Application of Parallel and Hybrid Metaheuristics for Graph Partitioning Problem: 9th International Conference, NMA 2018, Borovets, Bulgaria, August 20–24, 2018, Revised Selected P...

Chapter · January 2019

DOI: 10.1007/978-3-030-10692-8_14

2 authors, including:

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

SEE PROFILE

Application of parallel and hybrid metaheuristics for graph partitioning problem

Zbigniew Kokosiński and Marcin Pijanowski

Faculty of Electrical & Computer Eng., Cracow University of Technology,
ul. Warszawska 24, 31-155 Kraków, Poland
zk@pk.edu.pl

Abstract. In this paper parallel and hybrid metaheuristics for graph partitioning are compared taking into account their efficiency in terms of a cost function and computation time. Seventeen methods developed on the basis of evolutionary algorithm, simulated annealing and tabu search are implemented and tested against graph instances computed on the basis of queen graphs from DIMACS repository and a class of random R–MAT graphs. These graphs are supposed to model a class of digital circuits being subject of decomposition into a given number of modules. In partitioning process several additional constraints have to be satisfied in order to enable composition of original circuits from subcircuits by means of VLSI/FPGA modules.

Keywords: graph partitioning, circuit partitioning, parallel metaheuristics, hybrid metaheuristics, approximate algorithms, DIMACS graphs, R–MAT graphs

1 Introduction

The graph partitioning problem for an undirected graph $G = (V, E)$ is a division of V into k pairwise disjoint subsets (partitions) such that all partitions are of approximately equal size and the edge–cut, i.e., the total number of edges having their incident nodes in different subdomains, is minimized.

The partitioning of a digital circuit arises when the technology used for its implementation imposes constraints related to the circuit size, available chip resources (macro cells, interconnections), I/O pins, clock distribution, energy dissipation etc. In real design problems a straightforward approach is to decompose the original circuit into a number of sub–circuits (partition blocks) satisfying certain design requirements [4, 9, 12, 15]. Usually the expected number of blocks is known and the number of interconnections between blocks have to be minimized. In some cases the implementation of a circuit in many heterogenious FPGA can cause problems not only with splitting the original circuit but also with partitioning and modifying the existing test benchmarks [2, 7].

Digital circuits are usually modeled by graphs and many design problems can be performed by methods developed in graph theory. However, the task

of circuit partitioning falls into category of graph clustering/partitioning problems which are known to be intractable, i.e. they belong to the class of NP-hard problems. Therefore, the search for an optimal solution is not efficient and heuristics/metaheuristics providing an approximate solution are to be applied [10, 11].

Graph partitioning and clustering was the topic of the 10th DIMACS Implementation Challenge [8] and included both theoretical and real world problems. The Chalange goals were : identifying a standard set of benchmark instances and generators, establishing the most appropriate problem formulations and objective functions for a variety of applications, comparison of present methods in hopes of identifying the most effective algorithmic innovations that have been proposed.

In the article seventeen approximate methods developed on the basis of evolutionary algorithm (EA), simulated annealing (SA) and tabu search (TS) are investigated. Independently, in [4], a mixed SA and TS algorithm was successfully applied for topological partitioning in a parallel test-pattern generator.

The resulting computer application was used for testing these metaheuristics against a set of constructed problem instances and compared taking into account their efficiency in terms of cost function and computation time. Realistic modeling of benchmarks was the key issue in the conducted research. For the first time a family of modified DIMACS *queen* graphs [6] as well as recursively defined random R–MAT graphs [5] with various parameters were applied for testing. These graphs are supposed to adequately model a class of digital circuits being subject of decomposition into a given number of modules. Approximate solving of graph partitioning problem with a realistic cost function being minimized enables efficient decomposition of an original digital circuit by means of VLSI/FPGA modules.

2 Graph Partitioning

Graph Partitioning Problem (GPP) relies on clustering of graph $G(V, E)$ vertices into partition blocks (clusters). Let us intruduce the notation used throughout the paper.

