Evolutionary Design of OAB and AAB Communication Schedules for Interconnection Networks

Miloš Ohlídal[1], Jiří Jaroš[1], Josef Schwarz[1], Václav Dvořák[1]

[1] Brno University of Technology, Faculty of Information Technology, Department of Computer Systems, Božetěchova 2, 612 66 Brno, Czech Republic
phone: +420-541141149, fax: +420-541141270
{Ohlidal, Jarosjir, Schwarz, Dvorak}@fit.vutbr.cz

Abstract. Since chip multiprocessors are quickly penetrating new application areas in network and media processing, their interconnection architectures become a subject of sophisticated optimization. One-to-All Broadcast (OAB) and All-to-All Broadcast (AAB) [2] group communications are frequently used in many parallel algorithms and if their overhead cost is excessive, performance degrades rapidly with a processor count. This paper deals with the design of a new application-specific standard genetic algorithm (SGA) and the use of Hybrid parallel Genetic Simulated Annealing (HGSA) to design optimal communication algorithms for an arbitrary topology of the interconnection network. Each of these algorithms is targeted for a different switching technique. The OAB and AAB communication schedules were designed mainly for an asymmetrical AMP [15] network and for the benchmark hypercube network [16] using Store-and-Forward (SF) and Wormhole (WH) switching.

1 Introduction

With parallel and distributed computing coming of age, multiprocessor systems are more frequently found not only in high-end servers and workstations, but also in small-scale parallel systems for high performance control, data acquisition and analysis, image processing, networking processors, wireless communication, and game computers. The design and optimization of hardware and software architectures for these parallel embedded applications have been an active research area in recent years. For many cases it is better to use several small processing nodes rather than a single big and complex CPU. Nowadays, it is feasible to place large CPU clusters on a single chip (multiprocessor SoCs, MSoCs), allowing both large local memories and the high bandwidth of on-chip interconnect.

One of the greatest challenges faced by designers of digital systems is optimizing the communication and interconnection between system components. As more and more processor cores and other large reusable components have been integrated on single silicon die, a need for a systematic approach to the design of communication part has become acute. One reason is that buses, the former main means to connect the components, could not scale to higher numbers of communication partners. Recently

the research opened up in Network on Chip (NoC) area, encompassing the interconnection/communication problem at all levels, from physical to the architectural to the OS and application level [1].

Presently, there are many different interconnection network topologies for general purpose multiprocessors, but new networks for specific parallel applications can still be created. Whereas the lower bounds on the time complexity of various group communications (in terms of required number of communication steps) can be mathematically derived for any network topology and the given communication pattern, finding a corresponding schedule of communication is more difficult and in some cases it is not known as yet. The rest of the paper addresses the quest for an optimal communication schedule based on evolutionary algorithms, provided that network topology and a communication pattern are given.

2 Models of Communications

Communications between two partners (p2p) or among all (or a subset) of partners engaged in parallel processing have a dramatic impact on the speedup of parallel applications. Performance modelling of p2p and group communications is therefore important in design of application-specific systems. A p2p communication may be random (input data dependent) as far as source-destination pair or a message length is concerned. However, in many parallel algorithms we often find certain communication patterns, which are regular in time, in space, or in both time and space; by space we understand spatial distribution of processes on processors. Communications taking place among a subset or among all processors are called group or collective communications. Examples of these may serve One-to-All Broadcast (OAB), All-to-All Broadcast (AAB), One-to-All Scatter (OAS, a private message to each partner), All-to-One Gather (AOG), All-to-All Scatter (AAS), permutation, scan, reduction and others [2]. Provided that the amount of computation is known, as is usually true in case of application-specific systems, the only thing that matters in obtaining the highest performance are group communication times.

The simplest time model of communication uses a number of communication steps (rounds): point-to-point communication takes one step between adjacent nodes and a number of steps if the nodes are not directly connected.

