A Survey of Estimation of Distribution Algorithms

Tian-Li Yu

Illinois Genetic Algorithms Laboratory and
Department of Computer Science
University of Illinois at Urbana-Champaign
tianliyu@illigal.ge.uiuc.edu

Abstract

This paper reviews the research on estimation of distribution algorithms. Several different approaches are categorized according to the domain that they act on and the complexity of the class of probabilistic models. Their algorithms are briefly introduced and their strength and weakness of are discussed.

1 Introduction to EDAs

Evolutionary computation algorithms have many different crossover and mutation operations to choose, e.g. single-point and uniform crossover. An inexperienced user may choose inappropriate operators for his problem. For example, single-point crossover is not a wise choice when building blocks of the problem are not tightly located in encodings (Thierens, 1995). This observation is especially highlighted when prior knowledge is unavailable. Based on this observation, learning the distribution of building blocks (linkage information) on the fly becomes urgent. Moreover, linkage information of distribution can be utilized to improve the performance of evolutionary computation.

This thought motivated Estimation of Distribution Algorithms (EDAs). EDAs include a class of algorithms and were first introduced by Mühlenbein and Paaß (1996). The template of EDAs can be described as following.

1. Initialize population (usually randomly).
2. Select promising individuals.
3. Estimate the distribution of those promising individuals (probabilistic model building).
4. According to the distribution (model), sample new solutions, and then go to step 2 until the halting criterion is meet.

The probabilistic model directly decides the performance of EDAs. More accurate models ensure better mixing and reproducing while they are usually more complicated and expensive to build. Clearly, there is a trade-off between efficiency and accuracy. Generally speaking, when the cost of fitness function evaluation is high, a more complicated but accurate model can be freely adopted, since the overhead of constructing such a model is comparably cheap.

The following sections describe different approaches in EDAs. The categorization style basically follows the way in Pelikan, Goldberg, and Lobo (1999), Larrañaga (2002), and Sastry (2002). In fact, this paper is a summary and an extension of these three excellent reviews. Firstly those approaches are divided by two groups (discrete and continuous) based on their acting domain. Then they are more detailed categorized again according to the complexity of their probabilistic models.

2 EDAs on Discrete Domain

This section focuses on EDAs on discrete domain. In this category, every individual can be considered as a vector in which every element can be assigned to some finite possible values. A very simple but vivid example is the onemax problem. In onemax problem, each individual is a fixed-length binary valued vector.

The following subsections are arranged based on the interdependencies between variables in individuals. Specifically, they are no dependencies, bivariate dependencies, and multiple dependencies.

2.1 No dependencies

The work in this category assumes that every variable is independent. Thus, the probability of an individual is just the product of the probability of every variable.

$$P(\bar{x}) = \prod_{i=1}^{n} P(x_i), \text{ where } \bar{x} = x_1 x_2 \cdots x_n$$

This assumption usually does not hold for difficult optimization problems, but the model is simple. Therefore, algorithms of this class perform excellently for linear problem and fail on complex problem.

The first proposed algorithm of this class is the Population Based Incremental Learning (PBIL) by Baluja (1994). In this algorithm, a solution is represented by a

fixed-length binary vector. For every generation, a probability vector (the probabilistic model) where every element is initially set to be 0.5 (usually) is kept. Then after promising solutions are selected out of population, the probability vector is modified by a Hebbian inspired rule (Hertz, Krogh, & Palmer, 1991). The PBIL algorithm can be summarized into the following equations.

$$\begin{cases} \bar{p}^{(0)} = (0.5, 0.5, \cdots, 0.5) \\ \bar{p}^{(t+1)} = (1-a)\,\bar{p}^{(t)} + a\,\dfrac{1}{s}\displaystyle\sum_{i=1}^{s} \bar{x}_i^{(t)} \end{cases}$$

where $0 < a \le 1$, $\bar{x}_i$ are selected individuals, and s is the number of selected individuals.

Mühlenbein and Paaß (1996) introduced the Univariate Marginal Distribution Algorithm (UMDA). For each generation, the frequency is computed for each variable from the selected individuals. Then, use those frequencies as the probabilities of variables to generate new individuals. Note that UMDA is in fact a special case of PBIL where a=1.

