Solving the Travelling Salesman Problem in Parallel by Genetic Algorithm on Multicomputer Cluster

Plamenka Borovska

Abstract: *The paper investigates the efficiency of the parallel computation of the travelling salesman problem using the genetic approach on a slack multicomputer cluster. For the parallel algorithm design functional decomposition of the parallel application has been made and the manager/workers paradigm is applied. Performance estimation and parallelism profiling have been made on the basis of MPI-based parallel program implementation.*

Key words: *Cluster Computing, Evolutionary Computing, TSP, Genetic Algorithms, Parallelism Profiling, Performance Estimation, Parallel Programming, MPI.*

INTRODUCTION

A wide spectrum of application areas demands time-consuming computation for solving various optimization problems i.e. finding local extrema, more often global extrema or near-optimal solutions. One of the most efficient approaches for global optimization is to utilize genetic algorithms [1, 2]. Genetic algorithms are powerful and promising technique for search problems to find out approximate solution to optimization. Genetic algorithms, being a particular class of evolutionary algorithms, imply the concepts of evolutionary biology (recombination, inheritance, natural selection, mutation). The most popular utilizations of GA is for computer simulation where candidates for an optimal solutions (individuals), presented abstractly by chromosomes, undergo evolution by performing recombination and mutation and the newly created populations tend to produce the optimal or near-optimal solution. The main goal of genetic algorithms is to search a solution space imitating the biological metaphor and, therefore, they are inherently parallel. Computer clusters [3, 4] are widely used nowadays for solving time-consuming problems.

The travelling salesman problem (TSP) is a well known combinatorial problem [5, 6, 7]. The idea of the TSP is to find the shortest tour of a group of cities without visiting any town twice, but, practically, it implies the construction of a Hamiltonian cycle within a weighted fully connected undirected graph. Therefore, this is a problem of combinatorial graph search. The TSP is maybe one of the most explored problems in computer science. The applications of the TSP problem are numerous – in computer wiring, job scheduling, minimizing fuel consumption in aircraft, vehicle routing problem, robot learning, etc.

The purpose of this paper is to investigate the performance of the parallel implementation of the TSP problem on a slack cluster of computers.

THE GENETIC APPROACH FOR SOLVING THE TSP PROBLEM

The standard TSP may be presented mathematically as finding the Hamiltonian cycle of minimal weight within a weighted fully connected undirected graph $G = (V, E)$ where the vertices present the cities, the edges denote the intercity paths, and the weights of the edges represent the intercity distances. The deterministic method to solve the TSP problem involves traversing all possible routes, evaluating corresponding tour distances and finding out the tour of minimal distance. The total number of possible routes traversing n cities is $n!$ and, therefore, in cases of large values of n it becomes impossible to find the cost of all tours in polynomial time. Parallel processing helps reducing the computation time of the TSP problem. Furthermore, our intention is to apply the genetic approach for finding the near-optimal solution of the TSP problem and to investigate the impact of the programming model on the performance of parallel system for that particular application.

The biological metaphor involves evolving populations where the size of the population is 40 and each individual is denoted by 40 chromosomes. Chromosomes present the order of cities in which the salesman will visit them. In our case permutation

encoding is used where every chromosome is a string of numbers that represent a position in a sequence. For the chromosomes permutation coding is used which is the best method for coding ordering problems. The fitness represents the length of the tour. The selection is performed following the rules of the roulette wheel method – the individuals of the highest fitness are selected for parents. The method of recombination is that of two crossover points – two parts of the first parent are copied, and the rest between these two parts is taken in the same order as in the second parent. The mutation applied is of the normal random type and involves changing of the city order. The parallel genetic computation of the TSP on the cluster platform has been performed for 40 generations, the population size being 40, the genome has been represented by 10 bits, the probability of crossover happening used is 0.75, and the probability of each bit being mutated has been 0.1. After a new city is added or deleted it is necessary to create and evolve new populations presented by the corresponding chromosomes and the genetic algorithm is restarted.

IMPLEMENTING THE FLAT PROGRAMMING MODEL

For the parallel genetic computation of the TSP functional decomposition is applied. For the flat (MPI-based) parallel program implementation the algorithmic paradigm is of the manager/workers type. The experimental parallel computer platform is a multicomputer platform comprising five workstations communicating via switch. The parallel model of computation is presented in Fig.1.

