267/2007

Raport Badawczy

Research Report

RB/7/2007

Parallel simulated annealing algorithm for graph coloring problem

S. Łukasik,
Z. Kokosiński, G. Świętoń

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences

POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 8373578

fax: (+48) (22) 8372772

Kierownik Pracowni zgłaszający pracę:
prof. dr hab. inż. Olgierd Hryniewicz

Warszawa 2007

Parallel Simulated Annealing Algorithm for Graph Coloring Problem

Szymon Łukasik[1], Zbigniew Kokosiński[2], and Grzegorz Świętoń[2]

[1] Systems Research Institute, Polish Academy of Sciences,
ul. Newelska 6, 01-447 Warsaw, Poland
Szymon.Lukasik@ibspan.waw.pl

[2] Department of Automatic Control, Cracow University of Technology
ul. Warszawska 24, 31-155 Cracow, Poland

Abstract. The paper describes an application of Parallel Simulated Annealing (PSA) for solving one of the most studied NP-hard optimization problems: Graph Coloring Problem (GCP). Synchronous master-slave model with periodic solution update is being used. The paper contains description of the method, recommendations for optimal parameters settings and summary of results obtained during algorithm's evaluation. A comparison of our novel approach to a PGA metaheuristic proposed in the literature is given. Finally, directions for further work in the subject are suggested.

Key words: graph coloring, parallel simulated annealing, parallel metaheuristic

1 Introduction

Let $G = (V, E)$ be a given graph, where V is a set of $|V| = n$ vertices and E set of $|E| = m$ graph edges. Graph Coloring Problem (GCP) [1, 2] is defined as a task of finding an assignment of k colors to vertices $c : V \rightarrow \{1, \ldots, k\}, k \leq n$, such that there is no conflict of colors between adjacent vertices, i.e. $\forall(u, v) \in E : c(u) \neq c(v)$ and number of colors k used is minimal (such k is called the graph chromatic number $\chi(G)$).

A number of GCP variants is used for testing printed circuits, frequency assignment in telecommunication, job scheduling and other combinatorial optimization tasks. The problem is known to be NP–hard [3]. Intensive studies of the problem resulted in a large number of approximate and exact solving methods. GCP was the subject of Second DIMACS Challenge [4] held in 1993 and Computational Symposium on Graph Coloring and Generalizations organized in 2002. The graph instances [5] and reported research results are frequently used in development of new coloring algorithms and for reference purposes.

Most algorithms designed for GCP are iterative heuristics [11], such as genetic algorithms [6], simulated annealing [7, 8], tabu or local search techniques [9], minimizing selected cost functions. At the time of this writing, the only parallel metaheuristic for GCP is parallel genetic algorithm [10, 12–15].

The purpose of the paper is to present a new algorithm capable of solving GCP, developed on the basis of Parallel Simulated Annealing (PSA) method [16]. Recent years brought a rapid development of PSA techniques. Classical Simulated Annealing [17] was transformed into parallel processing environment in various ways: most popular approaches involve parallel moves, where single Markov chain is being evaluated by multiple processing units calculating possible moves from one state to another. The other method uses multiple threads for computing independent chains of solutions and exchanging the obtained results on a regular basis. Broad studies of both techniques can be found in [18, 19]. The PSA scheme for GCP, proposed by the authors, includes above strategies with the rate of current solution update used as a distinctive control parameter.

The paper is organized as follows. Next section is devoted to the description of the proposed PSA algorithm. Besides its general structure, details of cooling schedule and cost function being used as well as neighborhood solution generation procedure are given. Subsequent part of the paper presents results of algorithm's experimental evaluation. Final part of the contribution gives general comments on the performance of PSA algorithm, and possible directions for future work in the subject.

2 Parallel Simulated Annealing

PSA algorithm for GCP introduced in this paper uses multiple processors working concurrently on individual chains and agreeing about current solutions at fixed iteration intervals. The aim of the routine is to minimize chosen cost function, with storing the best solution found. The coordination of the algorithm is performed in master–slave model – one of processing units is responsible for collecting solutions, choosing the current one and distributing it among slave units.

The exchange interval e_i is a parameter which decides which PSA scheme is being used. Setting $e_i = 1$ is equivalent to producing single chain of solutions using multiple moves strategy. Increasing the interval e_i leads to creating semi-independent chains on all slave processors starting at each of concurrent rounds with the same established solution. Setting e_i to infinity results in performing independent simulated annealing runs.

