EXPERIMENTAL STUDY: HYPERGRAPH PARTITIONING BASED ON THE SIMPLE AND ADVANCED GENETIC ALGORITHM BMDA AND BOA

Josef Schwarz, Jiří Očenášek

Technical University of Brno
Faculty of Engineering and Computer Science
Department of Computer Science and Engineering
CZ - 61266 Brno, Božetěchova 2
e-mail: schwarz@dcse.fee.vutbr.cz
e-mail: xocena00@stud.dcse.fee.vutbr.cz

Abstract: *This paper is an experimental study on hypegraph partitioning using the simple genetic algorithm (GA) based on the schema theorem and the advanced algorithms based on the estimation of distribution of promising solution. Primarily we have implemented a simple GA based on the GaLib library[Gal94] and some hybrid variant included a fast heuristics to speed up the convergence of the optimization process. Secondly we have implemented the Univariate Marginal Distribution algorithm (UMDA) and the Bivariate Marginal Distribution algorithm (BMDA), both have been published even recently[Pel98] and used a share version of a superior new program BOA based on the Bayesian Optimization Algorithm [Pel99]. We have also extended the BMDA algorithm to a new version with finite alphabet encoding of chromozomes and new metric that enables the m-way partitioning graphs. The aim of our paper is to test the efficiency of new approaches for discrete combinatorial problems represented by hypergraph partitioning.*

Key words: *decomposition, hypergraph partitioning, simple and hybrid GA, estimation of distribution algorithm, Bayesian network.*

1 Introduction

Hypergraph partitioning is a well known problem of graph theory. In case of so called 2-way partitioning the bisection term is used. The graph representation can be used for many application problem e.g. for the system segmentation, network partitioning and VLSI layout.

More formal, the particular partitioning problem can be defined as follows: Let us assume a hypergraph $G=(V,E)$, with $n = |V|$ nodes and $t = |E|$ edges. Let m be a natural number. A m-way partition is specified by disjoint partitions of nodes $A_0, A_1,\ldots,A_{m-1}$ with equal or predefined cardinality. The cost of the partition is defined as a function of the hyperedges having nonempty intersection with at least two partitions from the partitions $A_0, A_1,\ldots,A_{m-1}$. We call these hyperedges external ones. From the previous premises it follows that an unconstraint balanced partitioning problem is solved.

Many heuristics are used to solve this NP-complete problem. We can refer to the recent paper [Oom96] where a good overview of the known local search techniques is done and automaton-based algorithm is described in more detail. The hybrid genetic algorithm is described in [Pat95],[Bui96]. The mentioned simple genetic algorithms are based on the schema theory [Gol86]. It is known that during evolving a new population standard genetic operators often cause disruption of schemata mainly of large defining length. To prevent the problem a techniques of reordering of graph nodes in chromosomes was proposed by [Bui96]. We focus in this paper on another promising approach based on an estimation of the joint distribution of promising solutions proposed in [Pel98], [PGC98], [MUE98].

2 Problem formulation

The hypergraph is often modelled by bigraph (see Fig 1), where a bigraf $G(V,S)$ with the set of node*s* V and set of nets S is presented. This is an example of 3-way partitioning problem with a 9 nodes and a 10 nets. Each partition/assembly A_0, A_1, A_2 contains 3 nodes. The cost/objective function L is based on external nets that connect nodes from different assemblies:

$$L = \sum_{s_i \in S_e} [D(s_i) - 1] \qquad (1)$$

where $D(s_i)$ is the degree of the external nets $S_e = \{s_1, s_2, s_3\}$.

The incidence of external nets to each assembly is represented by dashed lines. The minimal set *{L1, L2, L3, L4}* of external connections is represented by full lines produced by the spanning tree technique. In the following 3-way partitioning the cost *L* equals to 4.
Another way how to calculate the external connection is based on the number of nets that incident with *i-th* assembly. The S_i items can be simply stated : $S_0=5,\ S_1=5,\ S_2=4;$ for the known total number of nets $t=10$ we get $L=4$.

