Evolutionary Algorithms based on Probabilistic Modeling

CHUNG-YAO CHUANG, Carnegie Mellon University
HSIEN-TANG KAO, Carnegie Mellon University

Evolutionary algorithms (EAs) are population-based meta-heuristic optimization algorithms that imitate the process of natural evolution. EAs generate solutions to optimization problems using mechanisms inspired by biological evolution, such as selection, reproduction, mutation and recombination. Recently, there is a growing interest in developing EAs that utilize probabilistic modeling in their search mechanism. These approaches are called Estimation of Distribution Algorithms (EDAs). By using probabilistic models to summarize the information, advanced EDAs are able to incorporate techniques from machine learning and statistics to automatically discover the multivariate interactions between problem variables, which leads to an approximation of problem decomposition and recognition of important substructures that constitute the promising solutions. For this project, we implement an EDA called Extended Compact Genetic Algorithm (ECGA) and use it to study some aspects of this paradigm.

Categories and Subject Descriptors: I.2.6 [**Artificial Intelligence**]: Learning—*Parameter learning*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*; G.1.6 [**Numerical Analysis**]: Optimization

General Terms: Algorithms

Additional Key Words and Phrases: Estimation of distribution algorithms, EDAs, marginal product models, extended compact genetic algorithm, ECGA, evolutionary algorithms, genetic algorithm, evolutionary computation.

ACM Reference Format:
Chung-Yao Chuang and Hsien-Tang Kao, 2013. Evolutionary Algorithms based on Probabilistic Modeling. *ACM* V, N, Article A (May 2013), 11 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Evolutionary algorithms (EAs) [Goldberg 1989; De Jong 2006; Eiben and Smith 2008] are population-based meta-heuristic optimization algorithms that imitate the process of natural evolution. EAs generate solutions to optimization problems using mechanisms inspired by biological evolution, such as selection, reproduction, mutation and recombination. Candidate solutions to the optimization problem play the role of indi-

Author's addresses: Chung-Yao Chuang, Robotics Institute, Carnegie Mellon University; Hsien-Tang Kao, Department of Mechanical Engineering, Carnegie Mellon University.

DOI: http://dx.doi.org/10.1145/0000000.0000000

viduals in a population, and the fitness function determines the environment within which the solutions "live." The evolution usually starts from a population of randomly generated solutions and proceeds in iterations. In each iteration, the fitness of every solution in the population is evaluated, better solutions are stochastically selected from the current population, and each solution is modified (through mutation and recombination) to form a new population. The new population is then used in the next iteration of the algorithm.

Recently, there is a growing interest in developing EAs that utilize probabilistic modeling in their search mechanism. These approaches are called Estimation of Distribution Algorithms (EDAs) [Mühlenbein and Paaß 1996; Larrañaga and Lozano 2001; Pelikan et al. 2002]. EDAs introduce a new paradigm into the framework of evolutionary algorithms in that it replaces the traditional variation operators, such as mutation and crossover, with the procedure of building a probabilistic model on promising solutions, and sampling the constructed model to generate new candidate solutions. Early EDAs, such as the population-based incremental learning (PBIL) [Baluja 1994] and the compact genetic algorithm (cGA) [Harik et al. 1999], assume no interaction between problem variables, i.e., variables are assumed independent of each other. Subsequent studies started from capturing pairwise interactions, such as mutual-information-maximizing input clustering (MIMIC) [de Bonet et al. 1997], Baluja's dependency tree approach [Baluja and Davies 1997], and the bivariate marginal distribution algorithm (BMDA) [Pelikan and Mühlenbein 1999], to modeling multivariate interactions, such as the extended compact genetic algorithm (ECGA) [Harik 1999], the Bayesian optimization algorithm (BOA) [Pelikan et al.], and the estimation of Bayesian network algorithm (EBNA) [Etxeberria and Larrañaga 1999]. By using probabilistic models to summarize the information, advanced EDAs are able to incorporate techniques from machine learning and statistics to automatically discover the multivariate interactions between problem variables, which leads to an approximation of problem decomposition and the recognition of important substructures that constitute the promising solutions.

