Adaptive Combinatorial Optimization Algorithm based on UMDA, EO and LA

Mitra Hashemi[۱]; **Mohammad Reza Meybodi**[۲]

ABSTRACT

UMDA algorithm is a type of Estimation of Distribution Algorithms. This algorithm has better performance compared to others such as genetic algorithm in terms of speed, memory consumption and accuracy of solutions. It can explore unknown parts of search space well. It uses a probability vector and individuals of the population are created through the sampling. Furthermore, EO algorithm is suitable for local search of near global best solution in search space, and it does not stuck in local optimum. Hence, combining these two algorithms is able to create interaction between two fundamental concepts in evolutionary algorithms, exploration and exploitation, and achieve better results of this paper is used adaptive version of τ -EO algorithm called EO-LA. In this method the task of choosing a replacement component is assigned to Learning Automata. During the implementation of this algorithm, according to the suitability of produced solutions, feedback signals are sent to Learning Automata until adapt selected replacement component well. In this paper, results represent the better performance of the proposed algorithm (combination of three methods) on a Graph Bi-partitioning, NP-hard problem.

KEYWORDS

Univariate Marginal Distribution Algorithm, Extremal Optimization, Learning Automata, Estimation of Distribution Algorithm, optimization problem

۱. INTRODUCTION

During the ninetieth century, Genetic Algorithms (GAs) helped us solve many real combinatorial optimization problems. But the deceptive problem where performance of GAs is very poor has encouraged research on new optimization algorithms. To combat these dilemma some researches have recently suggested Estimation of Distribution Algorithms (EDAs) as a family of new algorithms [۲, ۳, ٤]. In contrast to GAs which considers crossover and mutation operators as essential tools to generate new populations, EDAs replace those operators by the estimation and sampling of the joint probability distribution of the selected individuals. This method has many advantages which can be illustrated by avoiding premature convergence and use of a compact and short representation.

In ۱۹۹٦, Muhlenbein and PaaB [۲۰۳] have proposed the Univariate Marginal Distributions Algorithm (UMDA), which approximates the simple genetic algorithm. It consists of a genetic algorithm with gene pool recombination, without mutation. Remarkable is the new view of genes. Choosing the bits from the parents can also be seen as sampling from a probability distribution. UMDA uses a very simple distribution: the product of the distributions for the single bits. This assumes no dependencies between bits. One problem of GA is that it is very difficult to quantify

[۱] Teacher, Payamnur university, East Azarbayjan, Iran, mitra.hash@yahoo.com .

[۲] Professor, Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran, mmeybodi@aut.ac.ir and thus analyze these effects. UMDA is based on probability theory, and its behavior can be analyzed mathematically.

Self-organized criticality has been used to explain behavior of complex systems for such different areas as geology, economy and biology. To show that SOC [۶،۷] could explain features of systems like the natural evolution, Bak and Sneepen developed a simplified model for an ecosystem in which species are placed side by side on a line with periodic boundary conditions. To each species, fitness numbers is assigned randomly, with uniform distribution, in the range [۰،۱]. The least adapted species, one with the least fitness, is then forced to mutate, and a new random number assigned to it. The change in the fitness of the least adapted species alters the fitness landscape of their neighbors, and to cope with that new random numbers are also assigned to them, even if they are well adapted after some iteration, the system evolves to a critical state where all species have fitness above a critical Threshold.

The ability of EO in exploring search space was not as well as its ability in exploiting whole search space; therefore, combination of two methods, UMDA and EO (UMDA-EO)[۱], could be very useful in exploring unknown area of search space and for exploiting area of near global optimum. Combination of three methods, UMDA, EO and LA (UMDA-EO-LA) is better than combination of two methods in exploring and exploiting of search space and find better solution.

This paper has been organized in five major sections: section II briefly introduces UMDA algorithm; in section III EO algorithm will be discussed; in section IV briefly introduces LA algorithm, in section V suggested algorithms will be introduced; section VI contains experimental results; finally, section VII concludes the paper.

