Parallel Combinatorial Search on Computer Cluster: Sam Loyd's Puzzle

Plamenka Borovska

Abstract: *The paper investigates the efficiency of parallel branch-and-bound search on multicomputer cluster for the case of parallel solving Sam Loyd's puzzle. Performance estimation and analysis as well as parallelism profiling have been made for MPI implementation developed on the basis of the manager/workers parallel algorithmic paradigm. The impact of the number of the processors and the computational workload – the board size – over the performance of the parallel system has been investigated.*

Key words: *Combinatorial Search, Parallel Branch and Bound, Cluster Computing, Parallelism Profiling, MPI Programming, Manager/Workers Algorithmic Paradigm, Distributed Load Balancing.*

INTRODUCTION

The increasing demands of greater and available computing power set up a steady modern tendency of constructing computer clusters [1]. The idea of clustering the available computing resources within one ore more buildings and utilizing them as a single computing resource is very attractive and gives the opportunity to solve time-consuming applications in reasonable time [2]. The easiest way to acquire considerable computing power is to construct a slack cluster of multicomputer architecture, integrating the available computers and use the message passing programming model [3]. In order to increase the level of the exploited parallelism in the case of heterogeneous computer platforms we can combine the flat programming model with multithreading to achieve greater speedup [4].

One of the application areas demanding greater computational power and consuming great computational time is combinatorial search [5]. Combinatorial search is the process of finding "one or more optimal or suboptimal solutions in a defined problem space" [6] and has been used for minimizing the layout of VLSI circuits, for minimizing the traveled distance in robot's motion, proving theorems and playing games. An algorithm that solves an optimization problem must find a solution that is an extreme of an objective function.

This paper investigates the opportunities and possible advantages of parallel combinatorial search on a cluster of computers. The specific problem under investigation is solving in parallel Sam Loyd's puzzle which is an example of the branch-and-bound search technique and may be used as a benchmark for estimating the performance of parallel systems in combinatorial search efficiency. Furthermore, the goal is to explore the correspondence of parallel architectural and algorithmic spaces for combinatorial search problems.

THE PROBLEM OF SOLVING SAM LOYD'S PUZZLE

The well-known 15-puzzle invented by Sam Loyd [7] consists of 15 tiles, numbered 1 to 15, arranged on a 4x4 board. Fifteen locations contain exactly one tile. The sixteenth location is empty. The goal of the puzzle is to repeatedly fill the hole with a tile adjacent to it in the horizontal or vertical direction until the tiles are in row-major order until the tiles are correctly ordered. From our point of view this is an optimization problem – the aim is to solve the puzzle in the least number of moves.

Search problems are represented by state space trees. This state space tree presents the board positions that can be reached from the initial position. Taking into consideration that the goal is to examine as few alternative moves as possible it is a good idea to associate a weight with each state, denoting the minimum number of tile moves made so far needed to solve the puzzle. The weight function adds the number of tile moves made so far to the Manhattan distance between each out-of-place tile and its correct location. The Manhattan distance between two points is the shortest path between these points in case all movements should be in horizontal or vertical directions only.

We consider the case of applying the branch-and-bound technique of search where the initial problem is decomposed into a set of two or more problems of smaller size. The decomposition process is repeated recursively until each unexamined problem is decomposed, solved, or proven that it does not lead to an optimal solution of the original problem. The objective function in the case is the number of moves necessary to order the tiles. Part of the state space tree for solving the 15-puzzle of Sam Loyd by the best-first branch-and-bound search is shown in Fig.1. Obviously, the state space tree is highly unbalanced and, therefore, the parallel solution will require some load balancing in order to achieve good speedup.