$$L = \sum_{i=0}^{m-1} S_i - t \qquad (2)$$

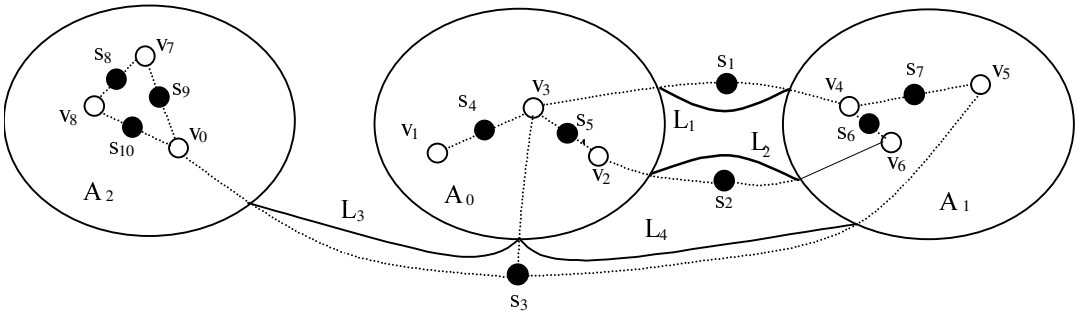


Fig.1 The 3- way partitioning case of a hypergraph.

In case that each net connects only 2 nodes the hypergraph can be reduced to a simple graph *G(V,E,W),* where the connection matrix *W=[wij]* represents the weight of edge/connection between node *i* and *j.*

3 Genetic algorithms

To solve the partitioning problem we have implemented 2 types of genetic algorithms- simple genetic algorithm SGA and its heuristic versions, advanced BMDA algorithm and used the BOA program. For all of them the following ordinary string/chromosome encoding is used:

Genotype		Meaning
Trisection	**0 0 0 1 1 1 2 2 2**	Gene value /Assembly number
Bisection	**1 1 0 0 1 0 1 0 0**	
	0 1 2 3 4 5 6 7 8	Locus/Node number

Table 1. The ordinary string encoding for bisection and trisection of hypergraph.

The gene value represents the assembly number, the index of locus specifies the node number**.** The efficiency of the BMDA and BOA algorithm is determined by the level of gene dependency. That is why it is useful to express the cost function using the string encoding. For the simplest case of 2 - way partitioning/bisection of a simple graph *G(V,E,W)* we derived on the binary string $X=(x_0, x_1, .., x_{n-1})$ the following quadratic cost function:

$$L = \sum_{\substack{i=0 \\ j>i}}^{n-1} w_{ij}(x_i + x_j - 2x_i x_j) \qquad (3)$$

with the balance condition $\sum_{i=0}^{n-1} x_i = \sum_{i=0}^{n-1} (1 - x_i)$,

where the coefficient $w_{ij}=1$ in case the net/edge exists between node *i* and *j*, else $w_{ij}=0$. In case of m-way partitioning of a single graph the function is not binary because the string is alphabetic:

$$L = \sum_{\substack{i=0 \\ j>i}}^{n-1} w_{ij}\ f_k(x_i, x_j)\ ;\quad f_k(x_i, x_j) = 1 \quad for \quad x_i \neq x_j\ ;\ f_k(x_i, x_j) = 0 \quad for \quad x_i = x_j \qquad (4)$$

In case of m-way partitioning of hypergraphs the useful term for cost is quite complex and is beyond the scope of this paper.

3.1 Simple genetic algorithm

We have implemented a simple genetic algorithm SGA described in [Gol86] using the well known GaLib library [Gal94]. The fitness function is based on the cost *L*: *Fitness = 1/(L+1).* The selection scheme is roulette wheel with the linear-scaled fitness. Offspring are generated by the one-point crossover operator. To implement

mutation the values of several genes are changed to any of the possible allele values (*flip mutator).* The genetic operators described do not guarantee the equal size of all assemblies. After crossover or mutation completion, strings should be balanced/normalized randomly to keep equal or predefined number of different alleles. In the replacement stage the principle of elitism is used. We designed also SGAN version with advanced/positive normalization of string contributed to cost decrease. To improve the convergence speed of the optimization process for more complex problems a heuristic procedure was added to the SGA algorithm to get a hybrid genetic algorithm SGAH. It is based on the reconfiguration of some offspring in the current population using the pairwise interchange of genes with cost decrement. The procedure is activated seldom only during an epoch H_e formed by a number of generations. It is possible to set the intensity H_i and the range H_r of the reconfiguration.

