An Evolutionary Approach to Collective Communication Scheduling

Jiří Jaroš
Brno University of Technology
Božetěchova 2
612 66, Brno, CZ
+420 54114-1207
jarosjir@fit.vutbr.cz

Miloš Ohlídal
Brno University of Technology
Božetěchova 2
612 66, Brno, CZ
+420 54114-1264
ohlidal@fit.vutbr.cz

Václav Dvořák
Brno University of Technology
Božetěchova 2
612 66, Brno, CZ
+420 54114-1149
dvorak@fit.vutbr.cz

ABSTRACT

In this paper, we describe two evolutionary algorithms aimed at scheduling collective communications on interconnection networks of parallel computers. To avoid contention for links and associated delays, collective communications proceed in synchronized steps. Minimum number of steps is sought for the given network topology, wormhole (pipelined) switching, minimum routing and given sets of sender and/or receiver nodes. Used algorithms are able not only re-invent optimum schedules for known symmetric topologies like hyper-cubes, but they can find schedules even for any asymmetric or irregular topologies in case of general many-to-many collective communications. In most cases does the number of steps reach the theoretical lower bound for the given type of collective communication; if it does not, non-minimum routing can provide further improvement. Optimum schedules may serve for writing high-performance communication routines for application-specific networks on chip or for development of communication libraries in case of general-purpose interconnection networks.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Design.

Keywords

Collective communications, communication scheduling, evolutionary optimization.

1. INTRODUCTION

The importance of communication among CPU cores, processors and computers and of related interconnection networks is recently steadily growing. More often than not, processing nodes access the network according to a global, structured communication pattern. The performance of these collective communications (CC for short) has a dramatic impact on the overall processing efficiency, because communication times, software- or hardware-related alike, add up to an overhead of parallel processing.

An interconnection network connects a group of nodes, with each node containing its own processor, memory, and a router. The router specifically handles the communication functions to free up the application processor for time critical message processing. Typically, only one pair of internal channels (1-port model) is used for the router to pass messages to the processor. To reduce the communication latency, some systems are implemented by using multiple pairs of internal channels (*k*-port model). These internal channels are to be differentiated from the external channels, which are responsible for the passing of messages between nodes. In the all-port model, the number of ports equals a number of external channels. The interconnection network can be direct (as 2D-torus or a hyper-cube), or indirect, with some routers (switches) without processors (as in Fig. 6a). A class of interconnection networks of interest in this paper aims at high performance; it makes use of nearly distance insensitive pipelined message transmission (wormhole switching WH or incidentally circuit switching CS) over full duplex links (two channels in opposite directions) and deterministic source-based routing algorithms. We therefore exclude adaptive algorithms with HW support for broadcast or multicast message passing techniques.

Collective communication [1] involves communication among subsets or among all processors; provided that there is 1:1 mapping between processors and processes, we can equivalently talk about communicating process groups. Generally we have two process groups: T – the subset of senders (transmitters) and R – the subset of receivers. The subsets T and R can be overlapping and can be as large as the full set of P processes. Collective communication may be categorized as one to one, one to many, many to one or many to many, with many being also all:

1. $T \cap R = \emptyset$, non-overlapping sets of processes.

   A. One-to-all, $|T| = 1$, $|R| = P - 1$. (One-to-all broadcast OAB, one-to-all scatter OAS).

   B. One-to-many, $|T| = 1$, $|R| < P - 1$. (Multicast)

   C. All-to-one, $|T| = P - 1$, $|R| = 1$, e.g. gather (AOG) or reduce (AOR).

   D. Many-to-many, $|T| = M$, $|\mathrm{R}| = N$, $M, N < P$. Non-overlapping sets of processes (Many-to-many broadcast MNB, many-to-many scatter MNS).

Conference Gecco'07, July 7–11, 2007, London, England.

2. $|T \cap R| \geq 1$. Many-to-many communication with overlapping sets of processes.

3. $|T \cap R| = P$. All-to-all communications such as permutation, all-to-all scatter (AAS), broadcast (AAB), reduce (AAR), and others.

In one to all, one process in group *T* is the sender (transmitter) and all the other processes in group *R* are receivers. This category of group communication has two distinct services, broadcast (multicast) and scatter. Broadcast (multicast) is when the same message is delivered to all the other processes in the group and scatter delivers a different message to each receiver in the group. For all to one group communication, all the processes in group *T* are identified as senders and the only one process in group *R* is identified as the receiver. A service offered in this category is gather, which occurs when different messages from the senders are received by the receiving process. This is the opposing process of scatter. And finally, all to all communication is when each process in group *T* performs its own one to all message passing. A service provided in this category is all broadcast, which allows for each sender to deliver the same message to all the receivers in the group. This is also known as gossiping. The other service of this group is all scatter, which means a different message is sent to each receiving process from the sending process. All scatter is also referred to as complete exchange.

