Theoretical and Practical Issues of Parallel Simulated Annealing

Agnieszka Debudaj-Grabysz[1] and Zbigniew J. Czech[1,2]

[1] Silesia University of Technology, Institute of Computer Science
Akademicka 16, 44-100 Gliwice, Poland
{agrabysz,zczech}@polsl.pl
[2] Silesia University, Sosnowiec, Poland

Abstract. Several parallel simulated annealing algorithms with different co-operation schemes are considered. The theoretical analysis of speedups of the algorithms is presented. The outcome of the theoretical analysis was verified by practical experiments whose aim was to investigate the influence of the co-operation of parallel simulated annealing processes on the quality of results. The experiments were performed assuming a constant cost of parallel computations, i.e., searching for solutions was conducted with a given number of processors for a specified period of time. For the experiments a suite of benchmarking tests for the vehicle routing problem with time windows was used.

Keywords: Simulated annealing, parallel processing, co-operation of processes, vehicle routing problem with time windows.

1 Introduction

The paper presents several co-operation schemes for parallel algorithms of simulated annealing (SA) which is a popular heuristic method of optimization. The algorithms with periodical and hybrid communication are compared with a no communication algorithm. The distinctive feature of the algorithm with hybrid communication is that it is intended to be executed on clusters of shared-memory nodes (SMP), combining the benefits of both shared and distributed memory systems. The theoretical analysis of speedups of the algorithms is presented. It considers the computational complexity of SA trials under the assumption that the algorithms stop after having performed a specified number of trials.

The outcome of the theoretical analysis was verified by practical experiments whose aim was to investigate the influence of the co-operation of parallel SA processes on the quality of results. The experiments were performed assuming a constant cost of parallel computations, i.e. searching for solutions was conducted with a given number of processors for a specified period of time. Such an approach guarantees linear speedup which might be desirable in practical applications. For the experiments a suite of benchmarking tests for the vehicle routing problem with time windows (VRPTW) was used.

R. Wyrzykowski et al. (Eds.): PPAM 2007, LNCS 4967, pp. 189–198, 2008.

Simulated annealing is a heuristic method of optimization employed in the cases when the solution space is too large to be exhaustively explored within a reasonable amount of time. The VRPTW is an example of such a problem. Other examples are: school bus routing, newspaper and mail distribution, delivery of goods to department stores etc. The optimization of routing lowers distribution costs, whereas parallelization makes possible to find better routes if the computation time is limited.

The SA bibliography focuses mainly on the sequential versions of the algorithm [2,15]. However parallel versions are also investigated, since the sequential method is considered to converge slowly as compared with other heuristics [16]. In [1,3,11,12,13] some recommendations for parallelization of SA are given. The VRPTW, formulated by Solomon [14] who also proposed a suite of benchmarking tests, has a rich bibliography [16]. The research whose results are presented in this paper is a continuation of our efforts described in [8,9,10,5,6], where parallel SA algorithms to solve the VRPTW are discussed. The VRPTW is a bicriterion optimization problem. Solving it consists in finding a solution with minimum number of routes (first optimization criterion) and then minimizing the total travel distance travelled by vehicles (second optimization criterion). This paper focuses on the first optimization goal, i.e. on minimizing the number of solution routes.

Section 2 presents the theoretical foundation of sequential and parallel SA algorithms. Section 3 describes a few variants of the co-operation of processes in parallel simulated annealing. Section 4 is devoted to the theoretical analysis of the algorithms. The results of the experiments are described in section 5. The last section contains conclusions.

2 Sequential and Parallel Simulated Annealing

Simulated annealing searches for the optimal state, i.e. the state which minimizes (or maximizes) the *cost function*. This is achieved by comparing the current solution with a random solution taken from a specific *neighborhood*. If a neighbor solution has lower cost than the current solution then it is accepted. Worse solutions can also be accepted with some probability, what prevents the algorithm from being prematurely stuck in local optima. The probability of accepting a worse solution decreases over the process of annealing, and it is governed by a control parameter called *temperature*. An outline of the SA algorithm is presented in Figure 1. A single execution of the innermost loop is called a *trial*. The final solution which is returned is the best one ever found. Simulated annealing can be modelled in terms of Markov chains. Namely, the algorithm is considered to generate a sequence of Markov chains, where each chain consists of trials executed in the same temperature.