A partition $C = \{C_1, C_2, \ldots, C_k\}$ of V is called clustering of G and C_i, $1 \leq i \leq k$, are called clusters. C is called trivial if either $k = 1$, or all clusters C_i contain only one element. We will indentify cluster C_i with the induced subgraph G_i of G, i.e. $G_i = (C_i, E(C_i))$, 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 $E \backslash E(C)$ — the set of inter–cluster edges. The graph density is denoted by $d(G) = 2|E|/|V|(|V| - 1)$.

The purpose of the optimization problem is to find a clustering of G into k clusters providing that the edge–cut $ext = |E \backslash E(C)|$ is minimal. Sometime there are additional constraints on the maximum number of inter–cluster edges E_i coming from a single cluster C_i. In some cases clustering is expected to be equitable in the sense of either the number of $|V_i|$ or $|E_i|$.

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

2.1 Definition of the cost function

According to the design requirements solution quality is defined by the following cost function (subject of minimization):

$$f = a \cdot ext + b \cdot bn + c \cdot |bnp|^3 + d \cdot bsp^2 + e \cdot pp^2, \tag{1}$$

where:
a, b, c, d – are integer coefficients;
ext – is the sum if inter–cluster edges (the edge–cut);
bn – is the assumed number of clusters called *block number*;
bnp – is the difference between bn and the actual number of blocks called *block number penalty*;
bsp – is the difference between the assumed block size bs and the actual block size called *block size penalty*;
pp – is the difference between the assumed number of block' I/O pins pn and the actual number of block' I/O pins called *pin penalty*.

Exponents of bp and pp were determined experimentally and reflect the relative importance of the corresponding terms of cost function f. In any solution satisfying the design assumptions bp, bsp and pp should be zeroed.

Terms of the cost function f provide the designer additional informations about solution quality. When design constraints are not fully satisfied this information helps the designer to choose a right way to complete the design process.

2.2 Test instances

Graph coloring instances [6] were originally designed and collected in DIMACS repository for the purpose of testing and comparing graph coloring algorithms. In graph coloring problem (GCP) partition blocks are assumed to be independent sets (ISs). For most DIMACS graphs their chromatic numbers are already known. In the complementary problem, i.e. partitioning into cliques (PIC), partition blocks are cliques. In both problems the number of partition blocks k is minimized. For the given graph $G(V, E)$ and its complementary graph $G'(V, E) = G(V, E')$ solutions for GCP and PIC, respectively, are equivalent with minimum $\chi(G) = k$.

PIC problem resambles GPP (circuit partitioning) in which intra–connection density in any partition block is maximal. The PIC problem instance G' is obtained from the instance G of GCP. Under assuption $k \in \{\chi(G) - 1, \chi(G) - 2\}$, the partition into k cliques is not existing, making the corresponding GPP even harder to solve.

The queen graph G_q of size $n \times n$ has the squares of two-dimentional chessboard for its vertices and two such vertices are adjacent if, and only if, queens placed on the two squares attack each other. For interesting introduction to queen graph coloring one may refere to Chvátal's article [3]. This class of graphs was chosen for generation of our test instances.

Basic DIMACS graphs selected for construction of test instances were queen graphs [6]:
1. *queen6.6*, $|V|$ =36, $|E|$ =290, $\chi(G)$ =7, $d(G)$ = 0,460 ;
2. *queen7.7*, $|V|$ =49, $|E|$ =475, $\chi(G)$ =7, $d(G)$ = 0,404 ;
3. *queen8.8*, $|V|$ =64, $|E|$ =728, $\chi(G)$ =9, $d(G)$ = 0,361 ;
4. *queen9.9*, $|V|$ =81, $|E|$ =1056, $\chi(G)$ =9, $d(G)$ = 0,326 ;
5. *queen10.10*, $|V|$ =100, $|E|$ =1470, $\chi(G)$ =11, $d(G)$ = 0,297.