Some researchers have taken a topology independent approach to the design of deadlock free wormhole routed CC algorithms. E.g. the postal model (similar to sending a batch of letters through the postal service at one time) demonstrates the ability for a sending node to transmit multiple messages before the receiving node receives the first message. In this paper, we will rather take a topology-aware approach and will assume that CC in WH networks proceeds in synchronized steps. In one step of CC, a set of simultaneous message transfers takes place along complete disjoint paths between source-destination node pairs. If the source and destination nodes are not adjacent, the messages go via some intermediate nodes, but processors in these nodes are not aware of it; the messages are routed automatically via routers attached to processors. We will assume that all messages in CC have identical size and are not combined or partitioned by network nodes (the non-combining model).

The paper is structured as follows. In Section 2 we present formally the scheduling problem and recent solutions of its certain instances. Our novel approach based on evolutionary algorithms is explained in Section 3. Solutions to sample communication problems on selected networks are given in Section 4, where the quality of resulting schedules is also discussed. In Conclusion we give the range and scope of scheduling problems solvable by the presented approach and possible future extensions.

2. MATHEMATICAL FORMULATION OF THE SCHEDULING PROBLEM AND ITS PARTIAL SOLUTIONS

Any collective communication is composed of set *Com* of pair-wise communications (transfers, messages, paths)

$$x_i = \{c_1, c_2, c_3, \ldots, c_L\}, \qquad (1)$$

where c_i are unidirectional channels along the minimum path from the source to destination node. (We will restrict ourselves to minimum routing for practical reasons given later). Cardinality of set *Com* may be quite high, e.g. all-to-all communication among *P* processors gives $|\,Com\,| = P\,(P - 1)$ messages; for $P \in <8, 128>$ we have $P\,(P - 1) \in <56, 16256>$.

The problem with broadcast (OAB, AAB) is that set *Com* is not known in advance, since the informed nodes become sources for further transfers. Broadcast in WH meshes and torus networks were scheduled by extending the use of dominating sets from graph theory [2], [3] to WH routing. In this approach, a subset of nodes (dominating set) may pass messages to the remaining nodes on the network in one step. Generally in case of OAB on *k*-port network *G*, having *m* nodes informed, we are trying to inform recursively $k \times m$ nodes in the next step, multiplying the number of informed nodes. Optimal algorithms reaching the theoretical lower bound of steps are not known even for familiar hyper-cubes of higher dimensions ($d > 7$).

For many-to-many broadcast (MNB) type of communication, regardless whether we use WH or SF routing, the number of steps is limited by *k* messages that can be absorbed by any node in one step. Therefore we can inform only adjacent nodes in one step (as in SF routing) and still develop optimum scheduling. The task is easier in symmetric networks; it is sufficient to find the so called time-arc disjoint broadcast tree (TADT), which translated to all source nodes, creates no conflicts in any step of AAB communication. However, for asymmetric or irregular (non-constant *k*) networks, no similar systematic approaches exist.

When the set *Com* is known in advance, the goal of scheduling is to pack messages in *Com* into the minimum number of groups such, that there is no conflict within a group. In wormhole routing a conflict means that two messages scheduled in the same step share one or more channels. If they don't, they are compatible. Compatibility relation γ on set *Com* can thus be defined:

$$x_i \,\gamma\, x_k \equiv \exists! c_e \,\{c_e \in x_i \text{ and } c_e \in x_k\} \qquad (2)$$

This relation defines a cover of *Com* by maximum-size compatibility classes. A group of messages in one compatibility class can start transmission simultaneously and we therefore schedule each such group in one communication step. Obviously we want to find a minimum number of compatibility classes still covering set *Com*. The final step is to transform this minimum cover of *Com* to a partition, compatibility classes to blocks, by eliminating messages in more than one class and possibly simultaneously balancing the size of classes.

Exact solution of the above problem can be obtained by MILP method (Mixed Integer Linear Programming), but very long solutions are required for network size of practical interest. The communication scheduling can also be formulated as a graph coloring problem [4]. Elements of *Com* can be represented by graph nodes and incompatibility relation (two nodes sharing a channel) by graph edges. Minimum number of colors needed to color the graph gives the optimum number of compatibility classes (communication steps); nodes with the same color belong to one compatibility class. MILP as well as exact or heuristic graph coloring yield only a suboptimal solution. The reason is the existence of multiple minimum paths for some source-destination pairs; it is not clear which minimum path should be selected for *Com*. On the other hand, inclusion all of them may produce more compatibility classes than necessary, aside from complex removal of redundant elements. Another approach, recursive division of set *Com* described in [4], is supposed to be exact, but suffers the following restrictions:

- only non-overlapping sets of processes $T \cap R = \varnothing$ are assumed,

- routing from src to dst is unique and prescribed,

- one-port model is assumed.

In our approach we will be able to relax all above restrictions.

3. EVOLUTIONARY APPROACH

The design of conflict-free schedules using evolutionary optimization has been carried out in two directions.