Two types of switching are used in this article. The first one is distance-sensitive Store-and-Forward (SF). Each intermediate node on the path firstly receives the whole message and then sends it to adjacent node in the next possible communication step. The second type of switching is called wormhole (WH) switching. Here several p2p messages between source-destination pairs, not necessarily neighbours can proceed concurrently and can be combined into a single step if their paths are disjoint. Of course, for simplicity, we assume no contention for channels and no resulting delays. An example of these switching techniques is shown in Fig. 1.

Further, we have to distinguish between unidirectional (simplex) channels and bidirectional (half-duplex, full-duplex) channels. The number of ports that can be engaged in communication simultaneously (1-port or all-port models of routers) has also

an impact on number of communication steps and communication time, as well as if nodes can combine/extract partial messages with negligible overhead (combining model) or can only retransmit/consume original messages (non-combining model).

We used all-port non-combining model in our experiments. The goal was to find communication algorithms whose time complexity is as close as possible to mathematically derived lower bounds on number of communication steps.

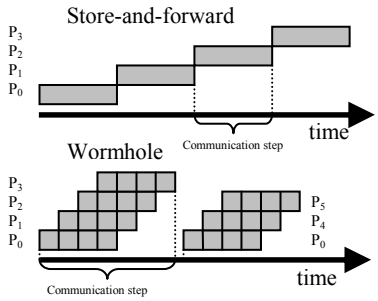

Fig. 1. Basic type of switching techniques

In our experimental runs mostly the well known hypercube [16] and AMP network [15] topologies were tested. Optimal schedules for the former topology are known and can therefore be used to evaluate quality of used algorithms; the feature of the latter topology (for which optimal schedules are unknown) is, that the number of nodes with degree *d* that can be connected in a network is maximum.

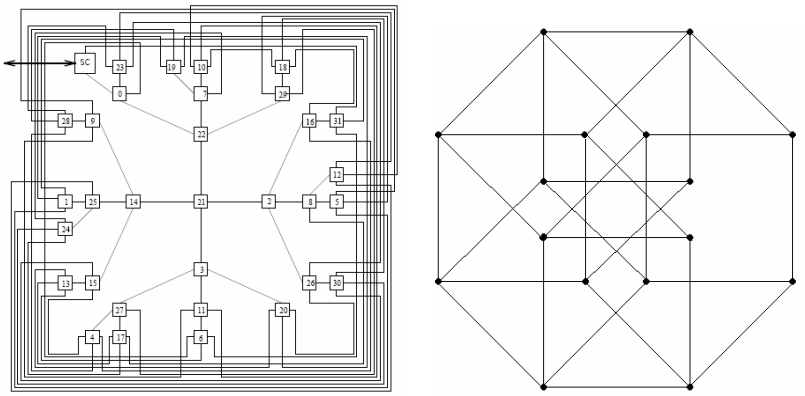

Fig. 2. 32 processors AMP topology and 16 processors hypercube topology

3 Discrete Optimization Algorithms

Combinatorial search and optimization techniques in general are characterized by quest for a solution to a problem from among many potential solutions. For many search and optimization problems, exhaustive search is infeasible and some form of guided search is undertaken instead. In addition, rather than only the best (optimal) solution, a good non-optimal solution is often sought.

3.1 Standard Genetic Algorithm (SGA)

Genetic algorithm [3] is a powerful, domain-independent search technique. SGA is a population-based computational model that uses selection and recombination operators to generate new samples in the search space. A chromosome, consisting of genes, represents one encoded solution from the search space. The values of genes are referred to as alleles. The chromosomes form population, which changes through the evolution process. The reproduction process is performed in such a way that chromosomes, which represent better solutions, are given more chances to reproduce than those chromosomes, which represent poorer solutions. The fitness function (a measure of quality) of chromosomes is defined in the frame of the population. The fitness function is applied to genotype (chromosomes) for evaluating phenotype (decoded form of the chromosome).

One point crossover and integer bound mutation were used as recombination operators and tournament selection as selection operator.

