Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, Baoding, 12-15 July 2009

HIGH-ORDER EDA

JIN ZENG[1], QING-SHENG REN[2]

[1] Department of Mathematics, Shanghai Jiao Tong University, Shanghai 200240, P. R. China
[2] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, P. R. China
E-MAIL: zengjin@sjtu.edu.cn, ren-qs@cs.sjtu.edu.cn

Abstract:

In this paper, we investigate the usage of history information for estimation of distribution algorithm (EDA). In EDA, the distribution is estimated from a set of selected individuals and then the estimated distribution model is used to generate new individuals. It needs large population size to converge to the global optimum. A new algorithm, the high-order EDA, is proposed based on the idea of filter. By the usage of history information, it can converge to the global optimum with high probability even with small population size. Convergence properties are then discussed. We also show the application for constrained optimization problems.

Keywords:

Estimation of distribution algorithm (EDA); High-order EDA; Convergence; Constraint optimization

1. Introduction

Genetic algorithms (GAs) are population-based optimization methods based on the evolutionary theory. Most of the theory of GAs deals with the building blocks [1]. The genetic algorithms implicitly manipulate a large number of building blocks by mechanisms of selection and re-combination, i.e., reproduce and mix building blocks. But two crucial factors of the GA success, a proper growth and mixing of good building blocks, are often not achieved [2]. The fixed mapping from the space of solutions into the internal representation of the solutions in the algorithm and simple two-parent recombination operators soon showed to be insufficiently powerful even for problems that are composed of simpler partial sub-problems. The problem-independent recombination operators often break partial solutions what can sometimes lead to losing these and converge to a local optimum.

Various attempts to prevent the disruption of important partial solutions have been done. An interesting direction is to estimate the distribution of the promising solutions and use this estimate in order to generate new individuals. A general scheme of the algorithms based on this principle is called the Estimation of Distribution Algorithm (EDA) [3], which is showed as:

1) t=0, randomly generate initial population $P(0)$
2) select a set of promising solutions $P^s(t)$ from $P(t)$
3) estimate the distribution of the selected set $P^s(t)$
4) create a new population $P(t+1)$ according to the estimate, $t = t+1$
5) if the termination criteria are not met, go to 2)

EDAs differ in the way of estimating the distribution and using this estimate for generation of new individuals. The simplest way is to assume that the variables are independent and to look at the values of each variable regardless of the remaining solutions, like UMDA [3]. This kind of algorithms works well on linear problems where the variables are not mutually interacting. Pairwise models allow covering some interactions in a problem and are very easy to learn. The algorithms based on pairwise models, such as MIMIC [4], reproduce and mix building blocks of order two very efficiently, and therefore they work very well on both linear and quadratic problems. In order to deal with the problems with multivariate or highly-overlapping building blocks, we need more complex models, such as FDA [5] and BOA [6].

Although EDA is used successfully in many kinds of application, it has the shortcoming of large population size. In order to achieve optima, EDA always needs a critical population size. The determination of a sufficient size of the population turned out to be difficult. Like GAs, EDA only use the information of the selected population and do not use any information of the population before selection. In this paper we will introduce a new algorithm which is called high-order EDA. By using the information of the evolution process, this new algorithm can achieve good results with small population size. The paper is organized as follows. The framework of the high-order EDA is proposed in Section 2. Section 3 gives some convergence results for high-order EDA. In Section 4, we will show how to use high-order EDA to solve constraint optimization problems. Section 5 summarizes the paper.

978-1-4244-3703-0/09/$25.00 ©2009 IEEE

2. High-order EDA

2.1. Frame of high-order EDA

In EDA, the new population is normally generated by the distribution of $P^s(t)$, i.e.,

$$P(x,t) = P^s(x,t-1)$$

From this equation we can find that the new population only uses the information of the selected population and has no relation with the last generation. Some researchers noticed this point and proposed some new algorithms, such as IUMDA [7], PBIL [8] and HCwL [9]. Although the realization of these algorithms is different, the basic ideas are the same as the following:

$$P(x,t) = \lambda P^s(x,t-1) + (1-\lambda)P(x,t-1) \quad 0 \le \lambda \le 1 \quad (1)$$

i.e., the distribution of the new population not only use the information after selection; but also use the information before selection. When $\lambda = 1$, it belongs to the traditional EDA. When $\lambda = 0$, the individual probability keeps unchanged and loses the capability of finding optimal solution.

