Parallel Simulated Annealing for Solving the Room Assignment Problem on Shared and Distributed Memory Platforms

Milena Lazarova

Abstract: *The paper is aimed at investigating the efficiency and quality of solutions of parallel simulated annealing on shared memory and distributed memory computer platforms for solving the room assignment problem. The parallel computational model for shared memory system utilizes concurrent generation and evaluation of moves by OpenMP based multithreading. Asynchronous and synchronous moves generation based on combination of functional and data parallelism are utilized for parallel computations using message passing on a distributed memory platform. Parallelism profiling and scalability analyses both in respect of the size of the parallel platform and the problem workload are presented.*

Key words: *Parallel Simulated Annealing, Multithreading, Message Passing, Synchronous and Asynchronous Moves, Speedup, Solution Quality, Profiling.*

INTRODUCTION

Simulated annealing (SA) is a general-purpose optimization technique capable of finding an optimal or near-optimal solution in various applications using an analogy with the annealing in solids [6]. It is a Monte Carlo approach that implements an appropriate Metropolis algorithm [10] to simulate the behavior of large and complex systems by annealing process that attempts to achieve thermal equilibrium. SA can be applied for solving combinatorial optimization problems such as the traveling salesman problem, cell placement of circuit components in VLSI, prediction of protein structure, etc. [2, 4].

SA comprises lots of iterations (moves) that require large amount of computational resources and is time consuming procedure. Since the algorithm is very computationally expensive, lots of efforts have been put towards its parallelization. The reason for using parallelism in solving optimization problems by SA is twofold: it is aimed at speeding up the computations by reducing the time required for finding a solution near to the optimum and it also attempts to provide better solution quality for certain time by higher diversification and better exploration of the search space. Parallel SA can exploit functional parallelism: the use of multiple processors to evaluate different phases of a single move, implementing several Markov chain computations in parallel [1], carrying out parallel computations for several states of the same Markov chain [15]. The data parallelism in SA is employed by the use of different processors or group of processors that propose and evaluate moves by multiple independent runs [14].

Multiprocessor systems comprising several processors that share common address space provide a platform for parallel computation based on multithreading. OpenMP is an API for multiplatform shared-memory parallel programming based on compiler directives [3]. MPI (Message Passing Interface) is a standardized library developed for parallel programming on distributed memory systems [11]. Since most of the recent high-performance computer systems are clusters of shared memory nodes, parallel programming must combine the distributed memory parallelization on the node inter-connect with the shared memory parallelization inside each node [12].

The paper is aimed at investigating the efficiency and quality of solutions of parallel simulated annealing on shared memory and distributed memory computer platforms for the case study of the room assignment problem. The parallel computational model for multiprocessor system utilizes concurrent generation and evaluation of moves by multithreading based on OpenMP parallel loops. The strategies for asynchronous and synchronous move generation are utilized for parallel computations using message passing on a multicomputer platform. Parallelism profiling and scalability analyses both in respect of the size of the parallel platform and the problem workload are presented.

SIMULATED ANNEALING FOR SOLVING ROOM ASSIGNMENT PROBLEM

SA is used to solve complex optimization problems attempting to minimize given cost function. SA is based on an iterative procedure of generating new solutions of the problem (moves) based on a random modification of the current solution. The moves that reduce the cost function are accepted with high probability whereas those that increase the cost are accepted with low probability. A temperature parameter controls the probability of acceptance of the uphill moves and it slowly decreases as the SA progresses according to certain cooling schedule. SA follows sequential algorithm with strong dependence between the consequent iterations. The general algorithm of the simulated annealing can be presented as follows:

```
Generate initial solution at random and initialize temperature T
while (T > 0) do
        while (thermal equilibrium not reached) do
                Generate a random neighbor state
                Evaluate the change in the energy level
                Update current state with new state if energy decreases
                Update current state with new state with calculated probability if energy
                increases
        end
        Decrease temperature T according to the annealing schedule
end
```

Instead of repeating the annealing process until the temperature parameter approaches zero other various stopping criteria can be applied providing deterministic or non-deterministic running time of the algorithm such as fixed number of iterations to be executed or the objective function value to not decrease for large number of consecutive iterations.

