A Survey of Optimization by Building and Using Probabilistic Models

Martin Pelikan, David E. Goldberg, and Fernando Lobo

IlliGAL Report No. 99018
September 1999

Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign
117 Transportation Building
104 S. Mathews Avenue Urbana, IL 61801
Office: (217) 333-0897
Fax: (217) 244-5705

A Survey of Optimization by Building and Using Probabilistic Models

Martin Pelikan, David E. Goldberg, and Fernando Lobo
Illinois Genetic Algorithms Laboratory
Department of General Engineering
University of Illinois at Urbana-Champaign
{pelikan,deg,lobo}@illigal.ge.uiuc.edu

Abstract

This paper summarizes the research on population-based probabilistic search algorithms based on modeling promising solutions by estimating their probability distribution and using the constructed model to guide the further exploration of the search space. It settles the algorithms in the field of genetic and evolutionary computation where they have been originated. All methods are classified into a few classes according to the complexity of the class of models they use. Algorithms from each of these classes are briefly described and their strengths and weaknesses are discussed.

1 Introduction

Recently, a number of evolutionary algorithms that guide the exploration of the search space by building probabilistic models of promising solutions found so far have been proposed. These algorithms have shown to perform very well on a wide variety of problems. However, in spite of a few attempts to do so, the field lacks a global overview of what has been done and where the research in this area is heading to.

The purpose of this paper is to review and describe basic principles of the recently proposed population-based search algorithms that use probabilistic modeling of promising solutions to guide their search. It settles the algorithms in the context of genetic and evolutionary computation, classifies the algorithms according to the complexity of the class of models they use, and discusses the advantages and disadvantages of each of these classes.

The next section briefly introduces basic principles of genetic algorithms as our starting point. The paper continues by sequentially describing the classes of approaches classified according to complexity of a used class of models from the least to the most general one. In Section 4 a few approaches that work with other than string representation of solutions are described. The paper is summarized and concluded in section 5.

2 Genetic Algorithms, Problem Decomposition, and Building Blocks

Simple genetic algorithms (GAs) (Holland, 1975; Goldberg, 1989) are population-based search algorithms that guide the exploration of the search space by application of selection and genetic operators of recombination/crossover and mutation. They are usually applied to problems where the solutions are represented or can be mapped onto fixed-length strings over a finite alphabet.

The user defines the problem that the GA will attempt to solve by choosing the length and base alphabet of strings representing the solutions and defining a function that discriminates the string solutions according to their quality. This function is usually called *fitness*. For each string, the fitness function returns a real number quantifying its quality with respect to the solved problem. The higher the fitness, the better the solution.

GAs start with a randomly generated population of solutions. From the current population of solutions the better solutions are selected by the *selection* operator. The selected solutions are processed by applying *recombination* and *mutation* operators. Recombination combines multiple (usually two) solutions that have been selected together by exchanging some of their parts. There are various strategies to do this, e.g. one-point and uniform crossover. Mutation performs a slight perturbation to the resulting solutions. Created solutions replace some of the old ones and the process is repeated until the termination criteria given by the user are met.

By selection, the search is biased to the high-quality solutions. New regions of the search space are explored by combining and mutating repeatedly selected promising solutions. By mutation, close neighborhood of the original solutions is explored like in a local hill-climbing. Recombination brings up innovation by combining pieces of multiple promising solutions together. GAs should therefore work very well for problems that can be somehow decomposed into subproblems of bounded difficulty by solving and combining the solutions of which a global solution can be constructed. Over-average solutions of these sub-problems are often called *building blocks* in GA literature. Reproducing the building blocks by applying selection and preserving them from disruption in combination with mixing them together is a very powerful principle to solve the decomposable problems (Harik, Cantú-Paz, Goldberg, & Miller, 1997; Mühlenbein, Mahnig, & Rodriguez, 1998).

However, fixed, problem-independent recombination operators often either break the building blocks frequently or do not mix them effectively. GAs work very well only for problems where the building blocks are located tightly in strings representing the solutions (Thierens, 1995). On problems with the building blocks spread all over the solutions, the simple GAs experience very poor performance (Thierens, 1995). That is why there has been a growing interest in methods that learn the structure of a problem on the fly and use this information to ensure a proper mixing and growth of building blocks. One of the approaches is based on probabilistic modeling of promising solutions to guide the further exploration of the search space instead of using crossover and mutation like in the simple GAs.