The actual test instances are the corresponding complementary graphs G', that have the following characteristics:
Q1: *queen6.6* $'$, $|V|$ =36, $|E'|$ =340, $d(G)$ = 0,640, k =5, bs =8, pn =110 ;
Q2: *queen7.7* $'$, $|V|$ =49, $|E'|$ =701, $d(G)$ = 0,596, k =6, bs =9; pn =200 ;
Q3: *queen8.8* $'$, $|V|$ =64, $|E'|$ =1288, $d(G)$ = 0,639, k =7, bs =10; pn =350 ;
Q4: *queen9.9* $'$, $|V|$ =81, $|E'|$ =2184, $d(G)$ = 0,674, k =8, bs =11; pn =520 ;
Q5: *queen10.10* $'$, $|V|$ =100, $|E'|$ =3480, $d(G)$ = 0,703, k =9, bs =12; pn =800.

In addition a number of random R–MAT graphs was used in the experimental part of the paper [5]. R–MAT graphs with $|V| = 2^b$ are generated recursively with the required density in $(b-1$ steps. Initially, adjacency matrix is zeroed. Then the generation algorithm determines the position of a consecutive "1" by random choosing the input matrix (submatrix) quater (NW, NE, SW, SE) with given quater probabilities a, b, c and d, all greater then 0, which sum $a+b+c+d = 1$. Depending of the distribution of probalibilities a graph G with the corresponging distribution of vertices with a given degree is generated [5].

The set of R–MAT graphs G has the following characteristics:
R1: *rmat50_35*, $|V|$ =50, $|E|$ =429, $d(G)$ = 0,350, k =5, bs =10, pn =150 ;
R2: *rmat50_40*, $|V|$ =50, $|E|$ =490, $d(G)$ = 0,400, k =5, bs =10, pn =250 ;
R3: *rmat50_65*, $|V|$ =50, $|E|$ =796, $d(G)$ = 0,650, k =5, bs =10, pn =270 ;
R4: *rmat50_70*, $|V|$ =50, $|E|$ =858, $d(G)$ = 0,700, k =5, bs =10, pn =300 ;
R5: *rmat75_84*, $|V|$ =75, $|E|$ =2331, $d(G)$ = 0,840, k =5, bs =15, pn =800 ;
R6: *rmat100_90*, $|V|$ =100, $|E|$ =4455, $d(G)$ = 0,900, k =5, bs =20, pn =1500.

2.3 Metaheuristics

The basis heuristics, their combinations and parallel/hybrid versions established a testbed for experimental part of our research [1, 13]. The tested algorithms are:
sSA — sequential Simulated Annealing (SA),
mirSA — parallel SA with Multiple Independent Runs (MIR),
aSA — asynchronous SA,
sTS — sequential Tabu Search (TS),
mirTS — parallel TS with MIR,
aTS — asynchronous TS,
sEA — sequential Evolutionary Algorithm (EA),
mirEA — parallel EA with MIR,
iEA — island EA without migration,
ibmEA — island EA with migration of best individuals,
irmEA — island EA with migration of random individuals,
sSA-TS — sequential SA-TS,

mirSA-TS — parallel SA-TS with MIR,
aSA-TS — asynchronous SA-TS,
aEA-SA — asynchronous EA-SA,
aEA-TS — asynchronous EA-TS,
aEA-SA-TS — asynchronous EA-SA-TS.

Pseudocodes of the above algorithms are available from WWW site [16].

Initial values of basic parameters used in the above algorithms are the following: Number of iterations in a single step = 15, Stop criterion = 15, Initial temperature (SA) = 10, Size of the tabu list (TS) = 7, Number of candidates (TS) = 8, Population size (EA) = 30, Offspring number (EA) = 5, Crossover probability (EA) = 0,8, Mutation probability (EA) = 0,1. The assumed cost function f coefficients are: a = 1, b = 1, c = 5, d = 5, e = 5.