The general scheme of the algorithm is presented below:

Parallel Simulated Annealing for Graph Coloring Problem

```
if proc=master
   best_cost:=infinity;
if proc=slave
   // generate initial solution randomly at each slave
   solution[proc]:=Generate_Initial_Solution();
for iter:=1 to iter_no do
   if proc=slave
      // each slave generates a new solution
      neighbor_solution[proc]:=Generate_Neighbor_Sol(solution[proc]);
      // and accepts it as a current one according to SA methodology
      solution[proc]:=Anneal(neighbor_solution[proc],solution[proc],T);
   // all solutions are then gathered at master
   Gather_at_master(solution[proc]);
   if proc=master
      current_cost:=infinity;
      // find solution with minimum cost and set it as a current one
      for j:=1 to slaves_no do
         if Cost(solution[j])<current_cost
            current_solution:=solution[j];
            current_cost:=Cost(solution[j]);
         // update best solution found (if applicable)
         if current_cost<best_cost
            best_solution:=current_solution;
            best_cost:=current_cost;
            // if stop condition fulfilled - end main loop
            if best_cost<=target_cost
                break;
   // distribute periodically current solution among all slaves
   if iter mod e_i = 0
      solution[proc]:=Distribute(current_solution);
   // update annealing temperature if appropriate
   T:=Update_Temperature(T);
if proc=master
   return best_solution;
```

Detailed information about our Simulated Annealing algorithm like cooling scheme, representation of the solution, method for generation of neighborhood and cost calculation will be given in following subsections.

2.1 Cooling Schedule

Choosing proper temperature schedule is crucial for algorithm based on Simulated Annealing methodology since it influence the acceptance probability of positive transitions (i.e. when a cost difference $\Delta_{cost,i}$ between generated neighbor and initial solution is positive) given by Metropolis [20]:

$$P(\Delta_{cost,i}) = e^{-\frac{\Delta_{cost,i}}{T_i}} \ . \tag{1}$$

As a result of intensive studies in this area multiple cooling strategies were developed [21]. The cooling schedule used here is the exponential one:

$$T_{i+1} = \alpha T_i \ , \tag{2}$$

where α is the cooling rate (usually set at 0.80–0.99 level [22]) for each cooling step. Every SA step consists of M_i iterations. For the exponential schedule following holds:

$$M_{i+1} = \beta M_i \ . \tag{3}$$

In order to extend gradually SA runs at lower temperature levels constant β is chosen usually from the range $[1.01, 1.20]$.

In addition to proper cooling schedule one has to choose correct initial temperature T_0. The authors used the most common method that involves calculating average cost difference $\overline{\Delta}_{cost,0}$ from a set of pilot runs consisting of positive transitions from an initial state. Preliminary temperature assuring desired initial acceptance probability $P(\Delta_{cost,0})$ can be calculated afterwards from equation:

$$T_0 = -\frac{\overline{\Delta}_{cost,0}}{\ln P(\Delta_{cost,0})} . \tag{4}$$

The alternative approach could follow a universal method for initial temperature selection introduced in [23].

2.2 Solution Representation and Neighborhood Generation

A graph coloring c is represented by a sequence of natural numbers $c =< c[1], \ldots, c[n] >$, $c[i] \in \{1, \ldots, k\}$, which is equivalent to set partition representation with exactly k non–empty blocks.

A rule for generation of a neighbor solution can be selected out of a wide range of existing methods [24]. For the purpose of the presented algorithm the following form of restricted 1-exchange neighborhood is used:

Restricted 1-exchange neighborhood for Graph Coloring Problem

```
// check for vertices with color conflicts
conflict_vertices:=Find_Conflicting_Vertices(c);
if sizeof(conflict_vertices)>0 do
   // if conflicts were found choose a random conflicting vertex
   vertex_to_change:=random(conflict_vertices);
   // and replace its color randomly with one of the colors {1, ... ,k+1}
   c[vertex_to_change]:=random(k+1);
else
   // if no conflicts are found choose random vertex
   vertex_to_change:=random(n);
   // and replace its color randomly with one of the colors {1, ... ,k}
   c[vertex_to_change]:=random(k);
return c;
```

2.3 Cost Assessment

As a quality measure of a selected coloring c the following cost function was used [13]:

$$f(c) = \sum_{(u,v)\in E} q(u,v) + d + k , \tag{5}$$

where q – is a penalty function:

$$q(u,v) = \begin{cases} 2 \; when \; c(u) = c(v) \\ 0 \; otherwise \end{cases} \tag{6}$$

d – is a coefficient for solution with conflicts:

$$d = \begin{cases} 1 \; when \; \sum_{(u,v)\in E} q(u,v) > 0 \\ 0 \; when \; \sum_{(u,v)\in E} q(u,v) = 0 \end{cases} \tag{7}$$

and
k – is the number of colors used.

