Efficiency of Parallel Combinatorial Search on Multicomputer Platforms

P. Borovska

Key Words: *Multicomputers; parallel combinatorial search; parallel backtrack search; parallel branch and bound search; N Queens problem; Sam Loyd's puzzle; dynamic scheduling; parallel performance; parallel algorithms; parallel programming; MPI; OpenMP.*

Abstract: *In this paper the efficiency of parallel combinatorial search on multicomputer platform has been investigated on the basis of MPI+OpenMP parallel program implementations. The parallel algorithms under consideration are parallel versions of backtrack search and branch-and-bound search for the case studies of the N Queens problem and Sam Loyd's puzzle as benchmarks, respectively. Both parallel computational models are based on the manager/workers algorithmic paradigm implying dynamic strategy for load balancing in the parallel system. The performance parameters of the parallel system have been estimated and analyzed by means of parallelism profiling and benchmarking the hybrid parallel programs. The scalability of the parallel system has been investigated with respect to the parallel machine size and the parallel computational workload.*

1. The Problem Area

The goal of the algorithms for combinatorial search [1,2]] is to find one or more optimal solutions in a defined search space. An algorithm that solves an optimization problem must find a solution that is an extreme of an objective function.

The most popular applications of combinatorial search comprise VLSI design with minimal area, robot motion scheduling with minimal traveled distances, designing minimum distance cable interconnects of local area networks, theorem proving, games, etc.

Practically the search space is presented as a state space tree. The root of the tree denotes the original problem, while the other nodes represent subproblems. The type of the search tree and its depth depend on the problem specifics. The search tree may be either of the AND, OR or of the hybrid AND/OR type.

In order to find the optimal solution combinatorial algorithms have to make an extensive and time consuming search for possible solutions of the problem. The search space and the time for search increase exponentially with respect to the size of the problem. Parallel computing gives the opportunity to speed up the search process significantly. Algorithms for combinatorial search are parallelizable because of the considerable inherent parallelism of the application area [1,2]. The space of the possible solutions can be searched concurrently resulting in improving the search time.

Backtrack search is a method for solving combinatorial problems through searching in depth the space tree for finding out alternative solutions. The N Queens problem is a combinatorial problem solved by backtrack search. For a sufficiently large size of the chess board the N Queens problem is used for benchmarking the performance and efficiency of backtrack search on computer platforms.

The branch-and-bound search is a variation of backtrack search. It implies optimization of the search process by analyzing the optimality of the possible solutions and ignoring search subareas that are not likely to give optimal solutions. Sam Loyd's puzzle is a combinatorial problem solved by branch-and-bound search. For a sufficiently large size of the board Sam Loyd's puzzle is used for benchmarking the performance and efficiency of branch-and-bound search on computer platforms.

The goal of this paper is to investigate the efficiency of parallel combinatorial search on parallel computers with hybrid types of parallelism. The inherent parallelism of the applications is dynamic and therefore, dynamic strategies for load balancing are needed to speedup the computation.

2. The Parallel Computational Models

For both methods of combinatorial search under investigation the manager/workers algorithmic paradigm is implied. The parallel computational model for the N Queens problem is shown in *figure 1*. The basic responsibility of the manager (Process 0) is to distribute the tasks dynamically among the workers at runtime. His activities are:

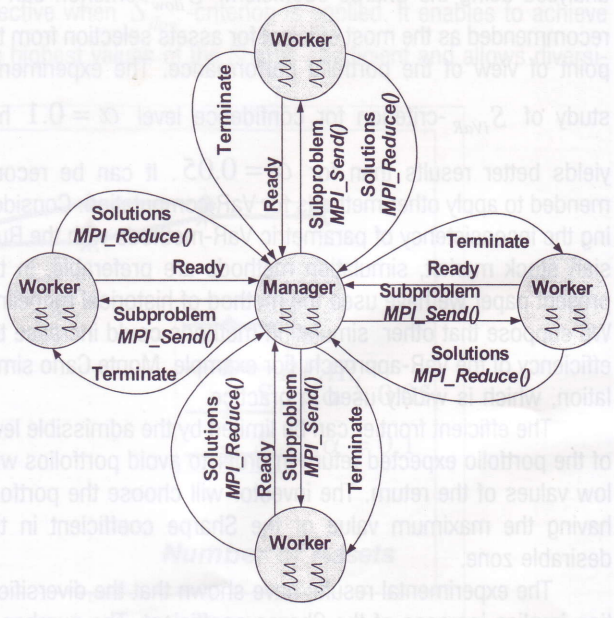


Figure 1. Parallel computational model for the N Queens problem

• Sends tasks to the workers for finding partial non-conflicting chess board configurations for a given number of queens to a specified depth of the space tree by the communication operation *MPI_Send()*;

