Univariate Marginal Distribution Algorithm in Combination with Extremal Optimization (EO, GEO)

Mitra Hashemi[1] and Mohammad Reza Meybodi[2]

[1] Department of Computer Engineering and Information Technology,
Islamic Azad University Qazvin Branch, Qazvin, Iran
Mitra.hash@yahoo.com
[2]Department of Computer Engineering and Information Technology,
Amirkabir University of Technology, Tehran, Iran
mmeybodi@aut.ac.ir

Abstract. The 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 dose 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 represent the performance of the proposed algorithm on two NP-hard problems, multi processor scheduling problem and graph bi-partitioning problem.

Keywords: Univariate Marginal Distribution Algorithm, Extremal Optimization, Generalized Extremal Optimization, Estimation of Distribution Algorithm.

1 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 [1, 2, 3]. Introduced by Muhlenbein and Paaβ, EDAs constitute an example of stochastic heuristics based on populations of individuals each of which encodes a possible solution of the optimization problem. These populations evolve in successive generations as the search progresses–organized in the same way as most evolutionary computation heuristics. This method has many advantages which can be illustrated by avoiding premature convergence and use of a compact and short representation.

In 1996, Muhlenbein and PaaB [1, 2] have proposed the Univariate Marginal Distributions Algorithm (UMDA), which approximates the simple genetic algorithm.

One problem of GA is that it is very difficult to quantify 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 in such different areas as geology, economy and biology. To show that SOC [5,6] could explain features of systems like the natural evolution, Bak and Sneepen developed a simplified model of an ecosystem to each species, a fitness number is assigned randomly, with uniform distribution, in the range [0,1]. The least adapted species, one with the least fitness, is then forced to mutate, and a new random number assigned to it. In order to make the Extremal Optimization (EO) [8,9] method applicable to a broad class of design optimization problems, without concern to how fitness of the design variables would be assigned, a generalization of the EO, called Generalized Extremal Optimization (GEO), was devised. In this new algorithm, the fitness assignment is not done directly to the design variables, but to a "population of species" that encodes the variables.

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/GEO(UMDA-EO, UMDA-GEO) , could be very useful in exploring unknown area of search space and also for exploiting the area of near global optimum.

This paper has been organized in five major sections: section 2 briefly introduces UMDA algorithm; in section 3, EO and GEO algorithms will be discussed; in section 4 suggested algorithms will be introduced; section 5 contains experimental results; finally, section 6 which is the conclusion

2 Univariate Marginal Distribution Algorithm

The Muhlenbein introduced UMDA [1,2,12] as the simplest version of estimation of distribution algorithms (EDAs). SUMDA starts from the *central probability vector* that has value of 0.5 for each locus and falls in the central point of the search space. Sampling this probability vector creates random solutions because the probability of creating a 1 or 0 on each locus is equal. Without loss of generality, a binary-encoded solution $x=(x_1,...,x_l)\in \{0,1\}^l$ is sampled from a probability vector p(t). At iteration t, a population S(t) of 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. Then the probability vector is updated by extracting statistics information from D(t) as follows:

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

The mutation operation always changes locus i={1,…,l}, if a random number r=rand(0,1)< p_m (p_m is the mutation probability), then mutate p(i,t) using the following formula:

$$p'(i,t) = \begin{cases} p(i,t)*(1.0-\delta_m), p(i,t) > 0.5 \\ p(i,t), p(i,t) = 0.5 \\ p(i,t)*(1.0-\delta_m)+\delta_m, p(i,t) < 0.5 \end{cases} \tag{2}$$

Where δ_m is mutation shift. After the mutation operation, a new set of samples is generated by 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 0.5 towards either 0.0 or 1.0, 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.

3 Extremal Optimization Algorithm

Extremal optimization [4,8,9] was recently proposed by Boettcher and Percus. 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. 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 is identified to alter. 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 are ranked according to the relevant fitness. Then each independent variable x_i to be moved is selected according to the probability distribution (3).

$$p = k^{-\tau} \tag{3}$$

Sousa and Ramos have proposed a generalization of the EO that was named the Generalized Extremal Optimization (GEO) [10] method. To each species (bit) is assigned a fitness number that is proportional to the gain (or loss) the objective function value has in mutating (flipping) the bit. All bits are then ranked. A bit is then chosen to mutate according to the probability distribution. This process is repeated until a given stopping criteria is reached .

4 Suggested Algorithm

We combined UMDA with EO for better performance. Power EO is less in comparison with other algorithms like UMDA in exploring whole search space thus with combination we use exploring power of UMDA and exploiting power of EO in order to find the best global solution, accurately.