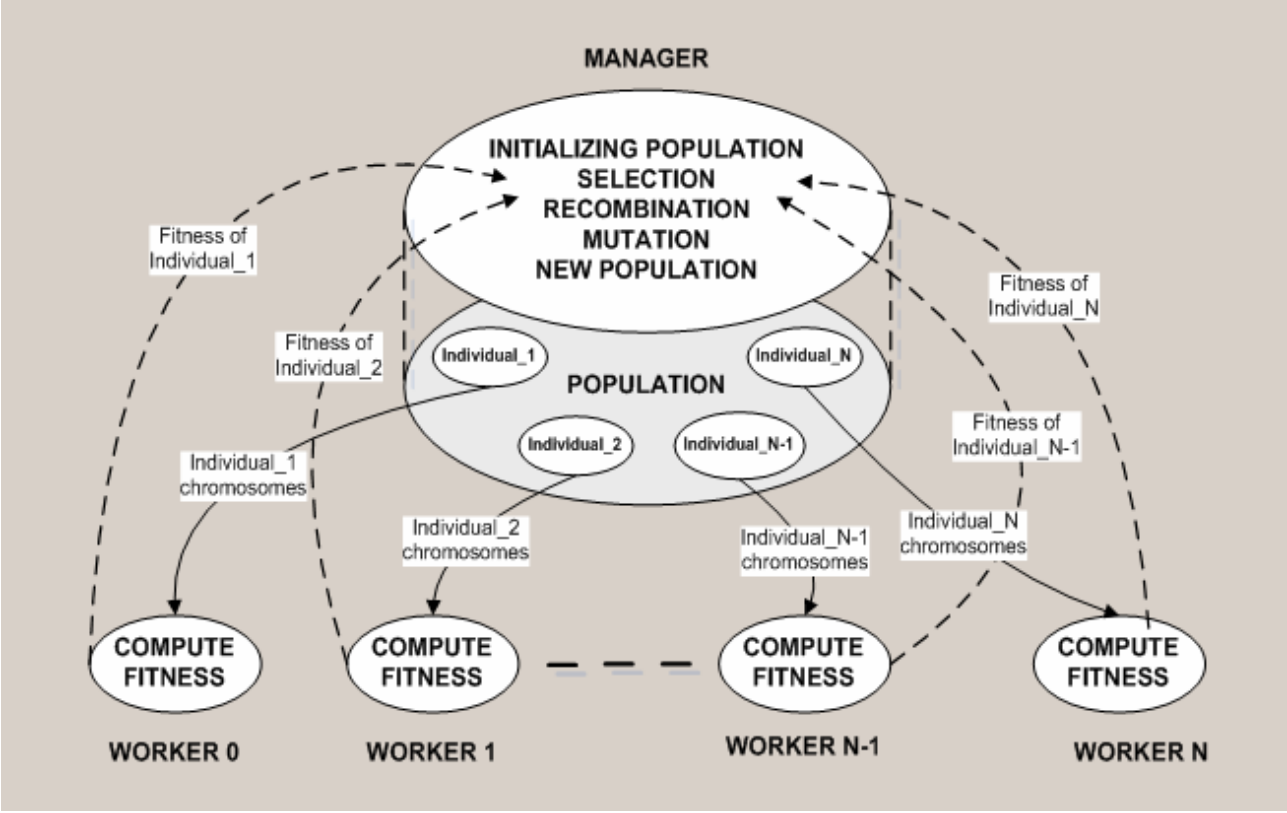

Fig.1. The flat parallel model of computing the TSP by genetic algorithm.

The manager process (rank 0) performs all genetic operations and distributes the computational load among the worker processes. It performs the following activities:

1. Initializes population
2. Sends a genome to be evaluated; Distributes the individuals among the workers by sending the chromosomes of the individuals (MPI_Send) to the workers to evaluate their fitness

3. Receives (MPI_Irecv) the evaluated fitness from workers
4. Evaluates the fitness of the entire population
5. Performs roulette wheel selection
6. Performs recombination (selects 2 points for crossover)
7. Performs normal random mutation (two cities are chosen and exchanged)
8. Generates new population
9. In case the number of the current generation is less than the maximum number - to 2
10. In case the number of the current generation equals the maximum number sends a termination message to all workers
11. Prints the computed shortest path

The operations of a worker process are as follows:

1. Signals the manager that it is ready for work
2. Waits for genomes to be sent by the manager until a termination message is received
3. Evaluates the fitness of each chromosome of the individual
4. Sends the computed fitness of the individual to the manager
5. Receives termination message from the manager

The manager sends the chromosomes to the workers by means of synchronous send transaction (MPI_Send) because of the insignificant delay. Once the manager has sent the genomes to all the worker processes it waits for them to finish by executing the MPI function MPI_Waitall. The manager gets the computed fitness values from the workers through a nonblocking communication transaction (MPI_IRecv) using request for keeping track of the completion of the communication transaction. Performing the communication transactions requires sending the size of the array being transferred, the type of the elements, the source of the message, the specific tag denoting the message, and the communicator specifying the group of processes taking part in the particular communication transaction (MPI_COMM_WORLD).

The first thing to do by the workers is to inform the manager that they are ready for work. Throughout the computational process they are waiting for genomes from the manager and get them by means of synchronous receive operation (MPI_Recv). Worker processes die upon receiving a terminating message from the manager.

PARALLELISM PROFILING AND PERFORMANCE BENCHMARKING

The parallel program (Visulal C + MPI) implementation is run in the programming environment of MPICH. Parallelism profiling is made for the case of 1000 cities utilizing jumpshot 3.0. The profiling of the communication transactions for the case study under investigation is shown in Fig.2. The connected states of the communication transactions during the parallel computational process are shown in Fig.3. The histogram of the SENDRECV operations is shown in Fig.4. Obviously, that communication is quite intensive among the manager and the workers exchanging data about individuals (chromosomes and fitness). The communication transactions are performed via switched media and so, the communication latency is low. The main shortcoming of the parallel genetic approach of computing the TSP with the manager/workers algorithmic paradigm is the sequential computation of the genetic operations – selection, mutation and reproduction – by the manager process. Gantt's chart for the parallel genetic computation of the TSP on a cluster of five workstations, showing the behavior of the parallel program implementation, is shown in Fig. 5.

The speedup of the parallel genetic computation of the TSP for 1000 cities on the computer cluster is shown in Fig. 6. The efficiency of the parallel system is: 83% for 5 processors, 89% for 4 processors, 90% for 3 processors and 96% for 2 processors.

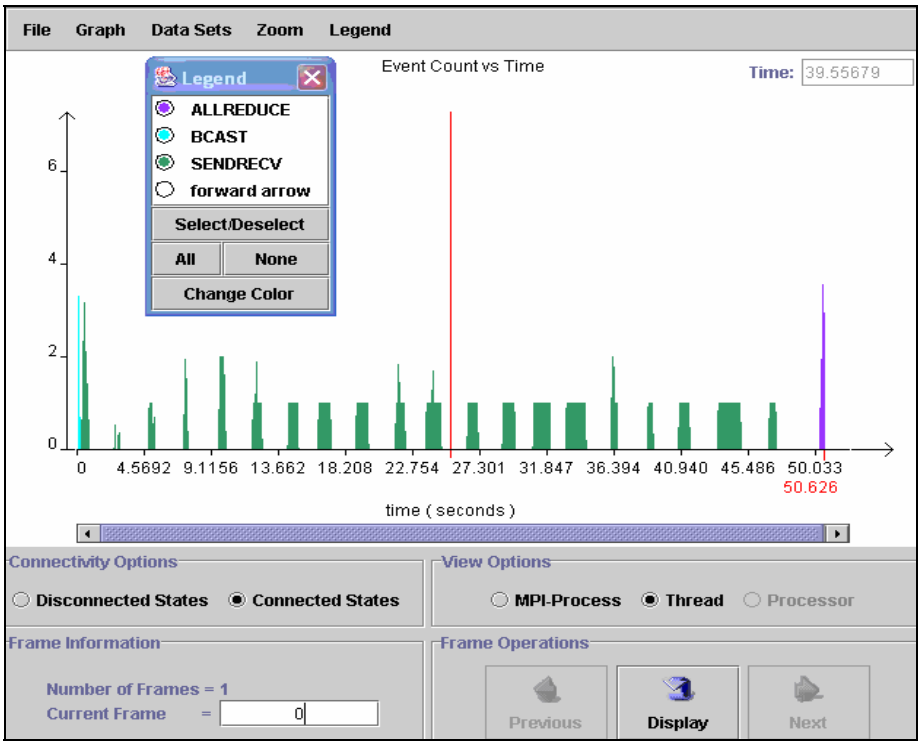

Fig. 2. Profiling the communication transactions.

