Solving Graph Partitioning Problems with Parallel Metaheuristics

Chapter *in* Studies in Computational Intelligence · June 2018

DOI: 10.1007/978-3-319-59861-1_6

Zbigniew Kokosiński
Cracow University of Technology
45 PUBLICATIONS **229** CITATIONS

Chapter 1
Solving graph partitioning problems with parallel metaheuristics

Zbigniew Kokosiński and Marcin Bała

Abstract In this article 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 concerning metaheuristics' performance that result from the conducted experiments are derived. The article provides a case study in which partitioning numbers $\psi_k(G)$, $k \geq 2$, of DIMACS graph coloring instances are evaluated experimentally by means of H-SP metaheuristic which is found to be the most efficient in terms of solution quality.

Key words: simulated annealing, tabu search, parallel metaheuristic, hybrid metaheuristic, graph partitioning problem, graph partitioning number

1.1 Introduction

Computational optimization attracts 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], [10]. Parallel and hybrid metaheuristics are among the most promisssing methods to be developed in the nearest time [2], [20]. Many new algorithms have been already designed and compared with existing methodologies [7], [14], but there is still a room for significant progress in this area.

Zbigniew Kokosiński
Cracow University of Technology, Faculty of Electrical and Computer Engineering, ul. Warszawska 24, 31-155 Kraków, Poland, e-mail: zk@pk.edu.pl

In this article we focus on a class of partitioning problems that appear in many application areas like data clustering [3], column-oriented database partitioning optimization [19], 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], [11], [13], [21], [22].

The paper describes two computer experiments. The aim of the first one is determining of efficient metaheuristics for the given problem taking into account both quality of the solution (cost function) and the computation time. In many cases the tradeoff is not easy to find, similarly as the best algorithms' settings. However, from our research some general recommendations can be derived. In the second experiment a single algorithm with the best solution quality is used for finding solution of the Graph Partitioning Problem (GPP) for a class of graphs from DIMACS graph repository [8, 9]. Originally, they were used as instances of Graph Coloring Problem (GCP), which is known to be NP-complete. The obtained computational results provide additional characteristic of this collection of graphs, since the objective function used represents the cost of graph partitioning into exactly k clusters (partition blocks), $2 \leq k \leq 5$. The cost function minima found experimentally are the upper bounds for the partitioning numbers $\psi_k(G)$, which represent cost of the optimal clustering.

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 sketched. The design assumptions and features of the developed solver application are described in section 4. Testing methodology and initial experimental results are shown in section 5. The experimental evaluation of $\psi_k(G)$ for 18 DIMACS graphs is described in detail in section 6. The conclusions of the article point out the directions of future research in this area.

The presented research was conducted within the frame of statutory activity (grant No. E3/627/2016/DS) 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.

1.2 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 $|V| = n$, $|E| = m$.

1.2.1 Graph partitioning problem (GPP)

A partition $C = (C_1, \ldots, C_k)$ of V is called a partitioning (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 of 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 general, for k-clustering problems in weighted graphs the total weight of the set $E \setminus E(C)$ shall be minimized.

Unweighted graph instances G, like DIMACS graphs investigated in section 6, can be characterized by the partitioning number $\psi_k(G)$ which equals to minimum $M(C)$.

1.2.2 Clique partitioning problem (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 [18]. 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.

1.2.3 Clique partitioning problem 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 [11].

1.2.4 Graph coloring problem (GCP)

Classical vertex coloring problem in a graphs is another formulation of graph partitioning into independent sets (ISs). 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 $c : V \longrightarrow \{1,\dots,k\}$, 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 [10].

1.2.5 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 [16].

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 [17].

1.3 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 [20]. Parallel algorithms can be splitted into three categories: parallel metaheuristics derived from SA, parallel metaheuristics derived from TS and hybrid methods.

1.3.1 Simulated annealing (SA)

Classical simulated annealing [20] 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.

1.3.2 Tabu search (TS)

Tabu search [20] 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.

1.3.3 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.

1.3.4 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.

1.3.5 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.

1.3.6 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 parallel metaheuristic 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 they 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 alternately 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 worse solution decreases in time.