The room assignment problem is a combinatorial optimization problem that solves the assignment of given number of persons in twice less number of rooms in a way that minimizes a cost function defined as a sum of conflicts between the roommates. SA can be applied for solving the room assignment problem by representing the solution as a random assignment of each person to a given room and calculating the cost function as simple sum of the dislike coefficients of all persons in the same room [8].

PARALLEL SIMULATED ANNEALING USING MULTHITHREADING

Parallelization of the SA on a shared memory platform utilizes global access to the common memory by all threads. In the case of the room assignment problem concurrent working threads generate moves independently without any special synchronization apart from control of the access to the shared data structures. OpenMP functions omp_test_lock and omp_unset_lock are used for synchronization of the access to each room by the multiple threads. Thus before generating a move that involves change of the assignment of certain rooms the thread locks the access to these rooms and makes sure no other thread will use these rooms until they are unlocked. The moves are generated in parallel using the compiler directive omp parallel for. In order to prevent race conditions between the threads in the update of the shared variables after each move generation, omp atomic directives are added:

```
#pragma omp parallel for private(nextMove, generate_start, generate_end)
  for (j = 0; j < iterations_steps; j++) {
          nextMove = generate_move(conf, NULL);
          #pragma omp atomic
          generated_moves++;
          if (nextMove.accepted) {
                  #pragma omp atomic
                  accepted_moves++; }
  }
```

PARALLEL SIMULATED ANNEALING USING MESSAGE PASSING

In distributed memory platforms SA can be parallelized either utilizing functional or data parallelism. Parallel computations of SA can also be distinguished as single trial or multiple trial parallelism but these methods are highly problem dependent and the speedup achieved depends on the targeted problem [13]. Each parallelization strategy implies certain tradeoffs among cost function accuracy, state generation, parallelism and communication overhead. In the case of the room assignment problem global information is required for evaluation of the cost function at each new move. The functional parallelism involves parallel generation and evaluation of moves according to the SA by several concurrent processes. In this case data communication and synchronization between the processes should provide the use of global configuration of the system corresponding to certain energy level. The parallel computational model for solving the room assignment by SA is based on a combination of two parallel programming paradigms: phase-parallel and master-slave. The phase-parallel paradigm consists of computation phase of local moves generation followed by communication phase for data exchange between the processes. Each process implements the following sequence of steps:

```
Computation phase:
    Generate local moves
Communication phase:
    Send the local moves to the master process
    Receive moves/configuration from the master process
Update the local configuration according to the received moves/configuration
Decrease the temperature
```

Two different communication approaches can be applied at the communication stage that lead to either synchronous or asynchronous update of the local configuration [7]. Synchronous moves strategy implies simultaneous generation and evaluation of several moves by the slave processes and global update of the state while the asynchronous moves allow calculation of the energy variations starting from partially outdated information [5]. The parallel computational model for solving the room assignment problem by SA on multicomputer using synchronous moves platform is given in Fig.1. The asynchronous moves allow the slave processes to continue with the moves generation and evaluation while the master process generates and broadcasts a new system configuration (Fig.2).