3 Experimental Evaluation

For testing purposes an implementation of the algorithm based on Message Passing Interface was prepared. All experiments with simulated parallelism were carried out on Intel® Xeon™ machine. As test instances standard DIMACS graphs, obtained from [5], were used. For experiments following values of SA control parameters were chosen: $\alpha = 0.95$ and $\beta = 1.05$. Initial temperature was determined from a pilot run consisting of 1% (relative to overall iteration number) positive transitions. The termination condition was either achieving the optimal solution or the required number of iterations.

Due to space limitations only most representative results are presented in the paper. The full set of simulation data can be found on the first author's web site (`http://www.pk.edu.pl/~szymonl`).

3.1 SA Parameters Settings

At first, the optimal values of SA parameters were under investigation. Essential results of those experiments are gathered in Table 1.

Table 1. Influence of Simulated Annealing parameters on algorithm's performance

Graph G(V,E)	Description	$P(\Delta_{cost,0})$ Results	Best P	T_f Results	Best T_f	k_0 Results	Best k_0
anna, $\chi(G) = 11$	best f(c)	11.12	70%	11.00	$0.04 \cdot T_0$	11.12	$\chi(G)$
$\|V\| = 138$	avg. f(c)	11.32		11.14		11.17	
$\|E\| = 493$	$\sigma_{f(c)}$	0.29		0.21		0.05	
queen8_8, $\chi(G) = 9$	best f(c)	11.33	60%	10.60	$0.06 \cdot T_0$	11.33	$\chi(G)$
$\|V\| = 64$	avg. f(c)	11.46		11.84		11.41	
$\|E\| = 728$	$\sigma_{f(c)}$	0.10		1.26		0.05	
mulsol.i.4, $\chi(G) = 31$	best f(c)	38.23	80%	31.03	$0.2 \cdot T_0$	37.63	$\chi(G) - 5$
$\|V\| = 197$	avg. f(c)	38.66		33.65		38.22	
$\|E\| = 3925$	$\sigma_{f(c)}$	0.46		1.64		0.36	
myciel7, $\chi(G) = 8$	best f(c)	12.95	60%	8.00	$0.2 \cdot T_0$	11.38	$\chi(G) - 5$
$\|V\| = 191$	avg. f(c)	13.22		9.47		12.93	
$\|E\| = 2360$	$\sigma_{f(c)}$	0.34		1.60		0.96	

Opening set of 500 runs with final temperature $T_f = 0.1$, $iter\_no = 10000$, randomly generated initial solution with $k = \chi(G)$ and the initial probability changing within the range 10%–90% proved that the best algorithm's performance, measured primarily by minimum average cost function (the second criterion was the iteration number), is achieved for high initial probabilities with an optimum found at about 60%–80%. It was observed, however, that the exact choice of $P(\Delta_{cost,0})$ in this range is not very significant for the overall algorithm's performance. Obtained results confirmed a hypothesis that for more complex problems it is advisable to use higher values of initial temperature.

In the next experiment the optimal final temperature (relative to T_0) was under examination. For fixed $P(\Delta_{cost,0}) = 70\%$, $iter\_no = 10000$ and $k = \chi(G)$ the best results were obtained for $T_f \in [0.01, 0.2] \cdot T_0$. Again, higher solution quality for more complex graph instances was achieved with increased temperature ratios.

The influence of initial number of colors k_0 on the solution quality was also determined experimentally. The range $[\chi(G) - 5, \chi(G) + 5]$ was under consideration with $P(\Delta_{cost,0}) = 70\%$, $T_f = 0.05 \cdot T_0$ and $iter\_no = 10000$. It was observed that using initial color number slightly different than chromatic number do not affect significantly the algorithm's performance. For some graph instances it is even recommended to start with colorings with k_0 lower than $\chi(G)$.