۲. UNIVARIATE MARGINAL DISTRIBUTION ALGORITHM

Muhlenbein introduced the UMDA [۲،۳،۱۳] as the simplest version from estimation of distribution algorithms (EDAs). Thereafter, there have been several modifications of UMDAs and have been applied to many optimization problems. In the binary search space, UMDAs evolve one probability vector p(t)=(p(۱,t),…,p(l,t)) where all the variables are assumed to be independent of each other.

SUMDA starts from the *central probability vector* that has value of ۰,۵ for each locus and falls in the central point in the search space. Sampling this probability vector creates random solutions because the probability of creating a ۱ or ۰ on each locus is equal. Without loss of generality, a binary-encoded solution x=($x_1,..., x_l$)= $\{0,1\}^l$ is sampled from a probability vector p(t) as follows: for each locus i, if randomly created number r=rand(۰،۱)<p(i,t), its allele x_i is set to ۱;otherwise,   is set to ۰. At iteration t, a population S(t) with n individuals are sampled from the probability vector p(t). The samples are evaluated and an interim population D(t) is formed by selecting μ (μ<n) best individuals, denoted $x_1(t),\ldots,x_\mu(t)$, from S(t). Then the probability vector is updated from extracting statistics information from D(t) as follows:

(۱)

$$p'(t) = \frac{1}{\mu}\sum_{k=1}^{k=\mu} x_k(t)$$

After the probability vector is updated according to D(t), in order to keep the diversity of generated samples in dynamic environments, a bitwise mutation is applied in SUMDA. The mutation operation always changes locus i={۱,…,l}, if a random number r=rand(۰،۱)< p_m (p_m is the mutation probability), then mutate p(i,t) using the following formula:

(۲)

$$p'(i,t) = \begin{cases} p(i,t)*(1.0-\delta_m),\ p(i,t) > 0.5 \\ p(i,t),\ p(i,t) = 0.5 \\ p(i,t)*(1.0-\delta_m)+\delta_m,\ p(i,t) < 0.5 \end{cases}$$

Where δ_m is mutation shift that controls the amount of mutation operation alters the value in each bit position. After the mutation operation, a new set of samples is generated from the new probability vector and this cycle is repeated.

As the search progresses, the elements in the probability vector move away from their initial settings of ۰،۵ towards either ۰،۰ or ۱،۰, representing samples of height fitness. The search stops when some termination condition holds, e.g., the maximum allowable number of iterations $t_{\max}$ is reached.

۳. EXTREMAL OPTIMIZATION ALGORITHM

Extremal optimization [٤،٨،٩] was recently proposed by Boettcher and Percus. It is inspired by self-organized critical models from co-evolution abstracted from the fundamental of ecosystem. The search process of EO eliminates components having extremely undesirable (worst) performance in sub-optimal solution, and replaces them with randomly selected new components iteratively. Finally, the high-quality solutions from hard optimization problems may be explored through such kind of local searches. Firstly, we interpret the novel algorithm by minimization problems. The basic algorithm operates on a single solution S, which usually consists of a number of variables $x_i (1 \le i \le n)$. At each update step, the variable x_i with worst fitness λ_i (individual cost contribution) is identified to alter. There are no parameters to be adjusted for the selection of better solutions. To improve the results and avoid the possible dead ends, Boettcher and Percus subsequently proposed τ -EO that is regarded as a general modification of EO by introducing a parameter. All variables x_i i are ranked according to the relevant fitness λ_i, namely find a permutation Π: $\lambda_1,..., \lambda_n$. Then each independent variable x_i to be moved is selected according to the probability distribution Eq.۳; $1 \le k \le n$ which is associated from the distinct ranks $k_1,...., k_n$. The choice of power law distribution ensures that no ranks get excluded for further evolution while maintaining a bias against variables with bad fitness.

(۳)

$$p = k^{-\tau}$$

EO and its derivatives have been extensively applied to solve numerous NP-hard combinatorial optimization problems. Simulation performance has been proved that EO outperforms other state-of-the-art algorithms from many applications, such as graph bi-partitioning, satisfiability (MAX-K-SAT), TSP problems and some industrial applications.

