Estimation of Distribution Algorithm Based on Mixture

Qingfu Zhang, Jianyong Sun, Edward Tsang, and John Ford
Department of Computer Science, University of Essex
CO4 3SQ, Colchester, Essex, U.K.

8th May, 2002, last modified: 13, May

Abstract

Keywords Random EDA, Mixture EDA

1 Introduction

Consider the following binary optimization problem:

$$\max f(x) \tag{1}$$

where $x = (x_1, x_2, \cdots, x_n) \in \Omega = \{0,1\}^n$, and the objective function $f : \Omega \rightarrow R^+$. This problem arises naturally in many applications. Estimation of Distribution Algorithms (EDAs) for solving this problem have attracted a lot of attention in evolutionary computation community in the past several years.The instances of EDAs include the Univariate Marginal Distribution Algorithm (UMDA) and the Population Based Incremental Learning (PBIL). Mutual Information Maximization for Input Clustering (MIMIC) , Combining Optimizers with Mutual Information Trees (COMIT), Factorized Distribution Algorithm (FDA) and Bayesian Optimization Algorithm (BOA), to name only a few.

Like other evolutionary algorithms, EDAs maintain and improve a population of trail points in the search space Ω. Let $Pop(t)$ be the population at generation t. EDAs work in the following recursive way:

Step 1 Selection Select a set of better solutions with higher objective function values from $Pop(t)$ to form the parent set $Q(t)$ by a selection method

Step 2 Modelling Build a new probabilty function $p(x)$ based on the statistical information extracted from the points in $Q(t)$.

Step 3 Sampling Randomly generate new trial points from the search space Ω according to the probability model $p(x)$, and creat a new population $Pop(t+1)$ by replacing points in $Pop(t)$ with new generated points fully or partially.

In the case where the size of population is infinite, if $p(x)$ is the same as the actual probability distribution of the points in $Q(t)$ and proportional, tournament or truncation selection is used in Step 2, then population will converge to the global optimal solutions. Therefore, in the design of EDAs, $p(x)$ should approximate the actual probability distribution of the points in $Q(t)$ as closely as possible. On the other hand, $p(x)$ should be as simple as possible so that it can be easily constructed and used to sample new trail points. Several different models for $p(x)$ have been proposed. In Univariate Marginal Distribution Algorithm (UMDA) and the Population Based Incremental Learning (PBIL), all the variables in $p(x)$ are independent, i.e.

$$p(x) = p(x_1)p(x_2)\cdots p(x_n,)$$

where $p(x_i)$ is the estimated marginal probability of x_i of the points in $Q(t)$. In order to sample a point $x' = (x'_1, x'_2, \cdots, x'_n)$ from the above $p(x)$, one can simply generate each x'_i from $p(x_i)$ independently. UMDA and PBIL igore all the variable dependencies in the probability distribution of points in $Q(t)$. These algorithms work well for the objective function $f(x)$ without significant interactions among the variables in $f(x)$. However, if there are high variable interactions in $f(x)$, UMDA and PBIL may fail in locating the global optimal point. To overcome this shortcoming, De. Bonet et al developed Mutual Information Maximation for Input Clustering (MIMIC) that utilizes the pair-wise dependencies in modelling $p(x)$. In MIMIC, The conditional dependencies of $p(x)$ is defined by a Markovian chain in which each variable is conditioned on the previous one. Very naturally, MIMIC attempts to find a optimal Markovian chain to minimize the cross-entropy between $p(x)$ and actural probability distribution of points in $Q(t)$. However, it is very time-consuming to find the global optimal tree.Therefore, MIMIC uses a greedy search to generate a chain. Bayesian Optimization Algorithm (BOA) can be regarded as an extension of MIMIC. It use a Bayesian network to model $p(x)$. However, finding a "optimal network" is often very time-consuming. As in MIMIC, heuristics have to be employed in learning the network struture in the implementation of BOA. We observed that MIMIC and BOA can utilize only part of pair-wise dependencies. Therefore, no matter what search methods are used in learning the network struture, the resulted $p(x)$ cannot approximate the actural probability of points in $Q(t)$ properly in these algorithms. In other words, the "optimal" structure should have little help in improving the performace of these algorithms.