For this project, we focus on studying multivariate EDAs. Specifically, we implemented the Extended Compact Genetic Algorithm (ECGA) [Harik 1999] and used our implementation to explore the capability and limitation of this meta-heuristic optimization paradigm. One thing we have noticed about the behavior of ECGA (and from the literature [Hauschild and Pelikan 2011; Yuan and Gallagher 2005; Lozano 2006], it's a characteristic/problem of other EDAs in general) is that the diversity of the population is decreased much faster comparing to the conventional GAs. While this could mean that ECGA is more efficient in optimizing the solutions, it might also correspond to a greater chance of premature convergence. In this project, we propose a diversity maintenance method that utilizes the information contained in the probabilistic models built by ECGA. To assess the effectiveness of our method, we performed experiments to compare our approach with a simple GA, and the original ECGA on the minimum vertex cover (MVC) problem. The instances of MVC problem were adopted

ALGORITHM 1: General Outline of Estimation of Distribution Algorithms

```
Initialize a population P with n solutions.
while the stopping criteria are not met do
  Evaluate the solutions in P.
  P' ← apply selection on P.
  M ← build a probabilistic model on P'.
  O ← generate new candidate solutions by sampling M.
  Incorporate O into P.
    end
```

from DIMACS benchmark set[1]. The results show that with our diversity maintenance mechanism, the modified ECGA performs better than the simple GA and the original ECGA.

The rest of this report is organized as follows: In Section 2, we review the background of EDAs. Following that, Section 3 describes probabilistic model used in ECGA and its model building mechanism. In Section 4, a method for maintaining diversity in ECGA is proposed. Section 5 introduce the minimum vertex cover problem and the fitness function for GAs. Section 6 presents and discuss the performance of GA, ECGA and ECGA-DM on a set of benchmark instances. Section 7 presents the conclusion of the paper with some observations and findings.

2. ESTIMATION OF DISTRIBUTION ALGORITHMS

Similar to other evolutionary algorithms, the procedure of EDAs, as listed in Algorithm 1, starts from initializing a population of solutions which can be randomly generated or produced with some heuristics. Then, the algorithm iterates through the selection process, which picks a set of promising solutions from the current population, and the creation of new candidate solutions, which recombines the set of selected solutions to form new solutions. The prominent feature of EDAs is that the recombination of solutions is done by building and sampling probabilistic models. From an abstract perspective, the selected set of solutions can be viewed as samples drawn from an unknown probability distribution. Knowing that distribution would allow the optimization method to generate new solutions that are somehow similar to the ones contained in the original selected set of promising solutions.

Although the actual probability distribution is unknown, there are techniques capable of estimating that distribution by using the set of selected solutions. Using these estimation techniques, we can obtain probabilistic models that reflect the possible formation of good solutions. In other words, an accurate distribution estimate is able to capture the structure of good solutions found so far. And by sampling this distribution, we can ensure an effective mixing and reproduction of those good structures constituting the promising solutions.

[1]We obtained those instances from `http://cs.hbg.psu.edu/benchmarks/vertex_cover.html`, which were converted from maximum clique problems.

Table I. An example of marginal product model

$[X_1]$	$[X_2\ X_4]$	$[X_3]$
$P(X_1 = 0) = 0.4$	$P(X_2 = 0, X_4 = 0) = 0.4$	$P(X_3 = 0) = 0.5$
$P(X_1 = 1) = 0.6$	$P(X_2 = 0, X_4 = 1) = 0.1$	$P(X_3 = 1) = 0.5$
	$P(X_2 = 1, X_4 = 0) = 0.1$	
	$P(X_2 = 1, X_4 = 1) = 0.4$	

An example of marginal product model that defines a joint distribution over four variables. The variables enclosed in the same brackets are considered dependent and modeled jointly. Each variable subset is considered independent of other subsets.