1. Store and Forward (SF) routing strategy, only moving messages to the adjacent nodes in one step, proved to be best also for MNB on WH networks. As the MNB lower bounds are equal for WH as well as SF routing, optimal MNB schedules on SF and WH networks coincide. MNB schedules were found by Hybrid parallel Genetic Simulated Annealing (HGSA) [6].

2. However, the situation is different with regard to OAB and MNS communications, as the SF and WH lower bounds differ. In these communications and in OAS, nearly distance insensitive WH routing was applied, moving messages several hops in one step. The schedules were obtained with the aid of MBOA [5] algorithm, because traditional genetic algorithms failed.

Based on recently published results [17], [18], where we tested also classical genetic algorithm, we chose only these two types of evolution algorithms HGSA and MBOA because they achieved the best solutions with the best success rate.

3.1 Many-to-many broadcast by means of HGSA

Scheduling MNB on SF (WH) networks was dividend into two separate tasks:

- creation of set *Com* with *all* minimum paths between any pair sender – receiver including paths that are a part of longer paths with the same sender. This is followed by detection and counting initial conflicts on all the channels.

- evolving a population of complete MNB schedules (chromosomes) burdened with conflicts towards a conflict-free schedule, taking into account the given target number of steps *S*. Time slots (steps) 1, 2, …, *S* assigned to channels on each selected path are re-arranged and the fitness value is computed from the number of conflicts in the whole schedule.

A conflict at SF routing arises when two messages want to use the same channel in the same step. Necessary (but not sufficient) conditions for a schedule to be conflict-free are:

1. Two paths can use the same channel in the same step only if they have a same source (sender). This is taken as a single use.

2. A channel can be used in CC *S*-times or less.

If *L* is a length of a path and *Z* is the position of the analyzed channel on the path, we can use this channel

in step 1 if $Z = 1$,
in step $\in \langle Z, S-(L-Z) \rangle$, if $Z \neq 1$.

At the beginning, the time step is assigned to a channel randomly from the above interval.

HGSA [12] is a hybrid method that uses parallel Simulated Annealing (SA) [14] with the operations used in standard genetic algorithms [16], see Fig. 1. In the proposed algorithm, several SA processes run in parallel. During communication, activated each 100's iteration of Metropolis algorithm (see paragraph bellow), each process sends their solution to a master. The master keeps one solution for himself and sends one randomly chosen solution to each slave. The selection is based on the roulette wheel, where the individual with the best value of the fitness function has the highest probability of selection.

After communication phase, each process has two individuals. Now starts the phase of a genetic crossover. Two additional children solutions are generated from two parent solutions using double-point crossover. The best solution from two parents and two children is selected and mutation is performed always (in the parent solution) or with a predefined probability (in the children solution). Mutation is performed by randomly selecting genes and by randomly changing their values. A new solution for each process is selected from the actual solution provided by SA and from the solution obtained by genetic manipulation. The selection is controlled by well-known Metropolis criterion.

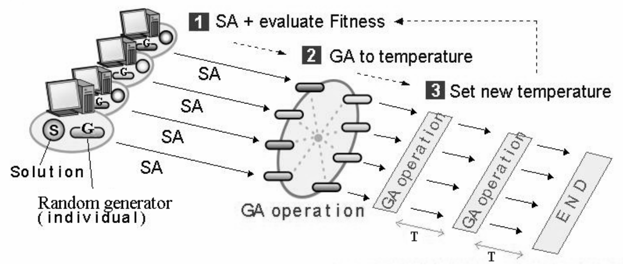


Figure 1. Hybrid parallel Genetic Simulated Annealing

The well-known Metropolis algorithm is a method of sampling a Boltzmann distribution [13]. A system with energy E_{old} is provisionally perturbed into a new state with energy E_{new}. Such a perturbation is called a "move". If $E_{\text{new}} < E_{\text{old}}$ the new state is accepted. If $E_{\text{new}} > E_{\text{old}}$ the new state is accepted with probability *exp* $-((E_{\text{new}}-E_{\text{old}})/T)$, where *T* is "temperature". If the new state is not accepted, the system remains in the old state. This algorithm tends to transform any distribution into a Boltzmann distribution and maintains a preexisting Boltzmann distribution [13].

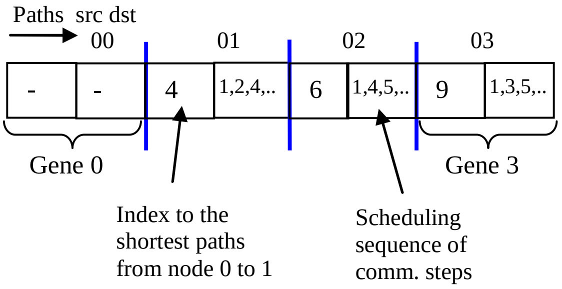

Paths src dst
00
01
02
03
-
-
4
1,2,4,..
6
1,4,5,..
9
1,3,5,..
Gene 0
Gene 3
Index to the shortest paths from node 0 to 1
Scheduling sequence of comm. steps

Figure 2. The structure of chromosome

Very simple encoding has been chosen for HGSA. Every chromosome consists of P^2 genes, where P is a number of nodes in a given topology. The gene's index represents both the sender and the receiver node index (sender node = gene's_index div P, receiver node = gene's_index mod P.