In this paper, we propose a new EDA for problem (1), called EDA Based on Mixtures (EDAM), in which the mixture of chains is used to model $p(x)$. We randomly pick several chains and set their mixture as the model of $p(x)$. Since EDAM does not involve any search for the optimal structure, the overhead at each iteration is reasonable small. We EDAM and MIMIC on a number of test functions. Our results show that the actual performance of EDAM is very similar to that of MIMIC.

2 Algorithms

2.1 MIMIC

In MIMIC, the probability model $p(x)$ in Step 2 is defined by is $\pi = i_1 i_2 \cdots i_n$, a permutation of $1, 2, \cdots, n$, In the following, we write it as $p_\pi(x)$. $p_\pi(x)$ is of the form:

$$p_\pi(x) = p(x_{i_1}|x_{i_2}) \cdots p(x_{i_{n-1}}|x_{i_n}) p(x_{i_n})$$

where $p(x_{i_{k-1}}|x_{i_k})$ is the estimated conditional probability of variable $x_{i_{k-1}}$ conditioned on $x_{i_{k-1}}$of the points in $Q(t)$. The dependency structure of $p_\pi(x)$ is the chain in which x_{i_k} is the $k-$th node. MIMIC seeks the permutation π that minimizes the Kullback-Liebler distance $d(p_\pi, p_{true})$ between $p_\pi(x)$ and the actural probability $p_{true}(x)$. $d(p_\pi, p_{true})$ is defined as

$$d(p_\pi, p_{true}) = \sum_{x \in \Omega} p_{true}(x)[\ln p_{true}(x) - \ln p_\pi(x)]$$

Finding the exact optimal permutation π is very computational expensive. Therefore, De Bonet et al proposed to use a heuristic algorithm to search a near-optimal π. The time complexity of their heuristic algorithm is $O(n^2)$.

There are $\frac{n(n-1)}{2}$ pairwise dependencis in $p_{true.}$ MIMIC can make use of only $n-1$ pairwise dependencis. Therefore, the gap between $p_\pi(x)$ and $p_{true}(x)$ cannot be eliminated in theory.

2.2 EDAM

In EDAM, we model $p(x)$ as a mixture of several chain based probabilities:

$$p(x) = \frac{1}{M} \sum_{j=1}^{M} p_{\pi_j}(x) \tag{2}$$

where $\pi_j = i_{j1} i_{j2} \cdots i_{jn}$ $(j = 1, 2, \cdots, M)$ are M different permutations, $p_{\pi_j}(x)$ is:

$$p_{\pi_j}(x) = p(x_{i_{j1}} | x_{i_{j2}}) \cdots p(x_{i_{j(n-1)}} | x_{i_{jn}}) p(x_{ji_n})$$

$p(x_{i_{j1}} | x_{i_{j2}})$ and $p(x_{ji_n})$ are estimated conditional probabilities and marginal probabilities of the points in $Q(t)$, respectively. To reduce the computational cost, we randomly generate M permutations in each iteration of EDAM. The sampling in Step 3 of EDAM is also very simple. To sample a point from the model (2), we can randomly pick a component probability function $p_{\pi_j}(x)$ with equal probability, then sample a point from $p_{\pi_j}(x)$.

3 Expirical Results

Jianyong, please clarify the following issues:

(1) the detailed information of the selection?
(2) how many runs per instances.

We have performed experiments to compare the performance of MIMIC and EDAM. In our experiments, we used trunction selection for both MIMIC and EDAM. For all the runs, the $r\%$ best points in $Pop(t)$ are selected to enter into $Q(t)$ in Step 2. M is set to 10 in EDAM.

3.1 Test Problem

We used binary quadractic optimization problems as test problems. Binary quadractic optimization problems are of form:

$$\max f(x) = x^T A x \tag{3}$$

where $A = (a_{ij})$ is an $n \times n$ symmetric matrix. These problems are known to be NP-hard. In our experiments, we randomly generated a_{ij} from $(10, -10)$.

3.2 Comparision

The following is the test results for problem 10. The matrix A is generated randomly. We generated different As which include different dimensions and zero percents and stored in some data files. For example, data file $a30.100.dat$ denotes that this matrix has dimension 30 and 10 zero-percentages and the last 0 of 100 means the index of the data files with dimension 30 and zero percentages 10. For every zero percentage (10, 30, 50, 70, 90), we have 10 randomly-generated data files. And there are two classes of data files of dimension 30 and 60. The data files are numbered from 1 to 100.

The following table lists the results when testing $a30.500.dat$.