• Receives messages for task completion and status „ready" by *MPI_Recv()*;

• Collects the solutions from the workers by means of *MPI_Reduce()*;

• Upon exhausting the workload sends termination messages to workers.

The activities of the workers are:

• Receive tasks from manager by means of the communication operation *MPI_Recv()*;

• Compute the task (for given chess board configuration and number of queens) by backtrack search;

• After completing the task send message to manager with tag „done" and indicate readiness for a new task by *MPI_Send()*;

• Terminate upon receiving termination messages from manager.

The parallel computational model for Sam Loyd's puzzle is shown in *figure 2*. It provides load balancing at run time. 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.

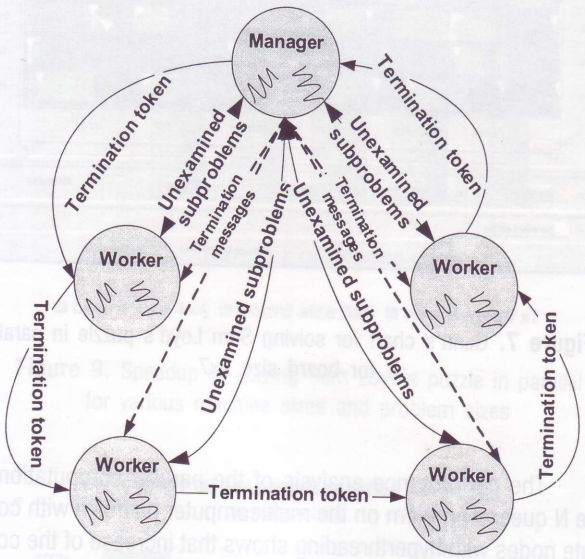


Figure 2. Parallel computational model for Sam Loyd's puzzle

The manager process is responsible for the following activities: initializing the primary configuration of tiles on the board, generates the original problem with the corresponding priority queue, dividing the original problem into two subproblems, distributing the unexamined problems to worker processes, sending termination token according to the requirements of the modified Dijkstra's distributed termination detection algorithm to worker processes in ring-like order, performing checks to identify the termination of the parallel algorithm, if it gets a white token and the message count is 0, sending a termination message to the worker processes. Each process maintains its own priority queue of unexamined subproblems. Initially the worker processes have empty priority queues expecting messages from other processes with unexamined subproblems. They receive messages, containing the termination token. 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.

3. The Experimental Framework

The experimental multicomputer platform comprised five workstations (Intel Pentium 4 3.1 GHz HT, RAM 514MB, Windows XP) interconnected by Fast Ethernet switch (100 Mbps, full duplex mode). The architecture of the target multicomputer platform is shown in *figure 3*.

