Reinforcement Learning Estimation of Distribution Algorithm

Topon Kumar Paul and Hitoshi Iba

Graduate School of Frontier Sciences, The University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-8656, Japan
{topon,iba}@miv.t.u-tokyo.ac.jp

Abstract. This paper proposes an algorithm for combinatorial optimizations that uses reinforcement learning and estimation of joint probability distribution of promising solutions to generate a new population of solutions. We call it Reinforcement Learning Estimation of Distribution Algorithm (RELEDA). For the estimation of the joint probability distribution we consider each variable as univariate. Then we update the probability of each variable by applying reinforcement learning method. Though we consider variables independent of one another, the proposed method can solve problems of highly correlated variables. To compare the efficiency of our proposed algorithm with other Estimation of Distribution Algorithms (EDAs) we provide the experimental results of the two problems: four peaks problem and bipolar function.

1 Introduction

After the introduction of Genetic Algorithm (GA) it has been widely used for the optimization of problems because it is easy to use and can be applied in a problem with little prior knowledge. The heart of GA is its selection and recombination of promising solutions; however, crossover and mutation of GA are problem specific and hence success depends on the choice of these parameters. They do not consider problem specific interactions among the variables called *linkage information*. As a result, the prediction of the movements of the populations in the search space is difficult in GA. Holland[12] introduced the idea that linkage information would be beneficial for Genetic Algorithms. Following his idea, Goldberg et al.[7], Harik [9]and Kargupta[13] extended GA by either changing the representation of the solutions or evolving recombination operators among individual solutions to process the building blocks. Recently, there have been proposed evolutionary algorithms based on the probabilistic model where selection and recombination of building blocks of Genetic Algorithm are replaced by generating new solutions by sampling the probability distribution which is calculated from the selected promising solutions. These algorithms are called Estimation of Distribution Algorithms (EDAs) [17] or Probabilistic Model-Building Genetic Algorithms (PMBGAs) [20]. As the EDAs try to capture the structure of the problem, EDAs are thought to be more efficient than GAs.

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1259–1270, 2003.

The purpose of this paper is to introduce a new Estimation of Distribution Algorithm (EDA) that uses Reinforcement Learning (RL) method and marginal probability distribution of selected individuals in order to generate new solutions. We call this algorithm Reinforcement Learning Estimation of Distribution Algorithm (RELEDA). The proposed method extends existing EDAs to solve difficult classes of problems more efficiently and accurately. We consider variables as univariate and calculate their marginal distributions accordingly. We then update the probability vector of the variables by reinforcement learning method using these marginal distributions and the best fit individual of a generation. New individuals (solutions) are generated by sampling this probability vector. The experiments were done with the four peaks problem and bipolar function which are very difficult for some EDAs to optimize. The experimental results show that our proposed algorithm is able to solve the tested problems more efficiently.

In Sects. 2 and 3, we provide the background needed to understand the motivation of our algorithm. Section 4 describes the proposed algorithm. In Sect- 5, a short review of different existing EDA approaches are presented. Section 6 describes the test functions and presents the experimental results on these problems. Section 7 addresses our future work. The summary and conclusion are provided in Sect. 8.

2 Optimizations by Free Energy Minimization

Before we proceed, we need to give some mathematical notations. The variable X denotes the set of n discrete binary variables $\{X_1, X_2, \ldots, X_n\}$ and x denotes the set of values of these variables. The search space is $S = \{0, 1\}^n$. $p(x)$ is the joint probability of X and $f(x)$ denotes the function to be maximized (minimized).

The Gibbs distribution of a function $f(x)$ at given temperature T is

$$p(x) = \frac{e^{-\frac{f(x)}{T}}}{Z} \tag{1}$$

where $Z = \sum_{y \in S} e^{-\frac{f(y)}{T}}$ is the partition function.