3.2 UMDA and BMDA algorithm

The following methods are based on probability theory and statistics. They use statistical information contained in the set of selected parents to detect gene dependencies. The estimated probability model is used to generate new promising solutions according to this distribution. Generally, UMDA, BMDA and BOA belong to an EDA class of algorithm (Estimation of Distribution Algorithm) [Mue98], which can be described as follows:

Generate initial population of size N (randomly);
While *termination criteria is false* **do**
begin
Select parent population P of M individuals according to a selection method (M≤N);
Estimate the distribution of the selected parents;
Generate new offspring (according to the estimated model);
Replace some individuals in current population by generated offspring;
end

UMDA [Pel98] (Univariate Marginal Distribution Algorithm) assumes that genes are mutually independent. Let us denote a chromosome length by n. For each gene position $i \in \{0..n-1\}$ and each possible value of this gene $x_i \in \{0,1\}$, the univariate marginal frequency $p_i(x_i)$ is defined as the frequency of strings that have x_i on *i-th* position in the parent population P: $p_i(x_i) = n_i(x_i)/N$, where $n_i(x_i)$ is a number of appearances of the allele x_i on *i-th* position. Each new individual $X = (x_0, x_1, ..., x_{n-1})$ is generated by UMDA according to the distribution

$$p(X) = \prod_{i=0}^{n-1} p_i(x_i), \tag{5}$$

so the value of *i*-th gene is set to value *a* with the probability equal to $p_i(a)$. UMDA is able to cover linear problems only.

BMDA [Pel98], [Oce99] (Bivariate Marginal Distribution Algorithm) is an extension of UMDA. In addition, the pair dependencies are allowed. The bivariate marginal frequency $p_{i,j}(x_i,x_j)$ is defined as the frequency of individuals in parent population P, that have values x_i and x_j on positions i and j at the same time: $p_{i,j}(x_i, x_j) = n_{i,j}(x_i, x_j)/N$. Conditional probability of occurrence of the value x_i on *i*-th position in the case of occurrence of x_j on *j*-th position is determined

$$p_{i,j}(x_i \mid x_j) = \frac{p_{i,j}(x_i, x_j)}{p_j(x_j)} \tag{6}$$

We extended the concept of UMDA and BMDA for the alphabet encoding, so that $x_i \in \{0,...,r_i-1\}$ and $x_j \in \{0,...,r_j-1\}$. For each pair of positions *i, j* the count of each combination of values can be summarized into following contingency table:

$x_i \backslash x_j$	0	1	...	$r_j - 1$	**Σ**
0	$n_{i,j}(0,0)$	$n_{i,j}(0,1)$	...	$n_{i,j}(0, r_j - 1)$	$n_i(0)$
1	$n_{i,j}(1,0)$	$n_{i,j}(1,1)$	...	$n_{i,j}(1, r_j - 1)$	$n_i(1)$
...	...	...	...	...	...
$r_i - 1$	$n_{i,j}(r_i - 1,0)$	$n_{i,j}(r_i - 1,1)$	...	$n_{i,j}(r_i - 1, r_j - 1)$	$n_i(r_i - 1)$
Σ	$n_j(0)$	$n_j(1)$	...	$n_j(r_j - 1)$	N

Gene dependencies are discovered by Pearson's chi-square statistics [Pel98], we use the following form of equation [Oce99]:

$$X^2_{i,j} = N\left(\sum_{k=0}^{r_i-1}\sum_{l=0}^{r_j-1}\frac{n^2_{i,j}(k,l)}{n_i(k)\,n_j(l)} - 1\right) \tag{7}$$

This metric is symmetrical, $X^2_{i,j} = X^2_{j,i}$, so for each couple of positions it has to be computed only once. Genes are considered to be independent if the result does not meet certain threshold. For example binary genes are independent for 95% if $X^2_{i,j} < 3.84$.

The dependency information is used to build up the acyclic dependency graph, which can be seen as the set of trees. The root nodes correspond to the positions where the values are generated using the univariate distribution, the values of positions connected to already generated positions in the graph are subsequently generated using the conditional probability. After chromosome generation the balancing procedure is used.

We have also implemented the following (non-symmetrical) metric giving as good results as Pearson's statistics:

$$K2_{i,j} = \left(\prod_{l=0}^{r_i-1}\frac{(r_j)!}{(r_j+n_i(l))!}\prod_{s=0}^{r_j-1}(1+n_{i,j}(l,s))!\right)\Bigg/\left(\frac{(r_j)!}{(r_j+N)!}\prod_{z=0}^{r_j-1}(1+n_j(z))!\right) \tag{8}$$

This equation is derived from K2 metrics used in BOA algorithm discussed in the next chapter.