If we consider equation (1) as the description of a linear system, we can get the transformation function of this linear system

$$H(z) = \frac{\lambda z^{-1}}{1-(1-\lambda)z^{-1}}$$

Obviously it is an IIR digital low pass filter. It can enhance the low-frequency content and restrain the high-frequency content. In this way the serge near the convergent point will be restrained and at the same time the convergence speed will be improved. Because $0 \le \lambda \le 1$, this filter is steady. But when λ tends to 0 the inertia will be increased and the function of the selection operator will be decreased. When $\lambda = 0$ the selection is useless; In order to prevent this problem, we can limit the range of λ. We also can use FIR filter. In this situation we let

$$P(x,t) = \sum_{i=1}^{m} \alpha_i P^s(x,t-i) \quad (2)$$

$$0 \le \alpha_i \le 1, \ \sum_{i=1}^{m} \alpha_i = 1$$

We call the algorithm based on equation (2) as high-order EDA and m is the order. The corresponding transformation function is

$$H(z) = \sum_{i=1}^{m} a_i z^{-i}$$

In the following, we let $m = 2$ for simplicity and it can be called as 2nd-order EDA. The core equation of 2nd-order EDA can be written as

$$P(x,t) = \lambda P^s(x,t-1) + (1-\lambda)P^s(x,t-2) \quad 0 \le \lambda \le 1$$

And the frame of 2-order EDA can be summarized as

1) $t = 0$, randomly generate initial population $P(0)$
2) select a set of promising solutions $P^s(0)$ from $P(0)$
3) estimate the distribution of the selected set $P^s(0)$
4) create a new population $P(1)$ according to the estimate, $t = 1$
5) select a set of promising solutions $P^s(t)$ from $P(t)$
6) estimate the distribution of the selected set $P^s(t)$
7) create a new population according to the estimate $P(x,t) = \lambda P^s(x,t-1) + (1-\lambda)P^s(x,t-2)$
8) if the termination criteria are not met, $t = t+1$, go to 5)

2.2. 2nd-order UMDA

In order to implement UMDA, estimates for the univariate marginal distributions are necessary. These estimates can be simply provided by the selected parents. In this case, the new individual $x = (x_1 x_2 \cdots x_n)$ is created according to the following equations:

$$P(x_i,t) = P^s(x_i,t-1)$$

$$P(x_1 x_2 \cdots x_n, t) = P(x_1,t)P(x_2,t)\cdots P(x_n,t)$$

But the previous marginal frequencies can also be taken into accounts as well. In IUMDA, the update rule for the i th gene is

$$P(x_i,t) = \lambda P^s(x_i,t-1) + (1-\lambda)P(x_i,t), \quad 0 \le \lambda \le 1$$

When $\lambda = 1$, IUMDA is identical to traditional UMDA.

If the UMDA is combined with 2nd-order EDA, the estimate for the univariate marginal distribution can be got as the following:

$$P(x_i,t) = \lambda P^s(x_i,t-1) + (1-\lambda)P^s(x_i,t-2), \quad 0 \le \lambda \le 1$$

When $\lambda = 1$, 2nd-order EDA is also identical to traditional UMDA.

Eg 1(OneMax) $\max \sum_{i=1}^{n} x_i$, $x_i = 0,1$

This problem, which is called as OneMax problem, is a very simple optimization problem. Because each variable is independent to each other, it can be solved by UMDA.

In Table 1, we compared UMDA with IUMDA and 2nd-order EDA. With the given population size N and λ, 100 independent experiments were made. In each experiment, the population can converge within 100

generations. The experimental results are expressed by the runs it converges the global optimum among these 100 independent runs.

Table 1. Successful rate of UMDA, IUMDA and 2nd-order EDA for OneMax

		λ	n=500	n=700	n=1000
N=100	UMDA		0	0	0
	IUMDA	0.7	1	0	0
		0.5	0	0	0
	2nd-order EDA	0.7	4	0	0
		0.5	20	1	0
N=150	UMDA		28	0	0
	IUMDA	0.7	45	6	0
		0.5	51	5	0
	2nd-order EDA	0.7	76	28	1
		0.5	89	68	11
N=200	UMDA		80	49	4
	IUMDA	0.7	90	68	12
		0.5	87	46	13
	2nd-order EDA	0.7	97	89	51
		0.5	100	97	71
N=300	UMDA		100	99	89
	IUMDA	0.7	99	99	92
		0.5	100	99	81
	2nd-order EDA	0.7	100	99	99
		0.5	100	100	99

From this table we can see that the performance of UMDA is quite good when the population size N is large enough respective to the problem size n. But with the decreasing of the population size, the performance became worse. When n=1000 and N=200, we can only converge to the global solution for 4 times. IUMDA performs better. Among 100 runs, we can get the solution in about 10% runs. But 2nd-EDA performs much better, the successful rate is as high as 71% when $\lambda = 0.5$. This means 2nd-EDA can get better results with smaller memory storage.

