Position Papers of the Federated Conference on Computer Science and Information Systems pp. 79–85 DOI: 10.15439/2016F414 ACSIS, Vol. 9. ISSN 2300-5963

Testing of parallel metaheuristics for graph partitioning problems

Zbigniew Kokosiński, Pawel Bala
Cracow University of Technology,
Faculty of Electrical and Computer Eng.
ul. Warszawska 24, 31-155 Kraków, Poland;
Email: zk@pk.edu.pl

***Abstract*—In this paper we describe computer experiments while testing a family of parallel and hybrid metaheuristics against a small set of graph partitioning problems like clustering, partitioning into cliques and coloring. In all cases the search space is composed of vertex partitions satisfying specific problem requirements. The solver application contains two sequential and nine parallel/hybrid algorithms developed on the basis of SA and TS metaheuristics. A number of tests are reported and conclusions resulting from the testing experiments are derived.**

***Index Terms*—simulated annealing, tabu search, parallel metaheuristic, hybrid metaheuristic, graph partitioning problem**

I. Introduction

COMPUTATIONAL optimization attracts for years researchers and practitioners interested in solving combinatorial problems by means of various computational methods and tools. In particular, many NPO problems require new versatile tools in order to find approximate solutions [1], [8]. Parallel and hybrid metaheuristics are among the most promisssing methods to be developed in the nearest time [2], [16]. Many new algorithms have been already designed and compared with existing methodologies [7], [11], but there is still a room for significant progress in this area.

In this paper we focus on a class of partitioning problems that appears in many application areas like data clustering [3], column-oriented database partitioning optimization [15], design of digital circuits, decomposition of large digital systems into a number of subsystems (moduls) for multi-chip implementation, task scheduling, timetabling, assiggnment of frequencies in telecommunication networks, etc. Partitionig problems are in general simpler than permutation problems but their search spaces are too huge for exhaustive search or extensive search methods [6], [9], [10], [17], [18].

The rest of the paper is organized as follows. In the next section the graph partitioning problems are defined and characterized. Then, in section 3, SA and TS algorithms as well as their parallelization and hybridization methods are presented. The design assumptions and features of the developed solver are described in section 4. Testing methodology and experimental results are shown in section 5. The final conclusions point out the directions of future research in this area.

II. Graph partitioning problems

In this section formulations of several partitioning problems are given that are to be solved by a collection of algorithms used in the experimental part of the paper.

We assume that $G = (V, E)$ is a connected, undirected graph. Let $|V| = n$, $|E| = m$.

A. Cluster partitioning problem (CPP)

A partition $C = (C_1, \ldots, C_k)$ of V is called a clustering of G and C_i clusters. C is called trivial if either $k = 1$, or all clusters C_i contain only one element. We will identify a cluster C_i with the induced subgraph of G, i.e. the graph $G_i = (C_i, E(C_i))$, where $E(C_i) = \{\{u, v\} \in E : u, v \in C_i\}$. Hence, $E(C) = \sum_{i=1}^{k} E(C_i)$ is the set of intra-cluster edges and $E \setminus E(C)$ the set of inter-cluster edges. [3]

The number intra-cluster edges is denoted by $m(C)$ and the number of inter-cluster edges by $M(C)$.

The $coverage(C)$ of a graph clustering C is a fraction of intra-cluster edges within the complete set of edges E: $coverage(C) = m(C)/m$. The larger the value of $coverage(C)$ does not necessarily mean the better quality of a clustering C.

Constructing a k-clustering with a fixed number of k, $k \geq 3$ of clusters is NP-hard [1].

In this paper we will consider k-clustering problems for weighted graphs, where the total weight of the set $E \setminus E(C)$ shall be minimized.

B. Clique partitioning problems (CPP)

A partition $C = (C_1, \ldots, C_k)$ of V is called a partition of G into cliques iff every subgraph $G_i = (C_i, E(C_i))$ induced by a cluster C_i is a clique, i.e. all vertices in C_i are pairwise connected. The goal is to find the minimal k, for which a partition into at most k cliques exists.

The clique partitioning problem is NP-complete [14]. The dual problem to CPP is graph partitioning into independent sets (ISs). It is equivalent to the CPP for $G(V, E')$, where E' is a complement of the set E.