Parameters of parallel algorithms are: Number of processors (mir)= 6, Communication rate = 10, Number of islands (iEA, ibmEA, irmEA) = 6, Migration size (ibmEA, irmEA) = 18, Migration rate (ibmEA, irmEA) = 10.

3 Computational experiments

The main purpose of the experimental part is graph $G(V, E)$ partitioning satysfying design assumptions related to the number of blocks (clusters) and simultaneously minimizing the number of interconnections between partition blocks.

Table 1. Computational test results for graph instances Q1, Q3, Q4 and Q5.

Graph	Q1			Q3			Q4			Q5		
Algorithm	f	ext	time [s]	f	ext	time [s]	f	ext	time [s]	f	ext	time [s]
sSA	286	254	8,12	1168	1093	4,81	1993	1898	9,57	3200	3079	5,07
mirSA	290	255	2,09	1160	1089	12,07	1932	1867	67,04	3140	4412	92,56
aSA	288	255	4,07	1172	1094	8,15	2000	1902	9,96	3211	3083	15,40
sTS	262	242	10,43	1096	1058	14,75	1902	1852	43,00	3074	3015	43,71
mirTS	**261**	**242**	**24,32**	1094	1056	73,25	1888	1845	204,4	3058	3009	327,3
aTS	263	243	13,10	1091	1054	58,67	1906	1851	193,5	3082	2987	161,4
sEA	**261**	**242**	**244,4**	1095	1050	595,5	1887	**1836**	**629,5**	3085	3011	1414
mirEA	**261**	**242**	**93,23**	1080	**1048**	**263,9**	1887	**1836**	**626,8**	3063	3000	1364
iEA	**261**	**242**	**95,79**	**1079**	1049	**620,8**	1881	1840	1229	**3056**	**2974**	**2811**
ibmEA	**261**	**242**	**102,9**	1092	1054	525,3	1889	1837	861,3	**3056**	2982	**2043**
irmEA	264	243	98,44	1082	1049	597,1	1895	1840	731,9	3060	2976	1355
sSA-TS	**261**	**242**	**2,57**	1093	1056	15,59	1886	1844	15,81	3082	2995	40,57
mirSA-TS	**261**	**242**	**15,45**	1088	1051	46,54	**1880**	1838	**94,40**	3070	3007	242,3
aSA-TS	**261**	**242**	**20,09**	1088	1051	53,43	1885	1842	142,6	3078	3016	244,4
aEA-SA	**261**	**242**	**31,45**	1086	1051	126,8	1903	1844	113,3	3073	3005	284,3
aEA-TS	263	243	34,70	1083	1050	233,9	1903	1844	274,3	3061	2992	592,2
aEA-SA-TS	**261**	**242**	**82,78**	1097	1058	136,9	1887	1843	465,1	3087	3020	122,0

The primary objective is to minimize the cost function f. The secondary objective is to minimize the computation time. In parallel algorithms computation times of parallel processors are added up (parallel execution of the algorithm is simulated).

In the first experiment we are searching for minimal values of f and min–cut ext. The esssential results for the four queen graphs are reported in Table 1. The best results of f and ext, the methods winning for at least one graph and the corresponding computation times are shown in the bold font. The iteration details and terms of f are not reported due to lack of space.

For Q1 graph many methods produce equivalent results, but the best computation time is obtained by sSA-TS. Relatively low running times are required for mirSA-TS and mirTS. For Q3 graph iEA provides the best f while mirEA finds a solution the best ext. However, mirEA uses only 42,5 % of iEA computation time. Many other parallel and hybrid algorithms can find quite satisfying solutions in a shorter time. For Q4 graph mirSA-TS outperforms EA-based methods in terms of computation time, providing optimal f and suboptimal ext. Similarly, the best solution quality for Q5 graph provide iEA and ibmEA but their computation times are hardly acceptable. Good alternatives with a significantly shorter computation times are mirTS and aEA-TS. Other results are more distant from the best solution found.