3 Evolutionary Algorithms Based on Probabilistic Modeling

The algorithms that use a probabilistic model of promising solutions to guide further exploration of the search space are called the estimation of distribution algorithms (EDAs) (Mühlenbein & Paaß, 1996). In EDAs better solutions are selected from an initially randomly generated population of solutions like in the simple GA. The true probability distribution of the selected set of solutions is estimated. New solutions are generated according to this estimate. The new solutions are then added into the original population, replacing some of the old ones. The process is repeated until the termination criteria are met.

The EDAs therefore do the same as the simple GAs except for that they replace genetic recombination and mutation operators by the following two steps:

(1) A model (an estimate of the true distribution) of selected promising solutions is constructed.

(2) New solutions are generated according to the constructed model.

Although EDAs process solutions in a different way than the simple GAs, it has been theoretically and empirically proven that the results of both can be very similar. For instance, the simple GA with uniform crossover which randomly picks a value on each position from either of the two parents works asymptotically the same as the so-called univariate marginal distribution algorithm (Mühlenbein & Paaß, 1996) that assumes that the variables are independent (Mühlenbein, 1997; Harik et al., 1998; Pelikan & Mühlenbein, 1999).

A distribution estimate can capture a building-block structure of a problem very accurately and ensure a very effective mixing and reproduction of building blocks. This results in a linear or subquadratic performance of EDAs on these problems (Mühlenbein & Mahnig, 1998; Pelikan et al., 1998). In fact, with an accurate distribution estimate that captures a structure of the solved problem the EDAs unlike the simple GAs perform the same as GA theory with mostly used assumptions claims. However, estimation of the true distribution is far from a trivial task. There is a trade-off between the accuracy and efficiency of the estimate.

The following sections describe three classes of EDAs that can be applied to problems with solutions represented by fixed-length strings over a finite alphabet. The algorithms are classified according to the complexity of the class of models they use. Starting with methods that assume that the variables in a problem (string positions) are independent, through the ones that take into account some pairwise interactions, to the methods that can accurately model even a very complex problem structure with highly overlapping multivariate building blocks.

An example model from each presented class of models will be shown. Models will be displayed as Bayesian networks, i.e. directed acyclic graphs with nodes corresponding to the variables in a problem (string positions) and edges corresponding to probabilistic relationships covered by the model. An edge between two nodes in a Bayesian network relates the two nodes so that the value of the variable corresponding to the ending node of this edge depends on the value of the variable corresponding to the starting node.

3.1 No Interactions

The simplest way to estimate the distribution of promising solutions is to assume that the variables in a problem are independent and to look at the values of each variable regardless of the remaining solutions (see figure 1). The model of the selected promising solutions used to generate the new ones contains a set of frequencies of all values on all string positions in the selected set. These frequencies are used to guide further search by generating new string solutions position by position according to the frequency values. In this fashion, building blocks of order one are reproduced and mixed very efficiently. Algorithms based on this principle work very well on linear problems where the variables are not mutually interacting (Mühlenbein, 1997; Harik et al., 1997).

In the population-based incremental learning (PBIL) algorithm (Baluja, 1994) the solutions are represented by binary strings of fixed length. The population of solutions is replaced with the so-called probability vector which is initially set to assign each value on each position with the same probability 0.5. After generating a number of solutions the very best solutions are selected and the probability vector is shifted towards the selected solutions by using Hebbian learning rule (Hertz, Krogh, & Palmer, 1991). The PBIL has been also referred to as the hill-climbing with learning (HCwL) (Kvasnicka, Pelikan, & Pospichal, 1996) and the incremental univariate marginal distribution algorithm (IUMDA) (Mühlenbein, 1997) recently. Some analysis of the PBIL algorithm can be found in Kvasnicka et al. (1996).

In the univariate marginal distribution algorithm (UMDA) (Mühlenbein & Paaß, 1996) the population of solutions is processed. In each iteration the frequencies of values on each position in the selected set of promising solutions are computed and these are then used to generate new solutions

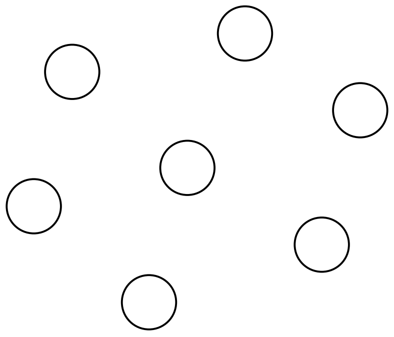


Figure 1: Graphical model with no interactions covered.

which replace the old ones. The new solutions replace the old ones and the process is repeated until the termination criteria are met. Some theory of the UMDA can be found in Mühlenbein (1997).