When $T \to 0$ the Boltzmann distribution converges uniformly to one of the global optima. The Metropolis algorithm, whose stationary distribution is precisely Boltmann distribution, samples from $p(x)$ to generate solution for optimization task. When it is not possible to calculate $p(x)$ precisely one approximates it with a target distribution $p_T(x)$ using Kullback Leibler (KL) divergence. The KL divergence is

$$D(p_T(x), p(x)) = \ln Z + \frac{1}{T}(E - TS) \tag{2}$$

where E and S are internal energy and entropy of the system and $F = E - TS$ is the free energy. Here F is a function of $p(x)$. So by minimizing F one can minimize KL divergence. In order to minimize the F we need to know all

probability distributions of the variables in the search space. Instead, the search to probability distributions can be reduced to a small number of parameters denoted by $\theta = \{\theta_1, \theta_2, \ldots, \theta_v\}$ where $\theta_i \in \mathbb{R}$ is a parameter related to the probability of the variable X_i through a function and the joint probability is denoted by $p(x, \theta)$ which is differentiable with respect to its parameters for each $x \in S$. Therefore, F is differentiable with respect to θ. Thus the problem has been transformed from the search in discrete domain to continuous domain. The update rule of the parameter θ can be found by defining a dynamical system by the ordinary differential equation: $\frac{d\theta}{dt} + \frac{\delta F}{\delta \theta} = 0$. After inserting the values of F we obtain by some calculations:

$$\frac{d\theta}{dt} + \sum_{x \in S} (f(x) + T(1 + \ln p(x, \theta))) \frac{\delta p(x, \theta)}{\delta \theta} = 0 \ . \tag{3}$$

Then the basic update rule proposed by Berny[4] for gradient descent is as follows:

$$\Delta\theta = -\alpha (f(x) + T(1 + \ln p(x, \theta))) \frac{\delta p(x, \theta)}{\delta \theta} \tag{4}$$

where $\alpha > 0$ is a small constant called learning rate.

3 Reinforcement Learning

In reinforcement learning an agent is connected to its environment through perception and action. At every step of interactions the agent gets some input signals and indications of current state of the environment, and then the agent chooses some actions to produce output. The action changes the state of the environment, and the value of this transition is sent back to the agent as a scalar reinforcement signal. Here learning is unsupervised, and the agent modified its behavior to maximize or minimize the reinforcement signal. Williams[22] in his REINFORCEMENT Algorithm for learning with associative networks has used the following incremental rule for the updating of the weight of the connection from input i to unit $j(w_{ij})$:

$$\Delta w_{ij} = \alpha_{ij} (r - b_{ij}) \frac{\delta \ln g_i}{\delta w_{ij}} \tag{5}$$

where α_{ij} is the learning rate, r is the reinforcement signal, b_{ij} is a reinforcement baseline and g_i is the probability of the output of unit i given the input and the weights to this unit. For combinatorial optimizations Berny[4] has used cost function (fitness) as reinforcement signal and low cost binary string as agent's output. His reinforcement algorithm is:

$$\Delta\theta_i = \alpha (f(x) - b_i) \frac{\delta \ln p(x, \theta)}{\delta \theta_i} \tag{6}$$

where $f(x)$ is the cost function, $p(x, \theta)$ the joint probability distribution and θ_i is a parameter related to the probability of i^{th} variable $(p_i(x_i))$, b_i is the baseline and α is the learning rate.

The baseline (b) is updated by the recursive formula:

$$b(t+1) = \gamma b(t) + (1-\gamma) f(x(t)) \tag{7}$$

where $0 < \gamma < 1$ is the baseline factor. He (Berny) has called this baseline update rule as expectation strategy where, if a randomly generated solution x has cost $f(x)$ lower than average cost, the agent takes the solution more likely.

4 Reinforcement Learning Estimation of Distribution Algorithm (RELEDA)

In RELEDA we consider no interactions among the variables. Then the joint probability($p(x, \theta)$) becomes:

$$p(x, \theta) = \prod_{i=1}^{n} p_i(x_i) \tag{8}$$

where $p_i(x_i)$ is the probability of $X_i = 1$.