Historically, EDAs were grown out of studying the limitation of genetic algorithms (GAs). GAs mimic the mechanism of natural evolution by introducing artificial selections and genetic operators to discover and recombine partial solutions. By properly growing and mixing promising partial solutions, which are often referred to as building blocks [Goldberg 1989; 2002], GAs are capable of efficiently solving a host of problems. The ability of implicitly processing a large number of partial solutions has been recognized as an important source of the computational power of GAs. Unfortunately, proper growth and mixing of building blocks are not always achieved. GA in its simplest form employing fixed representations and problem-independent recombination operators often breaks the promising partial solutions while performing crossovers. This can cause crucial building blocks to vanish during the optimization process.

In order to overcome this *building block disruption problem*, EDAs employ the concept of constructing probabilistic models on promising solutions to identify the relationship among partial solutions. Higher-order EDAs, such as ECGA [Harik 1999], Bayesian optimization algorithm (BOA) [Pelikan et al.], and the estimation of Bayesian network algorithm (EBNA) [Etxeberria and Larrañaga 1999], are able to discover multivariate interaction exhibited in the promising solutions. Using this information, EDAs can recognize important substructures that constitute the promising solutions, thus avoid the building block disruption problem.

3. EXTENDED COMPACT GENETIC ALGORITHM

The extended compact genetic algorithm (ECGA) [**?**] uses a product of marginal distributions on a partition of the variables. This kind of probability distribution belongs to a class of probabilistic models known as marginal product models (MPMs). In this kind of model, subsets of variables can be modeled jointly, and each subset is considered independent of other subsets. In this work, the conventional notation is adopted that variable subsets are enclosed in brackets. Table I presents an example of MPM defined over four variables: X_1, X_2, X_3 and X_4. In this example, X_2 and X_4 are modeled jointly and each of the three variable subsets ($[X_1]$, $[X_2\ X_4]$ and $[X_3]$) is considered independent of other subsets. For instance, the probability that this MPM generates a

sample $X_1X_2X_3X_4 = 0101$ is calculated as follows,

$$\begin{aligned} &P(X_1X_2X_3X_4 = 0101) \\ &= P(X_1 = 0) \times P(X_2 = 1, X_4 = 1) \times P(X_3 = 0) \\ &= 0.4 \times 0.4 \times 0.5. \end{aligned}$$

In fact, as its name suggests, a marginal product model represents a distribution that is a "product" over the marginal distributions defined over variable subsets.

In the ECGA, both the structure and the parameters of the model are searched and optimized with a greedy approach to fit the statistics of the selected set of promising solutions. The measure of a good MPM is quantified based on the minimum description length (MDL) principle [Rissanen 1978], which assumes that given all things are equal, simpler distributions are better than complex ones. The MDL principle thus penalizes both inaccurate and complex models, thereby, leading to a near-optimal distribution. Specifically, the search measure is the MPM complexity which is quantified as the sum of model complexity, C_m, and compressed population complexity, C_p. The greedy MPM search first considers all variables as independent and each of them forms a separate variable subset. In each iteration, the greedy search merges two variable subsets that yields the most $C_m + C_p$ reduction. The process continues until there is no further merge that can decrease the combined complexity.

The model complexity, C_m, quantifies the model representation in terms of the number of bits required to store all the marginal distributions. Suppose that the given problem is of length ℓ with binary encoding, and the variables are partitioned into m subsets with each of size k_i, $i = 1 \dots m$, such that $\ell = \sum_{i=1}^{m} k_i$. Then the marginal distribution corresponding to the ith variable subset requires $2^{k_i} - 1$ frequency counts to be completely specified. Taking into account that each frequency count is of length $\log_2(n+1)$ bits, where n is the population size, the model complexity, C_m, can be defined as

$$C_m = \log_2(n+1) \sum_{i=1}^{m} \left(2^{k_i} - 1\right) \ .$$

The compressed population complexity, C_p, quantifies the suitability of the model in terms of the number of bits required to store the entire selected population (the set of promising solutions picked by selection operator) with an ideal compression scheme applied. The compression scheme is based on the partition of the variables. Each subset of the variables specifies an independent "compression block" on which the corresponding partial solutions are optimally compressed. Theoretically, the optimal compression method encodes a message of probability p_i using $-\log_2 p_i$ bits. Thus, taking into account all possible messages, the expected length of a compressed message is $\sum_i -p_i \log_2 p_i$ bits, which is optimal. In the information theory [Cover and Thomas 1991], the quantity $-\log_2 p_i$ is called the *information* of that message and $\sum_i -p_i \log_2 p_i$ is called the *entropy* of the corresponding distribution. Based on the information the-