The compact genetic algorithm (cGA) (Harik, Lobo, & Goldberg, 1998) replaces the population with a single probability vector like the PBIL. However, unlike the PBIL, it modifies the probability vector so that there is direct correspondence between the population that is represented by the probability vector and the probability vector itself. Instead of shifting the vector components proportionally to the distance from either 0 or 1, each component of the vector is updated by shifting its value by the contribution of a single individual to the total frequency assuming a particular population size. By using this update rule, theory of simple genetic algorithms can be directly used in order to estimate the parameters and behavior of the cGA.

All algorithms described in this section perform similarly. They work very well for linear problems where they achieve linear or sub-quadratic performance, depending on the type of a problem, and they fail on problems with strong interactions among variables. For more information on the described algorithm as well as theoretical and empirical results of these see the cited papers.

Algorithms that do not take into account any interdependencies of various bits (variables) fail on problems where there are strong interactions among variables and where without taking into account these the algorithms are mislead. That is why a lot of effort has been put in extending methods that use a simple model that does not cover any interactions to methods that could solve a more general class of problems as efficiently as the simple PBIL, UMDA, or cGA can solve linear problems.

3.2 Pairwise Interactions

First algorithms that did not assume that the variables in a problem were independent could cover some pairwise interactions. The mutual-information-maximizing input clustering (MIMIC) algorithm (De Bonet, Isbell, & Viola, 1997) uses a simple chain distribution (see figure 2a) that maximizes the so-called mutual information of neighboring variables (string positions). In this fashion the Kullback-Liebler divergence (Kullback & Leibler, 1951) between the chain and the complete joint distribution is minimized. However, to construct a chain (which is equivalent to ordering the variables), MIMIC uses only a greedy search algorithm due to its efficiency, and therefore global optimality of the distribution is not guaranteed.

Baluja and Davies (1997) use dependency trees (see figure 2b) to model promising solutions. There are two major advantages of using trees instead of chains. Trees are more general than chains because each chain is a tree. Moreover, by relaxing constraints of the model, in order to find the best model (according to a measure decomposable into terms of order two), a polynomial maximal branching algorithm (Edmonds, 1967) that guarantees global optimality of the solution can be used. On the other hand, MIMIC uses only a greedy search because in order to learn chain distributions, an NP-complete algorithm is needed. Similarly as in the PBIL, the population is replaced by a probability vector which contains all pairwise probabilities.

In the bivariate marginal distribution algorithm (BMDA) (Pelikan & Mühlenbein, 1999) a forest (a set of mutually independent dependency trees, see figure 2c) is used. This class of models is even more general than the class of dependency trees because a single tree is in fact a set of one tree. As a measure used to determine which variables should be connected and which should not, Pearson's chi-square test (Marascuilo & McSweeney, 1977) is used. This measure is also used to discriminate the remaining dependencies in order to construct the final model.