Since in SA each new state is potentially a modification of the previous state, the process is often considered as inherently sequential. However, a few strategies to develop parallel SA were proposed. In our implementations the trials are executed in parallel by a number of processors. We assumed that the length of

```
01 S ← GetInitialSolution();
02 T ← InitialTemperature;
03 for i ← 1 to NumberOfTemperatureReduction do
04     for j ← 1 to ChainLength do
05         S' ← GetSolutionFromNeighborhood();
06         ΔC ← CostFunction(S') − CostFunction(S);
07         if (ΔC < 0 or AcceptWithProbabilityP(ΔC, T)) then
08             S ← S';    {the trial is accepted}
09         end if;
10     end for;
11     T ← λT;    {with λ < 1}
12 end for;
```

Fig. 1. Simulated annealing algorithm

Markov chains was fixed in such a way that the number of trials executed by all processors within each chain is the same as in the sequential algorithm.

3 Co-operation of Processes in Parallel Simulated Annealing

No communication algorithm (NC). The main assumptions for the no communication algorithm were formulated in [2], where the *division algorithm* was proposed. The method uses all available processors to run many sequential algorithms, where the original chain is split into sub-chains of *ChainLength* (see Figure 1) divided by the number of processors. At the end, the best solution found is selected as the final one. The use of a large number of processors can result in excessive shortening of the sub-chains length what in turn may negatively affect the quality of results.

Periodical communication algorithm (PC). The idea of periodically interacting searches was fully developed in [13]. As in the NC algorithm the length of the sub-chains is decreased. Additionally, processes communicate after performing a part of a chain called a *segment*, and the best solution is selected and mandated for all processes. In the PC algorithm the segment length is defined by the number of temperature drops. As suggested in [13] to prevent search paths from being trapped in local minima areas as a result of communication, the period of the information exchange needs to be carefully selected.

Hybrid communication algorithm (HC). The details of the method were described in [9]. The implementation is intended to run on clusters of SMP nodes, so the parallelization is accomplished using two levels. Intensively communicating operations are moved to the inner level where a shared memory environment is used. The remaining, comparatively rare, communication – the outer level – takes place in a distributed memory environment.

Outer-level parallelization. Each Markov chain of SA optimization is divided into sub-chains. Their length is equal to the length of the original chain divided by the number of sub-chains. The main idea is to assign a separate sub-chain to a cluster node to allow nodes to generate different sub-chains simultaneously. In this way the computational effort of generating a Markov chain is distributed among the available nodes. Having generated the first Markov chain, the generation of the next chains is performed with no communication among nodes. Each node takes the outcome of the last trial of the preceding sub-chain as the starting point for the subsequent sub-chain. At the end, the best solution found is produced as the final one. It is worth mentioning the sub-chains length is shortened by the number of nodes instead of the number of processors. That results in longer sub-chains as compared with the NC and PC algorithms.

Inner-level parallelization. Within a node a few threads communicate with each other while building a single sub-chain of the length determined at the outer level. The idea of parallelization consists in dividing the total number of trials in each sub-chain into rather small *sets of trials*. Each thread performs its part of the set independently of the other threads. Having completed a set, the master thread selects a solution among all solutions which have been accepted and the remaining ones are discarded. The selected solution is made common for all threads and it becomes the starting point for further computation.

4 Theoretical Analysis of Algorithms

The speedups of the algorithms are established through the analysis of the computational complexity of trials. It is assumed that the algorithms stop when a defined number of sub-chains are completed. The results of the analysis are presented in Table 1, where the parameters denote:

p — number of processors,
L — length of the Markov chain for the sequential algorithm (see Figure 1),
ω — period of communication (in the PC algorithm only),
λ — intensity of the Poisson's process of generation of trials, to be identified experimentally,
β — coefficient determining the duration of a broadcast type inter-node communication, to be identified experimentally,
M — number of Markov chains for the corresponding sequential algorithm,
d — size of the set of trials (in the inner level of the HC algorithm),
$\overline{t_1}$ — duration of the sequential part executed on the inner-level of parallelization, to be identified experimentally.