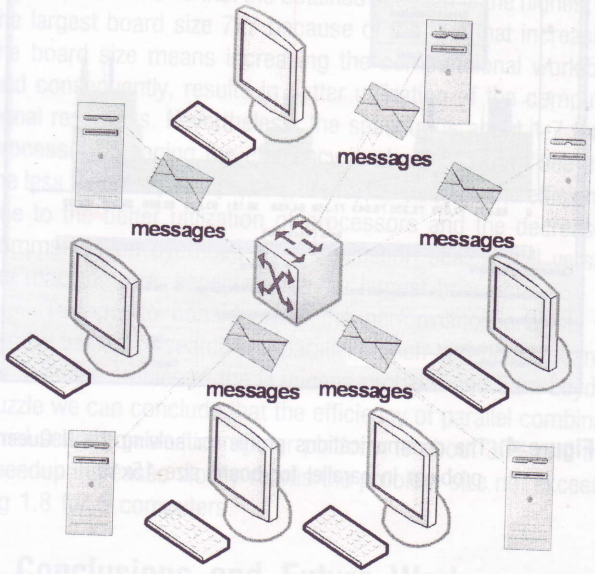


Figure 3. The architecture of the target multicomputer platform

Parallel program implementations based on hybrid MPI+OpenMP programming model have been run on the multicomputer platform. The processes are multithreaded utilizing the hyperthreading technology implemented in the computer nodes of the multicomputer. The parallel implementations for the N queens problem comprised different parallel workloads for various sizes of the chess board – 10x10, 13x13, and 15x15. For the specified sizes of the chess board the corresponding numbers of the non-conflicting configurations are *724, 73 712* and *2 279 184*, respectively. The parallel implementations for Sam Loyd's puzzle comprised different parallel workloads for various sizes of the board – 4x4, 5x5, and 7x7.

4. Performance Estimation and Parallelism Profiling

Parallelism profiling and performance parameters evaluation have been made for the 30 experiments in order to estimate the efficiency of the parallel combinatorial search as well as the scalability of both the machine size and the parallel applications. The communications profile and Gantt's chart for solving the N Queens problem in parallel for board size 15x15 are shown in *figure 4* and *figure 5*, respectively. The communications profile and Gantt's chart for solving Sam Loyd's puzzle in parallel for board size 7x7 are shown in *figure 6* and *figure 7*, respectively.

The values of the speedup of solving the N Queens problem and Sam Loyd's puzzle in parallel on multicomputer platform for the various multicomputer sizes and problem sizes are shown in *figure 8* and *figure 9*, respectively, while the values of the estimated efficiency - in *figure 10* and *figure 11*.

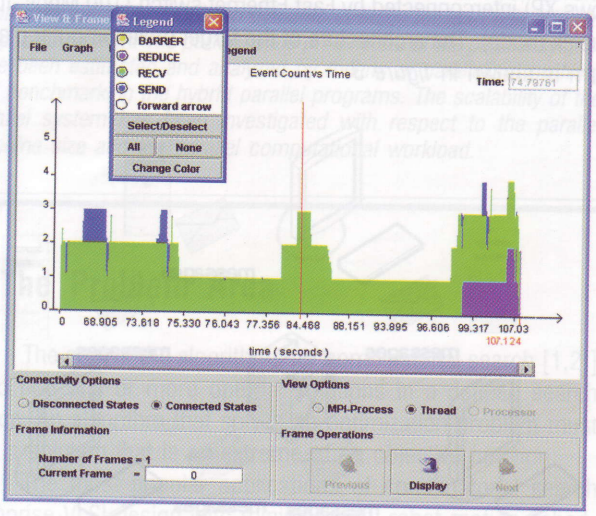


Figure 4. The communications profile for solving the N Queens problem in parallel for board size 15x15

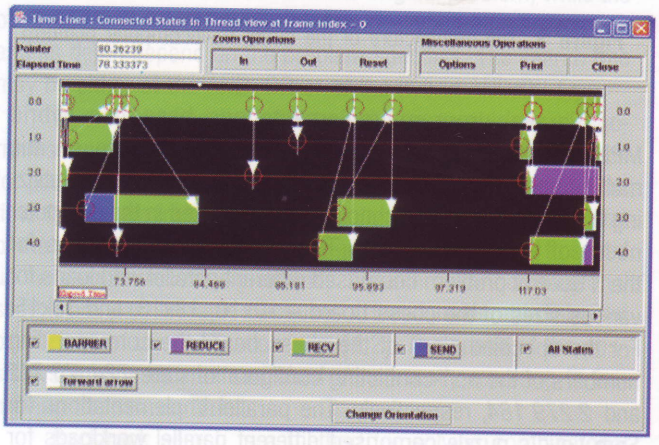


Figure 5. Gantt's chart for solving the N Queens problem in parallel for board size 15x15

