A Bivariate Probabilistic Model-building Genetic Algorithm for Graph Bipartitioning

Dirk Thierens
Dept. of Computer Science, Universiteit Utrecht
Utrecht, The Netherlands
dirk.thierens@cs.uu.nl

ABSTRACT

We investigate a bi-variate probabilistic model-building GA for the graph bipartitioning problem. The graph bipartitioning problem is a grouping problem that requires some modifications to the standard construction of the dependency tree. We also increase the computational efficiency of the Bi-PMBGA by restricting the dependency tree to the edges of the graph to be partitioned. Experimental results indicate that the Bi-PMBGA performs significantly better than the multi-start local search. Compared to a genetic local search algorithm the Bi-PMBGA performs slightly worse on some of the graphs considered here.

Categories and Subject Descriptors

I.2.8 [**Problem Solving and Search**]:

General Terms

Algorithms, Performance

Keywords

Probabilistic model-building EAs

1. INTRODUCTION

Probabilistic Model-Building Genetic Algorithms (PMBGAs) (also known as estimation of distribution algorithms (EDAs) or iterated density estimation algorithms (IDEAs)) are population-based, stochastic search algorithms that build a series of probabilistic models of promising solutions and generate new solutions by sampling these models [11]. In this paper we will look at a bi-variate PMBGA applied to the graph bipartitioning problem. The graph bipartitioning problem is a grouping problem so we cannot apply the standard BMDA algorithm [1] [10]. To improve the computational efficiency we also restrict the dependency tree to the edges of the underlying problem graph. In the next section we will first discuss PMBGAs. Section 3 will introduce the graph bipartitioning problem and Section 4 discusses the application of the bivariate PMBGA to this problem. Section 5 presents some experimental results and finally section 6 concludes the paper.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

2. PMBGA

During the GA search the solutions present in the population will progressively consist of ever better partial solutions. Selection increases the frequencies of these partial solutions in the population. In a traditional genetic algorithm it is the role of crossover to use these partial solutions to build new promising candidates. Basically we can distinguish two different ways for the crossover operator to exploit the current solutions. The first method is that crossover preserves the common partial structure of the two parent solutions. At the part of the solution where the two parents disagree a random sample is generated. This mechanism uses crossover purely as a biased sampling operator. The commonalities of the parents identify the subspace of the search space where promising new solutions may be sampled. We call this the type I search bias. The second method (type II search bias) is that crossover juxtaposes two different partial solutions from the two parents and integrates them in a new child solution. These two exploration mechanisms represent the inductive search bias of crossover for on a given representation. If the search bias is well matched to the underlying problem then the search process will be successful. Note that the two exploration methods are not mutually exclusive. It might well be that both mechanisms are being exploited during the search. If the search problem is such that the first exploration method is sufficient then in general it will not be too hard to select a good crossover operator and solution representation. If the search problem requires the more complex second exploration method then it might be rather challenging to design a suitable recombination operator and solution representation. The potential problem with recombination is that the crossover is too disruptive. Instead of mixing the partial solutions in new candidate solutions the partial solutions are being destroyed too often.

One possible way to circumvent this problem is the use of so called Probabilistic Model-building GAs (PMBGAs). PMBGAs replace the recombination of individual solutions by building a probabilistic model of selected solutions and generate new candidates by sampling from this distribution. The idea behind PMBGAs is that the probabilistic model represents the dependencies present in the partial solutions and that by sampling from the model these dependencies are now being respected.

To summarize, PMBGAs are based on the following observations:

- Probability distributions can be used to model dependencies between random variables from the partial solutions.
- Selection will make these fitness-based dependencies between the problem variables stand out.
- By estimating a probability distribution over the selected solutions, the dependencies are identified.
- Drawing new samples from the estimated probability distribution will respect the dependencies and therefore not disrupt the partial solutions.

3. GRAPH BIPARTITIONING

A combinatorial optimization problem is specified by a finite set of solutions S and a cost function f that assigns a numerical value to each solution: $f : S \rightarrow \Re$. The goal in combinatorial optimization is to find a solution s^* that is globally optimal. A solution s^* is called globally optimal if there exist no other solutions $s \in S$ that have a better cost value. For many combinatorial optimization problems there exist no polynomial-time algorithm to solve them. A particularly successful method to tackle these problems is multi-start local search. Local search algorithms iteratively try to improve the current solution by applying small changes. Only a limited set of solutions - the neighborhood set - can be reached from the current solution. Local search explores the neighborhood of the current solution and if a better solution is found this will become the new current solution. The search continues by exploring the neighborhood of the new solution. When no improvement is found in the neighborhood of the current solution the search process stops. In a multi-start local search (MLS) setting the search will be restarted from another random solution.