The details of the derivation of formulas showed in Table 1 can be found in [7]. Note that the formulas differ only in the denominators. By the definition the denominator is an inverse of the parallel efficiency η^{-1}. For arbitrarily and experimentally defined parameters the parallel efficiency as a function of the number of processors is presented in Figure 2. In terms of efficiency the NC algorithm seems to be the best. The HC algorithm is the worst although its

Table 1. Theoretical speedups

Algorithm	Speedup
PC	$\frac{1}{1+\sqrt{\frac{p(p-1)^2}{(2p-1)L\omega}+\frac{2\beta p\lambda}{L\omega}\log_2 p}}p$
NC	$\frac{1}{1+\sqrt{\frac{p(p-1)^2}{(2p-1)LM}+\frac{\beta p\lambda}{LM}\log_2}}p$
HC	$\frac{1}{1+\frac{d-1}{\sqrt{2d-1}}+\lambda\overline{t_1}+\frac{\beta p\lambda}{LM}\log_2\frac{p}{d}}p$

efficiency has a striking feature of being relatively independent from the number of processors. One could conclude that only the trivial parallelizations of the NC algorithm are recommended. Nevertheless, analysing the chart the following caveats should not be ignored:

- In a real life application a user can demand the linear speedup, or he/she can set a time limit for the execution of the algorithm. In other words, one might need to stop the algorithm after a specified period of execution time.
- The way of exploration of the search space is not taken into account.
- The influence of the co-operation of processes is not considered.

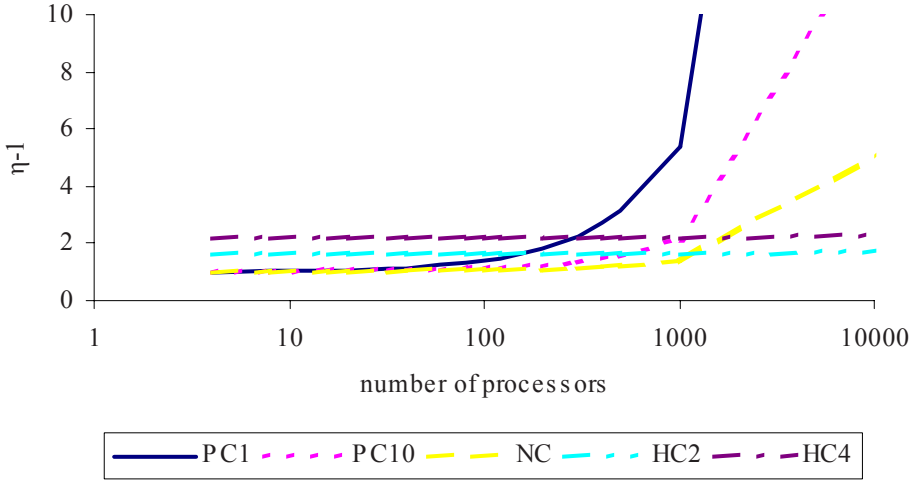

Fig. 2. Parallel efficiency vs. number of processors (PC1, PC10 – the PC algorithm with the period of communication 1 and 10; NC, HC2, HC4 – the NC and HC algorithms with 2 and 4 parallel threads)

The first note deserves explanation. As assumed at the beginning of this section, the chart concerns executions with no time limits (instead, the number of sub-chains to be generated is defined). In [7] the following theorem is proved:

Theorem 1. *The deterioration of the solution quality of the parallel algorithm with the time limit is a non-increasing function of the efficiency of the parallel algorithm without the time limit.*

The deterioration of the solution quality is measured by the difference between the expected solutions obtained by the parallel and sequential algorithms (the result of the sequential algorithm is taken as a reference). The theorem says that lower theoretical efficiency indicates a possibly worse quality of the solution achieved if the time limit is imposed. In the context of the investigated algorithms running with the time limit one may infer from the chart that it is necessary to take into account the number of available processors. More specifically, for a small number of processors one may expect that the HC and NC algorithms shall yield the worst and best results, respectively. On the other hand, for a large number of processors the HC algorithm is expected to show its superiority. We believe, however, that these theoretical conclusions should be verified in practice, what is a subject of the next section.