Table II. Illustration of MPMs built by ECGA when solving concatenated 4-bit trap function

Generation	Marginal Product Model
1	[1 2 3 4] [5 6 7 8] [9 10 11 12] [13 14 15 16]
2	[1 2 3 4] [5 6 7 8] [9 10 11 12] [13 14 15 16]
3	[1 2 3 4] [5 6 7 8] [9 10 11 12] [13 14 15 16]
⋮	... ...

Marginal product models built by ECGA when solving concatenated 4-bit trap problem. The variables are denoted by their indexes. Each group of variables represents a marginal model in which a marginal distribution resides.

ory, the compressed population complexity, C_p, can be derived as

$$C_p = n \sum_{i=1}^{m} \sum_{j=1}^{2^{k_i}} -p_{ij} \log_2 p_{ij} \ ,$$

where p_{ij} is the frequency of the jth possible partial solution to the ith variable subset observed in selected population.

To see how ECGA avoid the building block disruption problem, consider a 4-bit trap function that takes a binary string of length 4,

$$f_{trap_4}(x_1 x_2 x_3 x_4) = \begin{cases} 4, & \text{if u = 4;} \\ 3 - u, & \text{otherwise.} \end{cases} \ ,$$

where u is the number of ones in the string $x_1 x_2 x_3 x_4$. Suppose that we are dealing with a 16-bit maximization problem formed by concatenating four 4-bit trap functions,

$$f(s_1 s_2 \cdots s_{16}) = \sum_{i=0}^{3} f_{trap_4}(s_{4i+1} s_{4i+2} s_{4i+3} s_{4i+4}) \ ,$$

where $s_1 s_2 \cdots s_{16}$ is a solution string and the variables to the same sub-function are considered as strongly related. Assume that we use ECGA to tackle this problem. By observing subsequent generations of the optimization process, a series of models built by ECGA can be obtained like those listed in Table II. From the table, we can see that the built model capture the structure of the problem properly from the first generation (i.e., the model building algorithm reasoned a probabilistic model that has the structure consistent with the decomposition of the problem.) and it continued to identify the building blocks correctly. By sampling the probabilistic model that is consistent with the problem structure, new candidate solutions can be generated in a building-block-wise way. As a result, the building block disruption can be avoid.

4. DIVERSITY MAINTENANCE IN ECGA

One situation that directly affects the EDAs' ability to find the best solutions is the premature convergence to some local optimum due to diversity loss. Diversity loss is a

problem that affects the exploration ability of evolutionary algorithms in general, and increasing the diversity of a population can help such algorithms to better explore the space of candidate solutions and avoid stagnation in some local optimum.

However, if we introduce diversity blindly, the resulting new solutions are often of low fitness value and subsequently not selected to be included in the parent population. This is because the chance is usually very small that a randomly drawn solution will be better than the solutions in the current population. Thus, we think that it may be more beneficial to place our diversifying samples around the current promising region than spending our search effort on remote, and possibly low fitness area. In this way, the introduced diversity has a greater chance to be incorporated into the parent population.

This perspective fits nicely into the framework of EDA, because the probabilistic model built by EDA can be seen as a description of our current belief of where the promising region is. To produce the desired allocation of samples, i.e., dense around the current promising region and sparse in the distant area, we can generate samples using a distribution that is slightly diverted from the one encoded in the probabilistic model to cover the vicinity of the current promising region. In ECGA, we can achieve this in a principled way: assuming the MPM is composed of m marginal models (i.e., a solution is decomposed into m substructures), we can devise an "exploration" mechanism that creates some of the substructures in a solution *not* by sampling the constructed MPM, but by generating them uniformly randomly.