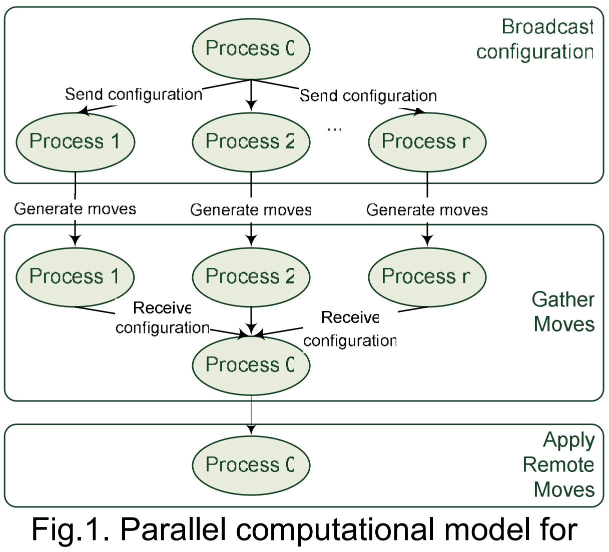

Fig.1. Parallel computational model for synchronous iterations

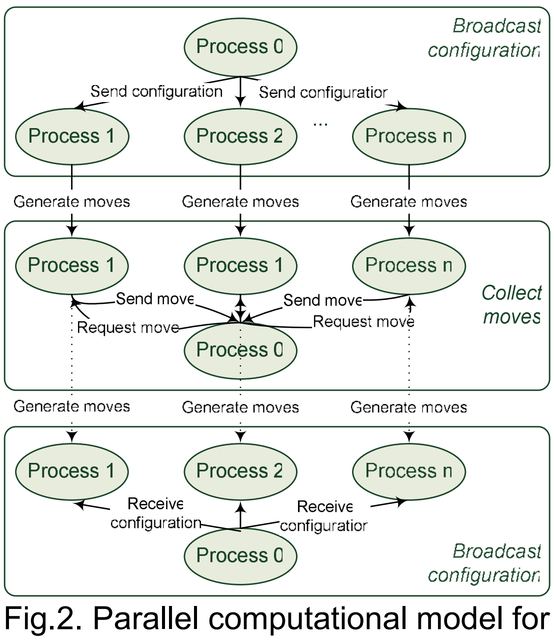

Fig.2. Parallel computational model for asynchronous iterations

In order to guarantee that the local moves generated by the concurrent working processes are independent and will not lead to invalid change of the system configuration each slave process is allowed to generate moves based on alterations of the assignment only for certain rooms. In this way if a local process generates a move that changes the assignment of the persons in given room the other processes will not be allowed to accept moves that involve assignment to a room owned by another process. Thus locally accepted moves will lead to similar energy change as the moves of the remote processes.

Since SA is a Monte Carlo based algorithm that relies on randomly generated moves and random evaluation of the probability of acceptance of the uphill moves, parallel random generator have to be utilized with the following features: large period of non-repeating generated numbers, normal distribution of the generated numbers, non-dependence between the consecutive numbers in the generated sequence. Due to its good characteristics Mersenne Twister random number generator (MTRNG) is utilized [9]. Parallel random number generator based on MTRNG is implemented by parameterization of the sequential generator and different seeds. The master process initializes the generator with an initial seed and then generates random numbers that are sent to the slave processes and used as seeds of their MTRNG.

PARALLELISM PROFILING AND PERFORMANCE ANALYSIS

The suggested parallel multithreaded and flat computational models for solving the room assignment problem using SA utilizing synchronous and asynchronous moves are implemented in C++ language and compiled using MS Visual Studio 2005. The shared memory parallelism is based on OpenMP API. MPICH-2 implementation of the MPI standard is utilized. Parallelism profiling is made utilizing Jumpshot v4.0.

The experimental evaluation of the performance parameters of the models is based on three configurations of the input data for 5000, 10000 and 20000 persons. The experiments were carried out on a multicomputer platform comprising 10 workstations (Intel Pentium 4, 3.2GHz, 1G RAM, Hyper-Threading) connected via Fast Ethernet switch (100 Mbps).

The results for the speedup and the efficiency of the multithreaded programming model executed with two threads and different number of persons are given in Fig.3. As can be seen the model scales well depending on the problem size with speedup of 1.7 and efficiency 85% for 20 000 persons. The difference in the energy obtained by the sequential and the parallel multithreaded model as a function of the number of moves for different problem size is presented on Fig.4. Obviously the parallel model outperforms the sequential in the change of the system energy due to the larger number of the generated moves.