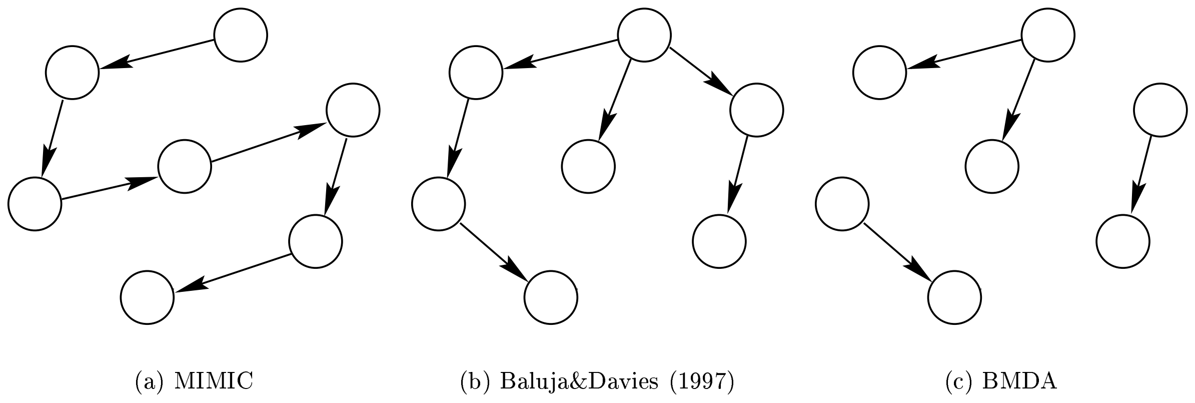


Figure 2: Graphical models with pairwise interactions covered.

Pairwise models allow covering some interactions in a problem and are very easy to learn. The algorithms presented in this section reproduce and mix building blocks of order two very efficiently, and therefore they work very well on linear and quadratic problems (De Bonet et al., 1997; Baluja & Davies, 1997; Mühlenbein, 1997; Pelikan & Mühlenbein, 1999; Bosman & Thierens, 1999). The latter two approaches can also solve 2D spin-glass problems very efficiently (Pelikan & Mühlenbein, 1999).

3.3 Multivariate Interactions

However, covering only some pairwise interactions has still shown to be insufficient to solve problems with multivariate or highly-overlapping building blocks(Pelikan & Mühlenbein, 1999; Bosman & Thierens, 1999). That is why research in this area continued with more complex models. On one hand, using general models has brought powerful algorithms that are capable of solving decomposable problems quickly, accurately, and reliably.

On the other hand, using general models has also resulted in a necessity of using complex learning algorithms that require significant computational time and still do not guarantee global optimality of the resulting models. However, in spite of increased computational time spent by learning the models, the number of evaluations of the optimized function is reduced significantly. In this fashion the overal time complexity is reduced. Moreover, on many problems other algorithms simply do not work. Without learning the structure of a problem, algorithms must be either given this information by an expert or they will simply be incapable of biasing the search in order to solve complex decomposable problems with a reasonable computational cost.

Algorithms presented in this section use models that can cover multivariate interactions. In the extended compact genetic algorithm (ECGA) (Harik, 1999), the variables are divided into a number of intact clusters which are manipulated as independent variables in the UMDA (see figure 3a). Therefore, each cluster (building block) is taken as a whole and different clusters are considered to be mutually independent. To discriminate models, the ECGA uses a minimum description length (MDL) metric (Mitchell, 1997) which prefers models that allow higher compression of data (selected set of promising solutions). The advantage of using the MDL metric is that it penalizes complex models when they are not needed and therefore the resulting models are not overly complex. To find a good model, a simple greedy algorithm is used. Starting with all variables separated, in each iteration current groups of variables are merged so that the metric increases the most. If no more improvement is possible, the current model is used.

Following from theory of the UMDA, for problems that are separable, i.e. decomposable into non-overlapping subproblems of a bounded order, the ECGA with a good model should perform in a sub-quadratic time. A question is whether the ECGA finds a good model and how much effort it takes. Moreover, many problems contain highly overlapping building blocks (e.g., 2D spin-glass systems) which can not be accurately modeled by simply dividing the variables into distinct classes. This results in a poor performance of the ECGA on these problems.

The factorized distribution algorithm (FDA) (Mühlenbein, Mahnig, & Rodriguez, 1998) uses a factorized distribution as a fixed model throughout the whole computation. The FDA is not capable of learning the structure of a problem on the fly. The distribution and its factorization are given by an expert. Distributions are allowed to contain marginal and conditional probabilities which are updated according to the currently selected set of solutions. It has been theoretically proven that when the model is correct, the FDA solves decomposable problems quickly, reliably, and accurately (Mühlenbein, Mahnig, & Rodriguez, 1998). However, the FDA requires prior information about the problem in form of its decomposition and its factorization. Unfortunately, this is usually not available when solving real-world problems, and therefore the use of FDA is limited to problems where we can at least accurately approximate the structure of a problem.