In the end it should be noted that above presented statements are to be treated as overall guidelines for SA parameters settings obtained from a relatively small set of graphs. The exact values for those parameters depend largely on the considered class of graph instances.

3.2 Influence of Parallelization Schemes

The second stage of the computing experiments involved examination of algorithm's performance with different parallelization schemes and comparison with results obtained with sequential Simulated Annealing algorithm. For PSA the configurations with $e_i = \{1, 2, 4, 6, 8, 10, \infty\}$ and slaves number from 2 to 18, were tested with various graph instances. To examine the effect of parallelization on the processing time the same number of iterations $iter\_no = 100000$ was set for both sequential SA and PSA algorithms (in PSA each slave performs only $iter\_no/slaves\_no$ iterations). For the temperature schedule following settings were applied: $P(\Delta_{cost,0}) = 70\%$ and $T_f = 0.05 \cdot T_0$.

Obtained results include mean values of the cost function, the number of conflict–free/optimal solutions, the number of iterations needed to find an optimal coloring (if applicable), and algorithm's execution time t[s] (until best solution has been found). The summary of the results is presented in Table 2. Best and worst parallel configurations, in terms of average $f(c)$ and processing time, with the obtained results are reported. As a reference average performance of the algorithm is given as well.

PSA clearly outperforms the sequential Simulated Annealing in terms of computation time. Moreover, applying parallelization improves the quality of the obtained solution. It can be seen though that it is important to select a proper configuration of the PSA algorithm to achieve its high efficiency. For most problem instances it is advisable to use multiple moves strategy with optimal, relatively small, number of slaves. There exists one exception to the presented statement - for the class of *mulsol.i* graphs significantly better results were obtained with fully independent SA runs.

The worst results of using Parallel Simulated Annealing were obtained when a high number of slaves was involved in the computations and parallelization scheme was far from the optimal one.

Table 2. Experimental evaluation of the PSA algorithm for GCP

Graph		SA	PSA Results			PSA Config.	
G(V,E)	Description	Results	Best	Worst	Average	Best	Worst
games120	avg. f(c)	9	9	9	9	7 slaves	18 slaves
$\chi(G) = 9$	c.-f. c /opt. c	100/100	100/100	100/100	100/100	$e_i = 1$	$e_i = \infty$
$\|V\| = 120$	avg. iter. /opt. c	477	78	258	176		
$\|E\| = 638$	avg. t[s] /best c	0.72	0.05	0.40	0.14		
anna	avg. f(c)	11	11	11.32	11.02	4 slaves	18 slaves
$\chi(G) = 11$	c.-f. c /opt. c	100/100	100/100	100/72	100/98	$e_i = 1$	$e_i = 1$
$\|V\| = 138$	avg. iter. /opt. c	5821	199	1177	462		
$\|E\| = 493$	avg. t[s] /best c	1.31	0.08	1.73	0.31		
myciel7	avg. f(c)	8	8	8.66	8.05	3 slaves	18 slaves
$\chi(G) = 8$	c.-f. c /opt. c	100/100	100/100	100/43	100/95	$e_i = 1$	$e_i = 1$
$\|V\| = 191$	avg. iter. /opt. c	7376	797	1524	1539		
$\|E\| = 2360$	avg. t[s] /best c	1.85	0.20	1.83	1.03		
miles500	avg. f(c)	20	20	20.1	20.01	6 slaves	17 slaves
$\chi(G) = 20$	c.-f. c /opt. c	100/100	100/100	100/90	100/98	$e_i = 1$	$e_1 = 1$
$\|V\| = 128$	avg. iter. /opt. c	38001	544	422	2842		
$\|E\| = 1170$	avg. t[s] /best c	4.71	0.21	0.58	1.06		
mulsol.i.4	avg. f(c)	31.04	31.19	38.24	34.74	2 slaves	17 slaves
$\chi(G) = 31$	c.-f. c /opt. c	100/96	100/81	100/0	100/1	$e_i = \infty$	$e_i = 1$
$\|V\| = 197$	avg. iter. /opt. c	19007	15908	-	13451		
$\|E\| = 3925$	avg. t[s] /best c	4.30	1.83	2.64	2.08		
queen8_8	avg. f(c)	9.97	9.81	10.05	9.97	5 slaves	16 slaves
$\chi(G) = 9$	c.-f. c /opt. c	100/3	100/19	100/0	100/3	$e_i = 1$	$e_i = \infty$
$\|V\| = 64$	avg. iter. /opt. c	66488	8831	-	8820		
$\|E\| = 728$	avg. t[s] /best c	1.66	0.64	3.82	1.65		
le450_15b	avg. f(c)	18.58	17.39	21.79	18.71	9 slaves	18 slaves
$\chi(G) = 15$	c.-f. c /opt. c	100/0	100/0	100/0	100/0	$e_i = 1$	$e_i = \infty$
$\|V\| = 450$	avg. iter. /opt. c	-	-	-	-		
$\|E\| = 8169$	avg. t[s] /best c	42.88	3.54	6.47	4.99		