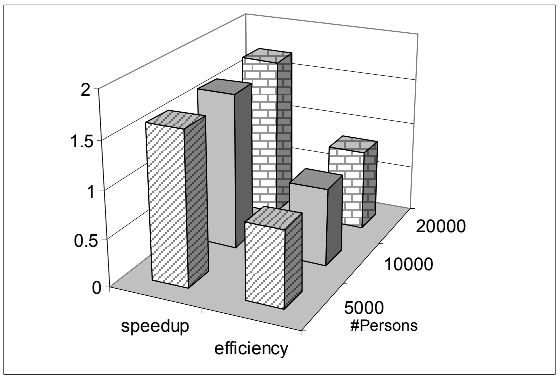

Fig.3. Speedup and efficiency of the multithreaded model

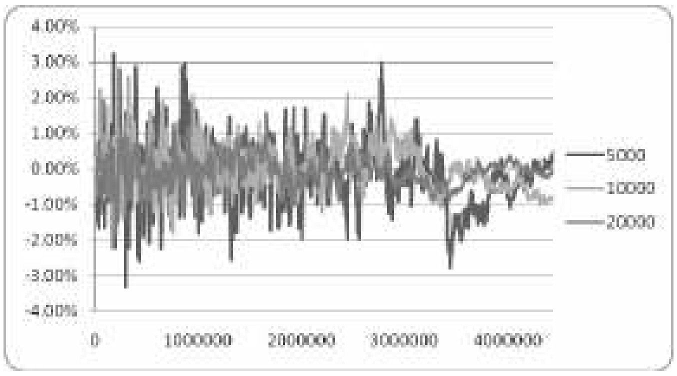

Fig.4. Energy difference between sequential and multithreaded model

The Gantt charts for the flat computational model with synchronous and asynchronous moves are presented in Fig.4 and Fig.5 respectively. On Fig.6 and Fig.7 the results for the speedup of the synchronous and asynchronous flat models are given. Comparison of the solution quality of the sequential and parallel computations is given on Fig.8 and Fig.9 presents a comparison of the energy change during the generation of the moves for a problem size of 5000 persons.

The results obtained show that the flat programming model for solving the room assignment problem by SA utilizing both asynchronous and asynchronous moves outperforms the serial algorithm in terms of the speedup achieved. The speedup of the parallel model with synchronous moves is slightly larger compared to the model with asynchronous moves: the speedup achieved for assignment of 20000 persons using 8 processors is 4.6 for the synchronous and 3.8 for the asynchronous algorithm. These results can be explained with the additional communication overhead and the more complex transaction exchange between the processes in the asynchronous algorithm based on the master-slave paradigm as can be seen in the sample Gantt's chart (Fig.5).

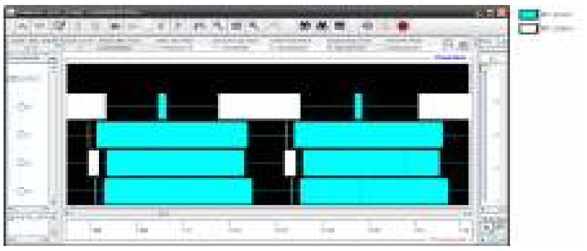
Fig.5. Gantt's chart of the parallel model with synchronous moves

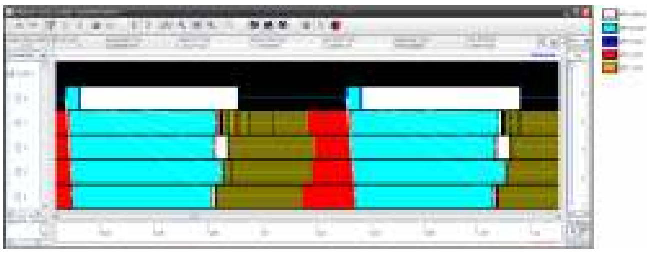
Fig.6. Gantt's chart of the parallel model with asynchronous moves