C. Clique partitioning problems with minimum clique size (CPP)

In the present paper a solution of clique partitioning problem is also searched for given clique size at least s: is there a graph partition into k cliques satisfying a condition related to the minimum clique size s? For given n and k the minimum size of cliques in G is $s = \lfloor n/k \rfloor$. Weighted version of the problem are also known, with additional conditions related to cliques' weights [9].

D. Graph coloring problem (GCP)

Classical vertex coloring problem in a graphs is another formulation of graph partitioning into independent sets. Such ISs can be assigned different colors, satisfying the property that all pairs of adjacent vertices in G are assigned nonconflicting colors. Formally:

For given graph $G(V,E)$, the optimization problem GCP is formulated as follows: find the minimum positive integer k, $k \leq n$, and a function $c : V \longrightarrow \{1,\ldots,k\}$, such that $c(u) \neq c(v)$ whenever $(u,v) \in E$. The obtained value of k is referred to as graph chromatic number $\chi(G)$.

GCP belongs to the class of NP-complete problems [8].

E. Restricted coloring problem (RCP)

In practical applications a conflict-free vertex/edge coloring is searched, often satisfying additional requirements. Therefore, a large number of particular coloring problems arised and has been investigated [12].

One well known example is vertex coloring with some restrictions set on available colors for the given graph vertex. In RCP each vertex is assigned a list of forbidden colors and a proper solution meeting such set of constraints is searched [13].

III. Sequential and parallel metaheuristics

The reported research is based on two sequential and nine parallel algorithms. The sequential metaheuristics include classical simulated annealing (SA) and tabu search (TS) that belong to the class of iterative methods [16]. Parallel algorithms can be splitted into three categories: parallel metaheuristics derived from SA, parallel metaheuristics derived from TS and hybrid methods.

A. Simulated annealing (SA)

Classical simulated annealing [16] is a well known technique widely used in optimization and present in most of the textbooks. It can be easily parallelized in various ways. Parallel moves enable single Markov chain to be evaluated by multiple processing units calculating possible moves from one state to another. Multiple threads compute independent chains of solutions and periodically exchange the obtained results. The key question in parallel implementation remains setting of algorithm's parameters like initial temperature, and a cooling schedule. For the problem at hand it is necessary to define an appropriate solution representation, cost function and a neighborhood generation scheme.

B. Tabu search (TS)

Tabu search [16] is an improvement of local search method in which so called tabu list contains a number of recent moves that must not be considered as candidates in the present iteration. This feature helps the method to escape from local minima what is impossible in local search. The question is to define the solution representation, cost function, neighborhood and a single move, the size of the neighborhood and the number of candidate moves, aspiration level which decides on the possibility to accept forbidden moves if it leads to a solution improvement etc.

C. MIR model of parallelization

Multiple independent runs (MIR) model is a very popular way of parallelization of iterative algorithms. A number of algorithm instances with different input data are executed simultaneously. All computational processes run independently and do not exchange data during computation. At the end, the best solution from all processes is selected. This simple model can be made more sophisticated by introducing an information exchange scheme, exchange rate etc.

D. MS model of parallelization

In Master-Slave (MS) model the master executes the sequential part of an algorithm, distributes computational tasks among slaves, collects results from slaves, process and aggregates this results. In certain versions of MS model the master splits the whole search space among slaves, synchronizes their work, checks the termination condition and collects the best solution from subspaces.

E. PA model of parallelization

Parallel asynchronous (PA) model provides maximum flexibility: various algorithms with different initial data search the whole search space in an asynchronous manner. Usually an efficient update scheme for the best solution must be implemented as well as occasional distribution of best solutions to asynchronous computational processes. One possibility is to employ a communication process. In some cases shared memory (SM) can be used for information updates and exchange. The second solution helps to avoid generation of interrupts in asynchronous processes. The processes communicate the SM in predictable moments of time.

F. Hybrid models

Hybrids models include : 1. two-phase algorithms, when each phase - restriction of the search space and solution refinement - is performed by a different method; 2. combined algorithms, when known elements of existing methods are composed in a single algorithm; 3. combined algorithms consisting original components like problem-oriented operations or heuristics; and 4. concurrent algorithm which is parallel execution of known methods with data exchange patterns.

In this paper three heuristic algorithms are used.

Parallel hybrid asynchronous (H-PA) algorithm splits computational processes into "even" performing SA and "odd" performing TS. Best solutions are updated via shared memory SM, where are immediately made available for all processes.