The correlation between $p_i(x_i)$ and θ_i is expressed through the sigmoid function:

$$p_i(x_i) = \frac{1}{2}(1 + \tanh(\beta\theta_i)) \tag{9}$$

where β is the sigmoid gain. Now

$$\begin{aligned} \frac{\delta \ln p(x,\theta)}{\delta \theta_i} &= \frac{1}{p_i(x_i)} \frac{\delta p_i(x_i)}{\delta \theta_i} \\ &= 2\beta(1 - p_i(x_i)) \ . \end{aligned}$$

Inserting this value into (6) we get:

$$\Delta\theta_i = 2\alpha\beta(f(x) - b_i)(1 - p_i(x_i)) \ . \tag{10}$$

To incorporate EDA we change these equations according to our needs. We shall decompose the equations into variable levels, i.e. we replace the cost function with the value of the variable of that position in the best fit individual. We rewrite the two equations (10) and (7) as follows:

$$\Delta\theta_i = \alpha(b_i - p_i(x_i))(1 - d_i) \tag{11}$$

[where we have put $\alpha = 2\alpha\beta$] and

$$b_i(t+1) = \gamma b_i(t) + (1-\gamma) x_i \tag{12}$$

where d_i is the marginal distribution of the variable X_i and x_i is the value of the variable X_i in the best individual in that generation. Marginal distribution d_i is calculated from the selected individuals in a generation by the formula:

$$d_i = \frac{\sum_{j=1}^{N} \delta_j(X_i = 1) + 1}{N + 2} \tag{13}$$

where

$$\delta_j(X_i = 1) = 1 \text{ if in } j^{th} \text{ individual } X_i = 1$$
$$= 0 \text{ otherwise.}$$

During calculation of marginal distributions we have used Laplace correction [8] in equation (13) to make sure that no probability is 0.

We update each θ_i by using equation (11) as follows:

$$\theta_i = \theta_i + \Delta\theta_i \ . \tag{14}$$

According to equation (11) each θ_i is updated by a small amount if either the marginal distribution is closing to 1 or the value of the the variable X_i is 0. Since the baseline is dependent on X_i, the reinforcement signal (baseline) b_i plays a role in controlling the directions of the probability of each variable.

4.1 Algorithm

Our resulting algorithm now is a combination of reinforcement learning and EDA. Here is the algorithm:

1. Initialize θ_i and b_i ($p_i(x_i)$ will be calculated according to equation(9)).
2. Generate initial population.
3. Select N promising individuals.
4. For $i = 1$ to n do
   a) Calculate marginal distribution (d_i) according to equation(13).
   b) Update b_i according to equation(12).
   c) Update θ_i by equation (14) and find $p_i(x_i)$ by equation (9).
5. Generate M offspring by sampling the probabilities calculated in the previous step.
6. Create new population by replacing some old individuals with offspring (M).
7. If termination criteria not met, go to step (3).

The initial parameter $p_i(x_i)$ should chosen such that $p(x, \theta)$ is uniform. The algorithm runs until either an optimal solution is found or the allowed maximum no. of generations have passed.

5 Other EDA Approaches

The two main steps of EDAs are to estimate the probability distribution of selected individuals (promising solutions) and generate new population by sampling this probability distribution. There is no simple and general approach to do these efficiently. Different EDAs use different models for the estimation of probability distribution.

Univariate Marginal Distribution Algorithm (UMDA) [15], Population Based Incremental Learning (PBIL)[1] and Compact Genetic Algorithm (CGA) [11]

treat variables in a problem as independent of one another. As a result, n-dimensional joint probability distribution factorizes as a product of n univariate and independent probability distributions. PBIL and CGA use a probability vector while UMDA uses both a probability vector and a population. The update rules of probability vector of PBIL and CGA are different.

Mutual Information Maximizing Input Clustering Algorithm (MIMIC) [5], Bivariate Marginal Distribution Algorithm (BMDA) [21] and Combining Optimizers with Mutual Information Trees (COMIT) [3] consider pairwise interactions among variables. These algorithms learn parameters as well as structure of the problems. They can mix building blocks of order one and two successfully.