3.3 BOA program

BOA [PGC98] (Bayesian Optimization Algorithm) uses Bayesian network to encode the structure of a problem. It is an analogy of BMDA dependency graph, but the higher order gene dependencies can be covered too. For each variable X_i a set of variables Π_{X_i} is defined it depends on, so the distribution of individuals is encoded as

$$p(X) = \prod_{i=0}^{n-1} p\left(X_i \mid \Pi_{X_i}\right) \tag{9}$$

Generally, the existence of directed edge from X_j to X_i in the network implies the belonging of the variable X_j to the set Π_{X_i}. To reduce the space of networks, number of incoming edges into each node is limited to *k*.

The Bayesian Dirichlet (BD) metric is used to measure the quality of the network [PGC98]. A special case of BD metric, so-called K2 metric, is used when no prior information about the problem is available. Actually, the equation (8) is derived from K2 metrics for *k=1* and alphabet encoding. It determines the relative metric improvement for one edge addition.

In the shared implementation of BOA [Pel99] a simple greedy algorithm is used to search for a good network. In each step the best edge is added. By the term 'best edge' we mean the edge giving the highest K2 metric for the network B' that can be constructed from the actual network B by adding this edge. It must also keep the network acyclic and meet the limit of incoming edges.
After network construction new instances are generated using the univariate and conditional probability in a similar way as for BMDA algorithm.

4 Experimental results

Numerous experiments were done to demonstrate the behaviour and the efficiency of the individual algorithms. The two types of graph structures are used:

1. Regular graphs RLXB with grid structure [Schw98], where the notion X specifies the number of nodes, B specifies the existence of bottle-neck in the graph structure; the global optimum is known. As an example a bisection of graph RL64B is represented in Fig.2a having a 2-edge bottle-neck in the dashed cut line. The regular graphs with bottleneck appear to be a proper test benchmark with many local optima.
2. Hypergraphs representing real circuits labelled by ICX [Schw86]. The global optima is not known. The structure of circuits can be characterized as a random logic. The hypergraph IC67 consists of 67 nodes and 134 edges/nets, the IC151 consists of 151 nodes and 286 edges/nets.

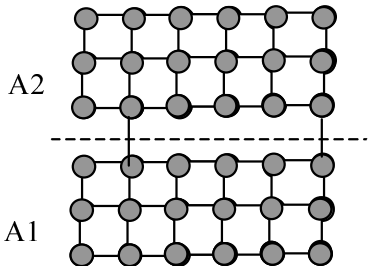


Fig.2 Regular graph RL36B.

The main experimental results are listed in Table 2. In case of regular graphs (see part A) the items in the table represent the average number of fitness for 5 successful runs with different initial random population. The parameters of the algorithms are set to the minimal possible values to get global optimum in all 5 runs. In case of graphs IC67 and IC151 (see part B) trials are focused on the goal to get a local minimum; the number of fitness evaluations is limited. The BOA and BMDA algorithms perform well, the SGA and SGAN converge slowly, the SGAH is the best of all algorithms.

Graphs	m [assemblies]	A. Average number of evaluations to get global optimum L					Global optimum L
		SGAH	SGAN	SGA	BMDA	BOA k=3	
RL36B	2	7115	1970	60942	5650	4242	2
RL36B	4	42812	3161	421827	28055	-	8
RL36B	6	65255	12133	1214150	293000	-	14
RL48B	2	4580	12850	*177933	20700	6036	2
RL48B	4	21375	138500	-	250320	-	8
RL64B	2	14900	8200	808571	64500	7905	2
RL64B	4	69830	306800	-	283200	-	10
RL100B	2	127825	105275	-	680000	19550	2
B. Cost/Average number of evaluations							
IC67	2	36/4050	41/22950	45/37900	41/15290	40/12375	unknown
IC67	4	69/16530	86/42460	103/39780	72/25850	-	unknown
IC151	2	58/19990	73/44220	110/46500	66/25465	67/22000	unknown
IC151	4	126/40760	218/43550	256/41510	208/26895	-	unknown

Tab.2. Experimental results for SGA variants, BMDA and BOA algorithms.