Hybrid serial-parallel algorithm (H-SP) process in parallel p threads in which SA and TS sections are performed alternatively starting from SA section. SA section modifies tabu list while TS section modifies current temperature for the next section, respectively. Swiching conditions are related to the progress achieved in improving best solution.

Parallel hybrid algorithm (H-P) is developed on the basis MIR method. Single step combines properties of both SA and TS: if new solution satisfies aspiration criterion (AC) it is always accepted, otherwise, it is accepted according to SA rules. This means that probability of acceptance of worst solution decreases in time.

IV. The solver

For all tests it is used the ”Partitioning problems solver” application. It is written in C++ (Visual Studio), while .NET Framework 3.5 provides necessary libraries and runtime environment.

The main program window contains three tabs: Program, Generator and Help. In appropriate fields of Program tab it is possible to select one of five basic problems (GPP, CPP, CPP-MIN, GCP, RGCP) and one of eleven algorithms. After that one can select the input file format and read input data. A numerous algorithm parameters and problem constrains must be filled in the forms including multiple runs, enabling statistics and write options. The cost of best solution and the total computation time are also displayed in this tab.

The Generator tab opens possibilities to generate input graphs or weighted input graphs after setting its parameters and lists of forbidden colors. The unweighted graphs are kept in .col format, weighted graphs are in .ecl format, which is extention of .col by adding edge weights as well as edge weight range (in the header). The type .rcp contains lists of forbidden colors for all vertices, if any. File formats .xpp and .xcp are used for preserving input graph and the partition being the best solution for the given problem together with its cost, respectively. Output data in CSV format are written to the .txt file and enable easy import of data into a spreadsheet.

V. Computational experiments

For experiments the Intel Pentium T2300 machine was used with two 1,66 GHz cores and 4GB RAM, running under Windows XP Pro SP2 and .NET Framework 3.5 platform.

All five problems were tested agaist all eleven algorithms with eight basic settings (stop criterion, no of iterations in a single step, initial temperature for SA, size of the tabu list). The specific setting that were selected in the initial phase of the experiment are shown in Table I.

TABLE I
Basic settings of algorithms

no.	stop criterion (it)	number of iterations/ step	SA - initial temperature	TS - size of tabu list
1	20	5	3	10
2	20	5	10	40
3	20	10	3	10
4	20	10	10	40
5	50	5	3	10
6	50	5	10	40
7	50	5	3	10
8	50	5	10	40

Other parameters are: coefficient of cost function = 1, no of parallel processes (if any) = 20, communication parameter = 20, no of algorithm repetitions = 20, no of clique extention trials = 5, no of repetitions for H-SP algorithm = 5.

In Tables II-XI computational data are presented. All experiments were conducted for random graph instances generated for each class of the graph partitioning problems in .ecl format. Relatively small graph instances were used with 20, 50 and 100 vertices and graph densities 10%, 20% and 30%. Cost functions from 20 trials are collected in Tables II-VI while the corresponding computation times in Tables VII-XI, respectively.

Average (Avg.) values for settings 1-8 in Tables II-XI are computed for parallel and hybrid methods only, serial methods SA i TS are excluded. Analysis of the results obtained for the five partitioning problems justifies several conclusions.

The shortest processing times are obtained by pure TS and SA methods. However, their solutions are not satisfactory. Parallelization and hybridization require additional computational work, and their aim is to improve search for a better suboptimal solution rather then providing significant speedup.

For GPP the fastest parallel algorithms are PTS metaheuristics. PSA and hybrid methods are less timeefficient. The slowest algorithm is H-SP, which is very time consuming. On the other hand H-SP finds the best solutions for all eight available settings. Average results of PSA-MIR and H-P algorithms are also outstanding and obtained approximately five times faster than by H-SP. The best setting in average is no 6 (minimum cost for six methods), but the best result for GPP is obtained with setting no 5. In terms of the computation time settings no 2 and 1 obviously win, and the fastest method is the PTS-A algorithm with moderate success in optimization.

For CPP the fastest parallel algorithms are PSA metaheuristics. Five other methods, except H-PS are also timeefficient. Among the parallel algorithm PSA-A is the fastest one with minimum time obtained for four settings. The slowest algorithm is again H-SP, which finds the best solutions for all eight parameter settings. The second result provides PTS-MIR which is eight times faster than H-SP. Setting no 8 provides the best solution quality for 7 parallel algorithms. In terms of the computation time settings no 2 and 1 win.