Each gene consists of two components. The first component is an index of the shortest path from the sender to the receiver. The second component is a sequence of communication steps (time slots) assigned to channels on the path, because the message can stop for one or more steps at some nodes along the path. Fig. 2 illustrates an example of this encoding.

3.2 Scheduling broadcast and scatter communication by means of MBOA

The general procedure of MBOA [5] belonging to the family of Estimation of Distribution Algorithm (EDA) [15] is similar to that of GA, but the classical recombination operators (crossover and mutation) are replaced by probability estimation followed by probability sampling. These algorithms use to advantage the statistical information contained in the set of promising solutions to discover the linkage between genes. New solutions are generated by sampling the constructed probabilistic model. A new feature of these algorithms is a global usage of the whole population in the process of model construction. One of the basic advantages is the capability to discover nonlinear interaction between genes, which allows solving complex nonlinear problems. The basic pseudo-code of EDA is shown in Fig. 3.

```
Generate initial population of individuals
D(0) of size N (randomly);
While termination criterion is false do
begin
   Select the parent population $D^s(t)$ of M
     individuals according to a selection
     method;
   Estimate the probability distribution of
     the selected parents $D^s(t)$;
   Generate new offspring O(t) according to
     the estimated probabilistic model;
   Replace part of D(t) by generated
     offspring O(t), yielding D(t+1);
end
```

Figure 3 Pseudo-code of EDA algorithm

We have used Mixed Bayesian Optimization Algorithm (MBOA) [5] for our task. MBOA is based on Bayesian Optimization Algorithm (BOA), whose probabilistic model is the Bayesian net. MBOA replaces this net by a set of binary decision trees/graphs. The MBOA differs from BOA also in the heterogeneous model parameters. The decision trees can be used also for continuous or mixed domains. MBOA uses variance adaptation for scaling variance in continuous domains. The integer bound mutation was newly added to this algorithm.

The OAS chromosome, shown in Fig. 4, uses P genes; each gene consists of two items: an index of one of the shortest source-destination path and a communication step number. (Now the whole path is traversed in a single time step). A chosen AAS chromosome encoding has a form of a matrix with P OAS chromosomes (vectors).

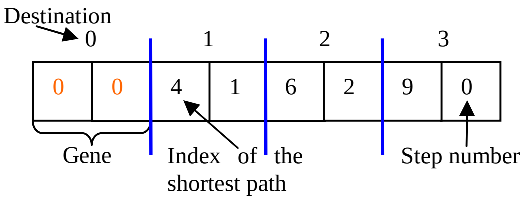


Figure 4. The structure of OAS WH chromosome

In the case of OAB with wormhole switching, we have used an indirect encoding; a chromosome does not include a broadcast tree, but only instructions how to create it. Each chromosome consists of P genes, one for each destination node, Fig.5. Individual genes are composed of three items: a source node index, the shortest path index, and a step number.

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 a correct broadcast tree cannot be obtained from it. E.g. the situation when in a certain step a node should receive a message from a node that did not get it yet. That is why admissibility has to be verified for each chromosome before evaluating fitness and if the need be, the chromosome would be restored. In Fig. 5, a chromosome for OAB on the 8-node WH ring topology is presented. The AAB chromosome is then a collection of P OAB chromosomes, a kind of a matrix chromosome.

A new heuristics for OAB chromosome restoration has been used. The restoration (a repair of the broadcast tree) proceeds in subsequent communication steps. A check is made for every node whether the node receives the message really from an informed node. If not so, the source node of this communication is randomly replaced by a node that already has the message. Also, it is necessary to check shortest paths already used. There is a finite number of shortest paths from each source to each destination node. If the second gene component (the path index) exceeds this value, the modulo operation will be applied to it.

The fitness function is again based on counting conflicts in a schedule when two paths share the same channel in the same step. The optimal schedule does not contain any conflict and the MBOA (with the given number of communication steps as an input parameter) was able to find it for common networks with up to 64 nodes [6].