In Fig.2 a comparison of BOA and BMDA algorithms is done for the case of bisection of graphs RL36B, RL48B and RL64B. The BOA algorithm for *k=3* performs very well. In a) part the dependency of the minimal population size on the problem size and in b) part the dependency of number of fitness evaluations on the problem size are shown.

In Fig.3a the performance of BMDA, BOA for k=3, SGA and SGAH is presented for the bisection of graphs mentioned above and for RL100B graph. It is evident that the BOA algorithm gains on all the algorithms used for bisection. BMDA converges much slower namely for RL100B graph. The SGAH algorithm is good enough, better than BMDA and worse than BOA. The SGA algorithm performs very purely with great number of fitness evaluations; in case of graph RL100B the global optimum was not reached during five independent trials. In case of graph RL48B the global optimum was reached only in three of five trials (the item in Table 2 was signed by asterisk).

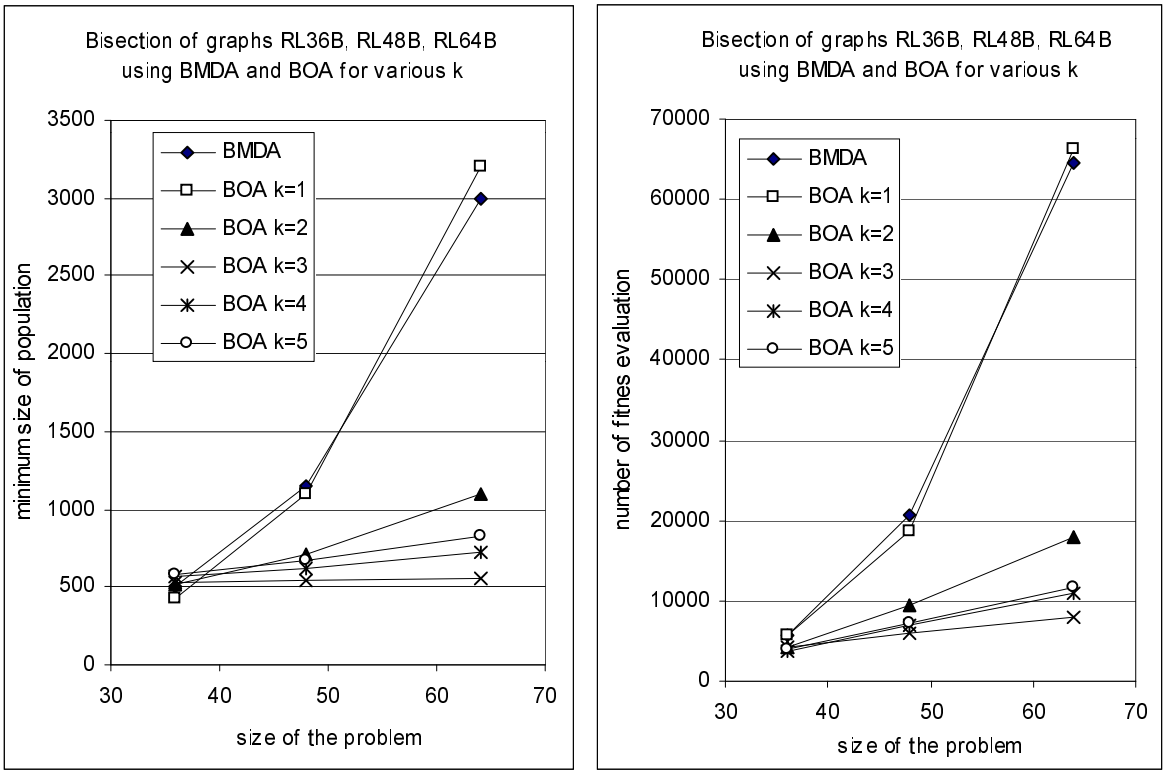
In Fig.3b the results of multi-way partitioning of graph RL36B are presented for m=2,4 and 6. The SGAN algorithm with positive normalization performs quite well, the SGA performs purely. BMDA and SGAH algorithms perform good enough for quadrisection but for 6-way partition the number of fitness evaluations rapidly increases. The relatively good results are due to the small size graph used for partitioning. The multi-way partitioning for *m>4* seems to be a hard problem for great size of problems.