1.4 The solver

For all tests the ”Partitioning problems solver” application was used. It was written in C++ (Visual Studio), while .NET Framework 3.5 provided 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 into the .txt file and enable easy import of data to a spreadsheet.

1.5 Computational experiments with metaheuristics

For experiments 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 1.1.

Table 1.1 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 (threads) = 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 1.2-1.11 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 1.2-1.6 while the corresponding computation times in Tables 1.7-1.11, respectively.

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

Table 1.2 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	271,7
2	296	237	254	253	330	286	285	282	246	228	234	266,5
3	293	243	257	259	338	283	285	288	254	234	238	270,2
4	293	238	257	257	331	282	285	283	252	232	235	267,7
5	292	234	249	247	341	276	286	288	245	226	237	265,5
6	286	235	245	244	338	274	284	281	238	227	235	262,5
7	289	242	252	256	332	280	286	286	249	238	242	268,4
8	289	240	252	252	329	271	280	283	252	235	239	265,6
Avg.	291,6	238,4	252,6	252,4	334,6	279,5	284,5	284,8	252,8	**231,2**	237,5	

Table 1.3 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	23,9
2	26	24	24	24	25	23	24	24	24	21	24	23,9
3	23	22	22	22	24	22	23	23	23	21	22	22,5
4	24	22	22	22	24	22	23	23	23	21	22	22,5
5	25	24	24	24	24	22	23	23	23	21	24	23,4
6	26	24	24	24	24	22	23	23	23	21	24	23,5
7	23	22	22	22	23	22	22	22	22	21	22	22,1
8	24	22	22	22	23	22	22	22	22	21	22	22,2
Avg.	24.6	23	23	23	24	22,3	23	23	23	**21**	23	

Table 1.4 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	108	108	147	126	135	134	134	101	106	120
2	115	107	108	108	144	127	135	134	110	101	106	118
3	111	105	105	105	137	122	129	129	108	100	104	114
4	112	105	105	105	137	123	129	128	107	100	105	114
5	112	105	106	106	140	114	131	130	108	101	106	114
6	113	106	106	106	139	114	130	130	108	101	106	114
7	112	105	105	105	133	113	126	126	107	99	105	112
8	112	104	105	105	132	112	126	126	107	99	104	112
Avg.	113	105	106	106	139	119	130	130	111	**100**	105	

additional computational work, and their aim is to improve search for a better sub-optimal solution rather then providing significant speedup.

For GPP the fastest parallel algorithms is P-TS metaheuristic. PSA and two hybrid methods (H-PA, H-P) are less time-efficient. 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 SA-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

Table 1.5 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	59,5
2	80	52	52	52	85	55	54	55	53	53	53	58,5
3	85	52	51	52	83	54	55	54	54	54	53	58,8
4	86	51	52	51	78	53	53	53	53	53	53	57,8
5	82	52	52	52	86	54	56	55	54	54	53	59,1
6	87	52	52	52	86	53	54	55	54	53	53	59,2
7	84	50	51	52	83	53	54	55	54	53	53	58,4
8	86	52	51	52	85	53	54	53	52	53	53	58,5
Avg.	84,5	**51,5**	51,6	52	83,8	53,8	54,4	54,4	53,6	53,5	53,1	

Table 1.6 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	29,8
2	37	26	26	26	37	28	29	29	27	26	28	29
3	36	26	26	27	39	28	29	28	28	26	28	29,2
4	36	27	27	27	36	28	28	28	27	26	28	28,9
5	37	26	26	26	38	28	29	29	27	26	28	29,1
6	36	26	27	26	38	27	28	29	27	27	28	29
7	36	26	27	27	37	28	28	28	27	26	28	28,9
8	36	26	27	27	36	27	28	28	27	26	27	28,6
Avg.	36,3	**26,1**	26,5	26,6	37,5	27,9	28,6	28,5	27,5	26,3	28	