5 Experimental Results

In the vehicle routing problem with time windows it is assumed that there is a warehouse "centrally" located to the customers. There is a road between each pair of customers and between each customer and the warehouse. The objective is to supply goods to all customers at the minimum cost. A solution with fewer routes (first goal of optimization) is better than a solution with a smaller total distance travelled (second goal of optimization). With each customer as well as with the warehouse a time window is associated. Each customer has its own demand for goods and should be visited only once. Every route must start and terminate at the warehouse and should preserve the maximum vehicle capacity. The sequential algorithm proposed in [4] was the starting point for parallelization.

The experiments were carried out on the NEC Xeon EM64T Cluster installed at the High Performance Computing Center in Stuttgart. Additionally, the NEC TX-7 (ccNUMA) system was used. Due to the lack of access to a genuine SMP cluster with 4 CPUs per node, the use of 4 threads per node was emulated (the NEC Xeon EM64T Clusters consisted of 2 CPU nodes).

The experiments were conducted on 5 tests from Solomon's benchmarking suite [14]: R108, R111, RC105, RC106 and RC108. It was shown in [5] that all the tests differ from each other substantially in terms of difficulty. This difficulty was measured by the factor Pr_1 – the probability that after an execution of the co-operating searches algorithm (CS, see [5] for details) a solution with the

minimum number of routes is found. In the current research we measured the same factor for algorithms PC, NC and HC. However:

- the following numbers of processors: $4, 8, 16, 20, 32, 40, 60, 80, 100, 120$ were used instead of $5, 10, 15, 20$,
- the limit on the execution time instead on the number of performed cooling stages was imposed.

It should be also stressed that the goal of parallelization in the PC, NC and HC algorithms is to achieve a high speedup with as small deterioration of solution quality as possible.

Table 2. Solomon's tests ranked by the factor Pr_1 obtained for the CS algorithm

Test	Algorithm					
	CS	PC1	PC10	NC	HC2	HC4
RC106	0.21	0.17	0.13	0.07	0.12	0.16
RC105	0.35	0.32	0.60	0.68	0.71	0.73
R111	0.66	0.47	0.30	0.23	0.31	0.48
R108	0.94	0.46	0.56	0.82	0.88	0.92
RC108	1.00	0.96	0.86	1.00	1.00	1.00

Table 2 shows the results of the experiments for the selected Solomon's tests ranked by the factor Pr_1 obtained by executing the CS algorithm. Our results confirm that the tests differ from each other substantially in terms of difficulty. It can be seen that in every case an algorithm with co-operating processes, either the PC or HC, outperforms the NC algorithm. Let us compare the results obtained for growing numbers of processors (Figure 3). For the hardest test RC106 the significant loss of the quality of results is observed as the number of processors increases. The co-operation of processes alleviates this loss, as in every case the results of the NC algorithm are worse as compared to other algorithms. Generally, the results of the PC1 algorithm are the best, although the results of the HC4 algorithm can also be described as satisfactory.

Considering the test RC105, the algorithms with periodic communication do not perform well. For every number of processors the NC algorithm generates results of better quality. All versions of the HC algorithm perform better than the versions of the PC. If the number of processors exceeds 40, the HC versions outperform the NC algorithm as well. Note that except of 4 processors the HC4 algorithm produces the results which are the closest to solutions generated by the sequential algorithm (number of processors = 1).

The positive influence of the co-operation of processes can be observed for the R111 test. The HC4 algorithm gives the best results and the NC the worst. The former algorithm gives also the best results for the test R108, although the NC algorithm outperforms both versions of the PC algorithm. In case of the test RC108 it is easy to find a solution with the minimum number of routes.