During all the experiments the following range of parameters was used: for BOA and BMDA the 50% truncation selection (a subset of population for estimation of distribution), for BOA the parameter *k=1-5* of Bayesian network. In case of SGA and SGAN algorithms the crossover rate R_c=0.5, mutation rate R_m=0.05 is used. In case of SGAH algorithm crossover rate Rc=0.5, mutation rate R_m=0.01- 0.05, H_e=5 -10, H_i = 10-30, H_r = 10-30 is used.

Generally, for SGA variants, the problems with proper setting of various heuristic parameters to prevent the premature convergence occur.

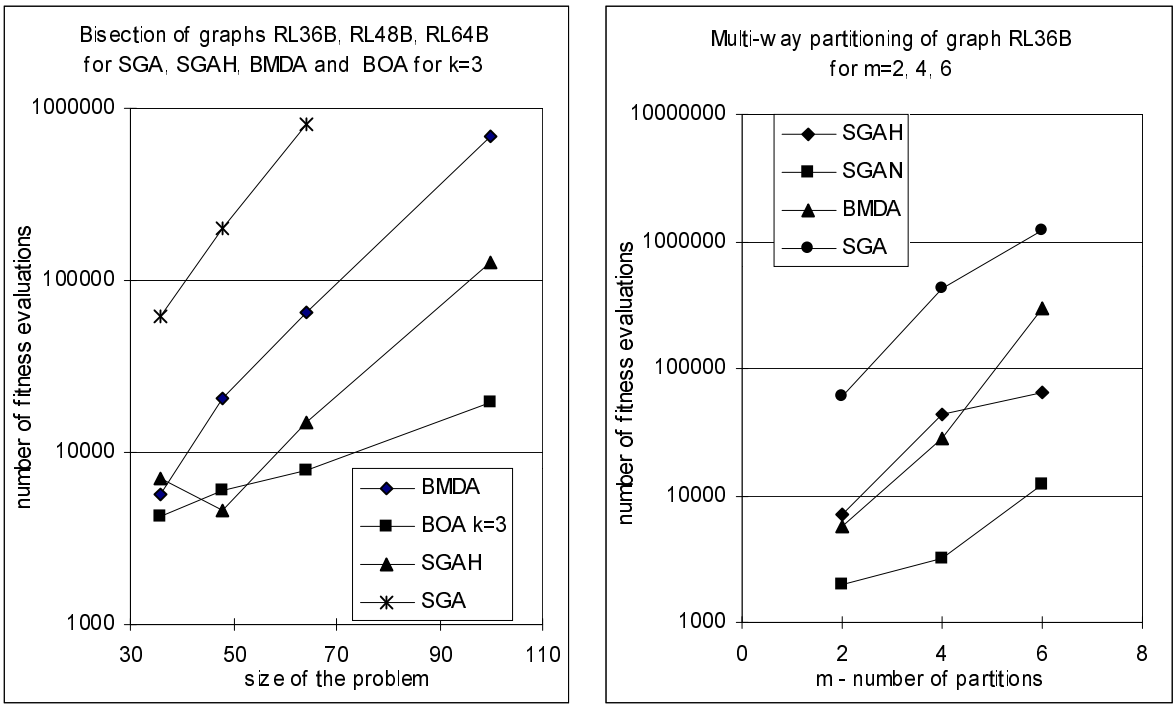
In case of partitioning of the hypergraphs IC67, IC151 (see Table 2, part B), the population size is set to a minimum value N=550 for BOA and BMDA as well; the number of evaluations is limited to 27500. For SGAH

algorithm R_m= 0.01, H_e=5, H_i = 20, H_r = 20, for SGA and SGAN R_m= 0.05, R_c= 0.5; the population size N=100, the number of evaluations is limited to 50000 for all the SGA variants.



a) b)

Fig2 a) The minimum size of population found to reach the global optimum for regular graphs RL36B, RL48B, RL64B, b) Number of fitness evaluations for bisection of regular graphs RL36B, RL48B, RL64B to reach the global optimum. The range of used population size was 500-3000 for BMDA and 430-3200 for BOA.



a) b)

Fig.3 a) Number of fitness evaluations for bisection of regular graphs RL36B, RL48B, RL64B, RL100B to reach the global optimum, using algorithms SGA, SGAH, BMDA and BOA for k=3,. The range of used population size was 200-700 for SGA, 100-200 for SGAH, 500-20000 for BMDA(3000 for RL36B) and 535-850 for BOA, b) Number of fitness evaluation as a function of number of assemblies m for graph RL36B, (for Y axis log-scaling is used).