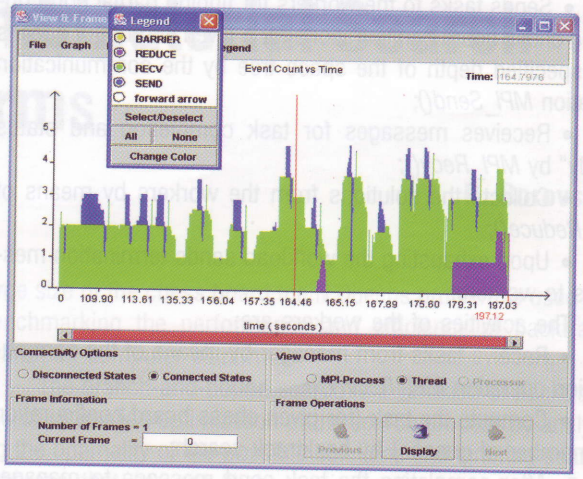


Figure 6. The communications profile for solving Sam Loyd's puzzle in parallel for board size 7x7

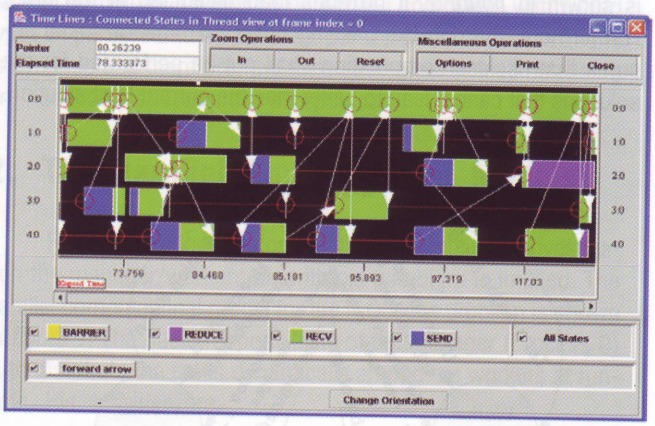


Figure 7. Gantt's chart for solving Sam Loyd's puzzle in parallel for board size 7x7

The performance analysis of the parallel computation of the N queens problem on the multicomputer platform with compute nodes with hyperthreading shows that increase of the computational workload (the chess board size and the number of queens, respectively) results in slowing down the speedup and reducing the efficiency of the parallel system due to the increased system overhead for dynamic load balancing demanding intensive communication. This effect is observed to be the greatest for the largest size of the parallel machine. We can conclude that the application scales sufficiently well in respect to the scaling of the multicomputer size in spite of the fact that the speedup increases slowly versus the increase of the parallel machine size.

The efficiency of the parallel system deteriorates when the machine size and problem size increase due to increased communication activities for load balancing. In all the experiments it is observed that increasing the size of the physical machine

results in degraded efficiency of the parallel system with about 20%. Varying the problem size and the machine size the value of the efficiency varies with about 6÷14%, the greatest degradation being observed for the maximum size of the problem and

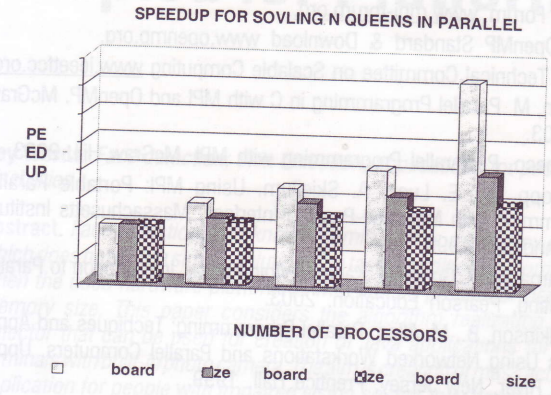


Figure 8. Speedup of solving N Queens problem in parallel for various machine sizes and problem sizes

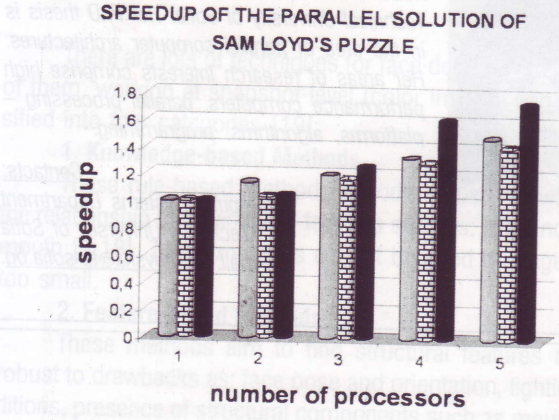


Figure 9. Speedup of solving Sam Loyd's puzzle in parallel for various machine sizes and problem sizes