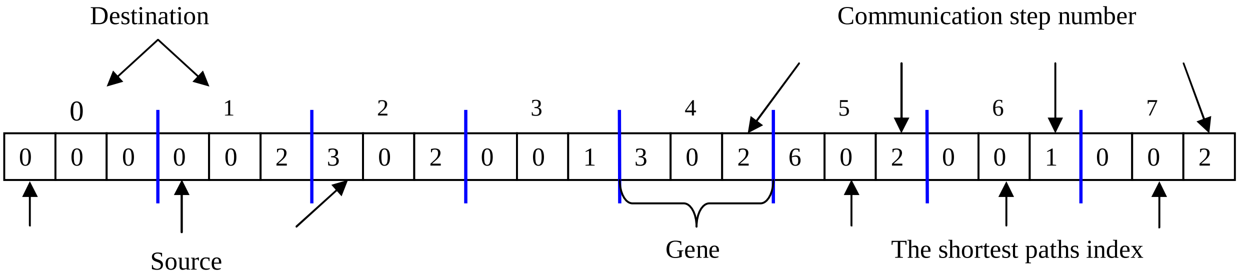


Figure 5. The structure of OAB WH chromosome

The advanced features of our algorithms follow:

- multiple minimum paths from source to destination nodes were accounted for in an easy way through mutation. As soon as the fitness was not improving in a certain period of optimization, replacement of one minimum path by another proved to be a good remedy.

- overlapping sets of senders and receivers are permissible without program modification

- networks with fat nodes (several processors connected to one router) are dealt with as simply as slim node networks.

- CC can be scheduled on direct as well as indirect interconnection networks.

4. RESULTS AND THEIR QUALITY

Time complexity of CCs will be determined in terms of the number of communication steps $\tau_{CC}(G)$ for the lower bound and $\tau^{CC}(G)$ for the upper bound. This figure of merit is in fact the number of start-ups (overhead t_s for each one) and does not take into account the message length. Since a nearly distance-insensitive wormhole switching has been assumed, the real communication times can be obtained approximately from the number of start-ups $\tau^{CC}(G)$ plus the serialization delay $m\ t_1$,

$$t_{CC} = \tau^{CC}(G)\ t_s + m\ t_1 \qquad (3)$$

neglecting the hardware overhead in routers along the traversed path. Possible synchronization overhead between communication steps, be it hardware or software-based, should be included in the start-up time t_s. According to frequency of CCs and an amount of computation in a certain application, efficiency of parallel processing can thus be estimated with a good degree of accuracy.

The lower bounds $\tau_{CC}(G)$ on number of CC steps can be found easily [1]. As far as the broadcast communication (OAB) on k ports is concerned, the lower bound on the number of steps

$$\tau_{OAB}(G) = s = \lceil \log_{k+1} P \rceil \qquad (4)$$

is given by the number of nodes informed in each step, that is initially 1, $1 + 1 \times k$ after the first step, $(k + 1) + (k + 1) \times k = (k + 1)^2$ after the second step, etc.,…, and $(k + 1)^s \geq P$ nodes after step s.

In case of AAB communication, since each node has to accept $P - 1$ distinct messages, the lower bound is $\lceil (P - 1) / k \rceil$ steps or (4), whichever is greater. A similar bound applies to OAS communication, because each node can inject into the network not more than k messages at a time. The lower bound for AAS can be obtained considering that one half of messages from each processor cross the bisection, whereas the other half do not. There will be altogether $2\ (P / 2)\ (P / 2)$ of such messages in both ways. If B_C is the network bisection width [1], not more than B_C messages can flow in one direction through the cut at a time. This gives $\lceil P^2 / (2\ B_C) \rceil$ or $\lceil (P - 1)/k \rceil$ communication steps, whichever is greater. Thus the lower bounds $\tau_{CC}(G)$ for the network graph G depends on three parameters: port number k, number of nodes P, and channel bisection width B_C, Table 1.

Table 1. Lower complexity bounds of selected CCs (any topology)

CC	WH, k-port, full duplex
OAB	$\lceil \log_{k+1} P \rceil = \lceil (\log P) / \log(k+1) \rceil$
AAB	$\mathrm{Max}(\lceil \log_{k+1} P \rceil, \lceil (P-1)/k \rceil)$
OAS	$\lceil (P-1)/k \rceil$
AAS	$\mathrm{Max}(\lceil P^2/(2\ B_C) \rceil, \lceil (P-1)/k \rceil)$

In order to see how powerful evolutionary algorithms are, we have started with scheduling CC on the well-known direct topology - an all-port WH hyper-cube interconnection network. A hyper-cube has been chosen because of its regular topology with known optimal scheduling so that it can serve as a convenient benchmark. Lower bounds for all all/one-to-all/one CC schedules shown in Table 1 are, except OAB, reachable by known optimal algorithms for any hyper-cube size. The double-tree algorithm for OAB [1] is optimal only for $d \leq 6$. Other known algorithms are nearly optimal (e.g. the algorithm by Ho-Kao is optimal up to $d \leq 7$ [1]). Ten optimization runs have been run for each configuration and the success rate (%) in reaching the lower bound (known to be the global optimum) is shown in Table 2.