3.2 Hybrid Parallel Genetic Simulated Annealing (HGSA)

HGSA [7] is a hybrid method that uses parallel Simulated Annealing (SA) [10] with the operations used in standard genetic algorithms [8]. In the proposed algorithm, several SA processes run in parallel. After a number of steps (after every ten iterations of Metropolis algorithm), the crossover is used to produce new solutions.

During communication, which is activated each 10th iteration of Metropolis algorithm, all processes 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, all processes have two individuals. Now the phase of genetic crossover starts. Two additional children solutions are generated from two parent solutions using double-point crossover. The solution with the best value of the fitness function is selected and mutation is performed: always in case of the parent solution, otherwise with a predefined probability. Mutation is performed by randomly selecting genes and by randomly changing their values. A new solution of each process is selected from the actual solution provided by SA process and from the solution, which was obtained after genetic mutation. The selection is controlled by well-known Metropolis criterion.

4 OAB and AAB Communication Patterns

OAB (One-to-All Broadcast) [4, 5] is a collective communication pattern. In this case, one node (initiator) distributes the same message to all other nodes in the interconnection network. If only node subset takes part in communication, we talk about multicast communication pattern (MC). This communication (as well as OAS [11, 12] with distinct messages to receiving nodes) can be performed by sequentially sending the message to particular nodes. This way is very inefficient because only one node sends the message in each communication step. However we can use a better technique using a broadcast tree when every node that received the message in previous communication step becomes an initiator of new multicast communication. Consequently, the number of informed nodes increases by d^k instead by d, where d is the node degree and k is number of communication step.

The goal of the proposed evolutionary algorithm is to find such a broadcast tree (communication schedule) that it will be possible to inform all nodes in the minimal number of communication steps. A resulting communication schedule has to be conflict-free, i.e. only one message can be transmitted via the same link in the same step and the same direction.

Optimal communication schedules for OAB communication pattern using store and forward and wormhole switching technique on eight nodes ring topology are shown on the left side of Fig. 3. Broadcast trees are shown on the right side.

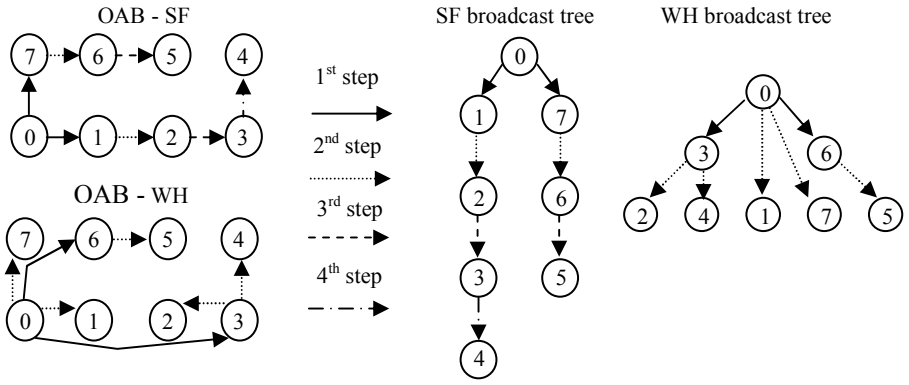

Fig. 3. The optimal OAB schedules for 8 nodes ring topology and the relevant broadcast trees.

The lower bounds on the number of communication steps for the all-port hypercube and AMP topology are shown in Tab. 1. Parameters of the interconnection network in Tab. 1 are: processors count P, network diameter D, node degree d, bisection width B_C, and average distance d_a.