Table 1.7 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	1,81
2	1,76	18,3	13,4	14,2	0,60	6,75	5,40	5,51	14,1	67,5	18,4	1,70
3	6,28	64,2	48,4	50,0	2,48	26,5	21,9	21,3	40,1	27,0	64,9	6,32
4	5,71	58,3	42,8	41,5	2,85	29,8	26,9	25,6	40,1	258	58,7	6,07
5	3,77	35,0	32,2	30,4	1,5	25,3	13,1	13,5	27,8	149	35,4	3,76
6	3,33	34,4	31,3	32,4	1,76	33,4	15,1	15,0	28,2	149	34,9	3,86
7	10,7	106	97,7	98,1	6,59	114	55,4	54,8	85,9	626	109	13,8
8	10,2	101	91,7	92,5	6,98	128	67,1	65,6	86,5	605	104	13,8
Avg.	5,4	548	46,5	46,8	2,92	463	26,3	25,8	41,0	274	57,5	

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 TS-PA algorithm with moderate success in optimization.

For CPP the fastest parallel algorithms are PSA metaheuristics. Two hybrid methods (H-PA, H-P) are also timeefficient. Among the parallel algorithm SA-PA is the fastest one with minimum 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 TS-MIR which is eight times faster than H-SP. Setting no 7 provides

Table 1.8 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	1,62
2	0,95	9,46	9,59	9,40	1,06	11,0	10,5	10,4	10,2	76,6	9,62	1,61
3	3,45	33,9	33,3	32,7	3,30	34,0	31,8	31,5	32,7	280	33,7	5,56
4	3,09	30,9	30,5	30,8	3,33	33,3	31,6	31,9	32,6	282	31,1	5,45
5	1,89	18,2	18,3	18,4	2,14	23,5	21,5	21,7	22,1	168	18,2	3,37
6	1,78	17,6	17,9	17,9	2,27	23,5	22,0	21,6	21,2	158	17,4	3,25
7	6,50	63,8	63,8	63,5	7,25	77,9	70,2	71,4	73,5	653	64,8	12,2
8	6,10	61,8	60,9	60,9	7,30	76,3	72,5	70,2	72,4	645	61,2	12,0
Avg.	3,10	30,7	30,5	30,4	3,46	36,3	33,8	33,7	34,4	292	30,8	

Table 1.9 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	20,1
2	1,76	17,0	16,7	17,3	1,28	17,5	13,4	13,6	16,6	89,4	16,6	20,1
3	3,89	39,9	39,6	39,4	3,82	53,3	40,5	39,3	40,0	337	39,7	61,6
4	3,90	38,4	38,2	39,2	3,90	52,1	39,4	39,5	40,1	332	38,4	60,5
5	2,93	27,6	29,6	29,3	2,60	50,5	26,7	26,4	30,0	193	26,9	40,5
6	2,85	27,4	29,9	29,3	2,58	49,6	27,1	27,3	29,7	185	27,4	39,9
7	7,24	72,3	72,1	73,4	8,60	169	89,1	83,0	77,6	795	71,9	13,8
8	7,33	71,9	73,1	71,8	9,04	167	90,4	86,6	76,6	840	71,9	14,2
Avg.	3,96	39,0	39,4	39,6	41,4	72,2	42,4	41,2	40,5	358	38,9	

Table 1.10 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	9,79
2	0,60	6,56	6,57	6,63	0,68	8,05	7,68	7,35	6,36	48,9	8,08	9.77
3	2,36	26,9	26,2	26,2	2,38	27,2	24,6	25,2	23,3	201	27,9	37,6
4	2,24	25,1	24,6	24,9	2,31	27,1	26,0	26,1	22,4	201	27,5	37,2
5	1,37	14,4	14,4	14,3	1,51	17,9	15,9	15,6	15,6	121	18,3	22,8
6	1,31	13,9	14,0	14,0	1,45	19,2	17,6	18,0	20,4	122	18,9	23,7
7	5,35	56,9	56,4	56,1	5,73	65,1	61,3	60,0	55,8	498	68,3	90,0
8	5,24	54,5	55,3	54,7	5,37	61,3	56,3	56,8	52,8	499	59,6	87,4
Avg.	2,39	25,7	25,6	25,5	2,51	29,1	27,1	27,0	25,4	218	29,6	

the best solution quality for all algorithms. In terms of the computation time settings no 2 and 1 win, and the fastest method is the SA-PA algorithm.