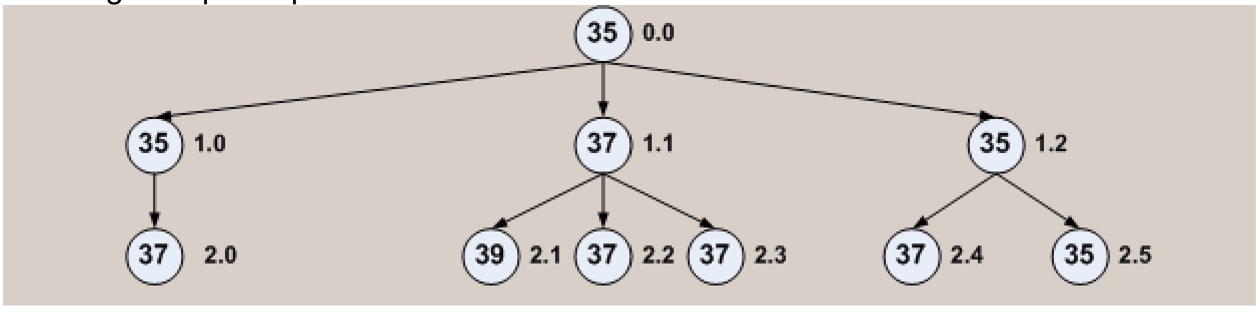


Fig.1. Part of the state space tree (3 levels) for solving the 15-puzzle of Sam Loyd by branch-and-bound search.

In Fig.1 each node of the graph is denoted by the value of the objective function which is a lower bounding function for each subproblem and is formed by the sum of the Manhattan distance and the moves made so far for the particular pattern of tiles. To the right of each node is denoted the consecutive number of the search level. In Fig.2 the computational model for solving the 15-puzzle of Sam Loyd by branch-and-bound search for depth up to three levels is shown. In the worst case, the lower bound function causes the algorithm to perform a breadth-first search of the state space tree without pruning. Consider the case when the optimal solution is found at level *k* of the state space tree with an average branching factor *b*, the worst case time complexity of best-first branch-and-bound search is $O(b^k)$. In that case, the priority queue containing all unexamined subproblems, inserts, on average, *b* nodes in the place of each node being removed.

THE PARALLEL VERSION OF THE SOLUTION

For the parallel solution of the problem the algorithmic paradigm “manager/workers” is applied. The manager process is responsible for the following activities: initializes the primary configuration of tiles on the board, generates the original problem with the corresponding priority queue, divides the original problem into two subproblems, distributes the unexamined problems to worker processes, sends termination token according to the requirements of the modified Dijkstra's distributed termination detection algorithm to worker processes in ring-like order, performs checks to identify the termination of the parallel algorithm, if it gets a white token and the message count is 0, sends a termination message to the worker processes. Each process maintains its own priority queue of unexamined subproblems. The worker processes initially have empty priority queues expecting messages from other processes with unexamined subproblems. They receive messages, containing the termination token. The format of the termination token is shown in Fig.3. In case a process receives a message and computes an unexamined problem with a lower bound less than that of the best solution found so far, it updates the color and the count fields, and the field containing the best solution so far. At last, the process compares the cost of the best solution found so far with the lower bound of the unexamined subproblem at the head of its priority queue. In case the cost of the current best solution is lower or equal to the lower bound of the head unexamined problem, the process empties its priority queue.