Harik, Lobo, and Goldberg (1998) proposed the compact Genetic Algorithm (cGA), which also contributes to this class. Like PBIL and UMDA, cGA also keeps a probability vector. For every generation, cGA samples two individuals from the probability vector. Then a competition between them two is held to select the more promising solution. cGA updates the probability vector according to the selected promising solution.

$$\begin{cases} \bar{p}^{(0)} = (0.5, 0.5, \cdots, 0.5) \\ \bar{p}^{(t+1)} = \bar{p}^{(t)} + \dfrac{1}{K}(\bar{x} - \bar{p}^{(0)}) \end{cases}$$

These three algorithms, PBIL, UMDA, and cGA, have many common features. They all perform on fixed-length binary vectors, and they keeps a probability vector in which each element estimates the probability of the corresponding bit. Figure 1 shows the graphical representation of the probability models in this class.

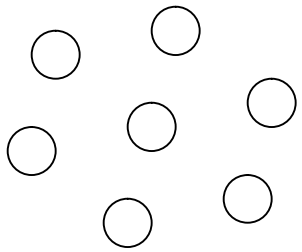

Figure 1 Graphical representation of probabilistic model without dependencies (PBIL, UMDA, cGA). Each circle stands for a variable.

2.2 Bivariate dependencies

The non-dependencies assumption is too far from real-world problems. The naturally next approximation is bivariate dependencies assumption. The algorithms of this class assume that dependencies are only between two variables. They usually use greedy methods to find the local optimal probabilistic model based on this assumption.

Mutual Information Maximization for Input Clustering (MIMIC) was proposed by De Bonet, Isbell, and Viola (1997). The probabilistic model they used is

$$p^p(\bar{x}) = p(x_{i_1} \mid x_{i_2}) \cdot p(x_{i_2} \mid x_{i_3}) \cdots p(x_{i_{n-1}} \mid x_n) \cdot p(x_{i_n})$$

where $p = (i_1, i_2, \cdots, i_n)$, a permutation of 1, 2, …, *n*. This forms a chain structure between variables (Figure 2). To find the (local) optimal p, they used a greedy search based on Shannon entropy.

Baluja and Davies (1997) introduced an algorithm based on a tree structure probabilistic (a Bayesian tree). Later they proposed the Combining Optimizers with Mutual Information Trees (COMIT) which combines a local searcher and the EDAs approach based on the tree representation. The Bayesian tree in COMIT is constructed by the Maximum Weight Spanning Tree (MWST) algorithm (Chow & Liu, 1968). The probabilistic model in COMIT is

$$p(\bar{x}) = \prod_{i=1}^{n} p(x_i \mid x_{parent(i)}).$$

where *parent* is a function which returns the parent node (variable). Here the conditional probability is generalized. That is, $p(x_i \mid x_{parent(i)})$ reduces to $p(x_i)$ if the parent of the *i*-th variable does not exist (node *i* is the root). Once this probability model is constructed from selected promising solutions, some solutions are sampled from that model. The best of they is treated as a initial point fed into a fast local searcher. Then both those solutions generated by the searcher and those sampled from the probabilistic model forms the new population. A graphical representation of the probabilistic model is shown in Figure 2.

Like COMIT, the probabilistic incremental program evolution (PIPE) (Salustowicz & Schmidhuber, 1997) also uses tree structure as its probabilistic model. In PIPE, computer programs are represented in tree structures as like in genetic programming. Instead of using crossover and mutation operators, PIPE tries to find the maximal possible program tree. Please note that PIPE acts on programs, not strings like other algorithms do.

Pelikan and Mühlenbein (1999) proposed the Bivariate Marginal Distribution Algorithm (BMDA). As its name suggests, the algorithm also assumes bivariate dependencies. In fact, this algorithm is the most generalized in bivariate class. It allows the no dependencies or bivariate dependencies between variables. In other words, the probabilistic model is of forest structure (Figure 2). The mathematical representation of BMDA is the same with that of COMIT, if the generalized conditional probability is adopted. The difference is that in BMDA, many roots are allowed. To construct the forest structure, first randomly choose a variable (node) and add the most greatly dependent node to it. The dependency is measured by Pearson's $?^2$ statistics. Next, add the most greatly dependent node to the newly added node. This procedure is repeated until the dependencies of the rest are below some pre-chosen threshold. Then randomly choose a node from those rest nodes and repeat until all nodes are processed.