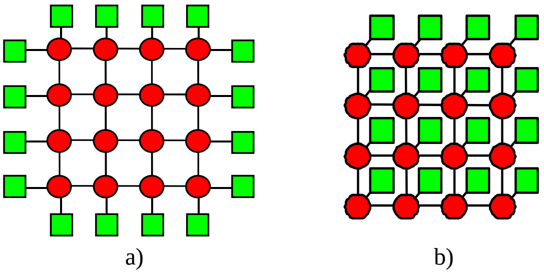
OAB schedules for WH hyper-cube have been obtained with the aid of WH-oriented MBOA only, because WH routing has different (better) lower OAB bounds than SF routing. On the other hand, WH AAB can be served best by SF-oriented HGSA; lower AAB bounds are identical and WH-oriented MBOA is too complicated in this case. Remaining OAS and AAS schedules were again obtained by WH-oriented MBOA. Results summarized in Table 2 were satisfactory and led us to application of MBOA and HGSA for scheduling CC on other network topologies, where optimal algorithms are not known. Among them AMP [7], twisted ladder, Moore graph [7], K-ring, also known as a folded hypercube [9], and other topologies were investigated.

Table 2. Lower bounds τ_{CC}(G) and success rate of reaching them for all-port WH hyper-cubes

P	OAB MBOA	OAS MBOA	AAB HGSA	AAS MBOA
8	2 / 100%	3 / 100%	3 / 100%	4 / 60%
16	2 / 100%	4 / 100%	4 / 100%	8 / 30%
32	2 / 100%	7 / 90%	7 / 100%	16 / 10%
64	3 / 70% *)	11 / 90%	11 / 80%	32 / 0%
128	3 / 50% **)	19 / 90%	19 / 0%	64 / 0%

*) 100% with HGSA **) 100% with HGSA

The illustrative examples of one indirect and one direct network are in Fig. 6 - coated Mesh (CM) [10] and 2D-Mesh (M). Only 4 x 4 meshes are presented for simplicity.



a) b)

Figure 6. a) 4 x 4 CM, b) 4 x 4 2D-M

The results of scheduling all-to-all communications are shown in Table 3. Optimum algorithms (lower bounds) have been obtained for OAB. AAS schedules require one step over the lower bounds $\tau_{AAS}(M) = \tau_{AAS}(CM) = 16$. With no way of decreasing the number of steps any further, the schedules may be optimal or only suboptimal.

Table 3. The number of communication steps τ^{CC}(G)

Network graph G	AAB HGSA	AAS MBOA
M 4x4, 1-port	15 (100%)	17 (60%)
M 4x4, all-port	8 (100%)	17 (25%)
CM 4x4, 1-port	15 (100%)	17 (20%)

Let us note that during the search for the optimum schedule, it may be necessary to include not only multiple minimum paths, but sometimes even non-minimum ones! Fig. 7 shows one example – one-to-all scatter communication in the mesh topology. To reach the minimum number of communication steps (the lower bound is 5 steps), 3 messages must be injected to a network in every step by the source node. The last step requires non-minimum routing.

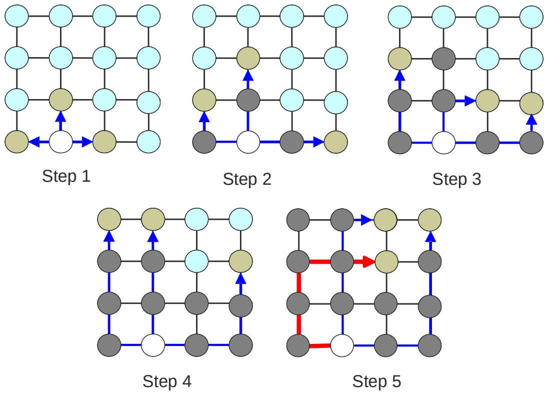
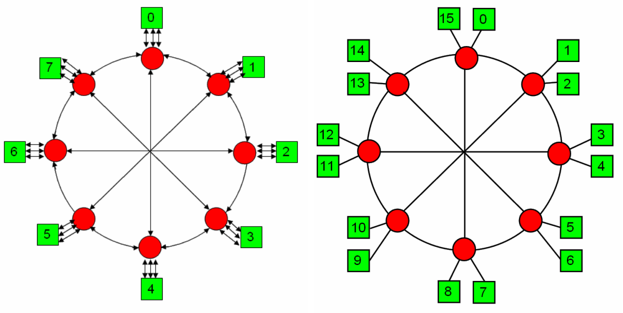


Figure 7. One-to-all scatter in 5 steps

Another interesting network topology is Octagon [8]. It is the novel on-chip communication network architecture suitable for the aggressive on-chip communication demands of System on Chips (SoCs), see Fig. 8. As a ring, it is not free from deadlock and virtual channels have to be used.



a) b)

Figure 8. a) Slim Octagon b) Fat Octagon topology

Collective communications on the generic 8-processor, symmetric Octagon network are easy. One-to-all communications are done the same way for every source node. OAB clearly can be done in 2 steps and OAS needs $\lceil 7/3 \rceil = 3$ steps. To implement AAB, we have to use such a broadcasting tree that is time-arc disjoint (TADT) and can be used by all nodes simultaneously without creating conflict.