Table 1. Lower bounds on number of communication steps (all-port models) [13]

	SF hypercube	WH hypercube	SF AMP	WH AMP
OAB	$D\ (= d)$	$\lceil d/\log(d+1) \rceil$	D	$\lceil \log_{d+1} P \rceil$
AAB	$\lceil (P-1)/d \rceil$	$\lceil (P-1)/d \rceil$	$\lceil (P-1)/d \rceil$	$\lceil (P-1)/d \rceil$

5 Design of Algorithms

The goal of proposed algorithms is to find a schedule of a group communication with the number of steps as close as possible to the above lower bounds. The solution of this optimization problem by means of evolutionary algorithms may be decomposed into several phases. In the first phase, it is necessary to choose a suitable encoding of the problem into a chromosome. The second step is a definition of the fitness function, which determines quality of a chromosome. The next phase is design of the input data structure for the evolutionary algorithm. The last phase includes experimental runs of the evolutionary algorithm and search for the best set of its parameters. The choice of parameters should speed-up the convergence of the algorithm and simultaneously minimizes a probability of getting stuck in local minima.

5.1 Solution encoding

Different encodings were used for each optimization algorithm according to the switching technique. We used an indirect encoding for OAB with wormhole switching optimized by SGA algorithm. Thus a chromosome does not include a decision tree, but only instructions how to create it from chromosome. Any chromosome consists of P genes. Every gene corresponds to one destination node. Individual genes include three integer values. The first one is a source node index. The second one determines the shortest path along which the message will by transmitted. The last one is a communication step number when the communication will be performed.

The main disadvantage of this encoding is formation of inadmissible solutions during process of genetic manipulation. We say that a solution is inadmissible if it is not possible to construct correct broadcast tree from it. An example of inadmissible solution can be a case when some node receives a message in a given step from a node that has not received the message yet. That is why admissibility verification has to be carried out for every solution before every fitness function evaluation and if the need be, the restoration will be accomplished. In Fig. 4, a chromosome for wormhole OAB communication patter for the 8-node ring topology is presented.

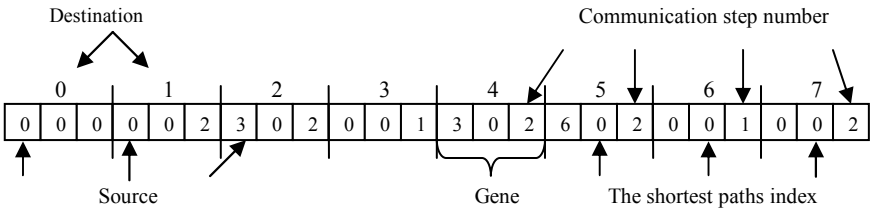

Fig. 4. Encoding of broadcast tree in chromosome for SGA case

Very simply encoding of SF OAB communication pattern has been chosen for HGSA. Every chromosome consists of *P* genes, where *P* is a number of processors in a given topology. The gene's index represents the destination processor for a message. The gene consists of two integer components. The first component is an index of one of the shortest path from source to destination. The second component is a sequence of communication links on the path. Fig. 5 illustrates an example of this encoding. The source processor has index 0. For completeness the chromosome includes also communication from source to source processor, but this communication is not realized. This gene is included only for the easier evaluation of the fitness function.

The main advantage of this encoding is a short chromosome and the absence of inadmissible solutions (every message is transmitted from the source to a destination). The main disadvantage is a large number of possible values of the first gene component. The number of the values rapidly increases with the distance from source to destination as there are more shortest paths between them.

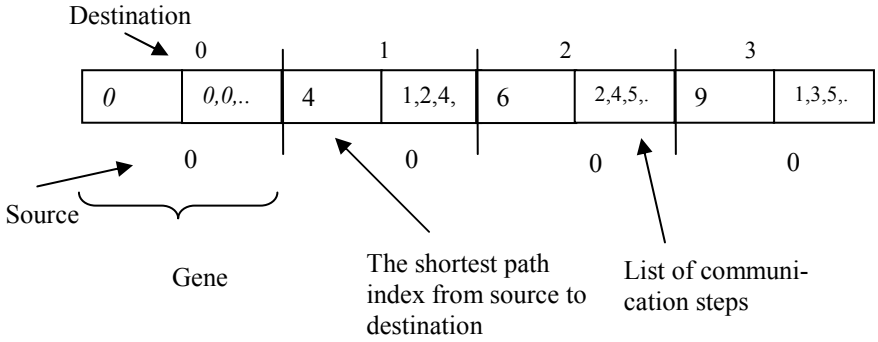

Fig. 5. The structure of chromosome of HGSA in case of OAB

The AAB chromosome is an extension of a vector to matrix for both optimization algorithms SGA and HGSA. An AAB chromosome is composed of *P* OAB chromosomes as every processor performs OAB.