For CPP-MIN the fastest parallel algorithms is one hybrid and all PSA metaheuristics. The winner is PTS-MS algorithm with setting no 1. The slowest algorithm is H-SP, which wins the quality competition for all eight parameter settings. PSA-MIR and H-P have been the most prospective challengers. Setting no 8 provides the best solution quality for eight algorithms. In terms of the computation time settings no 2 and 1 are the winners.

For GCP the fastest parallel methods are H-PA and all PSAs which provide also best approximate solutions (PSA-MIR wins for six out of eight settings). The fastest parallel algorithm is H-PA, the best setting for six algorithms is no 2. The slowest algorithm is H-SP, which is 5th in terms of solution quality. The best setting for cost-optimality is no 4 in average.

The final problem - RCP - brings also interesting results. The fastest parallel algorithms are H-P six winning settings and PSAs. The best settings for all methods are 1 and 2. The best solution in average is found by PSA-MIR (the winner for seven out of eight settings), the runner-up is H-SP which was about eight time s lower, the next positions are occupied by PSA-MS and PSA-A. Most good results (8) were obtained for the setting no 8.

VI. Conclusions

In this paper some research results related to parallel metaheuristics and their applications were reported. The conducted experiments gave certain limited insight to computational behaviour of parallel metaheuristics developed on the basis of SA and TS, and applied to a class of popular partitioning problems in graphs. Some algorithms were better then others for solving particular problems. We were focused mostly on solution quality, but computation time was the second factor in comparison. Many results were not obvious and difficult to predict without verification. We believe that the presented initial results justify further experiments with our solver for more elaborated input instances. For this purpose to chose and modify DIMACS graph coloring instances, which were used for generation of instances of such partitioning problems like sum coloring, robust coloring etc. In time we will improve the algorithms to obtain more accurate solutions.

VII. Acknowledgements

The presented research was conducted within the frame of statutory activity (DS/2016) at Faculty of Electrical and Computer Engineering, Cracow University of Technology and was financially supported by Ministry of Science and Higher Education, Republic of Poland.

References

[1] G. Ausiello. et all. Complexity and Approximation - Combinatorial optimization problems and their approximability properties, Springer-Verlag, 1999.
[2] C. Blum, A. Roli, E. Alba. An Introduction to Metaheuristic Techniques. [in:] Parallel Metaheuristics, Wiley-Interscience, 3-42, 2005
[3] U. Brandes, M. Gaertler, D. Wagner Experiments on Graph Clustering Algorithms, Proc. ESA'2003, LNCS 2832, 568-579, 2003 DOI: 10.1007/978-3-540-39658-1-52
[4] C-Y. Byun. Lower Bound for Large-Scale Set Partitioning Problems, *ZIB-Report*, Vol. 12, 1822, 2001
[5] I. Charon, O. Hudry. Noising methods for a clique partitioning problem. *Discrete Applied Mathematics*, Vol. 154 , 754-769, 2006 DOI: 10.1016/j.dam.2005.05.029
[6] L. Coslovich, R. Pesenti, W. Ukovich. Large-Scale Set Partitioning Problems. *Journal on Computing*, Vol. 13, 191-209, 2001
[7] B. Crawford, C. Castro. ACO with Lookahead Procedures for Solving Set Partitioning Problems and Covering Problems. Chile, 2004
[8] R. Garey, D.S. Johnson. Computers and intractability. A guide to the theory of NP-completeness. Freeman, San Francisco, 1979
[9] X. Ji, J.E. Mitchell. The Clique Partition Problem with Minimum Clique Size Requirement, *Discrete Optimization*, Vol. 4, 87-102, 2007
[10] B.W. Kernighan, S. Lin. An Efficient Heuristics Procedure for Partitioning Graphs. *The Bell System Technical Journal*, Vol. 49, 291-307, 1970
[11] Z. Kokosiński, K. Kwarciany, M. Kołodziej. Efficient Graph Coloring with Parallel Genetic Algorithms. *Computing and Informatics*, Vol. 24, No. 2, 109-121, 2005
[12] M. Kubale(Ed.) Graph Colorings, American Mathematical Society, 2004
[13] M. Kubale. Some results concerning the complexity of restricted colorings of graphs. *Discrete Applied Mathematics*, Vol. 36, 35-46, 1992
[14] E. Mujuni, F. Rosamond. Parameterized Complexity of the Clique Partition Problem. *The Australasian Theory Symposium*, 75-78, 2008
[15] A. Nowosielski, P.A. Kowalski, P. Kulczycki. The column-oriented database partitioning optimization based on the natural computing algorithms Proc. FedCSIS, 1035-1041, 2015 DOI : 10.15439/2015F262
[16] S.M. Sait, H. Youssef. Iterative computer algorithms with applications in engineering. IEEE Computer Society, Los Alamitos, 1999
[17] W-D. Tseng, I-S. Hwang, L-J. Lee, C-Z. Yang. Clique-partitioning connections-scheduling with faulty switches in dilated Benes network, *Journal of the Chinese Institute of Engineers*, Vol. 32, 853-860, 2009
[18] S. Zhou. Minimum partition of an independence system into independent sets, *Discrete Optimization*, Vol. 6, 125-133, 2009 DOI: 10.1016/j.disopt.2008.10.001