For CPP-MIN the fastest parallel algorithms are hybrid and PSA metaheuristics. The winner is H-P algorithm with setting no 2. The slowest algorithm is H-SP, which wins the quality competition for all eight parameter settings. SA-MIR and H-P has been the most prospective challengers. Setting no 8 provides the best solution

Table 1.11 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	1,10
2	0,72	7,15	7,25	7,07	0,71	7,80	7,35	7,24	6,20	49,1	8,01	1,10
3	2,78	28,7	2,89	29,4	2,51	27,1	25,7	25,2	23,1	199	28,1	4,26
4	2,64	26,9	26,7	26,7	2,57	28,7	26,9	26,2	22,9	199	28,9	4,23
5	1,51	14,9	15,2	15,3	1,58	18,3	15,6	16,1	14,1	121	18,4	2,54
6	1,45	14,3	14,5	14,6	1,68	19,2	17,5	16,5	14,6	121	19,2	2,57
7	5,84	57,9	58,6	58,7	6,26	72,3	63,6	62,7	55,0	497	72,4	10,2
8	5,61	56,4	56,1	56,7	5,79	66,0	63,6	61,9	55,3	496	66,6	9,93
Avg.	2,66	26,8	26,9	27,1	2,72	30,9	28,3	27,8	24,7	217	31,1	

quality for nine algorithms. In terms of the computation time settings no 1 and 2 are the winners.

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

The last problem - RGCP - brings also interesting results. The fastest parallel algorithms are PSAs, but the winner in this category is H-PA with seven winning settings. 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 slower, the next positions are occupied by PSA-MS and PSA-A. Most good results (9) were obtained for the setting no 8.

1.6 Graph Partitioning Problem - a case study

Graphs are often characterized by their combinatorial properties. For instance, in The second DIMACS Implementation Challenge: 1992-1993, chromatic numbers $\chi(G)$ were searched for a collection of hard to color graph instances by means of virtually all known computational techniques. Gradually, most graph instances from this repository were assigned these distinctive numbers. The first parallel metaheuristic used for searching chromatic numbers $\chi(G)$ was Parallel Evolutionary Algorithm (PEA) [14]. Another example of graph characteristics are graph chromatic sum $\sum(G)$ and graph chromatic sum number $s(G)$ [16]. In the research on sum coloring PEA as well as many new computational methods were highly successful [15, 12].

We believe that DIMACS graphs could be further characterized with respect to other graph partitioning problems. In this context GPP seems to be one of the the most important and promising candidates. In the next experiment we will investigate 18 DIMACS graphs in search for their clustering numbers $\psi_k(G)$, for different number of clusters $k \geq 2$. For our computations hybrid serial–parallel algorithm (H-SP) has been selected due to its efficiency in solving majority of partitioning problems. The time efficiency is not a priority, but the number of algorithm's runs shall be reasonably restricted.

The algorithm H-SP uses p threads, each with a pair of modified SA and TS algorithms. SA section creates a *tabu list* for TS on exit while TS section updates *temperature* for SA on exit. Starting from SA section, both H-SP components work by turns improving the current and the best solutions, respectively. Alternate runs are continued till the stop criterion is reached. Details of SA and TS algorithms are omitted here.

The pseudocode of the H-SP algorithm used in our experiment is as follows.

```
H-SP(input: p,stop_criterion,iter_number,initial_temp,
list_size, alternat_coeff; output: best_sol,cost(best_sol));
best_sol(1):= random_vertex_partition(1)
best_sol:= best_sol(1)
cost_best_sol:=cost(best_sol(1))
for j=1 to p do in parallel
current_sol(j):= random_vertex_partition(j)
best_sol(j):=current_sol(j)
tabu_list(j):=empty
T(j):=initial_temp
repeat
  alternat_counter:=0
  repeat
    for iter_counter=1 to iter_number do
      SA(T(j),temp_coeff,current_sol(j),best_sol(j));
      if (cost(best_sol(j)) < cost_best_sol)
        then best_sol:=best_sol(j)
             cost_best_sol:=cost(best_sol(j))
             iter_counter:=0
        else Inc(alternat_counter)
  until (alternat_counter > alternat_coeff)
  update(tabu_list(j))
  alternat_counter:=0
  repeat
    for iter_counter=1 to iter_number do
      TS(tabu_list(j),list_size,current_sol(j),best_sol(j));
      if (cost(best_sol(j)) < cost_best_sol)
        then best_sol:=best_sol(j)
             cost_best_sol:=cost(best_sol(j))
      iter_counter:=0
        else Inc(alternat_counter)
  until (alternat_counter > alternat_coeff)
  update(T(j))
  Inc(main_counter)
until (main_counter = stop_criterion)
```