5.2 The fitness function

The fitness function evaluation is the same for both proposed algorithms. It is based on testing of conflict-freedom. We say that two communication paths are in conflict if and only if they use the same communication link in the same time and in the same direction (see Fig. 6). The fitness function is based on conflict counting. The optimal communication schedule for the given number of communication steps must be conflict-free. If the conflict occurs, the schedule can not be used in real application.

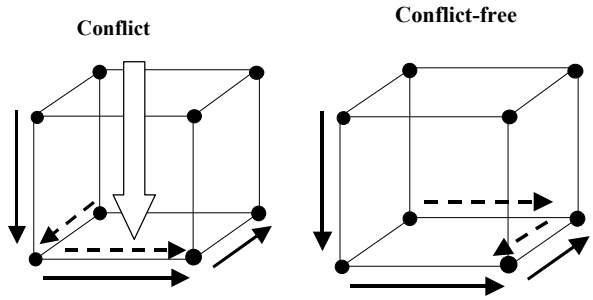

Fig. 6. Conflict in a communication schedule

5.3 The shortest paths algorithm

This algorithm generates all shortest paths and saves them in the operating memory into a specific data structure. The generating algorithm [6] is inspired by the breadth-first search algorithms BFS. BFS is based on the searching a graph, where the source processor is chosen as a root. The edges create a tree used in searching process. A tree is gradually constructed, one level at a time, from the root that is assigned an index of a source node. When a new level of the tree is generated, every node at the lowest level (leaf) is expanded. When a node is expanded, its successors are determined as all its direct neighbours except those, which are already located at higher levels of the tree (it is necessary to avoid cycles). Construction of the tree is finished when a value of at least one leaf is equal to the index of a destination node. Destination leaves' indices confirm the existence of searched paths, which are then stored as sequences of incident node indices.

5.4 Heuristics

In SGA a new heuristic for chromosome restoration was used. The restoration (correction of the broadcast tree) proceeds subsequently in particular communication steps. For every node we check if it receives the message from the node that has already received it in some previous communication step. As far as this condition is not satisfied, the source node of this communication is randomly replaced by a node that al-

ready has the message. Further, it is necessary to check already used shortest paths. There is a finite number of the shortest paths from every source to every destination node. If the second gene component (the index path) exceeds this value, the modulo operation will be applied to this gene component.

In HGSA two heuristics are used to speed up the convergence to a sub-optimal solution. They decrease the probability of being trapped in local optima during the execution. The idea is a simple reduction of the path length. The first heuristics is used after the initialization of HGSA and then after each application of Metropolis algorithm. The length of the path from the source to the destination node has some value. If the end node occurs in another gene with a smaller length, than the length and the path in the original gene are changed accordingly.

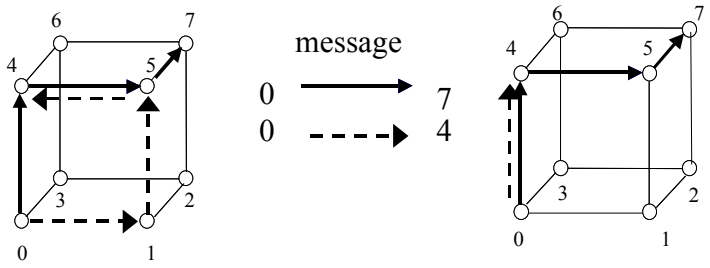

Fig. 6. : Reduction shorter path according to longer path

The second heuristics is used in all surveyed collective communications. It removes using proper setting of the communication step for several nodes incident with the examined path. It means really to endeavour after suspending message in the node during the usage of the same link by another message. However, this changing must not increase the number of the communication steps of the optimal schedule.