Table 2. Computational test results for graph instances R3, R4, R5 and R6.

Graph	R3			R4			R5			R6		
Algorithm	f	ext	time [s]	f	ext	time [s]	f	ext	time [s]	f	ext	time [s]
sSA	715	640	2,15	740	685	1,12	1924	1862	1,34	3681	3681	3,40
mirSA	703	643	19,5	740	685	4,42	1932	1866	13,40	3679	3612	24,23
aSA	711	638	7,50	748	686	4,04	1938	1872	12,50	3681	3616	20,76
sTS	679	622	6,93	708	669	11,04	1842	1824	13,12	3597	3574	33,75
mirTS	**677**	**621**	**85,57**	**704**	**667**	**67,78**	1840	1823	107,1	3587	3569	149,2
aTS	683	624	48,12	708	669	35,79	1844	1825	92,31	3583	3567	184,5
sEA	681	623	54,50	714	672	73,26	1854	1830	139,8	3595	3573	263,6
mirEA	681	623	346,5	710	670	406,7	1840	1823	1338	3597	3574	2755
iEA	679	622	380,3	708	669	276,8	1846	1826	1307	3745	2585	2274
ibmEA	683	624	320,8	712	671	202,5	1842	1824	1005	3579	3579	3132
irmEA	**677**	**621**	**463,4**	710	670	206,7	1854	1830	889,3	3589	3570	2028
sSA-TS	681	623	6,28	706	668	7,47	**1836**	**1821**	**38,73**	3591	3571	29,75
mirSA-TS	679	622	41,20	**704**	**667**	**38,76**	1840	1823	83,48	**3573**	**3562**	**257,6**
aSA-TS	**677**	**621**	**51,79**	706	668	47,48	1838	1822	97,09	3579	3565	259,1
aEA-SA	681	623	103,3	714	672	60,81	1840	1823	282,3	3601	3576	458,9
aEA-TS	681	623	151,8	708	669	115,8	1842	1824	324,8	3591	3571	265,9
aEA-SA-TS	683	624	91,68	706	668	186,3	1846	1826	255,3	3593	3572	626,4

In the second experiment, devoted to R–MAT graphs, we are also searching for minimal values of f and min–cut ext. The esssential results for four input graphs R3, R4, R5 and R6 are reported in Table 2. The best results of f and ext, the methods winning for at least one graph and the corresponding computation times are shown in the bold font.

For relatively easy R3 graph instance three methods find best values of f and ext : aSA-TS, mirTS and irmEA. They are listed in the increasing order of computation times. For R4 graph two methods: mirTS and mirSA-TS produce equivalent results in a reasonable time, but mirSA-TS is almost two times faster. The worst solution is found by aSA. For R5 graph the single winner is sSA-TS with acceptable computation time, a good alternative is sTS which is three times faster then sSA-TS still ensuring competitive resuls. Other efficient methods: aSA-TS, mirSA-TS, aEA-SA and aEA-TS require a longer time. The best solution quality for R6 graph ensures mirSA-TS within acceptable computing time. Other methods are worse in terms of f and ext.

Computer application *Electronic Circuit Decomposition* used for the presented research was written in C++/CLI within Visual C++ 2008, Express Edition environment. GUI was made in Windows Forms Application. For program execution .NET Framework 3.5 package is needed, supplied by ZedGraph library.

4 Conclusions

From the reported research one can conclude that for graph partitioning problem with modified queen and R–MAT graph instances the most valuable components for hybrid algorithms are TS and EA, which can be combined within one algorithm. This confirms earlier results on different set of benchmarks reported in [7]. EA usually involves longer computation time. The essential part of parallel algorithms is Multiple Independent Runs (MIR) model except mirSA algorithm.