Bayesian Optimization Algorithm (BOA) [19], Estimation of Bayesian Networks Algorithm (EBNA) [14], Factorized Distribution Algorithm (FDA) [16] and Extended Compact Genetic Algorithm (ECGA) [10] use models that can cover multiple interactions among variables. BOA and EBNA both use Bayesian Network as structure learning but different score metrics. BOA uses Bayesian Dirichlet equivalence (BDe) scores while EBNA uses K2+Penalization and Bayesian Information Criterion (BIC) scores to measure the likelihood of a model. FDA can be applied to additively decomposable problems. It uses Boltzmann selection for Boltzmann distribution. In ECGA the variables of a problem are grouped into disjoint sets and marginal distributions of these sets are used to compress the population of selected individuals. This algorithm uses the Minimum Description Length (MDL) approach to measure the goodness of a structure. All these multivariate algorithms can produce a good solution with the sacrifice of computation time.

A detailed overview of different EDA approaches in both discrete and continuous domains can be found in [14]. For applications of UMDA with Laplace corrections in binary and permutation domains, see [18].

6 Experiments

The experiments have been done for functions taken from [19] and [2]. In these functions of unitation variables are highly correlated. The following section describes the function and presents the results of the experiments.

6.1 Test Functions

Before we define our test functions, we need to define elementary function like deceptive function taken from [19].

A deceptive function of order 3 is defined as

$$f^3_{dec}(X) = \begin{cases} 0.9 & \text{if } u = 0 \\ 0.8 & \text{if } u = 1 \\ 0 & \text{if } u = 2 \\ 1 & \text{if } u = 3 \end{cases} \tag{15}$$

where X is a vector of order 3 and u is the sum of input variables.

A bipolar deceptive function of order 6 is defined with the use of deceptive function as follows:

$$f^6_{bipolar} = f^3_{dec}(|3-u|)\ . \tag{16}$$

Bipolar function has global solutions of either $(1,1,1,1,1,1)$ or $(0,0,0,0,0,0)$.

So non overlapping bipolar function for the vector $X = (X_1, X_2, \ldots, X_n)$ is defined as:

$$f_{bipolar}(X) = \sum_{i=1}^{n/6} f^6_{bipolar}(S_i) \tag{17}$$

where $S_i = (X_{6i-5}, X_{6i-4}, \ldots, X_{6i})$. Bipolar function has total $2^{\frac{n}{6}}$ global optima and $\binom{6}{3}^{\frac{n}{6}}$ local optima, making it highly multimodal. According to Deb and Goldberg [6] all the schema of order less than 6 misleads Simple Genetic Algorithm (SGA) away from the global optimum into a local one for this bipolar function.

Another test function is the four peaks problem taken from [2]. The function is defined as

$$f_{4\_peak}(X) = max(z(X), o(X)) + R(X) \tag{18}$$

where $z(X)$ is number of trailing 0s in the vector $X = (X_1, X_2, \ldots, X_n)$ and $o(X)$ is the number of leading 1s in X. $R(X)$ is a conditional reward defined with the threshold $s = \frac{n}{10}$ as

$$R(X) = \begin{cases} n & \text{if } z(X) > s \text{ and } o(X) > s \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

The function has global optimum of the form: first and last $(s+1)$ bits are all 1 and 0 respectively and the remaining $(n-2s-2)$ bits in the middle are either all 0s or all 1s. As n increases, local optima increase exponentially while global optima decrease in the same rate, making local search methods trapped into local optima [2].

6.2 Experimental Results

Here we present the experimental results running the algorithm on a computer with 1133 MHZ Intel Pentimum (III)$^{\text{TM}}$ Processor and 256 MB of RAM and in Borland C++ builder 6.0 environment. For all the problems, the experimental results of our algorithm of 50 independent runs are provided.

For all the problems, the algorithm runs until either an optimal solution is found or the maximum no. of generations has passed. This is somehow different from [19] where a population is said to be converged when the portion of some values in each position reaches 95%. In each run of the algorithm we apply truncation selection: the best half of the population is selected for the estimation of marginal probability distribution. Then we update the probability vector of the variables by reinforcement learning method using these marginal distributions and the best fit individual of a generation. For the bipolar function and

Table 1. No. of fitness evaluations required until an optimal solution of the bipolar function is found