3. Further discussion

3.1. Convergence

Suppose the search space is represented by

$$\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_n$$

where $|\Omega_i| = r_i$ and n is the length of the individual. The cardinality of the search space is $|\Omega| = r_1 \times r_2 \times \cdots \times r_n = m$. According to the multi-set sense, the population in the algorithm is a subset of size M of elements of Ω. Each population D_w can be represented as a vector

$$D_w = (d_{w1}, d_{w2}, \ldots, d_{wm})$$

where d_{wi} is the number of ith individuals in population D_w and $\sum_{i=1}^{m} d_{wi} = M$.

A 2nd -order EDA can be modelled using a finite Markov chain whose state space is formed from the different populations that the algorithm can take:

$$E = \{D_1, D_2, \ldots, D_v\}$$

where $v = C_{M+m-1}^{m-1}$ is the number of different populations.

A 2nd-order Markov chain model can be used here because the population at step t only depends on the populations at step $t-1$ and $t-2$. Moreover, neither operation used for the calculation of the transition probabilities depends on the step parameter t, so the Markov chain is homogeneous.

Theorem 1: Let A be an instance of 2nd-order EDAs such that

$$P(x,t) \geq \varepsilon > 0$$

for all $x \in \Omega$ and for all step $t = 1,2,\ldots$. Then A visits populations of D^*, which is the set of populations that contain a global optimum, infinitely often with probability one. If the selection is elitist, then the 2nd-order EDA converges to a population that contains the global optimum.

Proof: First suppose the selection is not elitist.

The probability of going from population D_p (at step $t-1$) and D_q (at step $t-2$) to D_r at step t is given by

$$P(D_r \mid D_p D_q) = \sum_{D_p^{sel} D_q^{sel}} P^s(D_p^{sel} D_q^{sel}) \frac{M!}{d_{r1}! \cdots d_{rm}!} \prod_{i=1}^{m} \underbrace{(\lambda P^p(x^{(i)}) + (1-\lambda) P^q(x^{(i)}))^{d_{ri}}}_{>0} > 0$$

where $P^s(D_p^{sel} D_q^{sel})$ is the probability to select $D_p^{sel} D_q^{sel}$ from D_p and D_q. $P(x, t-1)$ coincides with some $P^P(x^i)$ and $P(x, t-2)$ coincides with some $P^q(x^i)$.

Hence the Markov chain is irreducible, i.e., all the states are intercommunicated. And the chain will visit D^* infinitely often with probability 1.

If the selection is elitist, then the global optimum will never be lost when it is found. Therefore the algorithm converges to a population that contains the global optimum.

3.2. Convergence speed

Here we again use OneMax problem to show the relation between λ and convergence speed of 2nd-order EDA. Because the variables are independent with each other and the initial value is identical, we can give the following assumption without loss of generality

$$P(x_i = 1, t) \equiv p(t), \quad P(x_i = 0, t) \equiv q(t)$$

$$P^s(x_i = 1, t) \equiv r(t) \qquad i = 1, \ldots, n$$

Obviously

$$p(t) + q(t) \equiv 1$$

After the Roulette wheel selection，

$$r(t) = p(t) + \frac{1 - p(t)}{n} = \frac{1}{n} + (1 - \frac{1}{n}) p(t)$$

So

$$p(t) = \lambda r(t-1) + (1-\lambda) r(t-2)$$

$$= \lambda [\frac{1}{n} + \frac{1}{n} p(t-1)] + (1-\lambda)[\frac{1}{n} + \frac{1}{n} p(t-2)]$$

$$= \frac{1}{n} + (1 - \frac{1}{n})[\lambda p(t-1) + (1-\lambda) p(t-2)]$$

$$q(t) = (1 - \frac{1}{n})[\lambda q(t-1) + (1-\lambda) q(t-2)]$$

According to the theory of difference equation, we have

$$q(t) = c_1 x_1^t + c_2 x_2^t$$

where c_1, c_2 are constants and

$$x_1 = \frac{(1 - \frac{1}{n})\lambda + \sqrt{(1 - \frac{1}{n})^2 \lambda^2 + 4(1 - \frac{1}{n})(1-\lambda)}}{2} > 0$$

$$x_2 = \frac{(1 - \frac{1}{n})\lambda - \sqrt{(1 - \frac{1}{n})^2 \lambda^2 + 4(1 - \frac{1}{n})(1-\lambda)}}{2} \le 0$$

Obviously $|x_2| < |x_1| < 1$ and $\lim_{t \to \infty} q(t) = 0$. It means the algorithm could converge to the global optimum. Then let's see when the algorithm converge fast, i.e.,

$$\min_{\lambda} q(t)$$

By the optimization method we can find the algorithm converges fast when $\lambda = 1$. It means the speed of UMDA is faster than 2nd-order EDA. In the numerical experiment we find the same results, which can be found in table 2. In table 2 we just consider those runs that converge to the global optimum.