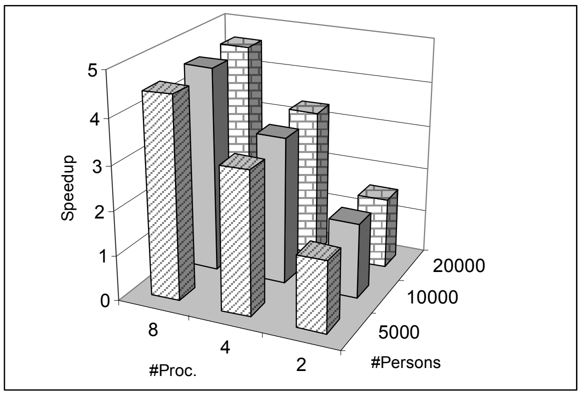

Fig.7. Speedup of the flat programming model with synchronous moves

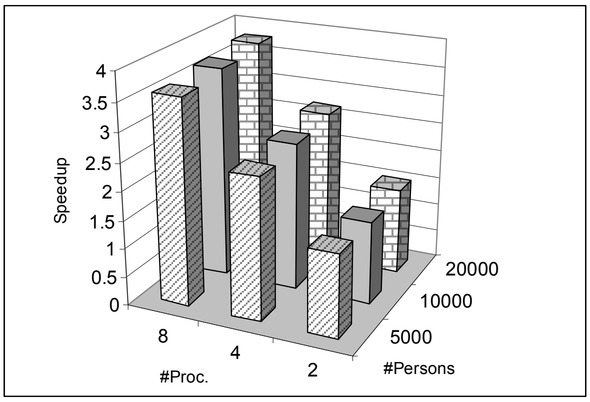

Fig.8. Speedup of the flat programming model with asynchronous moves

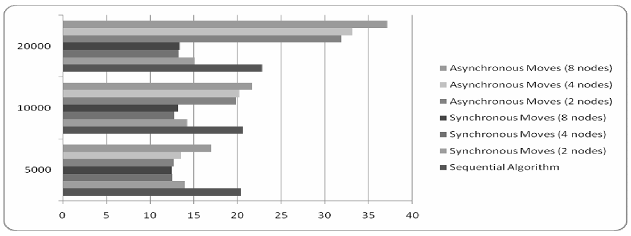

Fig.9. Comparison of the quality solution of the sequential algorithm and the parallel model with synchronous and asynchronous moves

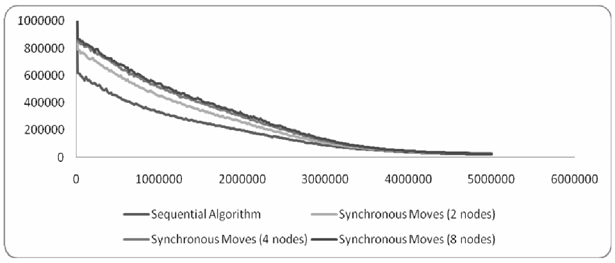

Fig.10. Comparison of the energy change during the generation of the moves for a problem size of 5000 persons

The results for the quality of the solution obtained for the flat parallel model show that both strategies with synchronous and asynchronous system configuration update provide better solution for certain number of moves compared to the sequential computations. The advantages of parallel computations are more obvious for smaller problem sizes. The larger the problem size is the more moves are generated by the asynchronous algorithm that are not synchronized with the global configuration and the moves generated by the other processes. Thus the solution quality decreases.