٤. LEARNING AUTOMATA

The automata approach to learning can be considered from the determining of an optimal action from a set of actions. An automaton can be regarded as an abstract object that has finite number from actions. It selects an action from its finite set of actions. This action is applied to an environment. The environment evaluates the applied action and sends a reinforcement signal to automata (i.e. favorable or unfavorable in a P-model environment).The response from environment is used from learning algorithm of automata to update its internal state. By continuing this process, the automaton learns to select an action, that leads to a favorable response. Variable structure learning automata are represented by the sextuple < β , λ ,α,P,G,T>, where β is a set of input actions, λ is a set of internal states, α is a set of outputs, P denotes state probability vector governing the choice of the state at each stage k; G is the output mapping, and *T* is learning algorithm. The learning algorithm is the recurrence relation and is used to modify the state probability vector. It is evident that the crucial factor affecting the performance from the variable structure learning automata is the learning algorithm for updating the action probabilities. The linear reward-penalty algorithm (L_{R-P}) is one of the various learning. Let α_i be an action chosen at time k as a sample realization from distribution $p(k)$. In a P-model environment where $\beta \in \{٠،١\}$, an L_{R-P} scheme for updating *P* is defined as Eq. (٤) when β=٠ (favorable input) and Eq. (٥) when β=١ (unfavorable input).

$$P_j(k+1) = \begin{cases} P_j(k) + a\,(1 - P_j(k)) & if\ i = j \\ P_j(k)(1-a) & if\ i \neq j \end{cases} \tag{٤}$$

$$P_j(k+1) = \begin{cases} P_j(k)(1-b) & if\ i = j \\ \dfrac{b}{r-1} + (1-b)P_j(k) & if\ i \neq j \end{cases} \tag{٥}$$

Where *a* and *b* represent reward and penalty step length respectively [١١،١٢].

٥. SUGGESTED ALGORITHM

Considering the advantages and disadvantages discussed in this paper and, in order to eliminate the disadvantages of a given algorithm, we combined UMDA with EO for better performance. Power EO is less in exploring whole search space in comparison with other algorithms like UMDA, so with combination, we use exploring power of UMDA and exploiting power of EO in order to find the best global solution, accurately. We combine UMDA, EO and LA for better performance because of LA power in adaptive behavior instead of EO lonely.

In the first place, the probability vector is initialized with ٠،٥ then we sample from probability vector and produce new population considering population size. In the second place, it as a temporary population play a role in updating probability vector. Proposed algorithm utilize the best person selected as the elite, after the change it is injected back into the population. Selecting the best individual in part of the search space, we tried to optimize the best solution on the population and applying a local search in landscape, then most qualified person earns and we use it in probability vector learning process.

According to the subjects described, the overall shape of proposed algorithms (UMDA-EO-LA) will be as follows:

١. Initialization
٢. Initialize probability vector with ٠،٥
٣. Sampling of population with probability vector
٤. Matching each individual with the issue conditions (equal number of nodes in both parts)

a. Calculate the difference between internal and external (D) cost for all nodes
b. If A> B transport nodes with more D from part A to part B
c. If B> A transport nodes with more D from part A to part B
d. Repeat steps until achieve an equal number of nodes in both

٥. Evaluation of population individuals
٦. Replace the worst individual with the best individual population (elite) of the previous population
٧. Improve the best individual in the population using *internal EO-LA*, and injecting to the population
٨. Select μ best individuals to form a temporary population
٩. Make a probability vector based on temporary population according Eq.١

١٠. Mutate in probability vector according Eq.٢
١١. Repeat steps from step ٣ until the algorithm stops

Internal EO-LA :

١. Initialize probability vector of Learning Automata

٢. Calculate fitness of solution components
٣. Sort solution components based on fitness as ascent

٤. Select an action chosen by Learning Automata and its corresponding component as replacement component
٥. Select the new value for exchange component according to the problem
٦. Replace new value in exchange component and produce a new solution

٧. Reward or penalize Learning Automata according to previous solutions and new solution (Eq.٤،٥)
٨. Repeat from step٢ until there are improvements.