For H-SP algorithm the setting no. 2 from Table 1 was chosen what is justified by results of the reported testing. Other program settings for H-SP were as follows: *no. of repetitions*=10, *no. of parallel processes*=20, *no. of algorithm's runs*= 10, 50.

Table 1.12 Evaluation of $\psi_k(G)$ for GPP (DIMACS graphs, algorithm H-SP, *no. of runs*=10,50)

Graph	No of blocks	10 runs			50 runs		
		best cost	iterations best/total	best run (s)	best cost	iterations best/total	best run (s)
anna	2	**139**	28056/75840	1.703	142	85440/133200	1.703
	3	151	56232/104160	2,343	**148**	104448/152400	2.828
	4	158	62832/110640	2,421	**150**	98088/145920	3.185
	5	157	99048/146880	3,156	**155**	87840/135600	3.015
david	2	141	39696/231360	2,468	**132**	80352/272160	2,703
	3	142	251712/443520	4,390	**136**	65976/257760	2,546
	4	148	205152/396960	3,937	**147**	55368/247200	2,468
	5	149	184776/376320	4,031	**147**	104976/296640	3,093
homer	2	**445**	576456/768000	278,6	n.a.	n.a.	n.a.
	3	**420**	356232/548160	187,2	n.a.	n.a.	n.a.
	4	**434**	439152/630720	215,9	n.a.	n.a.	n.a.
	5	**444**	453888/645600	219,7	n.a.	n.a.	n.a.
huck	2	61	56328/248160	1,765	**59**	101832/293760	2,078
	3	64	17232/208800	1,468	**63**	23880/215520	1,578
	4	69	139176/330720	2,625	**67**	193632/385440	2,765
	5	**72**	339000/530880	3,937	**72**	31440/223200	1,546
jean	2	52	188664/380640	3,125	**51**	38136/229920	2,031
	3	**53**	176856/368640	3,035	54	113400/305280	2,734
	4	60	240264/432000	3,890	**59**	50976/242880	2,187
	5	62	48096/230400	2,031	**61**	49320/240960	2,140
games120	2	133	65064/256800	4,640	**128**	48600/240480	4,641
	3	128	461424/653280	12,04	**123**	86256/277920	5,359
	4	**127**	282792/474720	8,718	136	301992/493920	9,594
	5	149	47592/239520	4,359	**138**	218184/409920	7,968
miles250	2	33	314592/506400	9,484	**29**	290568/482400	9,797
	3	34	59064/251040	4,859	**30**	307944/499680	9,859
	4	**35**	77136/268800	5,000	37	73968/265920	5,219
	5	**43**	92400/284160	5,250	46	75048/266880	5,770
miles500	2	159	89256/280800	6,906	**155**	39072/230880	5,826
	3	160	84504/276480	6,468	**156**	54696/246240	6,209
	4	171	256104/447840	10,93	**163**	128904/320640	8,125
	5	168	63960/255840	6,140	**167**	48792/240480	6,015
miles750	2	534	73776/265440	9,328	**524**	469200/660960	23,56
	3	529	278544/470400	16,82	**475**	101064/292800	10,56
	4	534	89040/280800	9,578	**530**	614904/806880	28,28
	5	539	132648/324480	11,59	**535**	338136/529920	18,26

For this experiment the machine with Intel Core i7 4700MQ CPU was used with four 2,4 GHz cores and 8 GB RAM, running under Windows 10 Home OS.

