

# Parallel Simulated Annealing Algorithm for Graph Coloring Problem

Szymon Łukasik<sup>1,2</sup>

Zbigniew Kokosiński<sup>2</sup>

Grzegorz Świętoń<sup>2</sup>

12 września 2007

---

<sup>1</sup>Polish Academy of Sciences, Systems Research Institute

<sup>2</sup>Cracow University of Technology, Department of Automatic Control

## Introduction

---

- Graph Coloring Problem (GCP)
- Parallel Simulated Annealing

Parallel Simulated Annealing for GCP

---

Experimental results

---

Final remarks

---

# Introduction

# Graph Coloring Problem (GCP)

## Introduction

### ● Graph Coloring Problem (GCP)

### ● Parallel Simulated Annealing

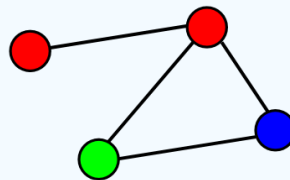
### Parallel Simulated Annealing for GCP

### Experimental results

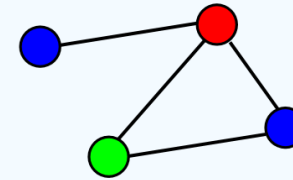
### Final remarks

Let:  $G = (V, E)$  be a given graph with  $n$  vertices  $V$  connected with  $m$  edges  $E$

Graph Coloring Problem: Find an assignment of  $k$  colors to vertices  $c : V \rightarrow \{1, \dots, k\}$ ,  $k \leq n$  such as  $\forall (u, v) \in E : c(u) \neq c(v)$  and  $k$  is minimal (graph chromatic number  $\chi(G)$ ).



1 conflict



conflict-free

GCP is one of the NP-hard problems.

Main heuristic solving methods:

- local and tabu search (Galinier & Hertz, 2006),
- genetic algorithms (Fleurent & Ferland, 1996),
- simulated annealing (Johnson et al., 1991),

+ one metaheuristics application - parallel genetic algorithm (Kokosiński et al., 2005).

# Parallel Simulated Annealing

## Introduction

### ● Graph Coloring Problem (GCP)

### ● Parallel Simulated Annealing

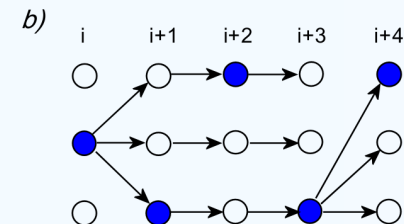
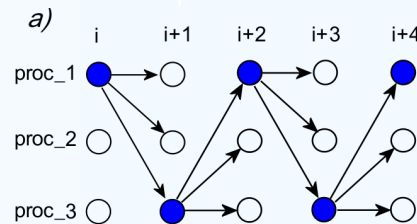
### Parallel Simulated Annealing for GCP

### Experimental results

### Final remarks

Two main concepts (Lee & Lee, 1996):

- parallel moves (trials) - single Markov chain is being evaluated by multiple processing units calculating possible moves from one state to another,
- multiple Markov chains - uses multiple threads for computing independent chains of solutions and exchanging the obtained results on a regular basis (synchronous or asynchronous).



Recent years brought rapid development of this technique (both in theory and practice). Some applications and improvements are:

- vehicle routing (Czech, 2001), flow shop (Wodecki & Bożejko, 2001), global optimization (Onbaşıoğlu & Özdamar, 2005)
- multisteps, backtrack jumps (Bożejko & Wodecki, 2004), genetic algorithm at higher level (Tomoyuki et al., 2000).

Introduction

---

Parallel Simulated Annealing  
for GCP

---

- Proposed Algorithm
- Solution representation & Cost assessment
- Neighborhood Generation - Algorithm

Experimental results

---

Final remarks

---

# Parallel Simulated Annealing for GCP

# Proposed Algorithm

## Introduction

## Parallel Simulated Annealing for GCP

### ● Proposed Algorithm

- Solution representation & Cost assessment
- Neighborhood Generation - Algorithm

## Experimental results

## Final remarks

```
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;
```

# Solution representation & Cost assessment

Introduction

Parallel Simulated Annealing  
for GCP

- Proposed Algorithm
- **Solution representation & Cost assessment**
- Neighborhood Generation - Algorithm

Experimental results

Final remarks

- A graph coloring  $c$  is represented by a sequence of natural numbers  $c = \langle c[1], \dots, c[n] \rangle$ ,  $c[i] \in \{1, \dots, k\}$ .
- Cost of solution is determined using following cost function (Kokosiński et al., 2005):

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

where  $q$  – is a penalty function:

$$q(u,v) = \begin{cases} 2 & \text{when } c(u) = c(v) \\ 0 & \text{otherwise} \end{cases}$$

$d$  – is a coefficient for solution with conflicts:

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

and  $k$  – is the number of colors used.