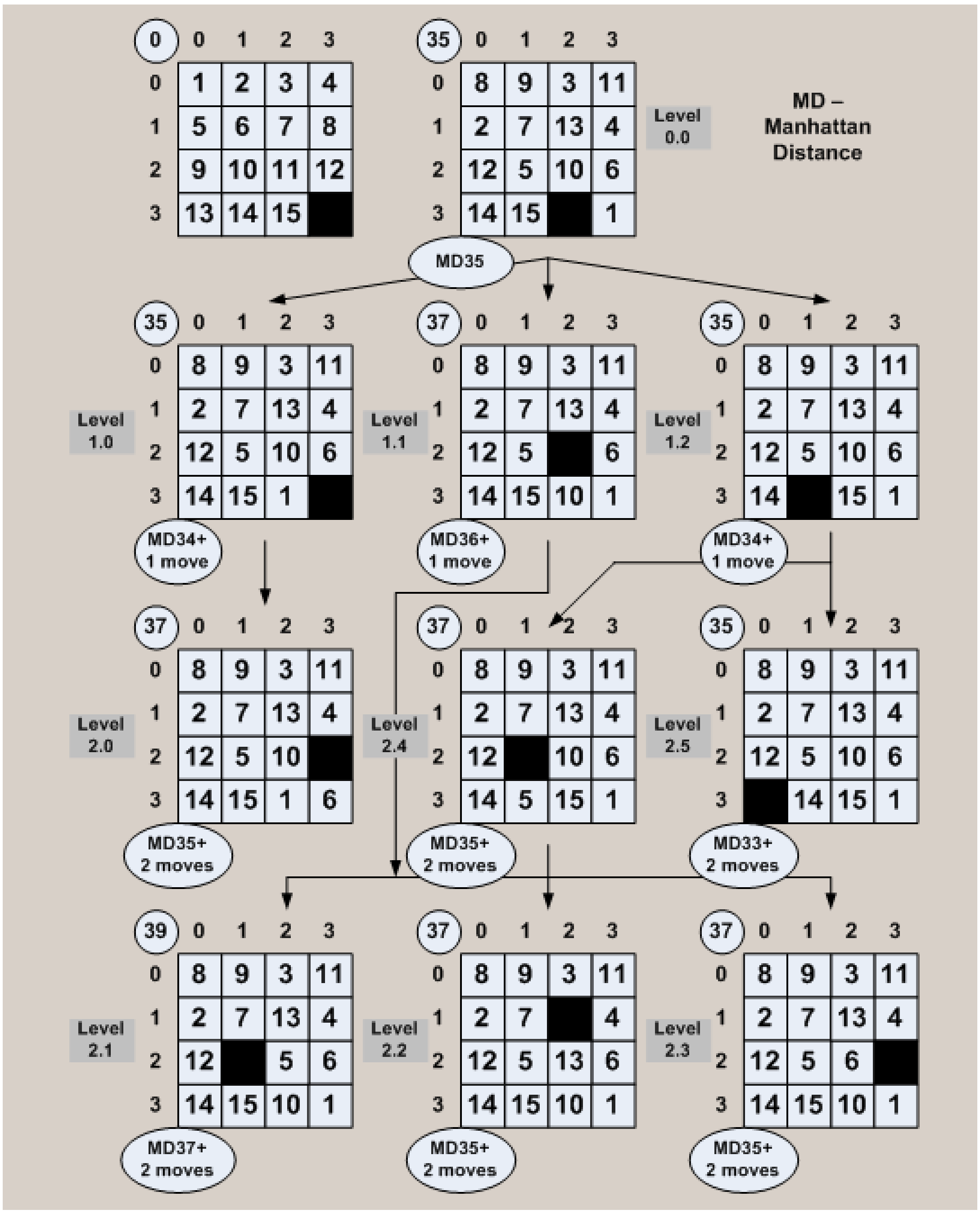
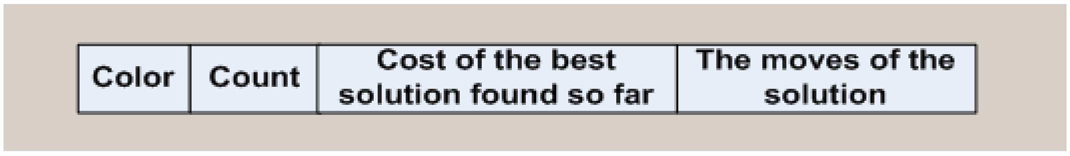


Fig.2. The computational model for solving the 15-puzzle of Sam Loyd by branch-and-bound search – depth up to three levels.



Color	Count	Cost of the best solution found so far	The moves of the solution

Fig.3. The format of the termination token.

The parallel computational model is shown in Fig.4. It provides opportunities for distributed load balancing at run time during the parallel branch and bound search.

Furthermore, the ring-like passing of the termination token ensures that useless computation shall not be performed for the solutions that cannot lead to better than the current best solution.