The probabilistic models of these three algorithms have some relationship. MIMIC uses a chain model which is a special case of tree structure used by COMIT. Then again, a tree structure is a special case of forest structures (used by BMDA) with only one root. Basically, bivariate model estimates distribution very well of those problems with nonlinearity of order two. The tree structure and BMDA have also been shown to efficiently solve the two-dimensional spin-glass problem (Pelikan and Mühlenbein, 1999). The model construction time of these three algorithms is quadratic to the number of variables. Among them, BMDA is the most generalized since a forest can a tree or a chain. Moreover, research in neural networks has already shown that elimination of unnecessary dependencies offers a better generalization. However, choosing a good threshold becomes an important issue.

2.3 Multivariate dependencies

Any approaches in EDAs that use probabilistic models using statistics of order greater than two belong to this category. The probabilistic models in this class are the most generalized and can accurately estimate many different distributions. There are two important issues here. One is to efficiently construct the probabilistic model. Since the model is complicated, to find the optimal model is time consuming. Instead of find the optimum, algorithms of this class usually adopt greedy methods to construct the model. The other issue is to determine a good threshold to get rid of unnecessary dependencies to achieve better generalization.

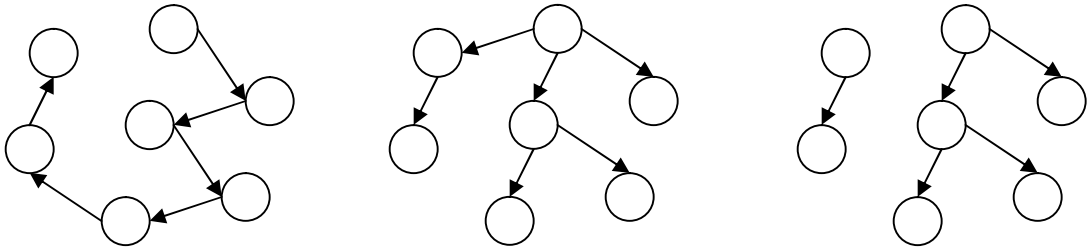

Figure 2 Graphical representations of probabilistic models of bivariate dependencies. The left is the structure in MIMIC. The middle is a tree structure used in COMIT and PIPE. The right is the structure in BMDA.

Harik (1999) introduced the Extended Compact Genetic Algorithm (ECGA). The basic idea is to group nodes into several independent clusters, and then with the marginal probability distribution, execute cGA on those clusters. The grouping is achieved by a greedy search. ECGA uses a minimum description length (MDL) metric (Mitchell, 1997) to measure the tightness of a cluster, then it partition the clusters until all clusters are tight enough. The probabilistic model of ECGA is

$$p(\bar{x}) = \sum_{c \in C} p(\bar{x}_c), \text{ where } C \text{ is a set of clusters.}$$

The Factorized Distribution Algorithm (FDA) (Mühlenbein & Mahnig, 1999) utilizes the additively decomposition property to factorize the joint probability distribution. It has been theoretically proven (Mühlenbein, Mahnig, & Rodriguez, 1999) that FDA is very effective when the problem is additively decomposable. However, FDA requires prior knowledge about the problem and impairs the blindness of GAs.

Soto et al. (1999) introduced the Polytree Approximation of Distribution Algorithm (PADA). A polytree is a tree with oriented edges. The structure is slightly more restricted the directed acyclic graph (DAG) which is to be discussed later. The main difference is that a polytree does not contain any cycle even after directions are removed. The structure is shown in Figure 3.

The Bayesian Optimization Algorithm (BOA) (Pelikan, Goldberg, & Cantú-Paz, 1999) uses the Bayesian networks (Figure 3). It uses the Bayesian Dirichlet equivalence (other measurements can also be used, e.g. MDL) to measure the goodness of network structures, and use greedy method to search a good structure. The probabilistic model is exactly an acyclic (more precisely, DAG) Bayesian network. Later, Pelikan and Goldberg

(2000a) introduced hierarchical BOA (hBOA) which efficiently solves hierarchical problems. A good survey concerning research on BOA can be found in Pelikan and Goldberg (2000b).