To design the most complex AAS schedule (yet unknown), the evolutionary approach has been used. Four steps were needed for AAS on Octagon with all-port (3-port) nodes, one step worse than the lower bound in Table 1. The optimum AAS schedule is given in Table 4. The sequences of digits denote the path of length one (src, dst) or two (src, via, dst). It can be seen that AAS communication is not performed the same way by all nodes - there is no analogy to the TADT.

Table 4. AAS communication schedule on the Octagon8 topology

step	AAS on Octagon
0	073, 104, 156, 21, 23, 267, 340, 432, 45, 512, 654, 701, 762
1	012, 07, 10, 265, 321, 34, 451, 437, 54, 567, 623, 73, 704, 76
2	01, 12, 15, 107, 234, 26, 32, 40, 456, 543, 670, 621, 765
3	04, 015, 076, 123, 210, 345, 326, 37, 43, 540, 51, 56, 62, 65, 67, 70

The suggested scaling strategy [8] based on bridge nodes connecting adjacent Octagons has a drawback of a very low bisection width B_C and therefore a poor performance in all-to-all and many-to many (MNB/MNS) traffic. Another scaling strategy extends the Octagon to the multidimensional space by linking corresponding nodes of several Octagons. This, however, increases the node degree, and is not always acceptable. Octagon can also be extended to a larger ring with P = 8, 12, 16,…, 4n nodes retaining the original topology [8], but congestion of wires in the middle may cause difficulties at manufacturing (in 2 dimensions, e.g. in Network on Chip). We have therefore used a fat Octagon with two CPU cores per node, Fig. 8b, not described in literature as yet. The results (upper bounds) of selected M-to-N broadcast and scatter schedules are given in Table 5.

Table 5. M-to-N communication, lower τ_{CC} and upper τ^{CC} bounds. Fat Octagon topology (P = 16)

FD, 1-port, Fat Octagon	MNB HGSA		MNS MBOA	
	τ_{MNB}	τ^{MNB}	τ_{MNS}	τ^{MNS}
8 to the same 8	7	7	7	10 /7*)
8 to other 8	8	8	10	10
8 to all 16	8	8	11	15
all 16 to all 16	15	15	15	17 +)

*) with non-minimum routing, +) non-minimum routing not found

5. DETAILS OF IMPLEMENTATION

The computational platform used was IBM BladeCenter® [11] with 12 HS20 blades, each fitted with 2 CPU Xeon 2,8GHz/533MHz, 1GB RAM, 40GB HD, interconnected by gigabit router-switch. Algorithms were coded in C/C++ and MPI and ran under Linux OS.

Parameters of HGSA (AAB problems) were set to the same values for all runs, i.e. 10 blades in the master slave architecture; the length of a communication interval between the master and each slave was each 100's iteration of Metropolis algorithm; population size was the same as the number of blades, starting temperature 100, number of iterations at one temperature value was 200, gradient of cooling 0.99. 20 runs of HGSA were performed for each topology.

OAB, OAS and AAS communication schedules have been sought using MBOA. Here the population size was determined such that the global optimum was reached in at least 50% of all runs (if possible). Tournament selection and replacement operator was used. Mutation rate was 100% (one random gene was mutated in each chromosome). Mutation rate is so high because the gene's item representing used shortest paths between source and destination, can take a huge number of different values.

The average time complexity of reaching global optimum (in terms of number of fitness function evaluations) is shown In Fig. 9 and 10 for several instances of hyper-cubes. The real execution time was from few seconds to several hours for the most complex problems.