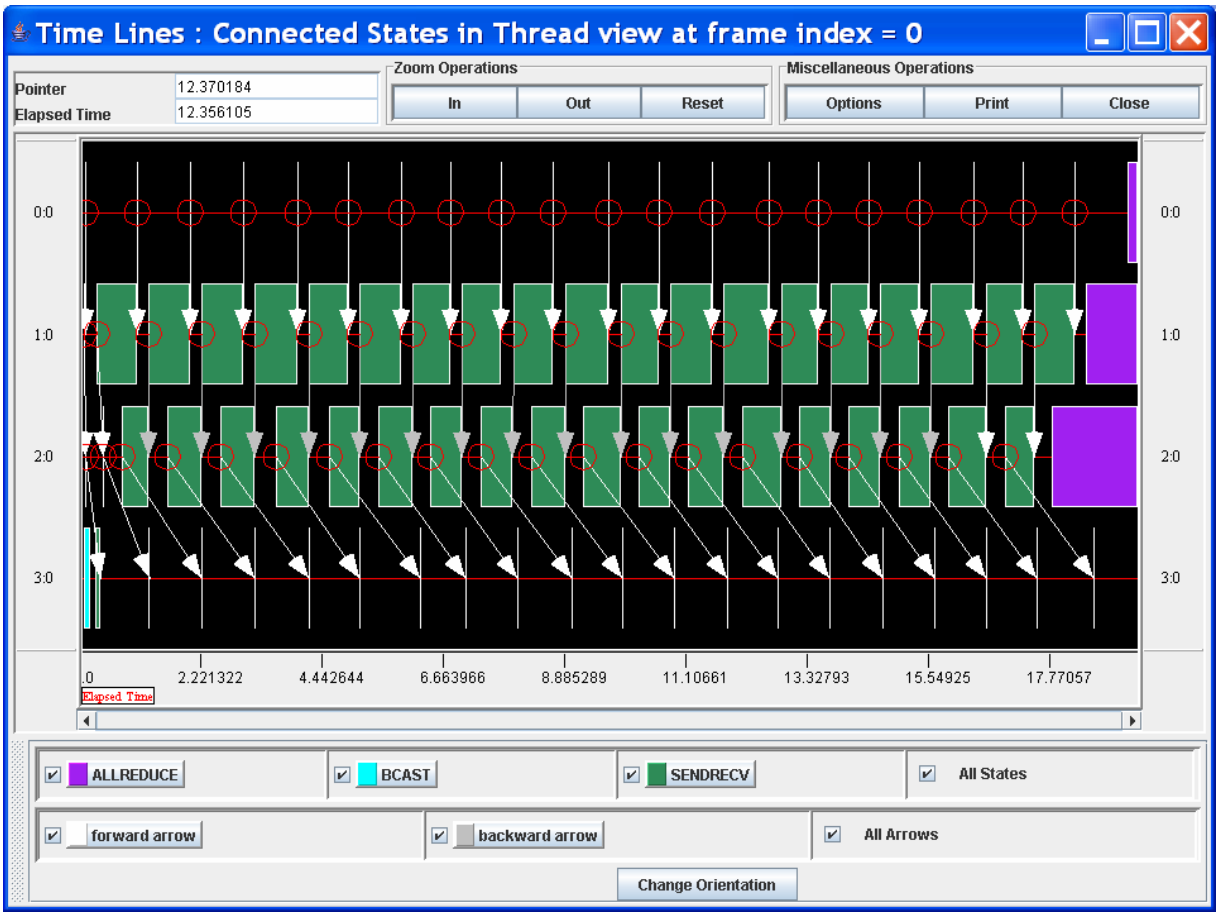

Fig. 3. Connected states of communication transactions.