Similarly, Estimation of Bayesian Networks Algorithm (EBNA) (Etxeberria & Larrañaga, 1999) also uses Bayesian networks. EBNA uses the Algorithm B (Buntine, 1991) to construct the network. The basic idea is starting with an arc-less graph, and then greedily adding one arc at one time until no improvement is possible. According to different scoring criteria, EBNA has many variations: $EBNA_{PC}$, $EBNA_{K2+pen}$, $EBNA_{BIC}$ (Larrañaga, Etxeberria, Lozano, & Peña, 2000a).

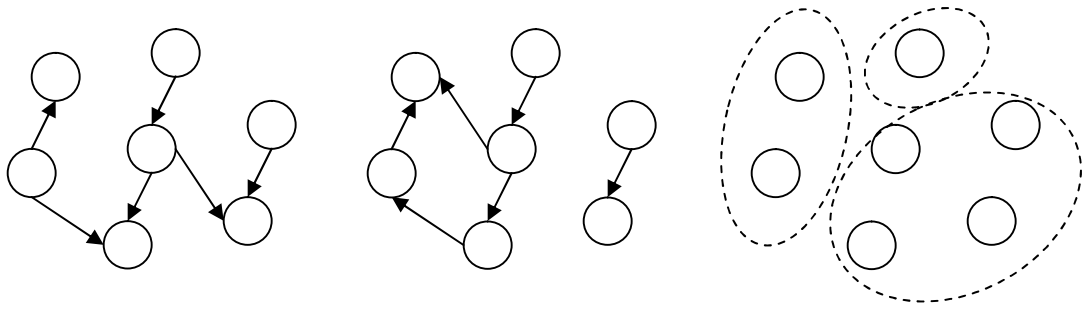

Figure 3 Graphical representations of probabilistic models of multivariate dependencies. The left is a polytree which is used in PADA. The middle is the structure in BOA and EBNA. The right is the structure in ECGA.

3 EDAs on Continuous Domain

In this section, approaches in EDAs that work on continuous domain are introduced. Many of them are modifications of those algorithms in the previous section. Usually continuous domain is hard to model. Two techniques are often used to cope with the continuity. One is to assume Gaussian distribution, and hence a real value can be modeled as a mean and a variance. The other technique is discretization. This technique desterilizes the continuous domain into finite number of possible values. Then those algorithms on discrete domain can be used as well.

3.1 No dependencies

The Stochastic Hill-Climbing with Learning by Vectors of Normal Distributions (SHCLVND) (Rudlof & Köppen, 1996) assumes Gaussian distribution and records a mean vector ($\bar{m}$) and a variance vector ($\bar{s}$) during the run. The mean vector is updated by

$$\bar{u}^{(t+1)} = (1-a)\cdot \bar{m}^{(t)} + a \cdot \bar{b}^{(T)}$$

where the $\bar{b}^{(t)}$ is the barycenter (center of gravity) of B (a predefined fixed number) best individual of selected promising solutions. The variance vector is updated by $\bar{s}^{(t+1)} = b\,\bar{s}^{(t)}$, where $0 < b < 1$. According to the equation, the variance vector is geometrically reduced, and tends to be a zero vector.

As extensions of PBIL and UMDA, $PBIL_C$ and $UMDA_C$ was introduced by Sebag and Ducoulombier (1998), and Larrañaga et al. (2000b), respectively. The basic idea in $UMDA_C$ is the same as that in UMDA, separately estimating probability for every variable. The difference is that $UMDA_C$ uses some statistical tests to find the best-fit probability distribution (hypotheses). Like SHCLVND, $PBIL_C$ also assumes Gaussian distribution and estimates means by

$$\bar{m}^{(t+1)} = (1-a)\bar{m}^{(t+1)} + a\,(\bar{x}_{best1} + \bar{x}_{best2} - \bar{x}_{worst})$$

where $\bar{x}_{best1}$, $\bar{x}_{best2}$, and $\bar{x}_{worst}$ is the best, the second best, and the worst individual of the selected promising solutions, respectively. In fact, $PBIL_C$ can be treated as an extension of SHCLVND where B is two (the best1 and best2) and the concept of punishment is added (the worst). There were several ways suggested in Sebag and Ducoulombier (1998) to estimate the variances. Basically, the variances reduce during the run so that the convergence of $PBIL_C$ is ensured.