Advantage of EO-LA algorithm in comparison with τ-EO algorithm is that the algorithm adapts with sample problem well. The algorithm initialize its probability vector with probability distribution vector of τ-EO algorithm. During execution of program this vector is adjusted by problem samples for better performance. Results on benchmark problem represent better performance of proposed algorithms in comparison with basis algorithms and EO-LA and UMDA-EO.

٦. EXPERIMENTS AND RESULTS

To evaluate the efficiency of the proposed algorithm and in order to compare it with other methods; one NP-hard problem, Graph Bi-partitioning Problem (GBP) is used. The graph bi-partitioning problem consists of dividing the set of its nodes into two disjoint subsets containing equal number of nodes in such a way that the number of graph edges connecting nodes belonging to different subsets (i.e., the cut size of the partition) is minimized. Samples of problems that the algorithms used to compare the performance can be found in reference [٨].

A. Graph Bi-partitioning Problem

We use bit string representation to solve this problem. ٠ and ١ in this string represent two separate part of graph. Also in order to implement EO for this problem, we use [٩] and [١٠]. In these references uses initial clustering. In this method to compute fitness of each component, we use ratio of neighboring nodes in each node for matching each individual with the conditions (equal number of nodes in both parts), using KL algorithm [١٣].

In the present study, we set parameters using calculate relative error in different runs. Suitable values for this parameters are as follows: mutation probability(٠،٠٢), mutation shift(٠،٢), pop size(٦٠), temporary pop size(٢٠) and maximum iteration number is ١٠٠.

In order to compare performance of methods, UMDA-EO-LA, UMDA-EO, EO-LA and EO, We set τ =١،٨ that is best value for EO algorithm based on calculating mean relative error in ١٠ runs. Fig.١ shows the results and best value for τ parameter.

This paper compare UMDA-EO-LA, UMDA-EO, EO-LA and τ-EO and see the change effects; the parameter value τ for all experiments is ١،٨.

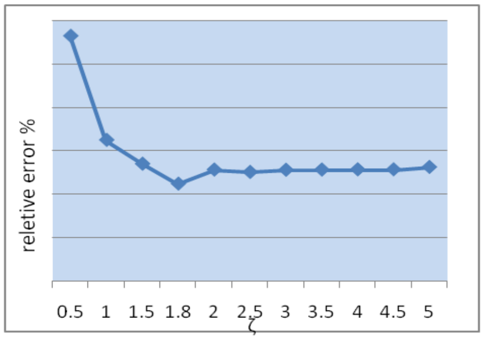

Figure ۱. Select best value for τ parameter

We calculate relative error generated by the algorithm for different parameter value of the reward (a), parameter value (۱۰) that has minimum error is considered as the optimum amount of it.

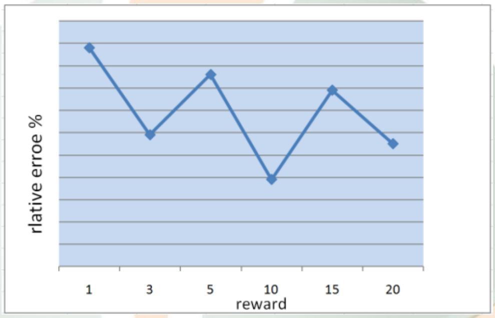

Figure ۲. Select best value for a (reward) parameter in LA

Table ۱ shows results of comparing algorithms for this problem. Observes proposed algorithm in most of instances has minimum and best value in comparing with other algorithms.

Comparative study of algorithms for solving the graph bi-partitioning problem is used from instances stated in the previous section. Statistical analysis of produced solutions by these algorithms are shown in Table ۱. As you can see in this table UMDA-EO-LA algorithm in almost all cases are better than rest of the algorithms. Compared with EO-LA (EO combined with learning automata) can be able to improve act of exploiting near areas of suboptimal solutions but do not explore whole search space well.

Fig.۳ also indicates that the average error in samples of graph bi-partitioning problem in suggested algorithm is less than other algorithms.

