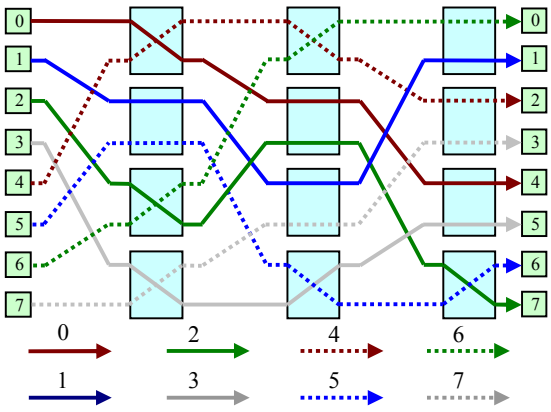


Figure 9. One of AAB steps on Omega network.

Second, we investigated the ability of proposed EA to discover optimal communication schedules for bidirectional MINs represented by binary (B-Tree), fat (F-Tree), and full binary tree (FB-Tree).

Binary trees represent suitable interconnection networks for Chip Multiprocessor because they need only very simple link arrangement on a 2D silicon chip. However, as we can see in Table 3, their performance rapidly decrease with the number of connected processing nodes. More importantly, the proposed EA is able to find optimal communication schedules for most tested binary trees and investigated communication patterns. An asterisk (*) indicates the fact that only a sub-optimal schedule has been discovered.

The fat tree topology eliminates the bottleneck of narrowing bandwidth towards the root. The height of the tree remains the same, but the number of bidirectional links proportionally increases. In this case, the EA was able to find optimal schedules for fat-trees with 4, 8, 16, and 32 leave, except AAS on 32-leave fat tree. There was achieved only a suboptimal solution with one step worse time complexity.

Finally, we completed our experimental work with full binary tree, where all switching elements integrate also a processing unit. Since full binary tree is an asymmetrical topology, several different situations depending on a level of source node (from a leave to the root), were investigated. In all cases, the theoretical lower bounds were reached.

Table 3. Performance of bidirectional MINs with *N* processing nodes.

Topology	OAB	AAB	OAS	AAS
B-Tree 4	2	3	3	4
B-Tree 8	3	8	7	16
B-Tree 16	4	20*	15	64
B-Tree 32	5	64*	31	256
F-Tree 4	2	3	3	3
F-Tree 8	3	7	7	7
F-Tree 16	4	15	15	15
F-Tree 32	5	31	31	32*
FB-Tree 7	3/2/2	7	6/4/3	12
FB-Tree 15	3/3/3/3	15	14/12/8/7	56
FB-Tree 31	4/4/4/ 4/4	31	30/28/24/ 16/15	240
FB-Tree 63	5/5/5/ 5/5/5	64	62/60/56/ 50/48/32	992

An example of designed optimal schedule for 8-leave fat-tree topology is shown in Table 4. The Fig. 10 shows the graphic visualization of the first communication step of this communication. Let us note, since interconnection links are bidirectional, the message need not to go through all intermediate stages. We can also see that not all links are utilized in the illustrated communication step. Finally, it should be mentioned, that the presented communication schedule is not unique; several optimal schedules can be found for a given CC.

Table 4. AAS in 7 steps on 8-node fat tree network.

	steps →						
Src	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	2	7	5	4	6	3	1
1	5	3	6	7	4	0	2
2	6	5	4	1	3	7	0
3	7	6	1	0	2	5	4
4	3	1	7	6	0	2	5
5	4	2	0	3	1	6	7
6	1	0	2	5	7	4	3
7	0	4	3	2	5	1	6

In the simplest linear time model of wormhole-switching communication in distributed memory systems, the real computational times of CC can be obtained as a sum of communication steps, each step composed of a start-up delay plus the serialization delay $m_i\, t_1$

$$t_{CC} = (t_0 + m_i t_1) \times \#\,\text{steps}, \qquad (1)$$

where m_i is a length (in bytes) of the longest message transported during step i, and t_1 is per byte transfer time. Start-up latency t_0 is the sw- and hw-based latency in the source and destination nodes for initializing the cache-to-cache or memory-to-memory DMA transfer and includes possible synchronization overhead. The hardware overhead in routers along the traversed path has been neglected in (1). Contention for links and associated delays are completely avoided in our schedules.