In the case that above presented way doesn't lead to improvement, it tries other way and it is endeavor after the fastest sending message from source to destination.

6. Experimental Results

Both sequential SGA and parallel HGSA have been implemented in C/C++. They use only standard C and C++ libraries to ensure good portability. HGSA implementation uses MPI [9] routines for message passing and can therefore be compiled and run on any architecture (clusters of workstations, MPPs, SMPs, etc.) for which an implementation of MPI standard is available.

The proposed algorithms were verified on some multiprocessor topologies (e.g. Midimew, K-Ring...). Two topologies were examined most intensively, namely five cases of hypercubes and five cases of AMP network topologies were used. Other topologies were tested only in 8-node configuration.

The theoretical time complexity in terms of a minimal number of communication steps can be derived for all examined topologies. Theoretical lower bounds of tested topologies are shown in Tab. 2.

Table 2. Theoretical lower bounds of tested topologies

Lower bounds	Hyper-8	Hyper-16	Hyper-32	Hyper-64	Hyper-128
OAB	2	2	2	3	3
AAB	3	4	5	6	7
	AMP-8	AMP-23	AMP-32	AMP-42	AMP-53
OAB	2	2	3	3	3
AAB	2	6	8	11	13
	K-ring	Midimew	Moore	Octagon	Ladder
OAB	2	2	2	2	4
AAB	2	2	3	3	4

Parameters of SGA were set to the same values for all runs, i.e. probability of crossover 70%, probability of mutation 5%. 10 runs of SGA were performed for each topology, whereas the size of population was set on the value, in which success rate was better than 50%.

Parameters of HGSA were set to the same values for all runs too, i.e. 10 computers in the master slave architecture, the length of communication interval between master and slave was each 10's iterations of Metropolis algorithm 10/10 (OAB/AAB), start temperature 100, number of iterations in each temperature phases was 10, gradient of cooling 0.9/0.99 (OAB/AAB). 15 runs of HGSA were performed for each topology.

We counted only the successful completions, i.e. those reaching the global optimum. The success rate of both algorithms (SGA and HSGA) was measured and compared. If we compare success rate (Tab. 3) of AMP-23 and AMP-32 topology, we see that the success rate is better for more complex topology. While for AMP-23 not rounded time complexity is 1.94 steps, for AMP-32 it is 2.15 steps. The time complexity of optimal communication schedule can not exceed two communication steps in the first case whereas it can be split into three steps in the second case. By comparing not rounded and rounded time complexities we can make a conclusion, that in the case of AMP-32 topology, much more interconnection links remain unused and the evolutionary algorithm has more space to find the optimal schedule. The same abnormality can be seen in some other topologies (hyper-32 and hyper-64). The success rate 100% was achieved for all other examined topologies.

The presented data of HGSA deserves some comments. Firstly, OAB (SF) is quite a simple operation and therefore the algorithm is likely to find an optimal solution even for larger architectures. Optimal solutions have already been found for topologies with up to 32 processors and acceptable results have been attained for AAB. A further improvement of these results can be expected in the future, because number of experiments, which could be carried out so far, was limited by the overall run time required for optimization (many hours if optimal solutions are sought). On the other hand, if we need an acceptable solution quickly, the proposed algorithms allow to

accept a larger number of communication steps and the solution is found in much shorter time.

Table. 3. Success rate in achieving the optimum schedule

	Hyper-8	Hyper-16	Hyper-32	Hyper-64	Hyper-128
SGA – OAB	100%	100%	50%	60%	50%
HGSA - OAB	100%	100%	100%	100%	100%
SGA – AAB	70%	20%	-	-	-
HGSA - AAB	100%	80%	-	-	-
	AMP-8	AMP-23	AMP-32	AMP-42	AMP-53
SGA – OAB	100%	50%	100%	60%	50%
HGSA - OAB	100%	100%	100%	100%	100%
SGA – AAB	70%	30%	10%	-	-
HGSA - AAB	100%	80%	10%	-	-

7 Conclusions

Optimization of communication schedules by means of the proposed evolutionary algorithms has been successful. Optimal communication schedules achieve the lower bounds of communication steps derived from graph-theoretical properties of interconnection networks. It is evident that optimum schedules can speed-up execution of many parallel programs that use collective communication as a part of their algorithm.