The Bayesian optimization algorithm (BOA) (Pelikan, Goldberg, & Cantú-Paz, 1998) uses a more general class of distributions than the ECGA. It incorporates methods for learning Bayesian networks (see figure 3b) and uses these to model the promising solutions and generate the new ones. In the BOA, after selecting promising solutions, a Bayesian network that models these is constructed. The constructed network is then used to generate new solutions. As a measure of quality of networks, any metric can be used, e.g. Bayesian-Dirichlet (BD) metric (Heckerman, Geiger, & Chickering, 1994), MDL metric, etc. In recently published experiments the BD scoring metric has been used. The BD metric does not prefer simpler models to the more complex ones. It uses accuracy of the encoded distribution as the only criterion. That is why the space of possible models has been reduced by specifying a maximal order of interactions in a problem that are to be taken into account. To construct the network with respect to a given metric, any algorithm that searches over the domain of possible Bayesian networks can be used. In recent experiments, a greedy algorithm has been used due to its efficiency.

The BOA uses an equivalent class of models as the FDA; however, it does not require any information about the problem on input. It is able to discover this information itself. Nevertheless, prior information can be incorporated and the ratio of prior information and information contained in the set of high-quality solutions found so far can be controlled by the user. Not only does the BOA fill the gap between the FDA and uninformed search methods but also offers a method that is efficient even without any prior information (Pelikan et al., 1998; Schwarz & Ocenasek, 1999; Pelikan et al., 1999) and still does not prohibit further improvement by using this. Another algorithm that uses Bayesian networks to model promising solutions, called the estimation of Bayesian network

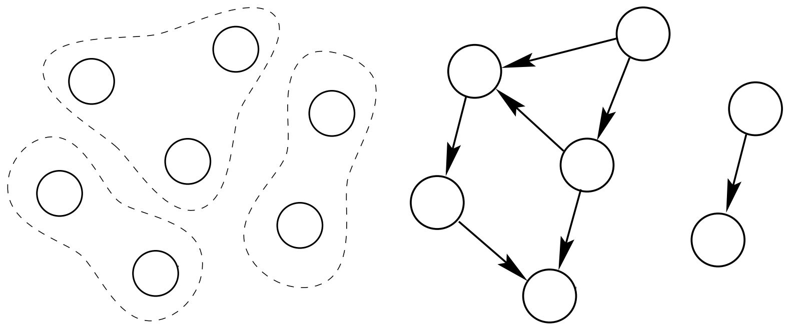


Figure 3: Graphical models with multivariate interactions covered.

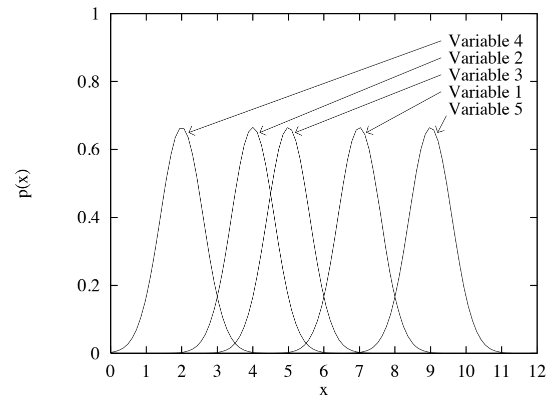
algorithm (EBNA), has been later proposed by Etxeberria and Larrañaga (1999).

The algorithms that use models capable of covering multivariate interactions achieve a very good performance on a wide range of decomposable problems, e.g. 2D spin-glass systems (Pelikan et al., 1998; Mühlenbein & Mahnig, 1998), graph partitioning (Schwarz & Ocenasek, 1999), communication network optimization (Rothlauf, 1999), etc. However, problems which are decomposable into terms of bounded order can still be very difficult to solve. Overlapping the subproblems can mislead the algorithm until the right solution to a particular subproblem is found and sequentially distributed across the solutions (e.g., see F_{0-peak} in Mühlenbein and Mahnig (1998)). Without generating the initial population with the use of problem-specific information, building blocks of size proportional to size of a problem have to be used which results in an exponential performance of the algorithms. This brings up a question on what are the problems we aim to solve by algorithms based on reproduction and mixing of building blocks that we have shortly discussed earlier in section 2. We do not attempt to solve all problems that can be decomposed into terms of a bounded order. The problems we approach to solve are decomposable in a sense that they can be solved by approaching the problem on a level of solutions of lower order by combining the best of which we can construct the optimal or a close-to-optimal solution. This is how we bias the search so that the total space explored by the algorithm substantially reduces by a couple orders of magnitude and computationally hard problems can be solved quickly, accurately, and reliably.