Table 2. Average convergence steps of UMDA and 2nd-order EDA for OneMax

		λ	n=500	n=700	n=1000
N=100	UMDA		--	--	--
	2nd-order	0.7	40.3	--	--
	EDA	0.5	45.8	56.0	--
N=150	UMDA		30.3	--	--
	2nd-order	0.7	39.3	47.2	59.0
	EDA	0.5	44.6	53.7	66.0
N=200	UMDA		29.0	35.3	44.3
	2nd-order	0.7	37.9	46.1	56.4
	EDA	0.5	43.3	52.3	64.1
N=300	UMDA		28.4	34.1	41.9
	2nd-order	0.7	37.5	44.8	54.7
	EDA	0.5	43.0	51.3	62.3

Although high order EDA converges slower than traditional EDA, the probability of convergence of high order EDA is higher, especially for small population size.

4. Application of constrained optimization

The constraint problem we discuss here is defined as:

$$\max f(X) = \sum_i f_i(S_i)$$

$$s.t.\ C_i(S_i) \le 0$$

where $X = \{x_1, \cdots, x_n\}$ and the value of i th variable belongs to the set $\{x_{i,1}, \cdots, x_{i,n_i}\}$. $C_i(S_i) \le 0$ stands for the i th constraint function (it may be equality or inequality)

and the variable set $S_i \subseteq X$ ($i = 1, \cdots, l$). The function $f(x)$ is called additively decomposed function (ADF). This class of functions is of great theoretical and practical importance. Optimisation of an arbitrary function in this space is NP complete.

4.1. The frame of the algorithm

To solve the problem, we must get a factorisation of the probability of distribution at first. For convenience, just suppose $X = \bigcup_{i=1}^{l} S_i$. Define

$$d_i = \bigcup_{j=1}^{i} S_j \qquad b_i = S_i \setminus d_{i-1} \qquad c_i = S_i \cap d_{i-1}$$

Set $d_0 = \phi$ and then we get the factorisation as

$$P(X) = \prod_{i=1}^{l} P(x_{b_i} \mid x_{c_i})$$

If

$$b_i \neq \phi, \quad \forall i = 1, \cdots l; \quad d_l = X$$
$$\forall i \geq 2, \quad \exists j < i \text{ such that } c_i \subseteq S_j$$

we say the factorisation satisfy the *running intersection property*. At this condition, $P(x_1, \cdots, x_n) = \prod_{i=1}^{l} P(x_{b_i} \mid x_{c_i})$ really holds [5]. If the running intersection property is violated, the factorisation might not be exact. But we will show the running intersection property is not necessary by the numerical examples.

We assume the factorisation of the probability distribution is given. The following is the frame of the algorithm 2nd-CFA (Constraint Factorisation Algorithm) to solve the constraint problem:

1) Get initial feasible population;
2) select a set of promising solutions;
3) Compute the probabilities $P^s(x_{b_i} \mid x_{c_i}, t-1)$ using the selected points;
4) Generate the new population according to $P(x,t) = \prod_{i=1}^{l} [\lambda P^s(x_{b_i} \mid x_{c_i}, t-1) + (1-\lambda) P^s(x_{b_i} \mid x_{c_i}, t-2)]$
5) if the termination criteria are not met, go to 2)

4.2. Feasibility

Before the further discussion, we should know if the algorithm 2nd-CFA is feasible. This means the final solution should be feasible. The following theorem gives us the guarantee:

Theorem 2: If the former generation is feasible, the new generation will be feasible too.