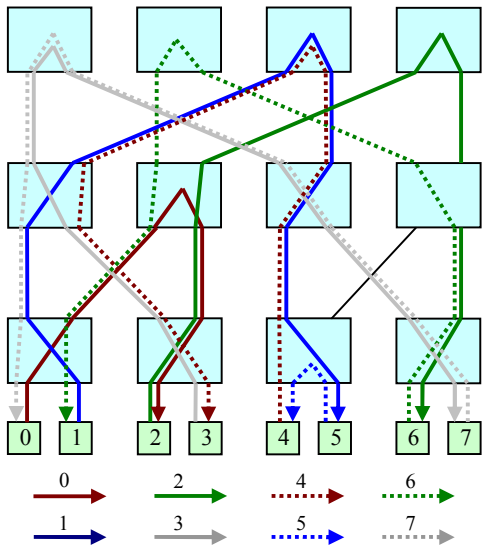


Figure 10. One of AAB steps on Omega network

For example, duration of one communication step in CC for typical cluster parameters [7] t_0=1 μs, t_1= 0.5 ns/byte and the size of the longest message of 1024 bytes has the value of 1.512 μs and the resulting CC times range from 3.02 μs (2 steps) up to 1499μs (992 steps). According to frequency of CCs and an

amount of interleaved computation in a certain application, efficiency of parallel processing can be estimated.

Table 5 shows the success rate (in percentage) of evolving an optimal schedule, if it was found. An asterisk (*) indicates the fact that only a sub-optimal schedule has been discovered. Ten experimental runs were executed for each topology and communication pattern. The success rates of OAS and OAB communications match the value of 100% in all tested topologies. The success rates of all-to-all CC embody also sufficient values. Only for the most complex benchmarks the success rate dropped bellow applicable values of 50% (trees with more then 16 communicating nodes).

Table 5. Success rate of proposed EA (calculated from 10 runs)

Topology	OAB	AAB	OAS	AAS
Omega 8	100	100	100	100
Omega 16	100	50	100	100
Butterfly 8	100	100	100	100
Butterfly 16	100	50	100	80
Clos 12	100	100	100	100
Clos 16	100	80	100	40
B-Tree 4	100	100	100	100
B-Tree 8	100	100	100	100
B-Tree 16	100	50	100	100
B-Tree 32	100	10	100	90
F-Tree 4	100	100	100	100
F-Tree 8	100	100	100	80
F-Tree 16	100	100	100	70
F-Tree 32	100	100	100	30
FB-Tree 7	100	100	100	100
FB-Tree 15	100	90	100	100
FB-Tree 31	100	80	100	100
FB-Tree 63	100	20	100	50

Table 6 shows average execution times of the EA obtained form successful runs (global optimum achieved). For OAB communication, the values are less than one second for simple network topologies. The longest execution time (FB-Tree 63) is about 62 seconds. OAS communication is relatively easy; a solution takes always less than one second. On the other hand, a suitable solution for all-to-all communication takes much longer time; especially for AAS communication. Evolution of an optimal plan for full binary tree with 63 nodes takes more than six days. An exponential increase of the execution time with network can be observed.

All experiments were realized on IBM Blade servers equipped with 2x dualcore AMD Opteron 275 processors and 4GB RAM.

6. CONCLUSIONS

The aim of the paper was granted. The UMDA evolutionary with embedded efficient heuristic was able to find out collective communication plans mostly close or equal to theoretical lower bounds. The only exception is AAS communication in larger networks, where the lower bounds are apparently too tight.

Table 6. Execution times of EA in seconds, minutes, hours and days (average values during successful runs, see table 5).

Topology	OAB	AAB	OAS	AAS
Omega 8	<1s	<1s	<1s	<1s
Omega 16	8s	23s	<1s	4m6s
Butterfly 8	<1s	<1s	<1s	8s
Butterfly 16	7s	16s	<1s	23m18s
Clos 12	3s	7s	<1s	16m53s
Clos 16	8s	2m6s	<1s	1h27s
B-Tree 4	<1s	<1s	<1s	<1s
B-Tree 8	<1s	32s	<1s	6s
B-Tree 16	4s	6m13s	<1s	26s
B-Tree 32	25s	35m14s	<1s	9m32s
F-Tree 4	<1s	<1s	<1s	<1s
F-Tree 8	<1s	8m41s	<1s	12s
F-Tree 16	5s	11m1s	<1s	6m16s
F-Tree 32	18s	52h4m	<1s	31m21s
FB-Tree 7	<1s	<1s	<1s	1s
FB-Tree 15	7s	13m4s	<1s	1h24m
FB-Tree 31	38s	2h11m	<1s	23h43s
FB-Tree 63	1m2s	13h42s	<1s	6d8h

From all MINs, Fat tree networks promise the best scalable performance, even though the node count can attain only a few values. However, the performance can be fine-tuned by the number of processors per node. Inter-node CC is then implemented by message passing, whereas intra-node CC can utilize either a synchronized access to the shared L2 cache by threads or again passing messages among processes [2]. CC schedules designed by the presented evolutionary technique are targeted for micro-programmed DMA engines residing in nodes of the network.