3.3 Comparison with Parallel Genetic Algorithm

The last stage of the testing procedure involved comparison of time efficiency of the PSA algorithm and Parallel Genetic Algorithm introduced in [12]. The implementation of the PGA for GCP used in [13] was applied. Both algorithms were executed on the same machine for selected DIMACS graph instances and computation time needed to find optimal coloring was reported. PGA was executed with 3 islands, subpopulations consisting of 60 individuals, migration rate 5, migration size 5 with the best individuals being distributed, initial number of colors 4 and operators: CEX crossover (with 0.6 probability), First–Fit mutation (with 0.1 probability). For PSA 3 slaves were used, $P(\Delta_{cost,0}) = 70\%$, $T_f = 0.05 \cdot T_0$, $iter\_no = 300000$ (for instance *mulsol.i.*1 to find optimal solution runs of length 1500000 iterations were needed). For most instances multiple moves strategy was applied. One exception was *mulsol.i* class of graphs where, according to earlier observations, independent SA runs were executed.

Results of the experiments, enclosed in Table 3, clearly demonstrate that PSA performance is comparable to the one achieved by the PGA. For some graph instances, like book graphs and *miles*500, the proposed algorithm was found to be superior. On the other hand, there exist a group of problems relatively easy to solve by PGA and, at the same time, difficult to solve by PSA (like *mulsol.i.*1).

Table 3. Comparison of time efficiency of PGA and PSA algorithms applied for GCP

Graph G(V,E)	t[s]		Graph G(V,E)	t[s]	
	PSA	PGA		PSA	PGA
anna, $\chi(G) = 11$ $\lvert V\rvert = 138, \lvert E\rvert = 493$	0.23	0.35	mulsol.i.4, $\chi(G) = 31$ $\lvert V\rvert = 185, \lvert E\rvert = 3946$	2.89	1.99
myciel7, $\chi(G) = 8$ $\lvert V\rvert = 191, \lvert E\rvert = 2360$	0.34	0.25	mulsol.i.1, $\chi(G) = 49$ $\lvert V\rvert = 197, \lvert E\rvert = 3925$	14.9	4.47
miles500, $\chi(G) = 20$ $\lvert V\rvert = 128, \lvert E\rvert = 1170$	0.48	18.0	games120, $\chi(G) = 9$ $\lvert V\rvert = 120, \lvert E\rvert = 638$	0.20	0.34

4 Conclusion

In the paper a new Parallel Simulated Annealing algorithm for GCP was introduced and evaluated.

First experiments revealed that its performance depends on choosing a cooling schedule and generation of the initial coloring suitable to the considered problem. Some general guidelines were derived for the algorithm's settings that ensure a better solution quality. Further research in the subject could concern adaptive cooling schedules and generation of initial solution by means of an approximate method.

Choosing an optimal number of processing units and parallelization scheme for the PSA was also under consideration. We found that problem specification essentially influence the proper choice of these elements. However, it can be stated as a general remark that the highest efficiency of the master–slave PSA algorithm is achieved for optimal, relatively small number of slaves.

During the performance evaluation the PSA algorithm was proved to be an effective tool for solving Graph Coloring Problem. The experiments showed that it achieves a similar performance level as PGA. The comparison results of both methods showed that none of them is superior. It encourages efforts for development of a new hybrid metaheuristics which would benefit from advantages of both PGA and PSA approaches. The overall concept of a hybrid algorithm could implement the idea presented in [25] and include some other improvements of the standard PSA scheme as proposed in [26].