TABLE II
Graph partitioning problem (GPP). Cost functions (11 algorithms, 8 settings, 20 runs)

	SA	**PSA**			TS	**PTS**			**Hybrid**			**Avg.**
		MIR	MS	A		MIR	MS	A	H-PA	H-SP	H-P	
1	295	238	255	251	338	284	285	287	286	230	240	262
2	296	237	254	253	330	286	285	282	246	228	234	256
3	293	243	257	259	338	283	285	288	254	234	238	260
4	293	238	257	257	331	282	285	283	252	232	235	260
5	292	234	249	247	341	276	286	288	245	**226**	237	254
6	286	235	245	244	338	274	284	281	238	227	235	**251**
7	289	242	252	256	332	280	286	286	249	238	242	259
8	289	240	252	252	329	271	280	283	252	235	239	256
Avg.	292	238	253	252	335	280	285	285	253	**231**	238	

TABLE III
Clique partitioning problem (CPP). Cost functions (11 algorithms, 8 settings, 20 runs)

	SA	**PSA**			TS	**PTS**			**Hybrid**			**Avg.**
		MIR	MS	A		MIR	MS	A	H-PA	H-SP	H-P	
1	26	24	24	24	25	23	24	24	24	**21**	24	23,6
2	26	24	24	24	25	23	24	24	24	**21**	24	23,6
3	23	22	22	22	24	22	23	23	23	**21**	22	22,2
4	24	22	22	22	24	22	23	23	23	**21**	22	22,2
5	25	24	24	24	24	22	23	23	23	**21**	24	23,1
6	26	24	24	24	24	22	23	23	23	**21**	24	23,1
7	23	22	22	22	23	22	22	22	22	**21**	22	**21,9**
8	24	22	22	22	23	22	22	22	22	**21**	22	**21,9**
Avg.	24,6	23	23	23	24	22,3	23	23	23	**21**	23	

TABLE IV
CPP with min. clique size (CPP-MIN). Cost functions x10^3 (11 algorithms, 8 settings, 20 runs)

	SA	**PSA**			TS	**PTS**			**Hybrid**			**Avg.**
		MIR	MS	A		MIR	MS	A	H-PA	H-SP	H-P	
1	114	107	109	108	147	127	136	134	134	102	106	118
2	115	108	109	109	145	127	136	134	111	102	107	116
3	112	105	106	105	138	123	129	130	108	100	105	112
4	113	106	106	106	137	124	129	129	108	101	106	113
5	113	106	106	106	140	114	131	130	109	102	106	112
6	114	106	106	107	140	114	130	130	109	102	106	112
7	113	105	105	105	134	114	127	127	108	**99,3**	105	111
8	112	106	105	105	133	113	126	126	107	**99,3**	105	**110**
Avg.	113	106	107	106	139	119	131	130	112	**101**	106	

TABLE V
Graph coloring problem (GCP). Cost functions (11 algorithms, 8 settings, 20 runs)

	SA	**PSA**			TS	**PTS**			**Hybrid**			**Avg.**
		MIR	MS	A		MIR	MS	A	H-PA	H-SP	H-P	
1	86	51	52	53	84	55	55	55	55	55	54	53,9
2	80	52	52	52	85	55	54	55	53	53	53	53,2
3	85	52	51	52	83	54	55	54	54	54	53	53,2
4	86	51	52	51	78	53	53	53	53	53	53	**52,4**
5	82	52	52	52	86	54	56	55	54	54	53	53,6
6	87	52	52	52	86	53	54	55	54	53	53	53,1
7	84	**50**	51	52	83	53	54	55	54	53	53	52,8
8	86	52	51	52	85	53	54	53	52	53	53	52,6
Avg.	84,5	**51,5**	51,6	52	83,8	53,8	54,4	54,4	53,6	53,5	53,1	