Problem	Average			Standard Deviation		
Size	RELEDA	PBIL	UMDA	RELEDA	PBIL	UMDA
12	750.00	897.60	784.80	532.98	645.31	569.83
24	13608.00	26940.00	31920.00	6502.86	14959.58	15607.80
36	77918.40	123577.20	121096.80	53929.71	36238.56	38886.05
48	256089.60	488745.60	512227.20	109425.70	132108.30	136246.98
60	529278.00	1521948.00	1418778.00	250798.44	548318.17	508102.96
72	1014084.00	3718238.40	3802471.20	621080.42	1310214.60	1268969.22
84	1855072.80	9372510.00	9164064.00	774195.88	869822.56	1432628.32
96	2366457.60	NA	NA	765031.96	NA	NA

four peaks problem, we set the baseline factor $\gamma = 0.1$, learning rate $\alpha = 0.9$, elite=50%, maximum no. of generations=30000 and population size=$10*n$ where n is the size of the problem. We set initial probability of each variable $p_i(x_i) = \frac{1}{2}$; hence $\theta_i = 0$. The baseline is initialized by setting each $b_i = 1$. Our replacement strategy is elitism so that the best individual of a generation is not lost, and the algorithm performs gradient ascent in search space.

To compare the experimental results of our proposed algorithm with those of PBIL and UMDA, we apply these algorithms to the bipolar function and only PBIL to the four peaks problem. We set the population size, maximum no. of generations and the rate of elitism for PBIL and UMDA with the same values as those for RELEDA. The best half of the population is selected for the calculation of marginal probabilities of the variables. We apply Laplace corrections during calculation of marginal probabilities. We also use the same selection and replacement strategy as those of RELEDA for PBIL and UMDA .

For the bipolar function, we set sigmoid gain $\beta = 0.1$ and learning rate for PBIL=0.9. In table 1 we show no. of fitness evaluations required for bipolar function by RELEDA, PBIL and UMDA until an optimal solution is found. In addition to convergence to a solution, RELEDA can discover a number of different solutions out of total $2^{\frac{n}{6}}$ global optima. For problem sizes of 12 and 24, it finds all the global optima only in 50 runs. For higher size problems, it discovers different global optima. So our algorithm can be used to find multiple solutions of a problem. For the problem size of 84, we provide results of 8 independent runs of PBIL and 15 of UMDA. For the problem size of 96, neither PBIL nor UMDA converged to any solution at all within the maximum no. of generations. We have indicated it in the table by inserting 'Not Available (NA)'.

Figures 1 and 2 show graphical views of average no. of fitness evaluations required by RELEDA vs PBIL and RELEDA vs UMDA respectively for the bipolar function. We show the experimental results in two graphs because in one graph it is very difficult to distinguish between the graph of PBIL and that of UMDA. Both PBIL and UMDA require almost the same no. of fitness evaluations to find an optimal solution of the bipolar function.

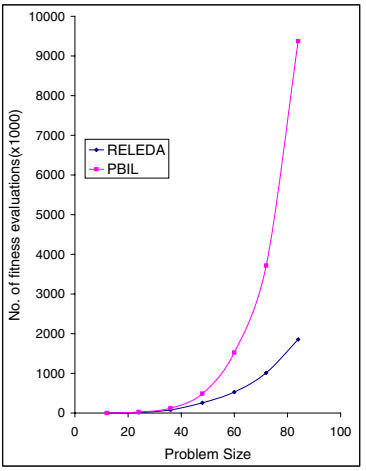

Fig. 1. Average no. of fitness evaluations required until an optimal solution of the bipolar function is found(RELEDA vs PBIL)

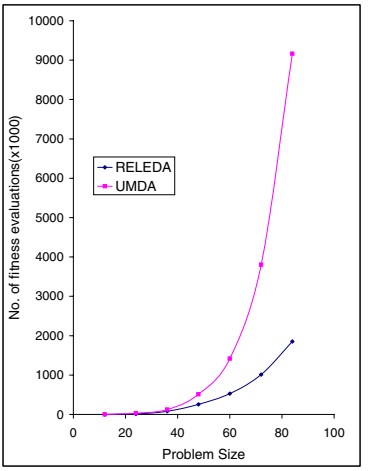

Fig. 2. Average no. of fitness evaluations required until an optimal solution of the bipolar function is found(RELEDA vs UMDA)

Table 2. No. of fitness evaluations required until an optimal solution of the four peak problem is found