References

1. Kubale, M. (Eds.): Graph colorings. American Mathematical Society, (2004).
2. Jensen T.R., Toft B.: Graph coloring problems. Wiley Interscience, New York (1995)
3. Garey R., Johnson D.S.: Computers and intractability. A guide to the theory of NP–completeness. W. H. Freeman, New York (1979)
4. Johnson, D.S., Trick, M.A. (Eds.): Cliques, Coloring and Satisfiability. DIMACS Series in Discrete Mathematics and Theoretical Science **26** (1996)
5. Graph coloring instances: `http://mat.gsia.cmu.edu/COLOR/instances.html`
6. Fleurent, C., Ferland, J.A.: Genetic and Hybrid Algorithms for Graph Coloring. Annals of Operations Research **63** (1996) 437–461
7. Chams, M., Hertz, A., de Werra, D.: Some Experiments with Simulated Annealing for Coloring Graphs. European Journal of Operational Research **32** (1987) 260–266
8. Johnson, D.S., Aragon, C.R., McGeoch, L., Schevon, C.: Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning. Operations Research **39/3** (1991) 378–406
9. Galinier, P., Hertz, A.: A Survey of Local Search Methods for Graph Coloring. Computers & Operations Research **33** (2006) 2547–2562
10. Alba E. (Ed.): Parallel Metaheuristics. John Wiley & Sons, New York (2005)
11. Szyfelbein, D.: Metaheuristics in graph coloring. In: Kubale M. (Ed.): Discrete optimization. Models and methods for graph coloring. WNT, Warszawa (2002) 26–52 (in Polish)
12. Kokosiński Z., Kołodziej M., Kwarciany K.: Parallel genetic algorithm for graph coloring problem. Proc. ICCS'2004, LNCS **3036** (2004) 215–222
13. Kokosiński, Z., Kwarciany, K., Kołodziej, M.: Efficient Graph Coloring with Parallel Genetic Algorithms. Computing and Informatics **24/2** (2005) 109–121
14. Łukasik, Sz.: Parallel Genetic Algorithms using Message Passing Paradigm for Graph Coloring Problem. Journal of Electrical Engineering **56/12/s** (2005) 123–126
15. Kokosiński Z., Kwarciany K.: On sum coloring of graphs with parallel genetic algorithms. Proc. ICANNGA'2007, LNCS **4431** (2007) 211–219
16. Azencott, R. (Ed.): Simulated Annealing: Parallelization Techniques. John Wiley & Sons, New York (1992)
17. Kirkpatrick, S., Gelatt, C.D., Vecchi, M. : Optimization by Simulated Annealing. Science **220** (1983) 671–680
18. Diekmann, R., Lüling, R., Simon, J.: Problem Independent Distributed Simulated Annealing and its Applications. Working Paper, University of Paderborn, Germany
19. Lee, S.-Y., Lee, K.G: Synchronous and Asynchronous Parallel Simulated Annealing with Multiple Markov Chains. IEEE Transactions on Parallel and Distributed Systems **7/10** (1996) 993–1008

20. Metropolis, N.A., Rosenbluth, A., Teller, A., Teller E.: Equation of State Calculations by Fast Computing Machines. Journal of Chemical Physics **21** (1953) 1087–1092.
21. Ingber, A.L.: Simulated Annealing: Practice versus Theory. Journal of Mathematical Computation Modelling **18/11** (1993) 29–57
22. Sait S.M., Youssef H.: Iterative computer algorithms with applications in engineering. IEEE Computer Society, Los Alamitos (1999)
23. Ben-Ameur, W.: Computing the Initial Temperature of Simulated Annealing. Computational Optimization and Applications **29** (2004) 369–385
24. Avanthay, C., Hertz, A., Zufferey, N.: A Variable Neighborhood Search for Graph Coloring. European Journal of Operational Research **151** (2003) 379–388
25. Tomoyuki, H., Mitsunori, M., Ogura, M.: Parallel Simulated Annealing using Genetic Crossover. Science and Engineering Review of Doshisha University **41/2** (2000) 130–138
26. Bożejko, W., Wodecki, M.: The New Concepts in Parallel Simulated Annealing Method. Proc. ICAISC'2004, LNCS **3070** (2004) 853–859

SEVENTH INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING AND APPLIED MATHEMATICS

Demo

Overview

User: slukasik

Contribution Details

Parallel Simulated Annealing Algorithm for Graph Coloring Problem

142

Main Track

S. Łukasik, Z. Kokosiński, G. Świętoń
submitted by: Szymon Łukasik

Topics: **"Parallel/distributed algorithms""Evolutionary computing and neural networks""Methods and tools for parallel solution of large-scale problems"**
Keywords: graph coloring, parallel simulated annealing, parallel metaheuristic

par_SA4.pdf (2007-05-07 11:48:49)

Review Result of the Program Committee

Congratulations! This contribution has been accepted.