TABLE VI
RESTRICTED GCP (RGCP). COST FUNCTIONS (11 ALGORITHMS, 8 SETTINGS, 20 RUNS)

	SA	**PSA**			TS	**PTS**			**Hybrid**			**Avg.**
		MIR	MS	A		MIR	MS	A	H-PA	H-SP	H-P	
1	36	**26**	**26**	27	39	29	30	29	30	27	29	28,1
2	37	**26**	**26**	**26**	37	28	29	29	27	26	28	27,2
3	36	**26**	**26**	27	39	28	29	28	28	**26**	28	27,3
4	36	27	27	27	36	28	28	28	27	**26**	28	27,3
5	37	**26**	**26**	**26**	38	28	29	29	27	**26**	28	27,2
6	36	**26**	27	**26**	38	27	28	29	27	27	28	27,2
7	36	**26**	27	27	37	28	28	28	27	**26**	28	27,2
8	36	**26**	27	27	36	27	28	28	27	**26**	27	**27,0**
Avg.	36,3	**26,1**	26,5	26,6	37,5	27,9	28,6	28,5	27,5	26,3	28	

TABLE VII
GRAPH PARTITIONING PROBLEM (GPP). COMPUTATION TIMES (11 ALGORITHMS, 8 SETTINGS, 20 RUNS)

	SA	**PSA**			TS	**PTS**			**Hybrid**			**Avg.**
		MIR	MS	A		MIR	MS	A	H-PA	H-SP	H-P	
1	1,70	19,4	14,6	15,3	0,57	6,83	5,23	5,13	5,49	70,0	34,2	19,6
2	1,76	18,3	13,4	14,2	0,60	6,75	5,40	5,51	14,1	67,5	18,4	**18,2**
3	6,28	64,2	48,4	49,8	2,48	26,5	21,9	21,3	40,1	27,0	64,9	67,5
4	5,71	58,3	42,8	41,5	2,85	29,8	26,9	25,6	40,1	258	58,7	64,7
5	3,77	35,0	32,2	30,4	1,52	25,3	13,1	13,5	27,8	149	35,4	40,1
6	3,33	34,4	31,3	32,4	1,76	33,4	15,1	15,0	28,2	149	34,9	41,5
7	10,7	106	97,7	98,1	6,59	114	55,4	54,8	85,9	626	109	150
8	10,2	101	91,7	92,5	6,98	128	67,1	65,6	86,5	605	104	149
Avg.	5,44	54,8	46,5	46,8	2,92	46,3	26,3	**25,8**	41,0	274	57,5	

TABLE VIII
CLIQUE PARTITIONING PROBLEM (CPP). COMPUTATION TIMES (11 ALGORITHMS, 8 SETTINGS, 20 RUNS)

	SA	**PSA**			TS	**PTS**			**Hybrid**			**Avg.**
		MIR	MS	A		MIR	MS	A	H-PA	H-SP	H-P	
1	1,03	9,86	10,0	9,82	1,05	10,8	10,0	10,5	10,5	76,4	10,0	17,5
2	0,95	9,46	9,59	9,40	1,06	11,0	10,5	10,4	10,2	76,6	9,62	**17,4**
3	3,45	33,9	33,3	32,7	3,30	34,0	31,8	31,5	32,7	280	33,7	60,4
4	3,09	30,9	30,5	30,8	3,33	33,3	31,6	31,9	32,6	282	31,1	59,4
5	1,89	18,2	18,3	18,4	2,14	23,5	21,5	21,7	22,1	168	18,2	36,7
6	1,78	17,6	17,9	17,9	2,27	23,5	22,0	21,6	21,2	158	17,4	35,2
7	6,50	63,8	63,8	63,5	7,25	77,9	70,2	71,4	73,5	653	64,8	134
8	6,10	61,8	60,9	60,9	7,30	76,3	72,5	70,2	72,4	645	61,2	131
Avg.	3,10	30,7	30,5	**30,4**	3,46	36,3	33,8	33,7	34,4	292	30,8	