Problem	Average		Standard Deviation	
Size	RELEDA	PBIL	RELEDA	PBIL
10	286	395	138.15	336.20
20	2366	38350	1027.30	8889.35
30	9219	189390	4765.65	63131.02
40	24904	462660	14858.31	75257.90
50	84085	643950	59619.60	52987.26
60	181224	1204290	159526.53	154901.78
70	316764	1337735	264597.58	92907.18
80	512280	1832760	425668.28	217245.50
90	715248	2283975	750726.15	262165.18

For the four peaks problem, we set sigmoid gain $\beta = 0.5$ and learning rate for PBIL=0.005 (as used in [2]). The results are shown in table 2. Graphical representation of average no. of fitness evaluations required by RELEDA and PBIL for the four peaks problem is shown in Fig. 3.

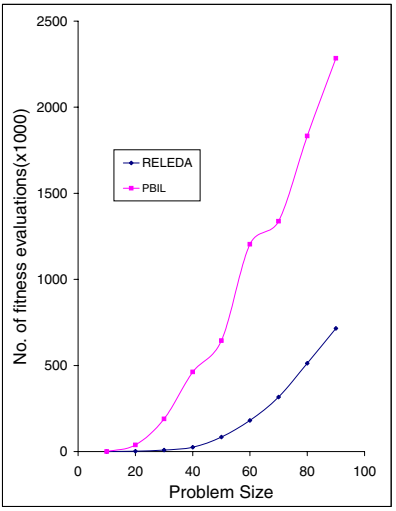

Fig. 3. Average no. fitness evaluations required until an optimal solution of the four peaks problem is found

6.3 Discussion on Experimental Results

The proposed algorithm finds optimal solutions of those problems which are very hard for simple GA (SGA) to solve. Deb and Goldberg [6] have said that in fully deceptive function with subfunctions of order k all the schema of order less than k mislead SGA away from the global optimum into a local one. In similar fashion, these deceptive functions are very hard for some EDAs. RELEDA finds optimal solutions of those problems in fewer no. of fitness evaluations because it involves a nonlinear function of hyperbolic tangent. Though the marginal distribution of a variable calculated from the selected individuals may be zero, the probability of that variable in RELEDA will not be zero. But in UMDA it will be zero. RELEDA actually shifts the probability of a variable in either directions depending on the values of the parameters by a small amount. Our algorithm has some biases to the best fit individual of a generation. As we start with the initial probability of a variable 0.5, if that variable in the best individual of the first generation is 0, the algorithm tends to produce 0 for that variable. That is why the algorithm finds optimal solutions of more 0s than 1s of the problems stated above quickly and sometimes terminates with no solutions at all within allowed maximum no. of generations.

BOA seems to be a promising algorithm for the functions of unitation, but it takes more time to find a good Bayesian network for the selected individuals.

And the learning of Bayesian Network is an NP-hard problem, but our proposed algorithm can quickly converge to an optimal solution.

7 Future Works

Our proposed algorithm is dependent on many parameters such as learning rate, sigmoid gain, baseline factor, population size etc. We are now investigating how to control these parameters for quick convergence and lesser fitness evaluations. Another question is how it can be applied to multiary search space. In this paper we have considered all variables as binary, but real world variables can take values from multiary alphabet as well as from continuous domains. We have to find some methods to make our algorithm generally applicable to all problems.

RELEDA assumes no dependency among variables, but in real life problems variables are highly correlated. Problems in permutation domains are very difficult to optimize. There are very few EDAs for combinatorial optimization in permutation domains. We shall investigate an algorithm for such kind of problems. And finally, we shall try to apply this algorithm to real world problems.

8 Summary and Conclusions

In this paper we propose the Reinforcement Learning Estimation of Distribution Algorithm (RELEDA). Our algorithm is a combination of the Estimation of Distribution Algorithm and Reinforcement Learning. We calculate marginal probability distribution of each variable using the selected individuals of a generation. Then the probability of a variable is updated using this marginal probability and the best fit individual of that generation. By sampling the probability vector of the variables, new solutions are generated.