The results of computations are shown in Tables 1.12-1.13. In Table 1.12 nine DIMACS graphs are gathered (5 book graphs, *games120* and 3 *miles* graphs). Due to excessive time *homer* graph was processed only with 10 algorithm's runs. Table 1.13 contains nine *queen* graphs. All computed upper bounds for $\psi_k(G)$, $2 \leq k \leq 5$, are distinguished in a bold font. Let us notice, that in several cases the bounds obtained with 10 algorithm's runs are better than with 50 runs. The presented computational

Table 1.13 Evaluation of $\psi_k(G)$ for GPP (DIMACS graphs, algorithm H-SP, *no. of runs*=10,50)

Graph	No of blocks	10 runs			50 runs		
		best cost	iteraton best/total	best run (s)	best cost	iteraton best/total	best run (s)
queen5.5	2	81	34080/225600	0,531	**74**	25704/217440	0,375
	3	**80**	41640/233280	0,546	**80**	31272/223200	0,375
	4	102	10512/202080	0,484	**90**	42456/234240	0,390
	5	102	27240/218880	0,515	**92**	4536/196320	0,343
queen6.6	2	141	15096/206880	0,703	**137**	54360/246240	0,843
	3	155	20496/212160	0,734	**149**	20736/212640	0,578
	4	158	106032/297600	1,062	**148**	71328/263040	0,718
	5	164	36264/228000	0,609	**157**	104976/296640	0,999
queen7.7	2	249	32016/223680	1,078	**238**	32256/224160	0,937
	3	247	35232/227040	1,093	**237**	164400/356160	1,484
	3	251	278160/469920	2,328	**244**	29088/220800	1,078
	4	253	22104/214080	0,890	**244**	16488/208320	1,046
queen8.8	2	373	88176/279840	1,984	**338**	125568/317280	2,218
	3	377	45048/236640	1,640	**360**	27360/218880	1,546
	4	369	296928/488640	3,531	**353**	102648/294240	2,062
	5	384	273576/465120	3,281	**376**	354576/546240	3,453
queen8.12	2	665	263808/455520	5,968	**664**	90360/282240	4,359
	3	**667**	130656/322560	5,164	670	219768/411360	6,562
	4	**670**	180264/372000	5,945	674	379728/571680	9,156
	5	687	103008/294720	4,709	**678**	47280/239040	3,781
queen9.9	2	532	339816/531360	5,562	**521**	62832/254400	2,859
	3	**498**	44280/236160	2,421	531	53304/245280	2,734
	4	540	217584/409440	4,250	**537**	60216/252000	2,609
	5	**533**	229440/420960	4,812	**533**	63192/254880	2,671
queen10.10	2	**725**	64512/256320	3,562	**725**	239352/431040	6,062
	3	735	51552/243360	3,718	**726**	39816/231360	3,265
	4	**730**	141360/333120	4,986	**730**	606768/798720	11,34
	5	738	139296/331200	4,671	**734**	48384/240000	3,531
queen11.11	2	974	137496/329280	6,984	**962**	244488/436320	9,437
	3	970	58272/250080	5,460	**967**	78384/270240	6,031
	4	**971**	105000/296640	6,459	**971**	126456/318240	6,750
	5	981	56808/248640	5,577	**974**	59328/251040	5,344
queen12.12	2	1254	258216/354000	11,36	**1242**	125544/221520	6,750
	3	1249	37416/133200	4,296	**1241**	264984/360960	11,12
	4	1247	187992/283920	8,750	**1236**	335544/431520	13,71
	5	1270	31128/126960	3,828	**1250**	233064/329040	10,09

results should be considered as the first attempt in evaluation of numbers $\psi_k(G)$ for DIMACS graph coloring benchmarks.

1.7 Conclusions

In this article some research results related to parallel metaheuristics and their applications were reported. The conducted experiments gave certain limited insight to computational behaviour of parallel and hybrid metaheuristics developed on the basis of SA and TS algorithms, and applied to a class of popular partitioning problems in graphs. Some algorithms were better than others for solving particular problems. We were focused mostly on solution quality, the computation time was the secondary factor in our comparison. Many obtained results were not obvious and difficult to predict without experimental verification.