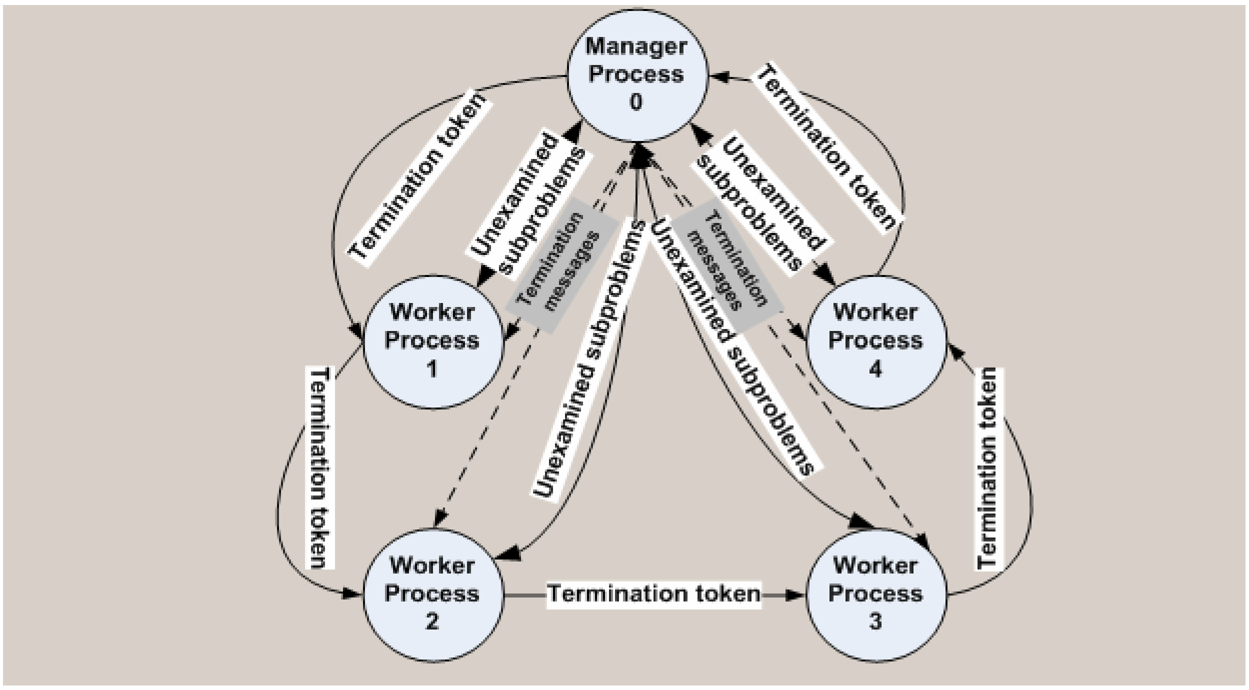


Fig. 4. The parallel computational model.

PERFORMANCE ESTIMATION AND PARALLELISM PROFILING

The experimental parallel computer platform comprised five workstations (Intel Pentium 4 1.5 GHz, RAM 256MB, Windows XP) interconnected by switch 100 Mbps. The message passing programming model was applied and MPI implementation of the parallel branch and bound search for solving Sam Loyd's puzzle was run for the cases of board sizes 4x4, 5x5 and 7x7. The speedup is calculated taking into account time for sequential processing on one workstation with centralized priority queue. The diagrams presenting the speedup and the efficiency as a function of the numbers of processors are shown in Fig. 5 and Fig.6, respectively.

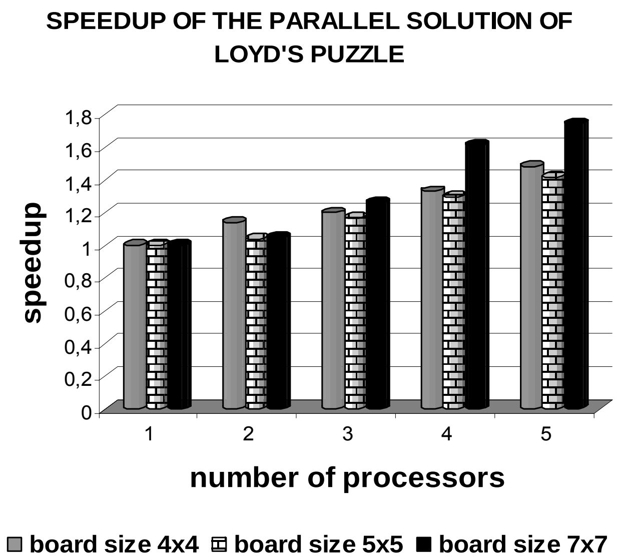


Fig.5. The speedup obtained.