The RELEDA is designed to solve problems in binary search space. It can solve problems with some correlated variables. With appropriate values of different parameters, it can solve problems of diverse directions. In the experiments we have shown that it can outperform some existing EDAs in terms of no. of fitness evaluations required to produce an optimal solution.

References

1. Baluja, S.: Population based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA (1994).
2. Baluja, S. and Caruana,R.: Removing the genetics from standard genetic algorithm. In A. Prieditis and S. Russell, editors, Proceedings of the International Conference on Machine Learning, Morgan Kaufmann, (1995) 38–46 .
3. Baluja, S. and Davies, S.: Using optimal dependency trees for combinatorial optimization: Learning the structure of search space. Technical Report No. CMU-CS-97-107, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA (1997).

4. Berny, A.: Statistical Machine Learning and Combinatorial Optimization. In Kallel, L., Naudts, B. and Rogers, A., editors, Theoretical Aspects of Evolutionary Computing, Springer (2001).
5. De Bonet, J.S., Isbell, C.L. and Viola, P.: MIMIC: Finding Optima by estimating probability densities. Advances in Neural Information Processing Systems, **9** (1997).
6. Deb, K. and Goldberg, D.E.: Sufficient conditions for deceptive and easy binary functions. Annals of Mathematics and Artificial Intelligence, **10** (1994), 385–408.
7. Goldberg, D.E., Korb, B. and Deb, K.: Messy genetic algorithms: Motivation, analysis and first results. Complex Systems **3(5)** (1989) 493–530.
8. González, C., Lozano, J.A. and Larrañaga, P.: Mathematical modeling of discrete estimation of distribution algorithms. In P. Larrañaga and J.A. Lozano, editors, Estimation of Distribution Algorithms: A New Tool for Evolutionary Optimization. Kluwer Academic Publishers, Boston(2001).
9. Harik, G.: Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. IlliGAL Report No. 97005, Illinois Genetic Algorithms Laboratory, University of Illinois, Urbana, Illinois, USA (1997).
10. Harik, G.: Linkage learning via probabilistic modeling in the ECGA. Illigal Report No. 99010, Illinois Genetic Algorithm Laboratory, University of Illinois, Urbana, Illinois, USA (1999).
11. Harik, G.R., Lobo, F.G. and Goldberg, D.E.: The compact genetic algorithm. In Proceedings of the IEEE Conference on Evolutionary Computation,(1998) 523–528
12. Holland, J.H.: Adaptation in Natural and Artificial Systems. The University of Michigan Press (1975).
13. Kargupta, H.: Revisiting the GEMGA: Scalable evolutionary optimization through linkage learning. In Proceedings of 1998 IEEE International Conference on Evolutionary Computation,IEEE Press, Piscataway, New Jersey, USA (1998) 603–608.
14. Larrañaga, P. and Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Optimization. Kluwer Academic Publishers, Boston, (2001).
15. Mühlenbein, H.: The equation for response to selection and its use for prediction. Evolutionary Computation, **5(3)** (1998) 303–346.
16. Mühlenbein, H. and Mahnig, T.: The Factorized Distribution Algorithm for additively decomposed functions. Proceedings of the 1999 Congress on Evolutionary Computation, IEEE press (1999) 752–759.
17. Mühlenbein, H. and Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature-PPSN IV, (1996) 178–187.
18. Paul,T.K. and Iba, H.: Linear and Combinatorial Optimizations by Estimation of Distribution Algorithms. 9th MPS Symposium on Evolutionary Computation, IPSJ Symposium **2003**,Japan (2002),99–106.
19. Pelikan, M., Goldberg, D.E. and Cantú-Paz, E.: Linkage Problem, Distribution Estimation and Bayesian Networks. Evolutionary Computation, **8(3)** (2000) 311–340.
20. Pelikan, M., Goldberg, D.E. and Lobo, F.G.: A survey of optimization by building and using probabilistic models. Technical Report, Illigal Report No. 99018, University of Illinois at Urbana-Champaign, USA (1999).
21. Pelikan, M. and Mühlenbein, H.: The bivariate marginal distribution algorithm. Advances in Soft Computing-Engineering Design and Manufacturing,(1999) 521–535.
22. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning **8** (1992) 229–256.