# Neighborhood Generation - Algorithm

Introduction

Parallel Simulated Annealing  
for GCP

- Proposed Algorithm
- Solution representation &  
Cost assessment
- **Neighborhood Generation  
- Algorithm**

Experimental results

Final remarks

```
// check for vertices with color conflicts
conflict_vertices := Find_Conflicting_Vertices(c);

if sizeof(conflict_vertices) > 0 do
    // conflicts were found – choose 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;
```



Introduction

---

Parallel Simulated Annealing  
for GCP

---

**Experimental results**

---

- Testing environment
- Simulated Annealing parameters settings
- Parallel Simulated Annealing for GCP performance evaluation
- Comparison with Parallel Genetic Algorithm

Final remarks

---

# Experimental results

# Testing environment

## Introduction

## Parallel Simulated Annealing for GCP

## Experimental results

### ● Testing environment

- Simulated Annealing parameters settings
- Parallel Simulated Annealing for GCP performance evaluation
- Comparison with Parallel Genetic Algorithm

## Final remarks

- Algorithm was implemented using MPICH-2 library and tested on LINUX-based system.
- All experiments were carried out on one Intel<sup>®</sup> Xeon<sup>™</sup> machine using simulated parallelism (number of iterations was inversely proportional to the number of slaves).
- As test instances standard DIMACS graphs were used.
- Initial temperature was determined from a pilot run consisting of 1% (relative to overall iteration number) positive transitions.
- For experiments following values of SA control parameters were chosen:  $\alpha = 0.95$  and  $\beta = 1.05$ .
- The termination condition was either achieving the optimal solution or the required number of iterations.

# Simulated Annealing parameters settings

## Introduction

## Parallel Simulated Annealing for GCP

## Experimental results

- Testing environment
- **Simulated Annealing parameters settings**
- Parallel Simulated Annealing for GCP performance evaluation
- Comparison with Parallel Genetic Algorithm

## Final remarks

Graph G(V,E)	Description	$P(\Delta_{cost,0})^1$		$T_f^2$		$k_0^3$	
		Results	Best $P$	Results	Best $T_f$	Results	Best $k_0$
anna, $\chi(G) = 11$ $ V  = 138$ $ E  = 493$	best f(c)	11.12	70%	11.00	$0.04 \cdot T_0$	11.12	$\chi(G)$
	avg. f(c)	11.32		11.14		11.17	
	$\sigma_{f(c)}$	0.29		0.21		0.05	
queen8_8, $\chi(G) = 9$ $ V  = 64$ $ E  = 728$	best f(c)	11.33	60%	10.60	$0.06 \cdot T_0$	11.33	$\chi(G)$
	avg. f(c)	11.46		11.84		11.41	
	$\sigma_{f(c)}$	0.10		1.26		0.05	
mulsol.i.4, $\chi(G) = 31$ $ V  = 197$ $ E  = 3925$	best f(c)	38.23	80%	31.03	$0.2 \cdot T_0$	37.63	$\chi(G) - 5$
	avg. f(c)	38.66		33.65		38.22	
	$\sigma_{f(c)}$	0.46		1.64		0.36	
myciel7, $\chi(G) = 8$ $ V  = 191$ $ E  = 2360$	best f(c)	12.95	60%	8.00	$0.2 \cdot T_0$	11.38	$\chi(G) - 5$
	avg. f(c)	13.22		9.47		12.93	
	$\sigma_{f(c)}$	0.34		1.60		0.96	

<sup>1</sup>500 runs,  $iter\_no = 10000$ ,  $T_f = 0.1$ ,  $k_0 = \chi(G)$

<sup>2</sup>500 runs,  $iter\_no = 10000$ ,  $P(\Delta_{cost,0}) = 70\%$ ,  $k_0 = \chi(G)$

<sup>3</sup>500 runs,  $iter\_no = 10000$ ,  $P(\Delta_{cost,0}) = 70\%$ ,  $T_f = 0.05 \cdot T_0$ ,

# Parallel Simulated Annealing for GCP performance evaluation

## Introduction

## Parallel Simulated Annealing for GCP

## Experimental results

- Testing environment
- Simulated Annealing parameters settings
- **Parallel Simulated Annealing for GCP performance evaluation**
- Comparison with Parallel Genetic Algorithm