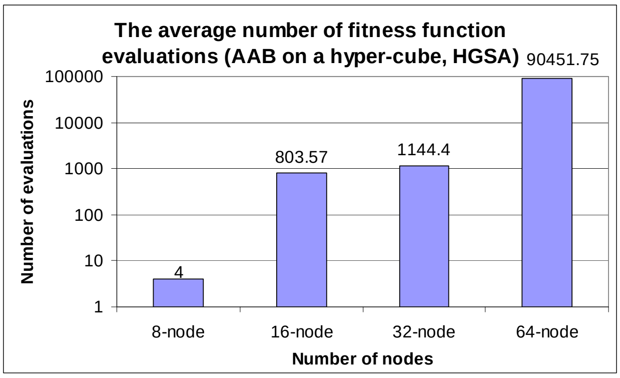


Figure 9. Time complexity of AAB HGSA

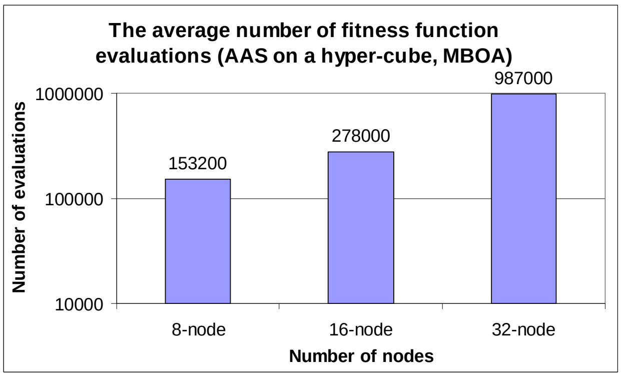


Figure 10. Time complexity of AAS MBOA

6. CONCLUSION

The evolutionary algorithms such as MBOA and HGSA have been applied successfully to several network topologies with slim or fat nodes and quite general collective communications with overlapping sets of senders and receivers. Scheduling CC in the minimum number of steps without creating a conflict (a common channel in two transfers in the same step) led to optimal solutions ($\tau^{CC}(G) = \tau_{CC}(G)$) or nearly optimal solutions. Of course, the fact that the lower bound may not be always reached by presented algorithms is to be expected because it may not be attainable in principle by any algorithm. Sometimes lower bounds can be obtained in schedules with non-minimum routing. However, only minimum routing has been considered in this paper because inclusion of the non-minimum routing would lead to an enormous increase of possible paths from sources to destinations and to the prohibitive computer memory and time requirements.

The results were derived for general case of M-to-N collective communications on WH interconnection networks. The application-oriented CCs of this kind are increasingly important in multiprocessor SoCs (System on Chips). One example is when one group of processors finishes a task and a group of different size continues and needs the intermediate results from the first group. The really obtained upper bounds $\tau^{CC}(G)$ were presented for the 2D-mesh, coated mesh and (fat) Octagon topology of small size for illustration only. The presented algorithms are at current form applicable to networks with up to around 64 nodes; however, if we put up with suboptimal schedules (giving the number of steps reasonably close to the lower bound), the network size can be substantially larger. Scalability limits of the both presented algorithms and their possible improvements could be a subject of future research.

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”, GACR, GA 102/07/0850, Grant Agency of Czech Republic, "Integrated approach to education of PhD students in the area of parallel and distributed systems", GACR, GD102/05/H050, Grant Agency of Czech Republic and was supported by the Research Plan No. MSM 0021630528 – “Security-Oriented Research in Information Technology”.

8. REFERENCES

[1] Duato, J., Yalmanchili, S. *Interconnection Networks, An Engineering Approach.* Morgan Kaufman Publishers, Elsevier Science, 2003

[2] Tsai, Y., McKinley P. K. *An Extended Dominating Node Approach to Broadcast and Global Combine in Multiport Wormhole-Routed Mesh Networks*. IEEE Trans. on Parallel and Distributed Systems, vol.8, no.1, 1997

[3] Tsai, Y., McKinley P. K. *A Broadcast Algorithm for All-Port Wormhole Routed Torus Networks*. IEEE Trans. on Parallel and Distributed Systems, vol.7, no.8, 1996

[4] Gabrielyan, E., Hersch, R. D. Efficient Liquid Schedule Search Strategies for Collective Communications. In *Proc. of the 12th IEEE International Conference on Network ICON 2004*, Singapore, Vol. 2, pp 760-766, Nov. 2004

[5] Ocenasek, J. *Parallel Estimation of Distribution Algorithms*. PhD. Thesis, Faculty of Information Technology, Brno University of Technology, Brno, Czech Rep., 2002

[6] Jaroš, J., Ohlídal, M., Dvořák, V. *Evolutionary Design of Group Communication Schedules for Interconnection Networks*. Lecture Notes in Computer Sciences 3733, Berlin, DE, Springer, pp. 472-481, 2005

[7] Chalmers, A., Tidmus, J. *Practical Parallel Processing*. International Thomson Computer Press, 1996

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

[9] Zomaya, A., *Parallel and Distributed Computing Handbook*. McGraw Hill, 1996

[10] Jantsch, A., Tenhunen, H, *Networks on Chip*, Kluwer Academic Publ. Boston, 2003

[11] IBM Blade system, URL: http://www-03.ibm.com/systems/bladecenter/

[12] Ohlídal, M., Schwarz, J. Hybrid parallel simulated annealing using genetic operations. In *Proc of Mendel 2004 10th International Conference on Soft Computing*, Brno, CZ, FSI VUT, pp. 89-94, 2004

[13] Metropolis, N., Rosenbluth, A., Rosenbluth M. N., Teller A., Teller, E. *Equation of state calculations by fast computing machines*, J. Chem. Phys. 21, 1087 (1953).

[14] Kita, H. Simulated annealing. In *Proceeding of Japan Society* for Fuzzy Theory and Systems, Vol. 9, No. 6, 1997

[15] Larrañaga P., Lozano J. A. *Estimation of Distribution Algorithms, A New Tool for Evolutionary Computation*, Kluwer Academic Publishers, 2002

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

[17] Ohlídal, M., Jaroš, J., Dvořák, V. Schwarz Josef: Evolutionary Design of OAB and AAB Communication Schedules for Interconnection Networks. In *Lecture Notes in Computer Science*, n. 3907, DE, pp. 267-278, ISSN 0302-9743, 2006

[18] 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, p. 4, ISBN 9639364487, 2005