TABLE IX
CPP WITH MIN. CLIQUE SIZE (CPP-MIN). COMPUTATION TIMES (11 ALGORITHMS, 8 SETTINGS, 20 RUNS)

	SA	**PSA**			TS	**PTS**			**Hybrid**			**Avg.**
		MIR	MS	A		MIR	MS	A	H-PA	H-SP	H-P	
1	1,74	17,6	16,3	17,2	1,25	18,1	13,0	13,6	13,3	91,2	18,0	24,3
2	1,76	17,0	16,7	17,3	1,28	17,5	13,4	13,6	16,6	89,4	16,6	**24,2**
3	3,89	39,9	39,6	39,4	3,82	53,3	40,5	39,3	40,0	337	39,7	74,4
4	3,90	38,4	38,2	39,2	3,90	52,1	39,4	39,5	40,1	332	38,4	73,1
5	2,93	27,6	29,6	29,3	2,60	50,5	26,7	26,4	30,0	193	26,9	48,9
6	2,85	27,4	29,9	29,3	2,58	49,6	27,1	27,3	29,7	185	27,4	48,1
7	7,24	72,3	72,1	73,4	8,60	169	89,1	83,0	77,6	795	71,9	167
8	7,33	71,9	73,1	71,8	9,04	167	90,4	86,6	76,6	840	71,9	172
Avg.	3,96	39,0	39,4	39,6	41,4	72,2	42,4	41,2	40,5	358	**38,9**	

TABLE X
GRAPH COLORING PROBLEM (GCP). COMPUTATION TIMES (11 ALGORITHMS, 8 SETTINGS, 20 RUNS)

	SA	**PSA**			TS	**PTS**			**Hybrid**			**Avg.**
		MIR	MS	A		MIR	MS	A	H-PA	H-SP	H-P	
1	0,62	7,09	7,14	6,97	0,64	7,06	7,01	6,88	6,84	49,3	8,13	**11,8**
2	0,60	6,56	6,57	6,63	0,68	8,05	7,68	7,35	6,36	48,9	8,08	**11,8**
3	2,36	26,9	26,2	26,2	2,38	27,2	24,6	25,2	23,3	201	27,9	45,4
4	2,24	25,1	24,6	24,9	2,31	27,1	26,0	26,1	22,4	201	27,5	44,9
5	1,37	14,4	14,4	14,3	1,51	17,9	15,9	15,6	15,6	121	18,3	27,5
6	1,31	13,9	14,0	14,0	1,45	19,2	17,6	18,0	20,4	122	18,9	28,7
7	5,35	56,9	56,4	56,1	5,73	65,1	61,3	60,0	55,8	498	68,3	109
8	5,24	54,5	55,3	54,7	5,37	61,3	56,3	56,8	52,8	499	59,6	106
Avg.	2,39	25,7	25,6	25,5	2,51	29,1	27,1	27,0	**25,4**	218	29,6	

TABLE XI
RESTRICTED GCP (RGCP). COMPUTATION TIMES (11 ALGORITHMS, 8 SETTINGS, 20 RUNS)

	SA	**PSA**			TS	**PTS**			**Hybrid**			**Avg.**
		MIR	MS	A		MIR	MS	A	H-PA	H-SP	H-P	
1	0,77	7,73	7,58	7,92	0,66	7,58	6,56	6,61	6,59	49,0	7,35	**11,9**
2	0,72	7,15	7,25	7,07	0,71	7,80	7,35	7,24	6,20	49,1	8,01	**11,9**
3	2,78	28,7	28,9	29,4	2,51	27,1	25,7	25,2	23,1	199	28,1	46,2
4	2,64	26,9	26,7	26,7	2,57	28,7	26,9	26,2	22,9	199	28,9	45,9
5	1,51	14,9	15,2	15,3	1,58	18,3	15,6	16,1	14,1	121	18,4	27,7
6	1,45	14,3	14,5	14,6	1,68	19,2	17,5	16,5	14,6	121	19,2	28,0
7	5,84	57,9	58,6	58,7	6,26	72,3	63,6	62,7	55,0	497	72,4	111
8	5,61	56,4	56,1	56,7	5,79	66,0	63,6	61,9	55,3	496	66,6	109
Avg.	2,66	26,8	26,9	27,1	2,72	30,9	28,3	27,8	**24,7**	217	31,1	