Of course, any good search algorithm will try to balance between exploration and exploitation. In ECGA, exploitation can be defined as sampling the constructed MPM, which corresponds to placing samples in the current promising region. To incorporate our mechanism of exploration, we introduce a tunable trade-off between exploration and exploitation. Suppose an "exploitation rate" ξ ($0 \leq \xi \leq 1$) is specified, then it could be saying that in the next generation, approximately ξ of the population should be created by sampling the MPM. Now let the probability of invoking the exploration mechanism be p, we have that

$$(1-p)^m = \xi.$$

which says that the probability that a solution is created entirely according to the built model is ξ. Base on that, we can derive

$$p = 1 - \xi^{1/m}.$$

In implementation, when composing a new solution, most of the time we generate each substructure by sampling the corresponding marginal model in the MPM, but with probability p as derived above (based on specified ξ), we place a partial solution that is generated uniformly randomly for that substructure. In this way, we can control the trade-off between exploration and exploitation (by specifying ξ), and make the "exploration points" dense near the current promising region but sparse in more distant area (on average, only mp substructures in a solution are not generated according to the constructed MPM.)

5. MINIMUM VERTEX PROBLEM

In order to examine the performance of ECGA, we runned the algorithm on minimum vertex cover(MVC) problem. A vertex cover of a graph is a set of verteces that includes at least one endpoint of each edge and a minium vertex cover is the vertex cover with smallest size. The minimum vertex problem belongs to the class of NP-hard optimization problem and its corresponding decision problem, vertex cover problem, is NP-complete.

The minimum vertex problem is formulated as following: consider a graph $G = (V, E)$ where $V = \{1, 2, ..., n\}$ is the set of vertices and E denotes the set of edges. There exists a smallest subset $V' \subseteq V$ such that $\forall \langle i, j \rangle \in E$, we have $i \in V'$ or $j \in V'$. V' is said to be a vertex cover of graph G. The objective of minimum vertex cover problem is to minimize the size $|V'|$ of the vertex cover and the optimal solution is a vertex cover V' that minimizes $|V'|$.

In genetic algorithm implementation, a vertex cover V' of the problem can be defined as a binary vector

$$V' = \vec{x} =< x_1, x_2, ..., x_n > .$$

where n is the number of vertices $|V|$ and

$$x_i = \begin{cases} 1, & \text{if the } i\text{th vertex is in vetex cover } V' \\ 0, & \text{if the } i\text{th vertex is not in vertex cover } V' \end{cases}$$

It is well known that choosing a proper fitness function is important for the efficiency and effectiveness of genetic algorithm. In this paper, we define the fitness function as:

$$f(\vec{x}) = \sum_{i=1}^{n} (x_i + \gamma(1 - x_i) \sum_{j=i}^{n} (1 - x_j) e_{ij}).$$

where γ is weighting factor and

$$e_{ij} = \begin{cases} 1, & \text{if } \langle i, j \rangle \in E \\ 0, & \text{if } \langle i, j \rangle \notin E \end{cases}$$

The first term $\sum_{i=1}^{n} x_i$ in the fitness function represents the size of vertex cover and the remaining term penalize the vertices that are not vertex cover by a factor of γ. It can be shown that if $|V'|$ is the minimum vertex cover, then $f(V') = |V'|$ and V' is the global minimum of fitness function. The penality of the fitness increased as the solutions departed from the minimum vertex cover.

Table III shows an example of marginal product model of graph (fig. 1) built in the first iteration. Based on the probabilistic model, the original graph are divided into four subgraph. It was observed that in order to solve the minimum vertex problem, a better way to generate candidate solutions is to sample from probabilistic model rather than arbitrarily choose the crossover point or mutation. The probabilistic approach took the whole graph structure into account and have higher probability to generate the better candidate solution for the optimization problem.