	$a30.500.dat$
MIMIC	5%
Random EDA	(1) 5% (2) 30% (3) 30% (4) 30%
Mixture EDA	30%

The percents in the table denote the percentages of best solutions found by these algorithms. For $a30.500.dat$, the best solution found is 522.1920. In the row of Random EDA, (1), (2), (3), (4) denote different chains used in the algorithm, which are 1, 5, 10, 100, 1000 (Mixture EDA), respectively in order to see the influence of different numbers of chains used. We can see that along with the increase of chains, the performance of Random EDA becomes better, then preserves constant, and 10 is the crucial point.

Random-Tree EDA: The previous Random EDA and Mixture EDA algorithm all use randomly-generated chains to generate new points in the next population. The Random-Tree EDA uses some randomly-generated trees to generate new points. Through doing some experiments ($a30.*.dat$, $a60.*.dat$), we find that the performance of Random-Tree EDA has no big difference with EDA who adopts randomly generated chains. Also, we find that EDA algorithms no matter using randomly-generated chains or trees has the similar performance as MIMIC algorithm.

The following tables list the best fitness values found by MIMIC and Random-EDA of problem 10 with different matrix As.

	1	2	3	4	5
MIMIC	1254.2859	411.3617	617.0026	502.4735	570.0186
Random EDA	1254.2859	411.3617	**617.2584**	502.4735	570.0186

	6	7	8	9	10
MIMIC	395.2712	899.7845	539.8989	457.7609	487.8002
Random EDA	395.2712	899.7845	539.8989	457.7609	487.8002
	11	12	13	14	15
MIMIC	411.1794	462.8862	597.8094	259.1725	546.3057
Random EDA	411.1794	462.8862	597.8094	259.1725	546.3057
	16	17	18	19	20
MIMIC	408.9334	381.5863	408.0520	462.8328	485.1526
Random EDA	408.9334	381.5863	408.0520	462.8328	485.1526
	21	22	23	24	25
MIMIC	522.1920	373.0384	322.3348	330.1824	387.9073
Random EDA	522.1920	373.0384	322.3348	330.1824	387.9073
	26	27	28	29	30
MIMIC	400.7901	416.5165	366.0421	315.1369	361.8236
Random EDA	400.7901	416.5165	366.0421	315.1369	361.8236
	31	32	33	34	35
MIMIC	275.9103	262.0890	238.9798	287.2588	244.8242
Random EDA	275.9103	262.0890	238.9798	287.2588	244.8242
	36	37	38	39	40
MIMIC	316.5128	453.3962	238.6005	282.3145	343.4582
Random EDA	316.5128	453.3962	238.6005	282.3145	343.4582
	41	42	43	44	45
MIMIC	116.4867	162.0989	132.4285	100.1149	164.9051
Random EDA	116.4867	162.0989	132.4285	100.1149	164.9051
	46	47	48	49	50
MIMIC	127.3539	109.6411	198.8599	129.6940	177.0543
Random EDA	127.3539	109.6411	198.8599	129.6940	177.0543
	51	52	53	54	55
MIMIC	1720.7623	1487.3300	1693.6855	1500.9944	1618.6859
Random EDA	1720.7623	1487.3300	1693.6855	1500.9944	1618.6859
	56	57	58	59	60
MIMIC	1158.6860	1350.9864	1784.7479	**1221.0650**	1457.0314
Random EDA	**1162.6267**	1350.9864	1784.7479	1208.3145	1457.0314
	61	62	63	64	65
MIMIC	1286.4973	1420.2423	1527.1575	1619.0562	1195.7518
Random EDA	1286.4973	1420.2423	1527.1575	1619.0562	1195.7518

	66	67	68	69	70
MIMIC	**1377.4330**	1150.3335	1444.7603	1513.9651	1296.1625
Random EDA	1373.2068	1150.3335	1444.7603	1513.9651	1296.1625

	71	72	73	74	75
MIMIC	1019.3873	1102.7991	**1281.7289**	1300.1639	898.3020
Random EDA	**1020.2873**	1102.7991	1278.8395	1300.1639	**906.6753**

	76	77	78	79	80
MIMIC	1139.5620	1352.2146	1094.1706	988.0009	**1063.5438**
Random EDA	1139.5620	1352.2146	1094.1706	988.0009	1056.9756

	81	82	83	84	85
MIMIC	875.5979	919.0432	899.0190	1122.0436	600.1423
Random EDA	875.5979	919.0432	899.0190	1122.0436	600.1423