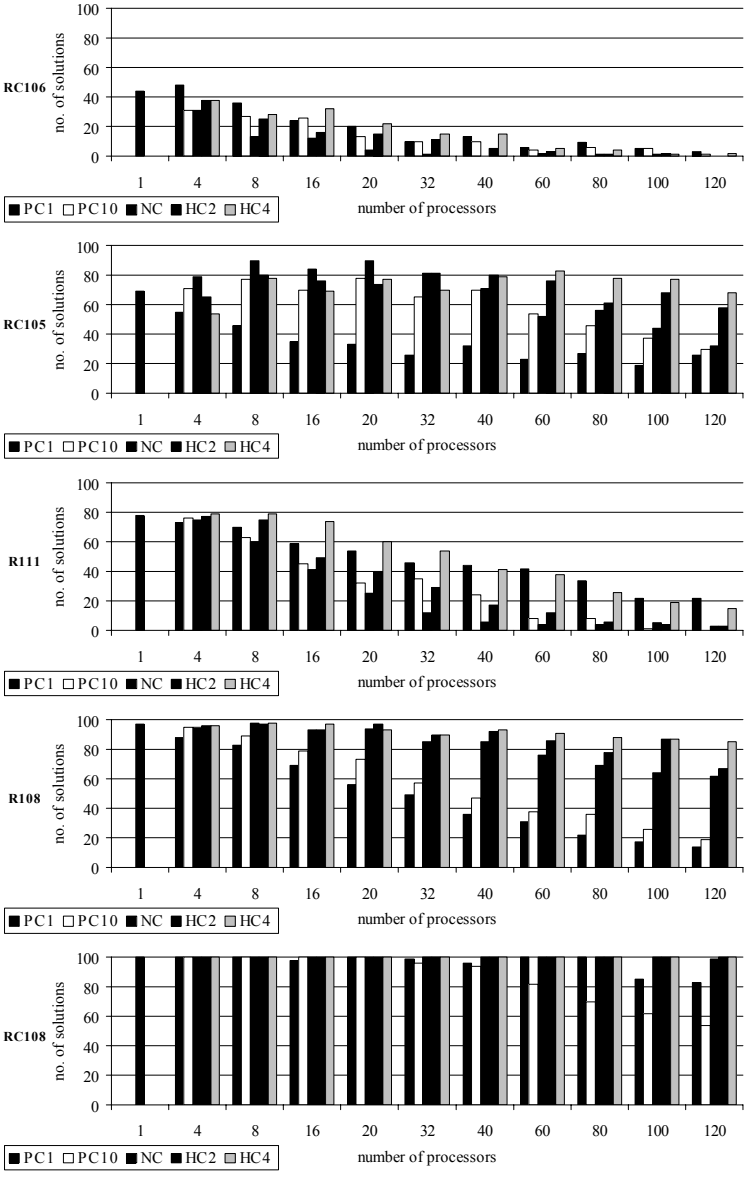

Fig. 3. Comparison of quality of results

Using the HC algorithms this can be done even for more than 400 processors (what is not presented in the figure). The quality of results generated by the NC algorithm decreases when the number of processors exceeds 120. Solutions of the PC algorithms lose their quality much earlier.

6 Conclusions

The theoretical analysis shows that the efficiency of the NC algorithm is the best among the algorithms under consideration. The experiments indicate that the co-operation of processes (not present in the NC algorithm) can improve the quality of solutions. The influence of periodic communication (PC) strongly depends on a test. Although for the test R111 the results of the PC1 algorithm were very good, for the test R108 they were the worst. The periodical co-operation of processes does not always compensate for shorter annealing sub-chains executed by processes. In the HC algorithms the inner level communication enables to extend the length of the sub-chains while preserving the same number of trials executed within a cooling stage. That is why the quality of results of these algorithms are generally better than the results of the NC algorithm. The HC algorithms can be also described as ,,balanced", since the quality of results is satisfactory for all investigated tests. Summing up, the HC algorithms are the advisable choice for solving the VRPTW by parallel simulated annealing.