Tsutsui, Pelikan, and Goldberg (2001) developed a histogram discretization technique. They introduced two models: fixed-width histogram and fixed-length histogram. Both showed good performance for problems which have no or low dependencies between variables.

3.2 Bivariate dependencies

$MIMIC_C$, an extension of MIMIC, was introduced by Larrañaga, P., Etxeberria, R., Lozano, J. A., and Peña, J. M. (2000b). $MIMIC_C$ also assumes Gaussian distribution. The method of learning the chain structure is two-fold. First, the variable (say, Y) with the smallest variance is chosen as a root. Next, the variable, X, where the entropy of X given

Y is the smallest, is attached to *Y*. This procedure is repeated until a full chain of all variables is complete.

3.3 Multivariate dependencies

Bosman and Thierens (2000) proposed an algorithm based on the Iterative Density Evolution Algorithm (IDEA) framework which is introduced by themselves in 1999. At each generation, IDEA truncates the population by the worst sample in the previous generation. Then after a search procedure (can be any that has been shown) finding the probabilistic density function, IDEA reproduces some new solution and replaces part of the population with them.

Larrañaga, Lozano, and Bengoetxea (2001) introduced the Estimation of Multivariate Normal Algorithm (EMNA). As its name implies, EMNA assumes Gaussian distribution. At every generation, a mean vector and a variance-covariance matrix is estimated. The estimation follows the maximum likelihood manner. Therefore, the mean vector is just estimated by sampling means, and so as the variances and covariance. Then the new solutions are generated based on these $O(n^2)$ parameters. According to different ways to compute those parameters, EMNA has many variations. For details, please refer to Larrañaga, Lozano, and Bengoetxea (2001) or Larrañaga (2002).

Pelikan, Goldberg, and Tsutsui (2001) extended BOA to continuous domain by using adaptive mutation technique in evolutionary strategy field. First, they discretize selected promising solutions. Then they use BOA to recombine those discretized solutions and map the results back to continuous domain. Finally, adaptive mutation is applied to generate the new generation.

4 Summary and Conclusions

In this paper, many different approaches in EDAs are reviewed. They are categorized into three groups according to their probabilistic modes: no dependencies, bivariate dependencies, and multivariate dependencies. Complicated models offer better accuracy, while the cost of their construction is higher. Greedy methods are often used for efficiently build these complicate models. To extend algorithms from discrete domain to continuous domain, Gaussian distribution approximation and discretization techniques are used.

References

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Pittsburgh, PA: Carnegie Mellon University (Technical Report No. CMU-CS-94-163).

Bosman, P. A. N., & Thierens, D. (2000). Continuous iterated density estimation evolutionary algorithms within the IDEA framework. In Wu, A. S., editor, Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program (pp. 197-200).

Buntine, W. (1991). Theory refinement in Bayesian networks. In Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence (pp. 52-60). Morgan Kaufmann.

Chow, C., & Liu, C. (1968). Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory, Volume 14 (pp. 462-467).

De Bonet, J. S., Isbell, C. L., & Viola, P. (1997). MIMC: Finding optima by estimating probability densities. In Mozer, M. C., Jordan, M. I., & Petsche, T. (editors), Advances in Neural Information Processing Systems, Volume 9 (pp. 424). The MIT Press, Cambridge.

Etxeberria, R. and Larrañaga, P. (1999). Global optimization with Bayesian networks. In II Symposium on Artificial Intelligence. CIMAF99. Special Session on Distributions and Evolutionary Optimization (pp. 332-339).

Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1998). The compact genetic algorithm. In Proceedings of the IEEE Conference on Evolutionary Computation 1998 (ICEG '98) (pp. 523-528). Piscataway, NJ: IEEE Service Center.

Harik, G. (1999). Linkage learning via probabilistic modeling in the ECGA. Urbana, IL: University of Illinois Genetic Algorithms Laboratory (IlliGAL Report No. 99010).

Hertz, J. A., Krogh, A. S., & Palmer, R. G. (1991). Introduction to the theory of neural computation. Reading, MA: Addison-Wesley.

Larrañaga, P., Etxeberria, R., Lozano, J. A., & Peña, J. M. (2000a). Combinatorial optimization by learning and simulation of Bayesian networks. In Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence (pp. 343-352). Stanford.