Overview of Reviews

Questions		**Review 1**	**Review 2**
Familiarity of the reviewers with the topic		10	6
Content	10%	6	6
Significance	10%	6	6
Originality	10%	6	6
Relevance	10%	10	6
Presentation	10%	8	6
Recommendation	50%	8	6
Total points (out of 100)		**76**	**60**

> index > paperDetails

Print View

Contact Address: roman@icis.pcz.pl
Conference: PPAM 2007

19° Seventh International Conference on Parallel Processing and Applied Mathematics 51°

PPAM 2007

Sponsors:

- call
- topics
- speakers
- program
- tutorials
- workshops
- committee
- submission
- dates
- location
- fee
- registration
- accommodation
- review
- gallery
- ppam 1999
- ppam 2001
- ppam 2003
- ppam 2005
- ppam 2009

: : CALL FOR PAPERS

FIRST ANNOUNCEMENT AND CALL FOR PAPERS

PPAM 2007
SEVENTH INTERNATIONAL CONFERENCE
ON PARALLEL PROCESSING AND APPLIED MATHEMATICS

Gdansk, POLAND,
September 9-12, 2007

http://ppam.pl

The PPAM 2007 conference, seventh in a series, will cover topics in parallel and distributed processing, including theory and applications, as well as applied mathematics. The focus will be on grid computing, and large-scale applications, as well as on software tools which facilitate efficient and convenient utilization of modern computing architectures.

PPAM is a biennial conference started in 1994, with the proceedings published in Springer's Lecture Notes in Computer Sciences. Next year the PPAM conference will take place in Gdansk, the thousand-year old city on the Baltic coast, the hometown of Hevelius, Fahrenheit, Schopenhauer, Grass and Walesa. One of social events will be in Malbork - the largest medieval brick castle in Europe.

Organized in-cooperation with the Society for Industrial and Applied Mathematics - SIAM.

Topics of interest include (but are not limited) to:

- Parallel/distributed and grid architectures, enabling technologies
- Multi-core parallel computing
- Cluster computing
- Pervasive, mobile,and ubiquitous computing
- Parallel/distributed algorithms
- Scheduling, mapping, load balancing
- Performance analysis and prediction
- Parallel/distributed/grid programming
- Tools and environments for parallel/distributed/grid processing
- Numerical linear algebra
- Methods of solving differential equations
- Evolutionary computing and neural networks
- Mathematical and computer methods in mechanics, material processing, biology and medicine, physics, chemistry, business, environmental modeling, etc.
- Applications of parallel/distributed/grid computing
- Methods and tools for parallel solution of large-scale problems

KEYNOTE SPEAKERS

David Bader	Georgia Institute of Technology, USA	abstract
Ben Bennett	ClearSpeed Technology	
Ewa Deelman	University of Southern California, USA	abstract
Jack Dongarra	University of Tennessee and ORNL, USA	abstract
Richard Dracott	General Manager HPC Intel Corporation	
Erik Elmroth	Umea University, Sweden	abstract
Fabrizio Gagliardi	Microsoft Research, USA	
Angel E. Garcia	Rensselaer Polytechnic Institute, USA	abstract
Fred Gustavson	IBM T.J. Watson Research Center, USA	abstract
Hans-Christian Hoppe	Intel Corporation	
Vladik Kreinovich	University of Texas at El Paso, USA	abstract
Jarek Nieplocha	Pacific Northwest National Laboratory, USA	

Jennifer Schopf	Argonne National Laboratory and eScience Institute	abstract
Masha Sosonkina	Ames Laboratory and Iowa State University, USA	abstract
Boleslaw K. Szymanski	Rensselaer Polytechnic Institute, USA	abstract
Jerzy Wasniewski	Technical University of Denmark	

TUTORIALS (preliminary list)

1. **"Globus Toolkit", Globus Team**
2. **"Intel tools for grid programming", Intel Corporation**
3. **"Grid Computing with GridWay on Globus Infrastructures: Porting Applications Using the DRMAA Standard", GridWay Team**
4. **"New Data Structures for the Cell Processor", IBM Thomas J. Watson Research Center**

The first day of PPAM 2007 is reserved for tutorials.