One way to improve on MLS is the use of a genetic algorithm to generate the new starting points of the local search. This method is known as genetic local search (GLS), hybrid genetic algorithms. or memetic algorithms. In this paper we want to investigate the use of a bi-variate PMBGA to generate the starting points of the local search. We are specifically interested in so called grouping problems like graph bipartitioning and graph coloring problems.

Here we will look at the graph bipartitioning problem. Assume an undirected graph with the set of vertices V and set of edges E. The number of vertices $|V| = n$ is even. The graph bipartitioning problem is to find a partitioning of the set of vertices V into 2 subsets A and B of equal size ($|A| = |B|$), such that the number of edges between vertices of A and B is minimal. Note that $(|A| = |B|) \wedge (A \cup B = V) \wedge (A \cap B = \emptyset)$ The task in the graph bipartitioning problem is to minimize the number of edges that are cut by a partitioning A and B. An efficient local search algorithm is the Fiduccia-Mattheyses (FM) heuristic [5]. When implemented efficiently FM adds, removes and replaces nodes in bucket arrays in constant time.

4. BI-PMBGA & GRAPH BIPARTITIONING

The bivariate PMBGA (Bi-PMBGA) models the pairwise dependencies between the random variables from the solution [10].

Although the graph bipartitioning problem might look like a positional defined problem it is really a grouping problem. As an example consider a problem with 10 vertices. Solutions to the graph bipartitioning problem can be represented by a binary string, for instance $s1$: 0000011111 and $s2$: 0011100011. Solution $s1$ has nodes {1,2,3,4,5} in set 0 and nodes {6,7,8,9,10} in set 1. Solution $s2$ has nodes {1,2,6,7,8} in set 0 and nodes {3,4,5,9,10} in set 1. Uniform crossover will preserve the set membership of nodes {1,2} and {9,10}. However the actual name of the partitioning set does not matter. If we would call in solution $s2$ {1,2,6,7,8} set 1 and {3,4,5,9,10} set 0 then uniform crossover will preserve the set membership of nodes {3,4,5} and {6,7,8}. Since we want to maximize the amount of information passed from the parents to the children we first calculate the hamming distance between the two parents and when this is larger than half the string length we take the binary complement of one of the solutions before crossing them.

The binary representation of a grouping problem poses a problem for a standard bivariate PMBGA. Calculating the pairwise frequencies of 0 and 1 in the population ignores the redundancy in our representation.

To take this into account we do not count the pairwise frequencies of 0 and 1 but we count the number of times each pair of nodes belongs to the same group in the selected solutions. Using these frequencies we can estimate the probability p_{ij} that two nodes i and j are in the same partitioning.

A bivariate PMBGA builds a dependency tree expressing the most significant probability values. In the graph bipartitioning problem the information that two nodes have a high probability of being in the same group is equally valuable than the information that two nodes have a very low probability of being in the same group. Therefore the dependency tree should be build by focusing on the largest and smallest probability values. The dependency values are calculated by taking the maximum of $(p, 1 - p)$.

A bi-variate PMBGA has to compute all the pairwise frequencies between all variables. It has therefore a quadratic complexity in the number of nodes. To reduce this complexity we restrict the dependency tree to the pairwise interactions between vertices that are also connected in the graph to be partitioned. This way the computational complexity is no longer quadratic in the number of vertices but linear in the number of edges. Note that this is also the computational complexity of the FM heuristic.

Building a dependency tree is equivalent to building a maximum spanning tree. Two well know algorithms can be used for this: Prim's algorithm and Kruskal's algorithm [8]. Here we have applied Kruskal's algorithm that builds the dependency tree by adding those edges with the highest dependency values that do not introduce a cycle to the current subtrees already selected. Kruskal's algorithm first sorts the edges in order of increasing weight. Since we only need the $n - 1$ edges with the largest dependencies we do not sort all the edges but place them into a heap datastructure. Using the heap we can keep asking for the next edge with maximum dependency $max(p, 1 - p)$ until all nodes are covered by the tree. Checking to see if an edge does not introduce a cycle is implemented efficiently by using a union-find algorithm [8].

Table 1: multi-start local search, genetic local search, and the bi-variate probabilistic model-building GA: each generating 1000 local optima.