We select the best individual in part of the search space, and try to optimize the best solution on the population and apply a local search in landscape, 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, UMDA-GEO) will be as follow:

1. Initialization
2. Initialize probability vector with 0.5
3. Sampling of population with probability vector

```
4. 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
 5. Evaluation of population individuals
 6. Replace the worst individual with the best individual population (elite) of the
previous population
 7. Improve the best individual in the population using internal EO (internal GEO),
and injecting to the population
 8. Select μ best individuals to form a temporary population
 9. Making a probability vector based on temporary population according (1)
10. Mutate in probability vector according (2)
11. Repeat steps from step 3 until the algorithm stops
```

Internal EO:

```
1. Calculate fitness of solution components
2. Sort solution components based on fitness as ascent
3. Choose one of components using the (3)
4. Select the new value for exchange component according to the problem
5. Replace new value in exchange component and produce a new solution
6. Repeat from step1 until there are improvements.
```

Internal GEO:

```
1. Produce children of current solution and calculate their fitness
2. Sort solution components based on fitness as ascent
3. Choose one of the children as a current solution according to (3)
4. Repeat the steps until there are improvements.
```

Results on both benchmark problems represent performance of proposed algorithms.

5 Experiments and Results

To evaluate the efficiency of the suggested algorithm and in order to compare it with other methods two NP-hard problem, Multi Processor Scheduling problem and Graph Bi-partitioning problem are used. The objective of scheduling is usually to minimize the completion time of a parallel application consisted of a number of tasks executed in a parallel system. Samples of problems that the algorithms used to compare the performance can be found in reference [11]. 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) are minimized. Samples of problems that the algorithms used to compare the performance can be found in reference [7].

5.1 Graph Bi-partitioning Problem

We use bit string representation to solve this problem. 0 and 1 in this string represent two separate part of graph. Also in order to implement EO for this problem, we use [8] and [9]. These references use 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 issue conditions (equal number of nodes in both parts), using KL algorithm [12].

In the present study, we set parameters using calculate relative error in different runs. Suitable values for this parameters are as follow: mutation probability (0.02), mutation shift (0.2), population size (60), temporary population size (20) and maximum iteration number is 100.

In order to compare performance of methods, UMDA-EO, EO-LA and EO, We set τ =1.8 that is best value for EO algorithm based on calculating mean relative error in 10 runs. Fig.1 shows the results and best value for τ parameter.

The algorithms compare UMDA-EO, EO-LA and τ-EO and see the change effects; the parameter value τ for all experiments is 1.8.

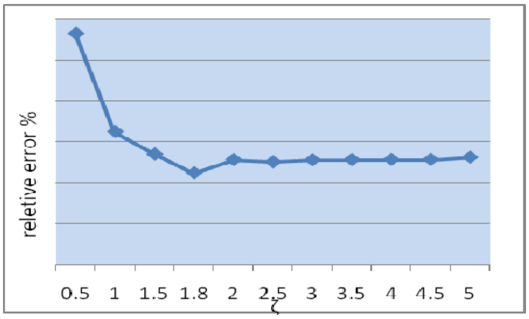


Fig. 1. Select best value for τ parameter

Table 3 shows results of comparing algorithms for this problem. We observe the 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 instances that stated in the previous section. Statistical analysis solutions produced by these algorithms are shown in Table 3. As can be UMDA-EO 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.2 also indicates that average error in samples of graph bi-partitioning problem in suggested algorithm is less than other algorithms.

Good results of the algorithm are because of the benefits of both algorithms and elimination of the defects. UMDA algorithm emphasizes at searching unknown areas in space, and the EO algorithm using previous experiences and the search near the global optimum locations and find optimal solution.

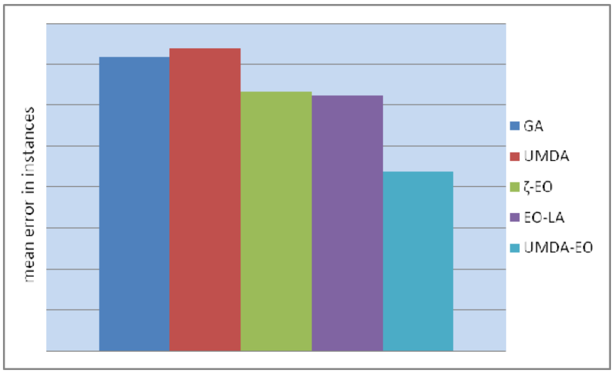


Fig. 2. Comparison mean error in UMDA-EO with other methods

5.2 Multiprocessor Scheduling Problems

We use [10] for implementation of UMDA-GEO in multiprocessor scheduling problem. Samples of problems that the algorithms used to compare the performance have been addressed in reference [11]. In this paper multiprocessor scheduling with priority and without priority is discussed. We assume 50 and 100 task in parallel system with 2,4,8 and 16 processor. Complete description about representation and etc. are discussed by P. Switalski and F. Seredynski [10].