	86	87	88	89	90
MIMIC	925.6242	598.2206	851.8573	843.4505	**604.5557**
Random EDA	925.6242	598.2206	**860.9593**	843.4505	599.6567

	91	92	93	94	95
MIMIC	421.9731	577.4069	331.3233	**475.1880**	548.7675
Random EDA	421.9731	577.4069	331.3233	474.0643	548.7675

	96	97	98	99	100
MIMIC	538.1555	490.2890	**488.0883**	526.8893	556.6000
Random EDA	538.1555	490.2890	487.2258	526.8893	556.6000

The bold number in the tables show the better fitness value found. From the above table, we can find that the best fitness values obtained by MIMIC and Random EDA are about the same. The following figures show the evolution graph of MIMIC and Random EDA for some different data. It also can be seen from these figures that the performance of MIMIC and Random EDA is about the same.

4 Conclusion

Many EDA algorithms assume that the structure of solutions reflects complex relationships between the different input parameters. Hence, many of the search work in EDAs is paid for finding the optimal structure out. In many cases, to find the optimal structure is a NP-hard problem. And obviously, no one tell us if the search for optimal structure is necessary. We also doubt if the structure can be clearly expressed by a single optimal structure like chain or tree.

In this paper, we empirically claim that the optimal chain is not necessary.

Figure 1: Evolution Graph of $F(x) = x^T Ax$ for different algorithm (MIMIC and Random-EDA) with data file $a30.100.dat$

Figure 2: Evolution Graph of $F(x) = x^T Ax$ for different algorithm (MIMIC and Random-EDA) with data file $a30.100.dat$

Figure 3: Evolution Graph of $F(x) = x^T Ax$ for different algorithm (MIMIC and Random-EDA) with data file $a30.300.dat$

Figure 4: Evolution Graph of $F(x) = x^T Ax$ for different algorithm (MIMIC and Random-EDA) with data file $a30.308.dat$

Figure 5: Evolution Graph of $F(x) = x^T Ax$ for different algorithm (MIMIC and Random-EDA) with data file $a30.500.dat$

Figure 6: Evolution Graph of $F(x) = x^T Ax$ for different algorithm (MIMIC and Random-EDA) with data file $a30.708.dat$

Figure 7: Evolution Graph of $F(x) = x^T Ax$ for different algorithm (MIMIC and Random-EDA) with data file $a60.100.dat$

Figure 8: Evolution Graph of $F(x) = x^T Ax$ for different algorithm (MIMIC and Random-EDA) with data file $a60.308.dat$

By comparison of the proposed Random EDA and MIMIC (a well-known EDA algorithm), we find that the performance of these two algorithms is about the same.

In the future work, we will devote to exploit if there exists some optimal structures (for example, some structures designed by uniform Latin squares or orthogonal Latin squares or golfer permutation) which can include enough complex relationships between the input parameters.

References

[1] Pelikan, M., Goldberg, D.E., & Cantu-Paz, E. (1998). *Linkage problem, distribution estimation, and Bayesian networks* (Technical Report 98013). Urbana, IL: University of Illinois at Urbana-Champaign.

[2] Pelikan, M., Goldberg, D.E., & Cantu-Paz, E. (1999). *Boa: The bayesian optimization algorithm* (Technical Report 99003). Urbana, IL: University of Illinois at Urbana-Champaign.

[3] Pelikan, M., & Muhlenbein, 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.

[4] De Bonet, J.S., Isbell, L.S Jr., Viola, P. (1997). *MIMIC: Finding Optima by Estimating Probability Densities.* Advances in Neural Information Processing Systems 1997 MIT Press, Cambridge, MA.

[5] Baluja, S. (1994). *Population=Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning, (*Carnegie Mellon Report, CMU-CS-94-163).

[6] Muhlenbein,H. (1998). *The Equation for Response to Selection and its Use for Prediction, Evolutionary Computation 5, 303-346.*

[7] Muhlenbein,H. and Paaβ,G. (1996). From Recombination of Genes to the Estimation of Distributions I. Binary Parameters, in H.M.Voigt, et al., eds., *Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature - PPSN IV, 178-187.*

[8] Muhlenbein,H. and Mahning,T. (1999b). The Factorized Distribution Algorithm for Additively Decomposed Functions, *Second Symposium on Artificial Intelligence. Adaptive Systems. CIMAF99, 301-313,* La Habana