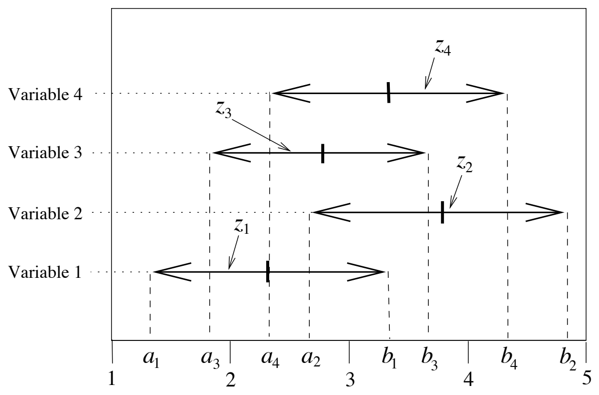
4 Beyond String Representation of Solutions

All algorithms described above work on problems defined on fixed-length strings over a finite alphabet. However, recently there have been a few attempts to go beyond this simple representation and directly tackle problems where the solutions are represented by vectors of real number or computer programs without mapping the solutions on strings. All these approaches use simple models that do not cover any interactions in a problem.

In the stochastic hill-climbing with learning by vectors of normal distributions (SHCLVND) (Rudlof & Köppen, 1996) the solutions are represented by real-valued vectors. The population of solutions is replaced (and modeled) by a vector of mean values of Gaussian normal distribution μ_i for each optimized variable (see figure 4a). The standard deviation σ is stored globally and it is the same for all variables. After generating a number of new solutions, the mean values μ_i are shifted towards the best of the generated solutions and the standard deviation σ is reduced to



(a) SHCLVND



(b) (Servet et al., 1998)

Figure 4: Probabilistic models of real vectors of independent variables.

make future exploration of the search space narrower. Various ways of modifying the σ parameter have been exploited in (Sebag & Ducoulombier, 1998). In another implementation of a real-coded PBIL (Servet, Trave-Massuyes, & Stern, 1997), for each variable an interval (a_i, b_i) and a number z_i are stored (see figure 4b). The z_i stands for a probability of a solution to be in the right half of the interval. It is initialized to 0.5. Each time new solutions are generated using the corresponding intervals, the best solutions are selected and the numbers z_i are shifted towards them. When z_i for a variable gets close to either 0 or 1, the interval is reduced to the corresponding half of it. In figure 4b, each z_i is mapped to the corresponding interval (a_i, b_i).

In the probabilistic incremental program evolution (PIPE) algorithm (Salustowicz & Schmidhuber, 1997) computer programs or mathematical functions are evolved as like in genetic programming (Koza, 1992). However, pair-wise crossover and mutation are replaced by probabilistic modeling of promising programs. Programs are represented by trees where each internal node represents a function/instruction and leaves represent either input variable or a constant. In the PIPE algorithm, probabilistic representation of the program trees is used. Probabilities of each instruction in each node in a maximal possible tree are used to model promising and generate new programs (see figure 5). Unused portions of the tree are simply cut before the evaluation of the program by a fitness function. Initially, the model is set so that the trees are generated at random. From the current population of programs the ones that perform the best are selected. These are then used to update the probabilistic model. The process is repeated until the termination criteria are met.

5 Summary and Conclusions

Recently, the use of probabilistic modeling in genetic and evolutionary computation has become very popular. By combining various achievements of machine learning and genetic and evolutionary computation, efficient algorithms for solving a broad class of problems have been constructed. The most recent algorithms are continuously proving their powerfulness and efficiency, and offer a promising approach to solving the problems that can be resolved by combining high-quality pieces of information of a bounded order together.

In this paper, we have reviewed the algorithms that use probabilistic models of promising

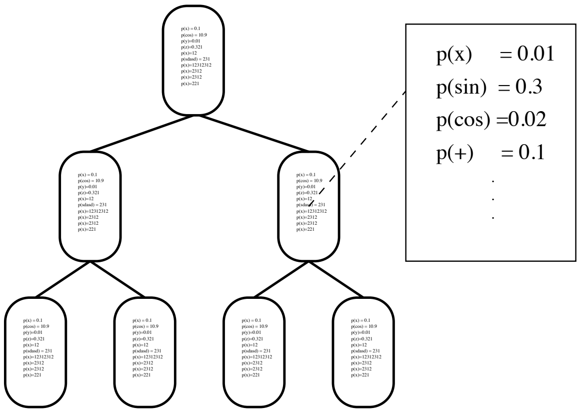


Figure 5: Graphical model of a program with no interactions covered used in PIPE.

solutions found so far to guide further exploration of the search space. The algorithms have been classified in a few classes according to the complexity of the class of models they use. Basic properties of each of these classes of algorithms have been shortly discussed and a thorough list of published papers and other references has been given.