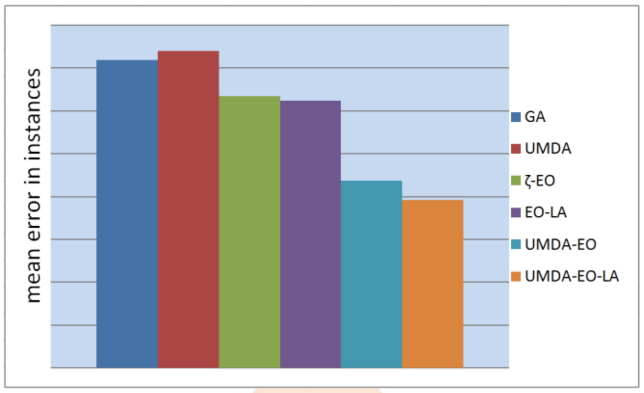

Figure ۳. Comparison of mean error in UMDA-EO with other methods

Good results of this algorithm are because of the benefits of both algorithms and elimination of the defects. UMDA algorithm emphasizes at searching unknown areas in space and does a kind act can be said the exploration , the EO-LA algorithm using previous experiences and the search near the global optimum locations and find optimal solution.

۷. CONCLUSION

Finding of the present study implies that, the suggested algorithm (UMDA-EO-LA) has a good performance in real-world problem, graph bi-partitioning problem. They combine the three methods and benefits and create a balance between two concepts of evolutionary algorithms, exploration and exploitation.

UMDA acts in the discovery of unknown parts of search space and EO search near optimal parts of landscape and EO with combination of LA is trying to find the global optimum solution accurately. So with combination of three methods can find global optimal solution accurately.

REFERENCES

[۱] M.Hashemi and M.R.Meybodi, " Combinatorial Optimization Algorithm based on UMDA and EO", CICIS'۱۱, IASBS, Zanjan, Iran, ۲۰۱۱.

[۲]S.Yang, "Explicit Memory scheme for Evolutionary Algorithms in Dynamic Environments." Studies in Computational Intelligence (SCI) ۵۱, springer , pp.۳-۲۸ , ۲۰۰۷.

[۳]Chen Tianshi, Ke Tang, Chen Guoliang, Xin Yao, "Analysis of Computational Time of Simple Estimation of Distribution Algorithms", IEEE Trans., Evolutionary computation, vol. ۱٤, No. ۱, ۲۰۱۰

[٤]Robin Hons, Estimation of Distribution Algorithms and Minimum Relative Entropy, phd. Thesis, university of Bonn, ۲۰۰۵.

[۵]S. Boettcher and A. G. Percus, "Extremal Optimization: An Evolutionary Local-Search Algorithm", http://arxiv.org/abs/cs.NE/ ۰۲۰۹۰۳۰
[٦] http://en.wikipedia.org/wiki/Self-organized _criticality
[۷] Bak Per, Tang Chao and K. Wiesenfeld, "Self-organized Criticality", Physical Review A, Vol. ۳۸, No. ۱, ۱۹۸۸.
[۸] http://staffweb.cms.gre.ac.uk/~c.walshaw/ partition
[۹]S. Boettcher, "Extremal Optimization of Graph Partitioning at the Percolation Threshold", Physics A, vol ۳۲, No.۲۸, pp.۵۲۰۱-۵۲۱۱, ۱۹۹۹.
[۱۰] S. Boettcher and A. G. Percus, "Extremal Optimization for Graph Partitioning", Physical Review E, vol. ٦٤, pp. ۰۲۱۱۱٤, ۲۰۰۱.

[۱۱]K. S. Narendra and M.A.L. Thathachar, "Learning Automata: An Introduction", Prentice Hall, ۱۹۸۹.

[١٢]M. A. L. Thathachar and P. S. Sastry, "Varieties of Learning Automata: An Overview", IEEE Transaction on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol. ٣٢, No. ٦, PP. ٧١١-٧٢٢, ٢٠٠٢.

[١٣]H. Mühlenbein and Th. Mahnig, “Evolutionary Optimization and the Estimation of Search Distributions with Applications to Graph Bipartitioning”, Journal of Approximate Reasoning, Vol. ٣١, ٢٠٠٢.