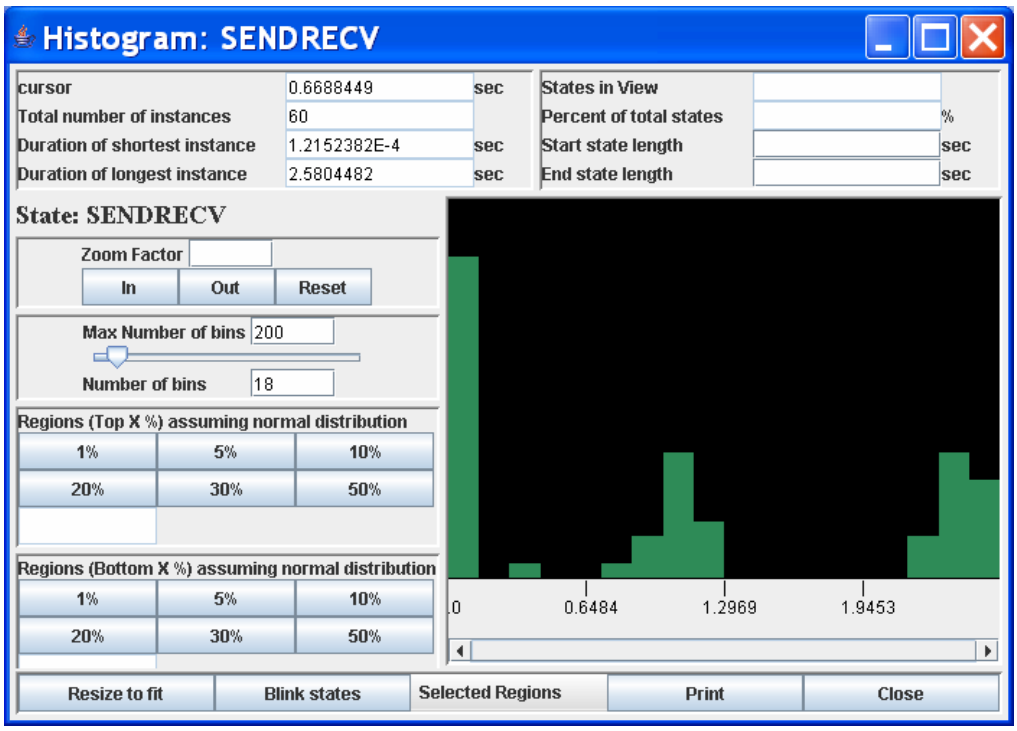

Fig. 4. Histogram of SENDRECV communication transactions.

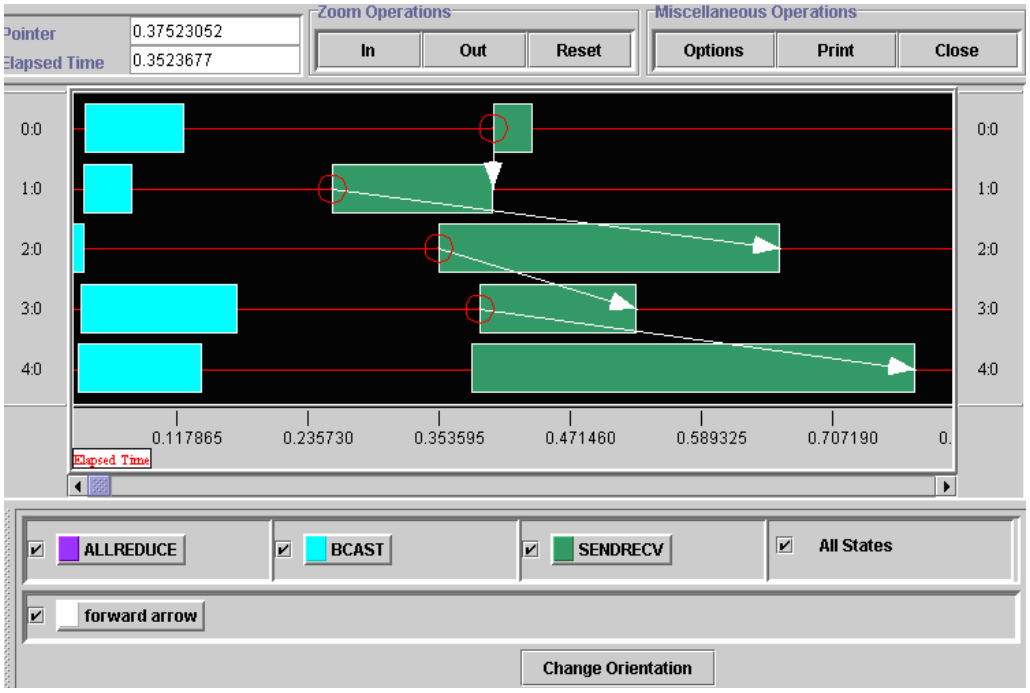

Fig.5. Gantt's chart for the parallel genetic computation of TSP.