6 Acknowledgments

The authors would like to thank Erick Cantú-Paz, Martin Butz, Dimitri Knjazew, and Jiri Pospichal for valuable discussions and useful comments that helped to shape the paper.

The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant number F49620-97-1-0050. Research funding for this project was also provided by a grant from the U.S. Army Research Laboratory under the Federated Laboratory Program, Cooperative Agreement DAAL01-96-2-0003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies and endorsements, either expressed or implied, of the Air Force of Scientific Research or the U.S. Government.

References

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

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

Bosman, P. A. N., & Thierens, D. (1999). Linkage information processing in distribution estimation algorithms. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., & Smith, R. E. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, Volume I (pp. 60–67). Orlando, FL: Morgan Kaufmann Publishers, San Fransisco, CA.

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

Edmonds, J. (1967). Optimum branching. *J. Res. NBS*, *71B*, 233–240.

Etxeberria, R., & Larrañaga, P. (1999, March). Global optimization using Bayesian networks. In *Second Symposium on Artificial Intelligence (CIMAF-99)* (pp. 332–339). Habana, Cuba.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

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

Harik, G., Cantú-Paz, E., Goldberg, D. E., & Miller, B. L. (1997). The gambler's ruin problem, genetic algorithms, and the sizing of populations. In *Proceedings of the International Conference on Evolutionary Computation 1997 (ICEC '97)* (pp. 7–12). Piscataway, NJ: IEEE Press.

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

Heckerman, D., Geiger, D., & Chickering, M. (1994). *Learning Bayesian networks: The combination of knowledge and statistical data* (Technical Report MSR-TR-94-09). Redmond, WA: Microsoft Research.

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

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Koza, J. R. (1992). *Genetic programming: on the programming of computers by means of natural selection.* Cambridge, MA: The MIT Press.

Kullback, S., & Leibler, R. A. (1951). On information and sufficiency. *Annals of Math. Stats.*, *22*, 79–86.

Kvasnicka, V., Pelikan, M., & Pospichal, J. (1996). Hill climbing with learning (An abstraction of genetic algorithm). *Neural Network World*, *6*, 773–796.

Marascuilo, L. A., & McSweeney, M. (1977). *Nonparametric and distribution-free methods for the social sciences.* CA: Brooks/Cole Publishing Company.

Mitchell, T. M. (1997). *Machine learning.* McGraw-Hill.

Mühlenbein, H. (1997). The equation for response to selection and its use for prediction. *Evolutionary Computation*, *5*(3), 303–346.

Mühlenbein, H., & Mahnig, T. (1998). *Convergence theory and applications of the factorized distribution algorithm.* To be published.

Mühlenbein, H., Mahnig, T., & Rodriguez, A. O. (1998). *Schemata, distributions and graphical models in evolutionary optimization.* Submitted for publication.

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

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1998). *Linkage problem, distribution estimation, and Bayesian networks* (IlliGAL Report No. 98013). Urbana, IL: University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory.

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. (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, Volume I (pp. 525–532). Orlando, FL: Morgan Kaufmann Publishers, San Fransisco, CA.

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

Rothlauf, F. (1999). Communication network optimization. Personal Communication.

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. (Eds.), *Machine Learning: ECML-97*, Volume 1224 of *Lecture Notes in Artificial Intelligence* (pp. 213–220). Springer-Verlag.

Schwarz, J., & Ocenasek, J. (1999, June). Experimental study: Hypergraph partitioning based on the simple and advanced algorithms BMDA and BOA. Brno, Czech Republic. In printing.

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). Berlin: Springer Verlag.

Servet, I., Trave-Massuyes, L., & Stern, D. (1997). Telephone network traffic overloading diagnosis and evolutionary computation techniques. In *Proceedings of the Third European Conference on Artificial Evolution (AE'97)* (pp. 137–144).

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