Table III. An example of marginal product model built in the first iteration

[1 2 3 4 5]	[6 7 8 9 10]	[11 12 13 14 16]	[15]
P(1 0 1 1 1) = 0.125000	P(1 0 1 1 1) = 0.092250	P(1 0 1 1 1) = 0.110500	P(0) = 0.543750
P(1 1 0 1 1) = 0.083250	P(1 0 0 1 1) = 0.087250	P(1 1 0 1 1) = 0.088750	P(1) = 0.456250
P(1 1 1 1 1) = 0.079500	P(1 1 1 0 0) = 0.083500	P(1 0 1 1 0) = 0.086250	
P(0 1 1 1 1) = 0.072500	P(1 1 0 1 0) = 0.071250	P(1 0 0 1 1) = 0.079500	
P(0 1 1 1 0) = 0.062000	P(1 1 1 1 1) = 0.069000	P(0 1 1 1 1) = 0.076000	
P(1 1 1 0 0) = 0.061500	P(0 1 1 1 1) = 0.067750	P(1 1 1 0 1) = 0.075250	
P(1 0 0 1 1) = 0.050000	P(0 1 1 0 1) = 0.067000	P(0 1 0 1 1) = 0.070250	
P(1 0 1 0 1) = 0.042250	P(1 1 1 1 0) = 0.064500	P(1 1 1 1 0) = 0.069250	
P(1 1 0 0 1) = 0.039500	P(1 1 0 1 1) = 0.060750	P(1 1 1 1 1) = 0.066500	
...	...	...	

The table shows the marginal product model that build in the first iteration which has lowest combined complexity. The variables represent the probability calculated after population initialization.

Table IV. Experimental results of different genetic algorithms using DIMACS benchmark instances

	Vertex	Edge	MVC	GA			ECGA			ECGA-DM		
				Success	Mean	Std Dev	Success	Mean	Std Dev	Success	Mean	Std Dev
hamming6-2	64	192	32	0	44.033	33.6905	0	39.6	3.2547	28	32.5	2.0129
hamming6-4	64	1312	60	12	60.767	0.7279	22	60.267	0.4498	30	60	0
johnson8-2-4	28	168	24	17	24.433	0.5040	29	24.033	0.1826	30	24	0
johnson8-4-4	70	560	56	0	61.833	0.8339	0	60.233	1.3309	17	57.5	1.8708
johnson16-2-4	120	1680	112	9	112.733	0.5208	22	112.267	0.4498	30	112	0

The variables in the table compared the number of sccesses, mean of vertex cover, standard deviation of vertex cover with different genetic algorithms (GA, ECGA and ECGA-DM) and DIMACS benchmark instances (hamming and johnson).

6. EXPERIMENTAL RESULTS

In our experiments, we describe the results by running GA, ECGA and ECGA-DM on DIMACS datasets maintained by Nguyen and Bui at PSU. The test instance Hamming and Johnson graphs are constructed based on algebraic coding theory and information theory. Our genetic algorithms were implemented in Matlab and runned 30 times on each instance.

Table IV tested on the benchmark dataset demonstrates the usefulness of maintaining diversity throughout the run. The performance of ECGA is better than GA with higher success rate, lower mean of vertex cover and lower standard deviation of vertex cover in 30 test runs, which shows that proper probabilistic model building approach can solve some difficult problem than simple GA. The experimental results also show that the ECGA-DM algorithm performed better than ECGA. The diversity maintenance technique increased the algorithm's exploration space of candidate solution. It can avoid premature convergence and prevent the solution converged to local optimum.

7. CONCLUSION

In this study, we looked into the mechansim of ECGA and applied it to solve the minimum vertex cover problem. In addition, we proposed a method to maintain the population diversity in ECGA. With the proposed diversity maintenance mechansim, ECGA is better in avoiding premature convergence. The results show that our approach yields a better performance compared to both conventional GA and ECGA.

REFERENCES

S. Baluja. 1994. *Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*. Technical Report. Pittsburgh, PA, USA.

S. Baluja and S. Davies. 1997. Using Optimal Dependency-Trees for Combinational Optimization. In *ICML '97*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 30–38.

Thomas M. Cover and Joy A. Thomas. 1991. *Elements of information theory*. Wiley-Interscience, New York, NY, USA.