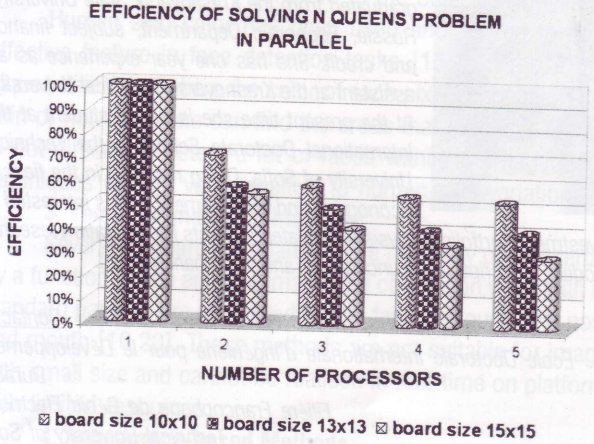


Figure 10. Efficiency of solving N Queens problem in parallel for various machine sizes and problem sizes

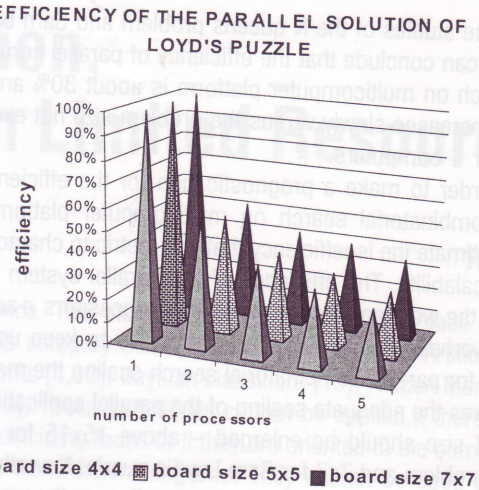


Figure 11. Efficiency of solving Sam Loyd's puzzle in parallel for various machine sizes and problem sizes

the machine. Nevertheless, the efficiency of the parallel system for 5 computers and board size 15x15 is 29.8%.

Performance analysis of the parallel computation of Sam Loyd's puzzle 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, results in 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 is, the better is the efficiency due to the better utilization of processors and the decreased communication overhead. The application scales well versus the machine size, especially for the largest board size.

Taking into consideration the performance analysis of parallel backtrack search and parallel branch-and-bound search for the case studies of the N queens problem and Sam Loyd's puzzle we can conclude that the efficiency of parallel combinatorial search on multicomputer platform is about 30% and the speedup increases slowly versus the problem size not exceeding 1.8 for 5 computers.

5. Conclusions and Future Work

In this paper the efficiency of parallel combinatorial search on multicomputer platform has been investigated on the basis of MPI+OpenMP parallel program implementations. The parallel algorithms under consideration are parallel versions of backtrack search and branch-and-bound search for the case studies of the N Queens problem and Sam Loyd's puzzle as benchmarks, respectively. Both parallel computational models are based on the manager/workers algorithmic paradigm implying dynamic strategy for load balancing in the parallel system. The performance parameters of the parallel system have been estimated and analyzed by means of parallelism profiling and benchmarking the hybrid parallel programs. The scalability of the parallel system has been investigated in respect with the parallel machine size and the parallel computational workload.

Taking into consideration the performance analysis of parallel backtrack search and parallel branch-and-bound search

for the case studies of the N queens problem and Sam Loyd's puzzle we can conclude that the efficiency of parallel combinatorial search on multicomputer platform is about 30% and the speedup increases slowly versus the problem size not exceeding 1.8 for 5 computers.