In general, the differences in computation time by various methods can be extremal, while the quality of all graph partitioning methods was good, usually not exceeding several percent of the values f and ext of the best solution. The research may be continued with focus on the outstanding mixed methods as well as larger and harder graph partitioning instances.

It would be also desirable to apply the best methods to circuit benchmarks resulting from engineering practice and verify their efficiency on real design problems.

5 Acknowledgements

This work was supported by the research grant No. E–3/611/2017/DS from Cracow University Technology.

References

1. E. Alba (ed.), Parallel Metaheuristics: A new Class of Algorithms, Wiley-Interscience, New Jersey 2005.
2. A. Chadha: Benchmark Creation for Circuit Partitioning Algorithms. Gauging the performance of circuit partitioning algorithms, Lambert Academic Publishing, 2015.
3. V. Chvátal: Colouring the queen graphs. Available at : http://users.encs.concordia.ca/ chvatal/queengraphs.html
4. M.C. Bhuvaneswari, M. Jagadeeswari: Circuit Partitioning for VLSI Layout. In: Bhuvaneswari M. (Eds.) Application of Evolutionary Algorithms for Multi-objective Optimization in VLSI and Embedded Systems. Springer, New Delhi, 37–46, 2014. https://doi.org/10.1007/978-81-322-1958-3_3
5. D. Chakrabarti, Y. Zhan, C. Faloutsos: R–MAT: A Recursive Model for Graph Mining, Proc. 2004 SIAM Int. Concerence Data Mining, 2004. DOI: 10.1137/1.9781611972740_43
6. COLOR web site. Available at : http://mat.gsia.cmu.edu/COLOR/instances.html
7. C. Gil, J. Ortega, M.G. Montoya, R. Banos: A Mixed Heuristic for Circuit Partitioning, Computational Optimization and Applications Vol. 23, 321–340, 2002. DOI: 10.1023/A:1020551011615
8. Graph Partitioning and Graph Clustering. 10th DIMACS Implementation Challenge, February 12-13, 2012, Atlanta, A. Bader *et all* (Editors), AMS, Contemporary Mathematics 588, 2013.
9. A. Khang, J. Liening, I. Markov, J. Hu, VLSI Physical Design: From Graph Partitioning to Timing Closure, Springer, 33–54, 2011.
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, M. Bała, Solving graph partitioning problems with parallel metaheuristics, [in:] Fidanova S. (ed.): Recent Advances in Computational Optimization, Studies in Computational Intelligence, Vol. 717, 89–105, Springer International Publishing, 2018. https://doi.org/10.1007/978-3-319-59861-1_6
12. S. Kozieł, W. Szczęśniak, Evolutionary Algorithm for Electronic System Partitioning and its Applications in VLSI Design, Proc. 6th IEEE Int. Conference on Electronics, Circuits and Systems, ICECS 1999, Pafos, Cyprus, September 5-8, 1999, Vol. 3, 1412–1414.
13. S. Sadiq, Y. Habib, Iterative Computer Algorithms with Applications in Engineering. Solving Combinatorial Optimization Problems, Wiley - IEEE Computer Society Press, 2000.
14. R.R. Swethaa, K.A.S. Devi, S. Yousef: Hybrid Partitioning Algorithm for Area Minimization in Circuits, Procedia Computer Science, Vol. 48, 692–698, 2015. DOI: 10.1016/j.procs.2015.04.203
15. W. Szczęśniak, Application of Adaptive Circuit Partitioning Algorithms to Reductions of Interconnections Length Between Elements of VLSI Circuit, Proc. 9th IEEE Int. Conference on Electronics, Circuits and Systems, ICECS 2002, Dubrownik, Croatia, September 15–18, 2002. DOI: 10.1109/ICECS.2002.1046259
16. Pseudocodes of algorithms from section 2.3. Available at : http://www.pk.edu.pl/˜zk/pubs/NMA18.zip