J. de Bonet, C. Isbell, and P. Viola. 1997. MIMIC: Finding Optima by Estimating Probability Densities. In *Advances in Neural Information Processing Systems*, Vol. 9. The MIT Press, 424–430.

Kenneth A De Jong. 2006. *Evolutionary computation: a unified approach*. The MIT Press.

Agoston E Eiben and James E Smith. 2008. *Introduction to evolutionary computing*. springer.

R. Etxeberria and P. Larrañaga. 1999. Global Optimization using Bayesian Networks. In *Proceedings of (CIMAF-99)*, A. Ochoa Rodriguez, M.R. Soto Ortiz, and R. Santana Hermida (Eds.). Habana, Cuba, 332–339.

David E Goldberg. 1989. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Professional.

David Edward Goldberg. 2002. *The design of innovation: Lessons from and for competent genetic algorithms*. Vol. 7. Springer.

G. Harik. 1999. *Linkage Learning via Probabilistic Modeling in the ECGA*. IlliGAL Report No. 99010. Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign.

G. R. Harik, F. G. Lobo, and D. E. Goldberg. 1999. The Compact Genetic Algorithm. *IEEE Transactions on Evolutionary Computation* 3, 4 (November 1999), 287.

Mark Hauschild and Martin Pelikan. 2011. An introduction and survey of estimation of distribution algorithms. *Swarm and Evolutionary Computation* 1, 3 (2011), 111–128.

Pedro Larrañaga and Jose A. Lozano. 2001. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Genetic algorithms and evolutionary computation, Vol. 2. Kluwer Academic Publishers, Boston, MA. ISBN: 0-7923-7466-5.

Jose A Lozano. 2006. *Towards a new evolutionary computation: Advances on estimation of distribution algorithms*. Vol. 192. Springer-Verlag New York Incorporated.

H. Mühlenbein and G. Paaß. 1996. From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. In *Proceedings of the PPSN IV*. Springer-Verlag, London, UK, 178–187.

Martin Pelikan, David E. Goldberg, and Erick Cantú-Paz. Vol. I. Morgan Kaufmann Publishers, San Fransisco, CA, Orlando, FL, 525–532.

Martin Pelikan, David E. Goldberg, and Fernando G. Lobo. 2002. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications* 21, 1 (2002), 5–20.

Martin Pelikan and Heinz Mühlenbein. 1999. The Bivariate Marginal Distribution Algorithm. In *Advances in Soft Computing - Engineering Design and Manufacturing*, R. Roy, T. Furuhashi, and P. K. Chawdhry (Eds.). Springer-Verlag, London, 521–535.

J. Rissanen. 1978. Modelling by shortest data description. *Automatica* 14 (1978), 465–471.

Bo Yuan and Marcus Gallagher. 2005. On the importance of diversity maintenance in estimation of distribution algorithms. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*. ACM, 719–726.

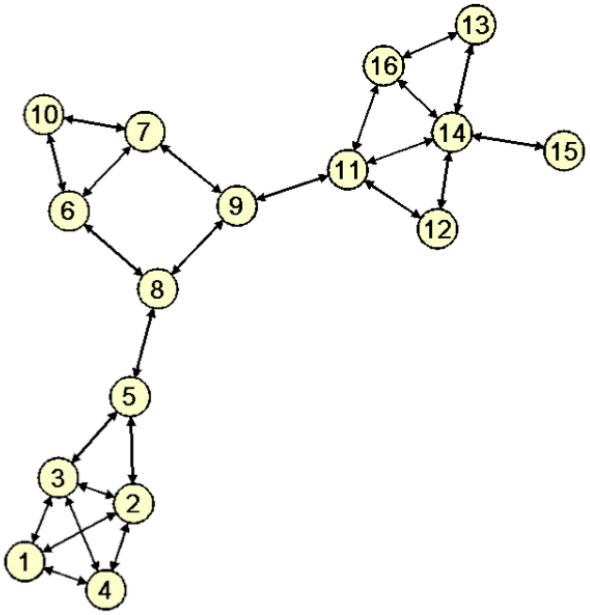

Fig. 1. Sample graph for Marginal Product Model Building