WORKSHOPS, MINISYMPOSIA, SPECIAL SESSIONS (preliminary list)

- **The Third Grid Applications and Middleware Workshop**
- **The Second Minisymposium on Novel Data Formats and Algorithms for Dense Linear Algebra Computations**
- **Combinatorial tools for parallel sparse matrix computations**
- **Workshop on Parallel Bio-Computing**
- **Minisymposium on Interval Analysis**
- **Workshop on Scheduling for Parallel Computing**
- **Workshop on Large Scale Computations on Grids**
- **Workshop on Language-Based Parallel Programming Models**
- **Workshop on Models, Algorithms and Methodologies for Grid-Enabled Computing Environments**
- **Performance Evaluation of Parallel Applications on Large-Scale Systems**
- **Workshop on High Performance Computing for Engineering Applications**
- Special Session on "Simulations with Particles"
- **Seminar on Remote Instrumentation**

PAPER SUBMISSION AND PUBLICATION

Original papers are invited for the conference. Authors should submit full papers (draft version, PDF file, together with abstract in ASCII format) to the conference office before April 30, 2007. Regular papers are not to exceed 10 pages (LNCS style). Papers will be refereed and accepted on the basis of their scientific merit and relevance to the conference topics. Abstracts of accepted papers will be available during the conference in the form of a brochure. Only papers presented at PPAM 2007 will be included in the proceedings, which once again will be published after the conference by Springer-Verlag in the LNCS series. Full camera-ready versions of accepted papers will be required by October 15, 2007.

IMPORTANT DATES

Submission of Papers: May 15, 2007
Notification of Acceptance: June 30, 2007
Camera-Ready Papers: Oct. 15, 2007

Roman Wyrzykowski Czestochowa University of Technology, Poland
CHAIR OF PROGRAM COMMITTEE
e-mail: roman@icis.pcz.pl
Phone: +48 668 476 840

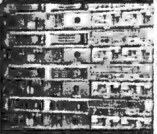
Boleslaw Szymanski Rensselaer Polytechnic Institute, USA
VICE-CHAIR OF PROGRAM COMMITTEE

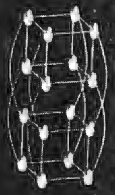
19°

Częstochowa

51°

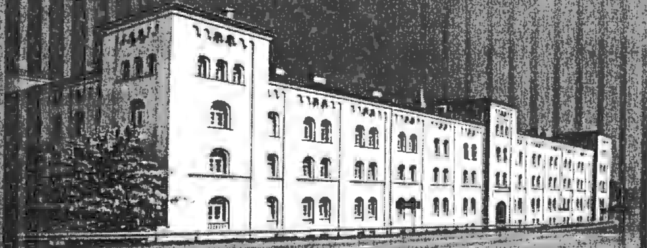
PPAM 2007





7th International Conference on Parallel Processing & Applied Mathematics

Gdańsk • Poland • September 9-12, 2007



TASK intel Microsoft IBM ACTION siam

Table of Contents

Parallel Simulated Annealing Algorithm for Graph Coloring Problem

Szymon Łukasik[1 2] , Zbigniew Kokosiński[1] , Grzegorz Świętoń[1]
[1]Department of Automatic Control
Cracow University of Technology, ul. Warszawska 24, 31-155 Cracow, Poland
[2]Systems Research Institute
Polish Academy of Sciences, ul. Newelska 6, 01-447 Warsaw, Poland
Szymon.Lukasik@ibspan.waw.pl

The paper describes an application of Parallel Simulated Annealing (PSA) for solving one of the most studied NP-hard optimization problems: Graph Coloring Problem (GCP). Synchronous master-slave model with periodic solution update is being used. The paper contains description of the method, recommendations for optimal parameters settings and summary of results obtained during algorithm's evaluation. A comparison of our novel approach to a PGA metaheuristic proposed in the literature is given. Finally, directions for further work in the subject are suggested.