CONCLUSIONS

Parallelization of the solving of the room assignment problem by parallel SA on shared memory system is suggested using concurrent generation and evaluation of moves by multithreaded model based on OpenMP parallel loops. The multithreaded model provides good synchronization in the parallel generation of moves by means of access to shared variables while the flat programming model uses exchange of the generated moves between the processes. The strategies for asynchronous and synchronous move generation are utilized for parallel computations using message passing on a multicomputer platform. The strategy of asynchronous moves requires additional communication overhead and leads to worse solution quality.

REFERENCES

[1] Aarts, E., F. de Bont, J. Habers, P. van Laarhoven, A parallel statistical cooling algorithm. Proc. of the 3rd Annual Symposium on Theoretical Aspects of Computer Science, LNCS Series, Vol.210, 1986, pp.87÷97.

[2] Aarts, E., J. Korst, P. van Laarhoven. Simulated annealing. In E. Aarts, J. Lenstra (edts.), Local Search in Combinatorial Optimization, Wiley, 1997, pp. 91–120.

[3] Chapman, B., G. Jost, R. van der Pas, Using OpenMP: Portable Shared Memory Parallel Programming, MIT Press, 2007.

[4] Dréo, J., A. Pétrowski, P. Siarry, E. Taillard. Metaheuristics for Hard Optimization, Springer-Verlag, Berlin, 2006.

[5] Durand, M., S. White, Trading Accuracy for Speed in Parallel Simulated Annealing with Simultaneous Moves, Parallel Computing, Vol.26, №1, January 2000, pp.135÷150.

[6] Kirkpatrick S., C. Gelatt, M. Vecchi, Optimization by Simulated Annealing, Science, Vol.220, №4598, 1983, pp.671÷680.

[7] Lee S., K. Lee, Synchronous and Asynchronous Parallel Simulated Annealing with Multiple Markov Chains, IEEE Trans. on Parallel Distributed Systems, Vol.7, №10, 1996, pp. 993÷1008.

[8] Martinez-Alfaro, H., J. Minero, G. Alanis, N. Leal, I. Avila, Using Simulated Annealing to Solve the Classroom Assignment Problem, Proc. of Int. Conf. on Intelligent Systems Technologies (ISAI/IFIS), 1996, pp.370÷377.

[9] Matsumoto, M., T. Nishimura, Mersenne Twister: A 623-dimensionally Equidistributed Uniform Pseudorandom Number Generator, ACM Trans. on Modeling and Computer Simulation, Vol.8, №1, 1998, pp.3÷30.

[10] Metropolis N., A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, Equation of State Calculations by Fast Computing Machines, Journal of Chemical Physics, Vol.21, 1953, pp.1087÷1092.

[11] Quinn M., Parallel Programming in C with MPI and OpenMP, McGraw Hill, 2003.

[12] Rabenseifner R., G. Hager, G. Jost, R. Keller, The Hybrid MPI and OpenMP Parallel Programming, LNCS Series: Recent Advances in Parallel Virtual Machine and Message Passing Interface, Springler-Verlag, Vol.4192, 2006.

[13] Ram, D., T. Sreenivas, K. Subramaniam, Parallel Simulated Annealing Algorithms, Journal of Parallel and Distributed Computing, Vol.37, 1996, pp.207÷212.

[14] Resende, M., P. Pardalos, S. Ekşioğlu, Parallel Metaheuristics for Combinatorial Optimization, in R. Correa et al. (edts,), Models for Parallel and Distributed Computation – Theory, Algorithmic Techniques and Applications, Kluwer Academic Publishers, 2002, pp.179÷206.

[15] Roussel-Ragot, P., P. Siarry, G. Dreyfus, A Problem Independent Parallel Implementation of Simulated Annealing: Models and Experiments. IEEE Trans. CADICS, Vol.9, 1990, pp.827÷835.

ABOUT THE AUTHOR

Assist. Prof. Milena Lazarova, PhD, Department of Computer Systems, Technical University of Sofia, Phone: +359 2 965 3285, E-mail: milaz@tu-sofia.bg