	MLS		GLS		Bi-PMB	
	mean	std	mean	std	mean	std
U500.05	8.62	1.90	4.24	1.35	3.62	0.98
U500.10	26.06	0.42	26.02	0.14	26	0
U500.20	178	0	178	0	178	0
U500.40	412	0	412	0	412	0
G500.005	53.28	0.85	51.3	0.46	51.48	0.57
G500.01	223.52	1.66	218.36	0.84	219.2	1.48
G500.02	630.64	1.48	627.68	1.05	627.94	1.17
G500.04	1749.98	2.45	1745.54	1.51	1746.5	1.85

Table 2: two-tail p-values for the unpaired t-test: values smaller than 0.05 indicate a statistical significant difference between the mean values of the best local optima found

	MLS/GLS	MLS/Bi-PMB	GLS/Bi-PMB
U500.05	**< 0.001**	**< 0.001**	**0.010**
U500.10	0.524	0.315	0.315
U500.20	-	-	-
U500.40	-	-	-
G500.005	**< 0.001**	**< 0.001**	0.085
G500.01	**< 0.001**	**< 0.001**	**< 0.001**
G500.02	**< 0.001**	**< 0.001**	0.245
G500.04	**< 0.001**	**< 0.001**	**0.005**

5. EXPERIMENTAL RESULTS

To test our Bi-PMBGA implementation for graph bipartitioning we take the widely used U500.d and G500.p test graphs. They consist of 500 vertices and have a geometric -resp. random - topology. The 4 geometric graphs U500.d have their vertices randomly chosen within the unit square and vertices within a distance smaller or equal than $\frac{d}{500\pi}$ are connected. This way the expected vertex degree is equal to d. The 4 random graphs G500.p have with probability p an edge between any pair of vertices. Their expected degree is therefore $p(500-1)$.

5.1 Performance

To test how well the Bi-PMBGA performs versus the multi-start local search (MLS) and the genetic local search (GLS) we allowed each algorithm to generate 1000 local optima. The GLS is a steady-state GA with population size 50, parents are selected randomly, the offspring solution replaces the current worst solution of the population if the offspring has a better fitness. The Bi-PMBGA has a population of size 100. The dependency tree is constructed from the 50 best solutions, 50 new solutions are being sampled and the best of the parents and offspring are used to construct the next dependency tree. Note that all solutions are obtained by applying the FM local search to the offspring solution. Table 1 shows the mean and standard deviation of the best partitioning averaged over 50 independent runs. Table 2 shows the two tail p-value for the unpaired t-test. Results smaller than 0.05 (shown in bold) are considered to be statistically significant.

A number of observations can be made:

- For the geometric graphs U500.10, U500.20, and U500.40 the local search heuristic FM has no trouble finding the optimal solution. It does not matter what the starting solution for FM was. Only for the graph U500.10 MLS and GLS did not find the optimal solution in all 50 runs but as shown in table 2 these results are not statistically significantly different from the results obtained by Bi-PMBGA.
- For the geometric graph U500.05 the results are statistically different. Both GLS and Bi-PMBGA outperform MLS. Bi-PMBGA also outperforms GLS.
- For the random graphs G500.p GLS and Bi-PMBGA outperformed MLS significantly. Clearly FM has more trouble with the random G500.p graphs than with the more structured geometric U500.d graphs.
- For graphs G500.005 and G500.02 GLS and Bi-PMBGA perform the same.
- For graphs G500.01 and G500.04 GLS outperforms Bi-PMBGA but the differences are rather small.

Table 3: multi-start local search, genetic local search, and the bi-variate probabilistic model-building GA: same run time.

	MLS		GLS		Bi-PMB	
	mean	std	mean	std	mean	std
U500.05	8.54	1.71	3.48	1.39	3.62	0.98
U500.10	26	0	26	0	26	0
U500.20	178	0	178	0	178	0
U500.40	412	0	412	0	412	0
G500.005	53.5	0.83	51.14	0.35	51.48	0.57
G500.01	223.48	1.71	218.4	1.11	219.2	1.48
G500.02	630.62	1.77	627.46	0.94	627.94	1.17
G500.04	1749.38	2.49	1745.34	1.35	1746.5	1.85

5.2 Efficiency

The previous subsection looked at the performance of MLS, GLS, and Bi-PMBGA when given the same number of calls to the FM local search operator. However each generation Bi-PMBGA needs to build a dependency tree which is obviously more computationally demanding than simply generating a random starting solution (MLS) or uniformly crossing two solutions (GLS). To compare the computational efficiency of the 3 algorithms we have compared them when given the same run time. The run time allowed is the time the Bi-PMBGA needed to generate the 1000 local optima in the previous set of experiments. On average the MLS can generate about 1250 local optima during this time while the GLS can explore about 1500. Note that GLS is faster than MLS because the FM local search operator will reach a local optimum much faster when started from a solution obtained by uniform crossover than from a randomly generated solution.

Table 4: two-tail p-values for the unpaired t-test: values smaller than 0.05 indicate a statistical significant difference between the mean values of the best local optima found

	MLS/GLS	MLS/Bi-PMB	GLS/Bi-PMB
U500.05	**< 0.001**	**< 0.001**	0.562
U500.10	-	-	-
U500.20	-	-	-
U500.40	-	-	-
G500.005	**< 0.001**	**< 0.001**	**< 0.001**
G500.01	**< 0.001**	**< 0.001**	**0.003**
G500.02	**< 0.001**	**< 0.001**	**0.026**
G500.04	**< 0.001**	**< 0.001**	**< 0.001**

Table 3 shows the mean and standard deviation of the best partitioning averaged over 50 independent runs. Table 4 shows the two tail p-value for the unpaired t-test. Results smaller than 0.05 (shown in bold) are considered to be statistically significant.