Acknowledgement

The authors thank Rolf Rabenseifner of the High Performance Computing Center in Stuttgart for his contribution to this work. The research was supported by the HPC-Europa project (contract No RII3-CT-2003-506079), the Minister of Education and Science of Poland grants 3 T11F 004 29 and BK-239/RAu2/2006. Computing time was provided within the framework of the HLRS-NEC co-operation and by the following computing centers: Academic Computer Centre in Gdańsk TASK, Academic Computer Centre CYFRONET AGH, Kraków (computing grant 027/2004), Poznań Supercomputing and Networking Center, Interdisciplinary Centre for Mathematical and Computational Modelling, Warsaw University (computing grant G27-9), Wrocław Centre for Networking and Supercomputing (computing grant 04/97).

References

1. Aarts, E., de Bont, F., Habers, J., van Laarhoven, P.: Parallel implementations of the statistical cooling algorithm. Integration, the VLSI journal, 209–238 (1986)
2. Aarts, E., Korst, J.: Simulated Annealing and Boltzman Machines. John Wiley & Sons, Chichester (1989)
3. Azencott, R. (ed.): Simulated Annealing Parallelization Techniques. John Wiley & Sons, New York (1992)
4. Czarnas, P.: Traveling Salesman Problem With Time Windows. Solution by Simulated Annealing. MSc thesis (in Polish), Uniwersytet Wrocławski, Wrocław (2001)
5. Czech, Z.J.: Speeding up sequential simulated annealing by parallelization. In: Proc. of the International Symposium on Parallel Computing in Electrical Engineering (PARELEC 2006), Białystok, pp. 349–356 (2006)
6. Czech, Z.J.: Co-operation of processes in parallel simulated annealing. In: Proc. of the 18th IASTED International Conference on Parallel and Distributed Computing and Systems, Dallas, Texas, pp. 401–406 (2006)

7. Debudaj-Grabysz, A.: Parallel simulated annealing algorithms. PhD thesis (in Polish), Silesian University of Technology, Gliwice (2007)
8. Debudaj-Grabysz, A., Czech, Z.J.: A concurrent implementation of simulated annealing and its application to the VRPTW optimization problem. In: Juhasz, Z., Kacsuk, P., Kranzlmuller, D. (eds.) Distributed and Parallel Systems. Cluster and Grid Computing. Kluwer International Series in Engineering and Computer Science, vol. 777, pp. 201–209 (2004)
9. Debudaj-Grabysz, A., Rabenseifner, R.: Nesting OpenMP in MPI to implement a hybrid communication method of parallel simulated annealing on a cluster of SMP nodes. In: Di Martino, B., Kranzlmüller, D., Dongarra, J. (eds.) EuroPVM/MPI 2005. LNCS, vol. 3666, pp. 18–27. Springer, Heidelberg (2005)
10. Debudaj-Grabysz, A., Rabenseifner, R.: Load balanced parallel simulated annealing on a cluster of SMP nodes. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1075–1084. Springer, Heidelberg (2006)
11. Greening, D.R.: Parallel Simulated Annealing Techniques. Physica D 42, 293–306 (1990)
12. Lee, F.A.: Parallel Simulated Annealing on a Message-Passing Multi-Computer. PhD thesis, Utah State University (1995)
13. Lee, K.–G., Lee, S.–Y.: Synchronous and Asynchronous Parallel Simulated Annealing with Multiple Markov Chains. IEEE Transactions on Parallel and Distributed Systems 7(10), 993–1008 (1996)
14. Solomon, M.: Algorithms for the vehicle routing and scheduling problem with time windows constraints. Operation Research 35, 254–265 (1987), `http://w.cba.neu.edu/~msolomon/problems.htm`
15. Salamon, P., Sibani, P., Frost, R.: Facts, Conjectures and Improvements for Simulated Annealing. SIAM, Philadelphia (2002)
16. Tan, K.C., Lee, L.H., Zhu, Q.L., Ou, K.: Heuristic methods for vehicle routing problem with time windows. In: Artificial Intelligent in Engineering, pp. 281–295. Elsevier, Amsterdam (2001)