## Final remarks

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

<sup>1</sup> $P(\Delta_{cost,0}) = 70\%$  and  $T_f = 0.05 \cdot T_0$  for both SA and PSA. SA tested with  $iter\_no = 100000$ . PSA executed for 2...18 slaves with  $e_i = \{1, 2, 4, 6, 8, 10, \infty\}$  and  $iter\_no/slaves\_no$  iterations

# Comparison with Parallel Genetic Algorithm

Introduction

Parallel Simulated Annealing for GCP

Experimental results

- Testing environment
- Simulated Annealing parameters settings
- Parallel Simulated Annealing for GCP performance evaluation
- **Comparison with Parallel Genetic Algorithm**

Final remarks

Graph $G(V,E)$	t[s]	
	PSA <sup>1</sup>	PGA <sup>2</sup>
anna, $\chi(G) = 11$ $ V  = 138,  E  = 493$	0.23	0.60
multsol.i.4, $\chi(G) = 31$ $ V  = 185,  E  = 3946$	2.89	3.00
myciel7, $\chi(G) = 8$ $ V  = 191,  E  = 2360$	0.34	1.40
multsol.i.1, $\chi(G) = 49$ $ V  = 197,  E  = 3925$	14.9	9.27
miles500, $\chi(G) = 20$ $ V  = 128,  E  = 1170$	0.48	18.0
queen8_8, $\chi(G) = 9$ $ V  = 64,  E  = 728$	51.8	0.87

<sup>1</sup>PSA: 3 slaves,  $P(\Delta_{cost,0}) = 70\%$ ,  $T_f = 0.05 \cdot T_0$ ,  $e_i = 1$  and  $e_i = \infty$  for *multsol.i*

<sup>2</sup>PGA as in (Kokosiński et al., 2005): 3 islands, subpopulations - 60 individuals, migration rate 5, best individuals migration size 5,  $k_0 = 4$  and operators: CEX crossover (with 0.6 probability), First-Fit mutation (with 0.1 probability)

Introduction

---

Parallel Simulated Annealing  
for GCP

---

Experimental results

---

**Final remarks**

---

- Comments and future work
- Final slide
- Bibliography

**Final remarks**

# Comments and future work

Introduction

Parallel Simulated Annealing  
for GCP

Experimental results

Final remarks

● **Comments and future  
work**

● Final slide

● Bibliography

- Algorithm performance depends on chosen cooling schedule and generated initial coloring. Further research could concern adaptive cooling schedule and generating initial solution by means of an approximate method.
- There exists optimal - relatively small number of slaves - for which highest algorithm performance is observed. In most cases parallel moves method outperform multiple Markov chains strategy.
- PSA algorithm was proved to be an effective tool for solving GCP - it achieves a similar performance level as PGA. Interesting result might be obtained with hybrid PSA-PGA algorithm.

**Thank you for your attention!**



# Bibliography

Introduction

Parallel Simulated Annealing  
for GCP

Experimental results

Final remarks

- Comments and future work
- Final slide
- **Bibliography**

- [1] Bożejko, W., Wodecki, M.: The New Concepts in Parallel Simulated Annealing Method. Proc. ICAISC'2004, LNCS **3070** (2004) 853–859
- [2] Czech, Z.J.: Three Parallel Algorithms for Simulated Annealing. Proc. PPAM'2001, LNCS **2328** (2001) 210–217
- [3] Fleurent, C., Ferland, J.A.: Genetic and Hybrid Algorithms for Graph Coloring. Annals of Operations Research **63** (1996) 437–461
- [4] Galinier, P., Hertz, A.: A Survey of Local Search Methods for Graph Coloring. Computers & Operations Research **33** (2006) 2547–2562
- [5] 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
- [6] Kokosiński, Z., Kwarciany, K., Kołodziej, M.: Efficient Graph Coloring with Parallel Genetic Algorithms. Computing and Informatics **24/2** (2005) 109–121
- [7] 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
- [8] Onbaşoğlu, E, Özdamar, L.: Parallel Simulated Annealing Algorithms in Global Optimization. Journal of Global Optimization **19** (2005) 27–50
- [9] Tomoyuki, H., Mitsunori, M., Ogura, M.: Parallel Simulated Annealing using Genetic Crossover. Science and Engineering Review of Doshisha University **41/2** (2000) 130–138
- [10] Wodecki, M., Bożejko, W.: Solving the Flow Shop Problem by Parallel Simulated Annealing Proc. PPAM'2001, LNCS **2328** (2001) 236–244