In order to make a prognostication for the efficiency of parallel combinatorial search on multicomputer platform we have to estimate the isoefficiency that is a metric to characterize system scalability. The efficiency of the parallel system E depends on the workload W, the number of processors n and the system overhead T_o i.e. $E=f(W,n,T_o)$. In order to keep up 30% efficiency for parallel combinatorial search scaling the machine size requires the adequate scaling of the parallel application i.e. the board size should be enlarged – above 15x15 for the N Queens' problem and 7x7 for Sam Loyd's puzzle. Nevertheless, we have to take into consideration that scaling up the workload will result in increasing the system overhead and eventually the efficiency might drop below 30%.

The future work should encompass investigation of the efficiency of parallel combinatorial search on computer cluster and the utilization of more efficient mechanisms for dynamic load balancing.

References

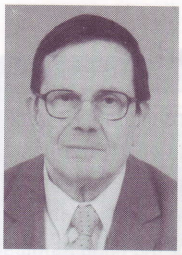
1. Raghavan, P. Parallel Combinatorial Search – Introduction, California, 2005.
2. Koip, P. Parallel Algorithms for Combinatorial Search Problems. University of Massachusetts, 2005.
3. Hwang, K., Z. Xu, Scalable Parallel Computing. McGraw Hill, 1998.
4. Borovska, P., H. Salem. Methodology for Balanced Design and Analysis of Parallel Algorithms on Multicomputer Platform. Proceedings of the Second International Conference on Challenges in Higher Education and Research in the 21st Century, Sozopol, Bulgaria, June 2004, Proceedings, Heron Press, Sofia, 241-244, ISBN 954-580-158-1.
5. MPI Forum www.mpi-forum.org.
6.The OpenMP Standard & Download www.openmp.org.
7. IEEE Technical Committee on Scalable Computing www.ieeetfcc.org.
8. Quinn, M. Parallel Programming in C with MPI and OpenMP. McGraw Hill, 2003.
9. Pacheco, P. Parallel Programming with MPI. McGraw Hill, 2003.
10. Gropp, W., E. Lusk, A. Skjellum. Using MPI: Portable Parallel Programming with Message-Passing Interface. Massachusetts Institute of Technology, 1999.
11. Grama, A., A. Gupta, G. Karypis, Vipin Kumar. Introduction to Parallel Computing, Pearson Education. 2003.
12. Wilkinson, B., M. Allen. Parallel Programming: Techiques and Applications Using Networked Workstations and Parallel Computers. Upper Saddle River, New Jersey, Prentice Hall, 1999.

Manuscript received on 16.03.2007

Assoc. Prof. ***Plamenka Borovska*** *is currently Head of Computer Systems Department of Technical University of Sofia. Her PhD thesis is in the area of parallel computer architectures. Her areas of research interests comprise high performance computers, parallel processing – platforms, algorithms, programming.*

Contacts:
Computer Systems Department
Technical University of Sofia
e-mail: pborovska@tu-sofia.bg.

continuation from 31



Geo I. Gatev *received his Electrical Engineering degree from the Czech Technical University, Prague (1960), and his PhD degree in Technical Cybernetics from the Moscow Power Institute, Moscow (1968). He was Senior Research Fellow and vice-director in the Central Research Institute of Complex Automation, Sofia (1971-1986). Since 1988 he has been Professor at the Technical University of Sofia. His research and teaching interests are in the fields of control systems, decision-making, complex systems. He is the author of six books and over 200 papers. He is Senior Member of IEEE.*

Contacts:
Department of Systems and Control
Technical University of Sofia
Phone: + 359 2 965 25 96
e-mail: gatev@tu-sofia.bg

Anna Malakhova *was born in 1981. She has graduated from the Krasnoyarsk State University, Russia, Economic Department, subject finance and credit. She has one year experience as an assistant at the Krasnoyarsk Technical University. At the present time she is a PhD student at the International Doctorate School in the Technical University of Sofia. Doing research in the field of Economics and Management, she is interested in Investment Portfolio Analysis and related subjects like operation research models, description of uncertainty and risk management.*

Contacts:
Ecole Doctorale Internationale d'Ingénierie pour le Développement Durable
Filiére Francophone de Génie Electrique
Technical University of Sofia
tel: +359-88-628-92-35
e-mail: anna_malahova@yahoo.fr