Some of the found CC schedules attain the theoretical lower bound on the number of communication steps and thus there is no way to improve them further. Future research may reveal limits on the size of networks that can be handled by parallel implementation of evolutionary techniques. Another direction for future research could explore a combining model for CC on MINs or their fault tolerance.

7. ACKNOWLEDGMENTS

This research has been carried out under the financial support of the research grants “Design and hardware implementation of a patent-invention machine”, GA102/07/0850 (2007-9), “Safety and security of networked embedded system applications”, GA102/08/1429 (2008-10), both care of Grant Agency of Czech Republic, and “Security-Oriented Research in Information Technology”, MSM 0021630528 (2007-13).

8. REFERENCES

[1] van der Steen, A. J., Dongarra, J. J. Overview of Recent Supercomputers. *TOP 500® Supercomputer Sites*, Nov. 2007 Edition, http://www.arcade-eu.org/overview/.

[2] Stewart, L. C., Gingold, D. *A New Generation of Cluster Interconnect*. White Paper, SiCortex Inc., Dec. 2006.

[3] Jaroš J., Ohlídal M., Dvořák V. An Evolutionary Approach to Collective Communication Scheduling, In: *2007 Genetic and Evolutionary Computational Conference*, New York, US, ACM, 2007,pp. 2037-2044.

[4] Duato, J., Yalamanchili, S. *Interconnection Networks – An Engineering Approach, Morgan Kaufman Publishers*, Elsevier Science, 2003.

[5] Hennessy, J. L., Patterson, D.A. *Computer Architecture - A Quantitative Approach*. 4th Edition, Morgan Kaufman Publishers, Inc., 2006.

[6] Karim, F., Nguyen, A. An Interconnect Architecture for Networking Systems on Chips. *IEEE Micro*, Sept. – Oct. 2002, pp.36-45.

[7] Mühlenbein, H., Paaß, G. From recombination of genes to the estimation of distributions I. Binary parameters. In *Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature – PPSN IV*, pp. 178-187, 1996.

[8] Jaroš, J., Dvořák, V. Speeding-up OAS and AAS Communication in Networking System on Chips, In: *Proc. of 8th IEEE Workshop on Design and Diagnostic of Electronic Circuits and Systems*, Sopron, HU, UWH, 2005, pp. 4, ISBN 9639364487.

[9] Ohlídal, M., Jaroš, J., Dvořák, V., Schwarz, J. Evolutionary Design of OAB and AAB Communication Schedules for Interconnection Networks, In: *Lecture Notes in Computer Science*, 2006, no. 3907, DE, pp. 267-278, ISSN 0302-9743.

[10] Larrañaga, P., Lozano, J. A. *Estimation of Distribution Algorithms*. Kluwer Academic Publishers, London 2002, ISBN 0-7923-7466-5.

[11] Goldberg D. G*enetics Algorithms in Search, Optimization, and Machine Learning*, Addision-Wesley Publishing Company, 1989.

[12] Dally, W., Towles, B. *Principles and Practices of Interconnection Networks*. The Morgan Kaufmann Series in Computer Architecture and Design, Morgan Kaufman Publishers, 2004.

[13] Szymanski T., Hamacher V. On the permutation capability of multistage interconnection networks. *IEEE Trans. Comp.*, C-36(7):810–822, Jul. 1987.

[14] Kruskal, C. P., Snir, M. The performance of multistage interconnection networks for multiprocessors. *IEEE Trans. Comput.*, C-32(12):1091–1098, Dec. 1983.

[15] Alleyne B. D.: *Methodologies for Analysis and Design of Data Routers in Large SIMD Computers*. PhD thesis, Princton Univ., June 1994.

[16] Lawrie, D. A.: Access and alignment of data in an array processor. *IEEE Trans. Comput.*, C-24(12):1145–1155, Dec. 1975.

[17] Tusch, D., Hommel, G. Multicast routing in Clos networks. In: *Proceedings of 2004 Design, Analysis, and Simulation of Distributed Systems*, Arlington, pp. 21-27, 2004

[18] Leiserson, C. E. Fat-trees: Universal networks for hardware-efficient supercomputing, *IEEE Transaction on Computers*, 34(10):892-901, October 1985.

[19] Yuanyuan, Y., Pan, Y., Wang, J. Abstract Permutation Capability of Optical Multistage Interconnection Networks, In: *Journal of Parallel and Distributed Computing*, pp. 72-91, 2000.

[20] Dijkstra, E. W.: A note on two problems in connection with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[21] Ni, L. M., Mckinley, P. K. A survey of wormhole routing techniques in direct networks. *IEEE Computer*}, vol. 26, pp. 62-76, 1993.