Proof: If $\exists x \in P(t)$ and x doesn't satisfy the k th constraint $C_k(S_k)$. Then

$$0 \neq P(x,t) =$$
$$\prod_{i=1}^{l} [\lambda P^s(x_{b_i} \mid x_{c_i}, t-1) + (1-\lambda) P^s(x_{b_i} \mid x_{c_i}, t-2)]$$

$$\rightarrow P^s(x_{b_k} \mid x_{c_k}, t-1) \neq 0 \text{ or } P^s(x_{b_k} \mid x_{c_k}, t-2) \neq 0$$
$$\rightarrow P^s(x_{S_k}, t-1) \neq 0 \text{ or } P^s(x_{S_k}, t-2) \neq 0$$
$$\exists \tilde{x} \in P(t-1), \quad \tilde{x}_{S_k} = x_{S_k} \quad \text{or} \quad \exists \hat{x} \in P(t-2), \quad \hat{x}_{S_k} = x_{S_k}$$
$$\therefore \tilde{x} \text{ or } \hat{x} \text{ doesn't satisfy } C_k(S_k)$$

And it is impossible because we suppose the former generation is feasible.

Although Theorem 2 guarantees that we can get a feasible population from a feasible former, we need a feasible initial population.

4.3. Numerical results

In the following examples, the experiment results are expressed by how many runs it gets the global optimum among 100 independent runs. The max generation is 100 and the population size is denoted as N.

$$\text{Eg2} \qquad \max \sum_{i=1}^{n} x_i$$
$$s.t. \; x_{2j-1} + x_{2j} + x_{2j+1} \leq 2$$
$$x_1 + x_{n-1} + x_n \leq 2$$

where $n = 2m$, $x_i \in \{0,1\}$ $i = 1, \cdots n$ $j = 1, \cdots, m-1$

The table 3 shows the results. This example is a linear programming problem. In order to achieve the same convergence rate, the population size needed for $\lambda = 0.5$ is much smaller than that of $\lambda = 1$. At the same time we can get the factorisation easily as

$$P(x_1, \cdots, x_n) =$$
$$P(x_1 x_2 x_3) P(x_4 x_5 \mid x_3) \cdots P(x_{n-2} x_{n-1} \mid x_{n-3}) P(x_n \mid x_1 x_{n-1})$$

Obviously it does not satisfy the running intersection property. But we can still get the global optimum.

Table 3. Successful rate for Example2

N	200		500		700		1000	
λ	1	0.5	1	0.5	1	0.5	1	0.5
n=200	1	12	81	88	99	100	99	100
n=300	0	0	45	80	79	97	99	100
n=398	0	0	41	85	88	100	98	100
n=502	0	0	8	58	60	98	95	100

Eg3 $$\max \sum_{i=1}^{n} x_i$$

$$s.t.\ 2 \le x_{2j-1}^2 + x_{2j}^2 + x_{2j+1}^2 \le 8$$

where $n = 2m+1$, $x_i \in \{0,\pm 1,\pm 2\}$ $i = 1,\cdots n$ $j = 1,\cdots,m$

The table 4 shows the results. This example is a non-linear programming problem with concave domain and the range of each variable is not small. It needs large population size. But the usage of λ can lead to better results with small population size.

Table 4. Successful rate for Example3

N	1000		2000		3000		4000	
λ	1	0.5	1	0.5	1	0.5	1	0.5
n=101	0	3	46	74	80	97	97	100
n=201	0	0	0	5	14	66	55	93
n=299	0	0	0	0	5	50	37	89

5. Conclusion

In this paper, a new algorithm, which is called as high-order EDA, is proposed. The basic character is the usage of the history information. From the numerical examples we find that the new algorithm can converge to the global optimum with small population size. Further research includes the choice of the order, the choice of the coefficients, etc.

Acknowledgements

This paper is supported by NSF 90820018.

References

[1] Goldberg D. E., Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, 1989.

[2] Thierens D., Analysis and design of genetic algorithms, Leuven, Belgium: Katholieke Universiteit Leuven, 1995.

[3] Mühlenbein H., Paaβ G., "From recombination of genes to the estimation of distributions I. Binary parameters", Parallel Problem Solving from Nature, PPSN IV, pp.178-187, 1996.

[4] De Bonet J.S., Isbell C.L., Viola, P., "MIMIC: Finding optima by estimating probability densities", Advances in Neural Information Processing Systems 9, pp.424-430, 1997.

[5] Mühlenbein H., Mahnig T., Rodriguez A. O., "Schemata, distributions and graphical models in evolutionary optimization", Journal of Heuristics, Vol 5, No.2, pp. 215-247, 1999.

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

[7] Mühlenbein H., "The equation for response to selection and its use for prediction", Evolutionary Computation, Vol 5, No.3, pp.303-346, 1997

[8] Baluja S., 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, 1994.

[9] Kvasnicka V., Pelikan M., Pospichal J., "Hill climbing with learning (An abstraction of genetic algorithm)", Neural Network World 6, 773-796, 1996.