CONCLUSIONS AND FUTURE WORK

In this paper the efficiency of the genetic approach for solving the TSP problem in parallel on multicomputer cluster has been investigated. The speedup has been evaluated on the basis of flat (Visual C+MPI) parallel program implementation and parallelism profiling has been performed (jumpshot 3.0). The future work should encompass investigation of the impact of the programming models (flat, multithreaded and hybrid) on the performance of the parallel system.

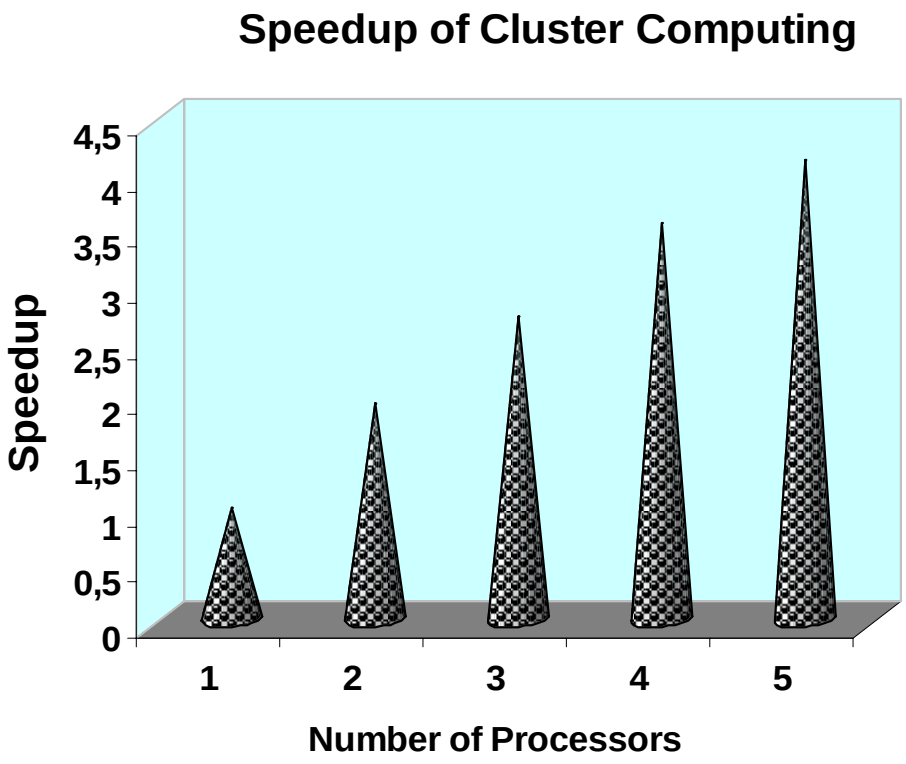

Fig.6. Speedup of cluster computing of TSP by GA for 1000 cities.

REFERENCES

[1] Winter G., J. Periaux, M. Galan. Genetic Algorithms in Engineering and Computer Science, John Wiley & Son Ltd., 1995.

[2] GA White Paper (http://www.manmach.com/information/white.html)

[3] Hennesy J., D. Patterson, Computer Architecture. A Quantative Approach, 3rd Edition, Morgan Kaufmann Pubishers, San Francisco, 2003.

[4] www.clustercomp.org

[5] http://www.cs.bham.ac.uk/~rmp/slide_book/node36.html

[6] http://www.ts.umu.se/~top/travel.html

[7] Borovska P., T. Ivanova, H. Salem. Efficient Parallel Computation of the Traveling Salesman Problem on Multicomputer Platform, Proceedings of the International Scientific Conference 'Computer Science'2004, Sofia, Bulgaria, December 2004, pp. 74-79

[8] Grama Ananth, Gupta Anshul, Kapyris George, Kumar Vipin, Introduction to Parallel Computing, Second Edition, PEARSON, Addison Wesley, 2003.

[9] Quinn Michael, Parallel programming in C with MPI and OpenMP, McGraw-Hill, 2003

[10] William G., E. Lusk, A. Skjellum. Using MPI Portable Parallel Programming with the Message-Passing Interface, MIT press, Cambridge, Massachusetts, London, England, second edition, 1999.

ABOUT THE AUTHOR

Assoc. Prof. Plamenka Borovska, PhD, Head of Computer Systems Department, Technical University of Sofia, Phone: +359 2 965 2524, E-mail: pborovska@tu-sofia.bg.