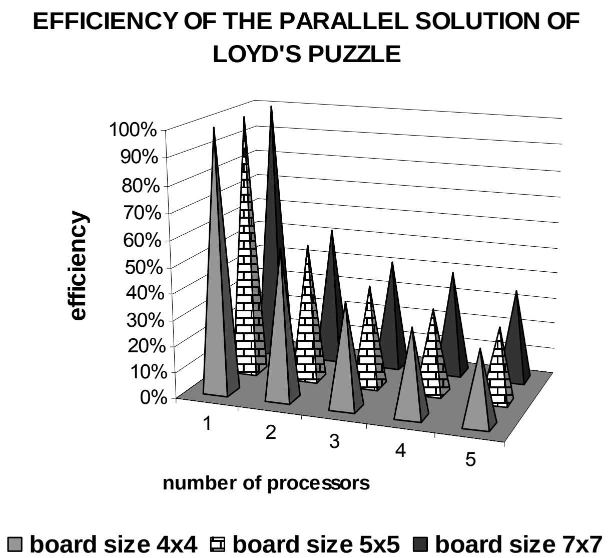
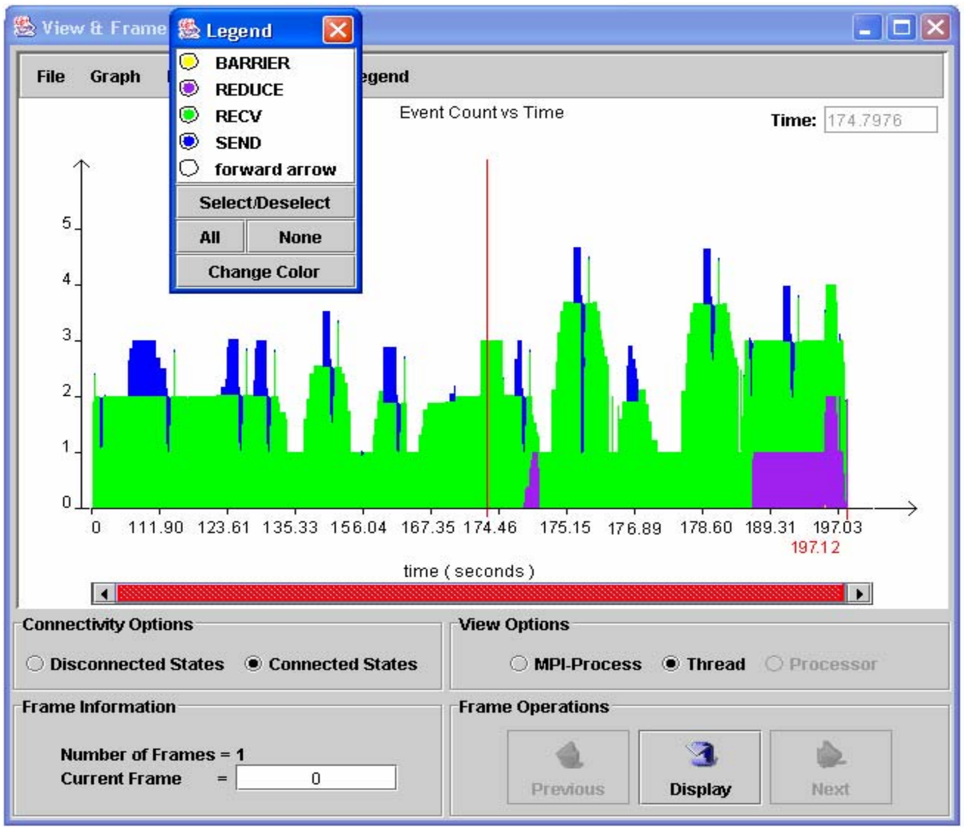


Fig. 6. The efficiency obtained.

The dynamics of the communication transactions of the MPI implementation of parallel branch-and-bound search for solving Loyd's puzzle is shown in Fig.7. Gantt's chart presenting the states of the processes is shown in Fig.8.