Larrañaga, P., Etxeberria, R., Lozano, J. A., & Peña, J. M. (2000b). Optimization in continuous domains by learning and simulation of Gaussian networks. In Wu, A. S. (editor), Proceedings of the 2000 Genetic and Evolutionary Computation Conference Workshop Program (pp. 201-204).

Larrañaga, P., Lozano, J. A., & Bengoetxea, E. (2001). Estimation of distribution algorithms based on multivariate normal and Gaussian networks. Department of Computer Science and Artificial Intelligence, University id the Basque Country (Technical Report KZZA-IK-1-01).

Larrañaga, P. (2002). A review on estimation of distribution algorithms. In Larrañaga, P., & Lozano, J. A. (editors), Estimation of Distribution Algorithms, Chapter 3 (pp. 57-100), Kluwer Academic Publishers.

Mitchell, T. M. (1997). Machine Learning, McGraw-Hill (section 6.6, pp. 171-173).

Mühlenbein, H., & Paaß, G. (1996). From recombination of genes to the estimation of distributions I. Binary parameters (pp. 178-187).

Mühlenbein, H., Mahnig, T., & Rodriguez, A. O., (1999). Schemata, distributions, and graphical models in evolutionary optimization. Journal of Heuristics, Volume 5 (pp. 215-247).

Mühlenbein, H., & Mahnig, T. (1999). Convergence theory and applications of the factorized distribution algorithm. Journal of Computing and Information Technology, Volume 7 (pp. 19-32).

Pelikan, M., & Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. I Roy, R., Furuhashi, T., & Chawdhry, P. K. (editors), Advances in Soft Computing – Engineering Design and Manufacturing (pp. 521-535). London: Springer-Verlag.

Pelikan, M., Goldberg, D. E., & Lobo, F. G. (1999). A survey of optimization by building and using probabilistic models. Urbana, IL: University of Illinois Genetic Algorithms Laboratory (IlliGAL Report No. 99018).

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: The Bayesian optimization algorithm. In Banzhaf, W., Daida,, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (editors). In Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, Volume I (pp. 525-532). Orlando, FL: Morgan Kaufmann Publishers, San Francisco, CA.

Pelikan, M., & Goldberg, D. E. (2000a). Hierarchical Problem Solving by the Bayesian Optimization Algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2000 (pp. 267-274), Las Vegas, Nevada. (also IlliGAL Report No. 2000002, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL.)

Pelikan, M., & Goldberg D. E. (2000b). Research on the Bayesian Optimization Algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2000 Workshop (pp. 216-219). (also IlliGAL Report No. 2000010, Illinois

Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL.)

Pelikan, M., Goldberg, D. E., & Tsutsui, S. (2001). Combining the Strengths of the Bayesian Optimization Algorithm and Adaptive Evolution Strategies, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL (Illi-GAL Report No. 2001023).

Rudlof, S., & Köppen, M. (1996). Stochastic hill-climbing with learning by vectors of normal distributions. Nagoya, Japan.

Salustowicz, R. P., & Schmidhuber, J. (1997). Probabilistic incremental program evolution: Stochastic search through program space. In van Someren, M., & Widmer, G. (editors), Machines Learning: ECML-97, Volume 1224 of Lecture Note in Artificial Intelligence (pp, 213-220). Spring-Verlag.

Sastry, K. (2002). Slides of presentation. Illinois Genetic Algorithms Laboratory (Illi-GAL), University of Illinois at Urbana-Champaign, Urbana, IL. File available at http://www-advancedgec.ge.uiuc.edu/presentations/GE493PMBGAsKumara.ppt.

Sebag, M., & Ducoulombier, A. (1998). Extending population-based incremental learning to continuous search spaces. In Parallel Problem solving from Nature – PPSN V (pp. 418-427). Springer-Verlag. Berlin.

Soto, M. R., Ochoa A., Acid S., & de Campos, L. M. (1999). Introducing the Polytree Approximation of Distribution Algorithm. In Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAF 99, (pp. 360-367). La Habana.

Thierens, D. (1995). Analysis and design of genetic algorithms. Doctoral dissertation, Leuven, Belgium.

Tsutsui, S., Pelikan, M., & Goldberg, D. E. (2001). Evolutionary Algorithm Using Marginal Histogram Models in Continuous Domain, Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop (pp. 230-233), San Francisco, CA.