A number of observations can be made:

- For the geometric graph U500.05 the difference in performance between GLS and the Bi-PMBGA now disappears. GLS makes up for its lack of performance relative to the Bi-PMBGA by exploring about 50% more local optima in the same computing time.
- For the random graphs G500.p MLS is still outperformed by GLS and Bi-PMBGA.
- GLS now outperforms Bi-PMBGA for all 4 G500.p graphs although the differences remain small.

Summarizing, there are a number of things to conclude:

- For those graphs were MLS was not able to find the optimal solution reliably it was shown that adding a metaheuristic to generate candidate starting solutions for the FM local search heuristic improves performance significantly.
- Only on the geometric graph U500.05 Bi-PMBGA outperforms MLS given the same number of local optima (= 1000) generated.
- When given the same computational time MLS slightly outperforms Bi-PMBGA. Viewing the ease of implementation of GLS versus the complex implementation of Bi-PMBGA the algorithm of choice should be the GLS. Apparently for the graph-bipartitioning problems investigated here the inductive search bias of uniform crossover that preserves the common groupings in two parent solutions is all what is needed to improve on MLS. As discussed in section 2 PMBGAs are particularly interesting with problems requiring a type I search bias. Considering the success of uniform crossover in the GLS algorithm it can be argued that for the graph bipartitioning problems considered here a type II search bias is sufficient to solve the problems efficiently. The added complexity of building a bivariate dependency model of selected solutions does not result in significant performance gains to offset the increased computational complexity.

6. CONCLUSION

We have investigated a bi-variate probabilistic model building GA for the graph bipartitioning problem. The graph bipartitioning problem is a grouping problem which necessitates the adaptation of the procedure to build the dependency tree. To increase the computational efficiency we restricted the dependency tree to the edges of the actual graph. Experimental results indicate that the Bi-PMBGA performs well as a metaheuristic to improve on multi-start local search using the FM local search heuristic. However a comparison to a genetic local search algorithm indicated that possible performance gains might be leveled out due to the increased computational overhead.

7. REFERENCES

[1] Baluja, S. & Davies, S. Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. *Proceedings of the 14th International Conference on Machine Learning*, pp. 30–38, 1997. Morgan Kaufmann.

[2] Bosman, P.A.N. & de Jong, E. D. Learning Probabilistic Tree Grammars for Genetic Programming. *Proceedings of Parallel Problem Solving from Nature - PPSN VIII*, pp. 192–201, 2004. Springer-Verlag.

[3] Bosman, P.A.N. & and Thierens, D. Permutation Optimization by Iterated Estimation of Random Keys Marginal Product Factorizations. *Proceedings of Parallel Problem Solving From Nature - PPSN VII*, pp. 331–340, 2002. Springer-Verlag.

[4] Falkenauer, E. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, Vol.2, pp. 5–30, 1996. Kluwer Academic Publishers.

[5] Fiduccia, C.M. & Mattheyses, R.M. A Linear-Time Heuristic for Improving Network Partitions. *Proceedings of the 19th Conference on Design Automation*, pp. 175–181, 1982. IEEE Press.

[6] Johnson, D.S. & Aragon, C.R. & McGeoch, L.A. & Schevon, C. Optimization by simulated annealing: an experimental evaluation, part I (graph partitioning). *Operations Research* 37, pp. 865–892, 1989.

[7] Kim, Y.K. & Moon, B.R. Investigation of the Fitness Landscapes in Graph Bipartitioning: An Empirical Study. *Journal of Heuristics*, Vol. 10, Nr. 2, pp. 111–133, 2004. Kluwer Academic Publishers.

[8] Levitin, A. *The Design and Analysis of Algorithms.* 2nd edition, 2007, Pearson.

[9] Merz, P. & Freisleben, B. Fitness Landscapes, Memetic Algorithms, and Greedy Operators for Graph Bipartitioning. *Journal of Evolutionary Computation*, Vol. 8, Nr. 1, pp. 61–91, 2000. MIT Press.

[10] Pelikan. M. & Mühlenbein, H. The Bivariate Marginal Distribution Algorithm. *Proceedings of Advances in Soft Computing - Engineering Design and Manufacturing*, pp. 521–535, 1999. Springer-Verlag.

[11] Pelikan, M. & Goldberg, D.E. & Lobo, F. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, Vol. 21, Nr. 1, pp.5–20, 2002. Kluwer.