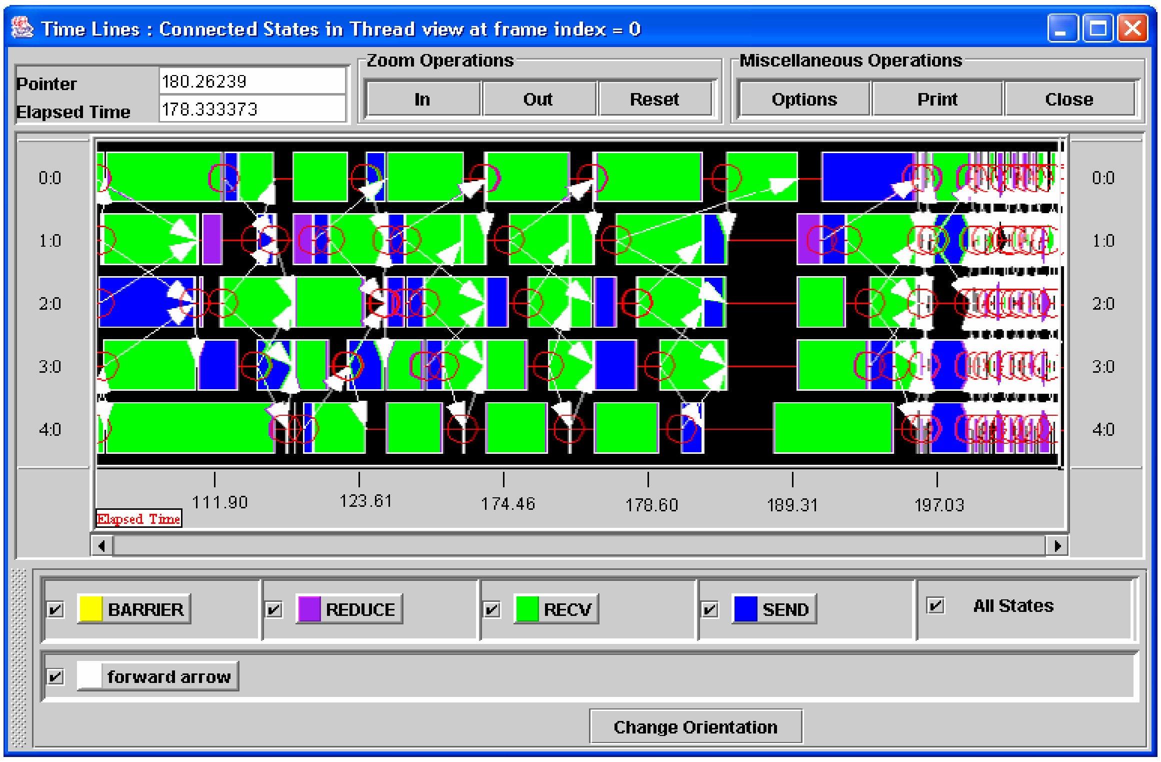


Connectivity Options: ( ) Disconnected States [✓ Connected States]

View Options: ( ) MPI-Process [✓ Thread] ( ) Processor

Number of Frames = 1
Current Frame = 0

Fig.7. The communication transactions of processes in the MPI implementation.



Pointer 180.26239
Elapsed Time 178.333373

✓ BARRIER ✓ REDUCE ✓ RECV ✓ SEND ✓ All States

✓ forward arrow

Fig. 8. Gantt's chart.

CONCLUSIONS AND FUTURE WORK

Performance analysis shows that the obtained speedup is the highest for the largest board size 7x7 because of the fact that increasing the board size means increasing the computational workload and consequently, better utilization of the computational resources. Nevertheless, the speedup is about 1.7 for 5 processors meaning the efficiency is about 35%. We see that the less the number of processors, the better the efficiency because of the better utilization of processors and the decreased communication overhead.

In order to improve the performance a more effective method for dynamic load balancing should be introduced. Whenever a process has an unexamined problem to send to another process it should have some system information about the parallel workload distribution within the parallel computer platform so that it can identify the process it is going to sent the newly generated workload. The length of the local priority queue can be used as a measure for the current workload of a process. A possible candidate for the parallel workload balancing is the gradient method and if it is applied, the additional incurred communication overhead should be considered.

The parallel computation of Sam Loyd's puzzle can be used as a benchmark for estimating the performance of parallel computer platforms for applications requiring combinatorial search.

The future work should involve investigating the performance of the parallel system in the cases of multithreading and hybrid programming models as well.

REFERENCES

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

[2] Wilkinson B., M. Allen, Parallel Programming: Techiques and Applications Using Networked Workstations and Parallel Computers, Upper Saddle River, New Jersey, Prentice Hall, 1999.

[3] www.clustercomp.org

[4] Quinn M., Parallel Programming in C with MPI and OpenMP, McGraw Hill Higher Education, International Edition, 2003.

[5] Grama A., A.Gupta, G.Kapyris, V.Kumar, Introduction to Parallel Computing, Second Edition, PEARSON, Addison Wesley, 2003.

[6] 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.

[7] Koip P., Parallel Algorithms for Combinatorial Search Problems, University of Massachusetts, 2005.

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.