We have tested two types of evolutionary algorithms. The first one is standard genetic algorithm SGA and the second one HGSA is a composition of parallel simulated annealing and the standard genetic algorithm. Both presented algorithms are able to find an optimal schedule of the given communication pattern for arbitrary network topology, each one with sufficient efficiency.

The future work will be focused on the communication patterns OAS and AAS in case of HGSA and OAB, AAB in case of Estimation of Distribution Algorithms (EDA) [14]. We will implement the multicriterial optimization in EDA algorithms (without the need not to enter the number of communication steps) and to design and implement more efficient heuristics for HGSA.

Importance and novelty of above goals should be emphasized. Algorithms, which would be able to find all types of collective communication on any regular or irregular topology, were not published so far in spite of a growing importance especially for multiprocessors on chips.

References

1. Jantsch, A., Tenhunen, H.: Networks on Chip, Kluwer Academic Publ., Boston, 2003, ISBN 1-4020-7392-5

2. Gabrielyan, E.: Hersch, R.D.: Network′s Liquid Throughput: Topology Aware Optimization of Collective Communication. Unpublished work, 2003
3. Goldberg, D.: Genetics Algorithms in Search, Optimization, and Machine Learning, Addision-Wesley Publishing Company, 1989
4. Defago, X., Schiper, A., Urban, P.: Total Order Broadcast and Multicast Algorithms: Taxonomy and Survey, technical report DSC/2000/036, 2003
5. Duato, J., Yalamanchili, S.: Interconnection Networks – An Engineering Approach, Morgan Kaufman Publishers, Elsevier Science, 2003
6. Staroba J.: Parallel Performance Modelling, Prediction and Tuning, PhD. Thesis, Faculty of Information Technology, Brno University of Technology, Brno, Czech Rep., 2004
7. Ohlídal, M., Schwarz, J.: Hybrid parallel simulated annealing using genetic operations, In: Mendel 2004 10th Internacional Conference on Soft Computing, Brno, CZ, FSI VUT, 2004, pp. 89-94
8. Goldberg D. E.: A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing, Complex Systems, 1990, pp. 445–460.
9. http://www-unix.mcs.anl.gov/mpi
10. Kita, H.: Simulated annealing. Proceeding of Japan Society for Fuzzy Theory and Systems, Vol. 9, No. 6, 1997
11. 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. 206-210
12. Jaroš, J., Ohlídal, M., Dvořák, V.: Evolutionary Design of Group Communication Schedules for Interconnection Networks, In: Proceedings of 20th International Symposium of Computer and Information Science, Berlin, DE, Springer, 2005, s. 472-481
13. Dvořák, V.: Scheduling Collective Communications on Wormhole Fat Cubes, In: Proc. of the 17th International Symposium on Computer Architecture and High Performance Computing, Los Alamitos, US, IEEE CS, 2005, pp. 27-34
14. Pelikan, M., Goldberg, E., Sastry, K.: Bayesian Optimization Algorithm, Decision Graphs, and Occams Razor.
15. Chalmers, A.-Tidmus, J.: Practical Parallel Processing. International Thomson Computer Press, 1996
16. http://www.sgi.com/products/remarketed/origin/pdf/hypercube.pdf

Acknowledgement

This research has been carried out under the financial support of the research project FRVS 2848/2006/G1 “Memetic Evolutionary Algorithms Applied to Designed of Group Communication between Processors”, FRVS 2983/2006/G1 “EDA Evolutionary Design of Group Communication Schedules”, (Ministry of Education) and of the research grant GA102/05/0467 “Architectures of Embedded Systems Networks” (Grant Agency of Czech Republic).