An original contribution of our research is evaluation of graph partitioning numbers $\psi_k(G)$, $2 \leq k \leq 5$, for 18 DIMACS graph coloring instences, which were so far characterized by chromatic number $\chi(G)$, chromatic sum $\sum(G)$ and graph chromatic sum number $s(G)$. Now, they have got also 72 experimentally computed upper bounds for $\psi_k(G)$.

We believe that the presented results justify further experiments with our solver. It would be interesting to extend our second experiment and include PSA-MIR algorithm which also provides quality results in a reasonable computation time. Another research direction is to continue evaluation of graph partitioning numbers for the remaining DIMACS graphs.

References

1. Ausiello G, et all.: Complexity and Approximation — Combinatorial optimization problems and their approximability properties. Springer-Verlag (1999)
2. Blum, C., Roli, A., Alba E.: An Introduction to Metaheuristic Techniques. In: Alba E. at all. (eds.) Parallel Metaheuristics, pp. 3-42. Wiley-Interscience (2005)
3. Brandes, U., Gaertler, M., Wagner, D.: Experiments on Graph Clustering Algorithms. Lect. Notes Comp. Sci. (2003) doi: 10.1007/978-3-540-39658-1-52
4. Byun, C.-Y.: Lower Bound for Large-Scale Set Partitioning Problems. ZIB-Report **12**, 18-22 (2001)
5. Charon, I., Hudry O.: Noising methods for a clique partitioning problem. Discrete Applied Mathematics (2006) doi: 10.1016/j.dam.2005.05.029
6. Coslovich, L., Pesenti, R., Ukovich, W.: Large-Scale Set Partitioning Problems. Journal on Computing **13**, 191–209 (2001)
7. Crawford, B., Castro, C.: ACO with Lookahead Procedures for Solving Set Partitioning Problems and Covering Problems. Chile (2004)
8. COLOR web site. http://mat.gsia.cmu.edu/COLOR/instances.html
9. DIMACS ftp site. ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/
10. Garey, R., Johnson, D.S.: Computers and intractability. A guide to the theory of NP-completeness. Freeman, San Francisco (1979)
11. Ji, X., Mitchell, J.E.: The Clique Partition Problem with Minimum Clique Size Requirement. Discrete Optimization **4**, 87–102 (2007)
12. Jin, Y., Hamiez, J.-P., Hao, J.-K.: Algorithms for the minimum sum coloring problem : a review. Artif. Intell. Rev. **47**, doi: 10.1007/s10462-016-9485-7
13. Kernighan, B.W., Lin, S.: An Efficient Heuristics Procedure for Partitioning Graphs. The Bell System Technical Journal **49**, 291–307 (1970)
14. Kokosiński, Z., Kwarciany, K., Kołodziej, M.: Efficient Graph Coloring with Parallel Genetic Algorithms. Computing and Informatics **24**, 109–121 (2005)
15. Kokosiński, Z., Kwarciany, K.: On sum coloring of graphs with parallel genetic algorithms. Lect. Notes Comp. Sci. (2007) doi: 10.1007/978-3-540-71618-1_24
16. Kubale, M. (Ed.): Graph Colorings. American Mathematical Society (2004)

17. Kubale, M.: Some results concerning the complexity of restricted colorings of graphs. Discrete Applied Mathematics **36**, 35–46 (1992)
18. Mujuni, E., Rosamond, F.: Parameterized Complexity of the Clique Partition Problem. The Australasian Theory Symposium, 75–78 (2008)
19. Nowosielski, A., Kowalski, P.A., Kulczycki, P.: The column-oriented database partitioning optimization based on the natural computing algorithms. Proc. FedCSIS (2015) doi: 10.15439/2015F262
20. Sait, S.M., Youssef, H.: Iterative computer algorithms with applications in engineering. IEEE Computer Society, Los Alamitos (1999)
21. Tseng, W.-D., Hwang, I.-S., Lee, L.-J., Yang, C.-Z.: Clique-partitioning connections-scheduling with faulty switches in dilated Benes network. Journal of the Chinese Institute of Engineers **32**, 853–860 (2009)
22. Zhou, S.: Minimum partition of an independence system into independent sets. Discrete Optimization (2009) doi: 10.1016/j.disopt.2008.10.001