5 Conclusions

The paper describes shortly three types of genetic algorithms. The first one (SGA) is based on the schemata theory, the second one (BMDA) on an estimation of the distribution of promising solution, the last one BOA uses the extra techniques to model data by Bayesian network.

We have focused on the implementation of SGA and BMDA version of genetic algorithm and adaptation of BOA program with the aim to compare their performance and efficiency. From the experiments it is evident that for bisection of regular test graphs the best performance offers the BOA algorithm with small amount of evaluations and size dependency. BMDA algorithm works well on the bisection of graphs up to 64 nodes; for RL100B graphs the number of evaluations increases dramatically. The SGA algorithm performs very purely, the number of evaluations is very high and it failed for the RL100B graph. The SGAH version with heuristics seems to be a good tool but it is very sensitive on setting various parameters and it often gets stuck in local optima. The bad performance of SGA seems to be caused by phenomenon of building block disruption.

The multi-way partitioning is a hard problem for all the algorithms. The BMDA algorithm works well enough but only for relatively small problems. The SGAN with positive normalization of partitions works best of all but for small problem and with the tendency to get stuck in local optima.

Our contribution can be seen in the extension of BMDA algorithm to the finite alphabet encoding of chromosomes to be able to solve the task of multi-way partitioning graphs. We have also proved the efficiency of the modified BMDA algorithm using the same heuristic procedure as in the SGAH algorithm. The performance of this version of BMDA is very similar to the original one. We have also implemented the modified K2 metric for BMDA instead of Pearson's statistics without remarkable influence.

The future work will be focused namely on an exploration of the BOA version with finite alphabet encoding for multiple partitioning graphs and placement problem. Other activities will be directed towards the usage of EDA algorithms with problem knowledge.

References

[Bui96] Than Nguen Bui, Byung Ro Moon: Genetic Algorithm and Graph Partitioning. IEEE Transactions on Computers, Vol.45, No.7, July 1996, pp. 841-855.

[Gol89] Goldberg E.,D.: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[Oce99] Očenášek J.: Advanced genetic techniques for problem optimization, Diploma Project, Department of Computer Science and Engineering, Technical University of Brno, 1999, pp.1-23, in Czech.

[Oom96] Oomen J. B., Croix V.: Graph partitioning using learning automata. IEEE Trans. on Computers, vol. 45, No2, February 1996, pp.195- 206.

[Mue98] Muehlenbein H., Rodriguez A. O.: Schemata Distributions and Graphical Models in Evolutionary Optimization. GMD Forschungs Zentrum Informationstechnik, 53754-St. Augustin, 1998, pp.1-21.

[Pat95] Ananta K.Majhi, L.M. Patnaik, Srilata Raman: A Genetic algorithm-based circuit partitioner for MCMs. Microprocessing and Microprogramming 41 (1995), pp. 83-96.

[Pel98] Pelikan M., Muehlenbein H.: Marginal Distribution in Evolutionary Algorithms. 4th International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets, June 24-26, 1998, Brno, Czech Republic, pp. 91-95.

[Pel99] Pelikan M.: A Simple Implementation of Bayesian Optimization Algorithm in C++(Version1.0). Illigal Report 99011, February 1999, pp. 1-16.

[PGC98] Martin Pelikan, David Goldberg, and Erick Cantú Paz: Linkage Problem, Distribution Estimation, and Bayesian Networks. IlliGal Report No. 98013, November 1998,pp. 1-25.

[Sch84] Schwarz J.: Design Automatization of Composition and Placement of Integrated Circuit. PhD. Theses, Technical University of Brno, Faculty of Engineering and Computer Science, Department of Computer Science and Engineering, Brno, 1984, 136 pp., in Czech.

[Sch98] Schwarz J.: Genetic algorithm for partitioning circuits. 4th International Mendel Conference on Genetic Algorithms, Optimization Problems, Fuzzy Logic, Neural Networks, Rough Sets, June 24-26, 1998, Brno, Czech Republic, pp.126-131.

[Gal96] Galib: A C++ Library of Genetic Algorithm Components, http://lancet.mit.edu/edu/ga/.