We set parameter using calculate relative error in different runs; suitable values for this parameter are as follow: mutation probability (0.02), mutation shift (0.05), pop size (60), temporary pop size (20) and maximum iteration number is 100.

To compare performance of methods, UMDA-GEO, GEO, We set τ =1.2; this is best value for EO algorithm based on calculating mean relative error in 10 runs.

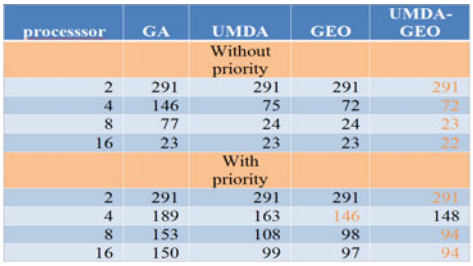
In order to compare the algorithms in solving scheduling problem, each of these algorithms runs 10 numbers and minimum values of results are presented in Tables 1 and 2. In this comparison, value of τ parameter is 1.2. Results are in two style of implementation, with and without priority.

Results in Tables 1 and 2 represent in almost all cases proposed algorithm (UMDA-GEO) had better performance and shortest possible response time. When number of processor is few most of algorithms achieve the best response time, but when numbers of processors are more advantages of proposed algorithm are considerable.

Table 1. Results of scheduling with 50 tasks

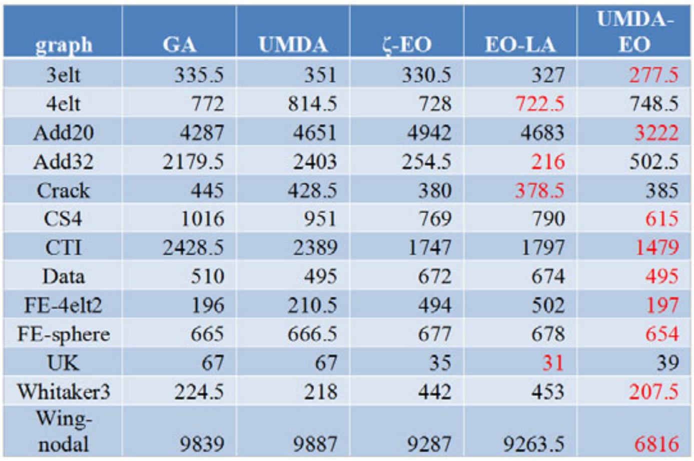
processor	GA	UMDA	GEO	UMDA-GEO
		Without priority		
2	131	131	131	131
4	66	33	33	33
8	35	17	18	16
16	21	12	10	10
		With priority		
2	131	132	131	131
4	78	74	66	68
8	57	55	55	55
16	55	55	55	55

Table 2. Results of scheduling with 50 tasks



processor	GA	UMDA	GEO	UMDA-GEO
		Without priority		
2	291	291	291	291
4	146	75	72	72
8	77	24	24	23
16	23	23	23	22
		With priority		
2	291	291	291	291
4	189	163	146	148
8	153	108	98	94
16	150	99	97	94

Table 3. Experimental results of graph bi-partitioning problem



graph	GA	UMDA	ζ-EO	EO-LA	UMDA-EO
3elt	335.5	351	330.5	327	277.5
4elt	772	814.5	728	722.5	748.5
Add20	4287	4651	4942	4683	3222
Add32	2179.5	2403	254.5	216	502.5
Crack	445	428.5	380	378.5	385
CS4	1016	951	769	790	615
CTI	2428.5	2389	1747	1797	1479
Data	510	495	672	674	495
FE-4elt2	196	210.5	494	502	197
FE-sphere	665	666.5	677	678	654
UK	67	67	35	31	39
Whitaker3	224.5	218	442	453	207.5
Wing-nodal	9839	9887	9287	9263.5	6816

6 Conclusion

Findings of the present study implies that, the suggested algorithm (UMDA-EO and UMDA-GEO) has a good performance in real-world problems, multiprocessor scheduling problem and graph bi-partitioning problem. They combine the two methods and both benefits that were discussed in the paper 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 to find global optimal solution; therefore, with combination of two methods can find global optimal solution accurately.

References

1. Yang, S.: Explicit Memory scheme for Evolutionary Algorithms in Dynamic Environments. SCI, vol. 51, pp. 3–28. Springer, Heidelberg (2007)
2. Tianshi, C., Tang, K., Guoliang, C., Yao, X.: Analysis of Computational Time of Simple Estimation of Distribution Algorithms. IEEE Trans. Evolutionary Computation 14(1) (2010)

3. Hons, R.: Estimation of Distribution Algorithms and Minimum Relative Entropy, phd. Thesis. university of Bonn (2005)
4. Boettcher, S., Percus, A.G.: Extremal Optimization: An Evolutionary Local-Search Algorithm, http://arxiv.org/abs/cs.NE/0209030
5. http://en.wikipedia.org/wiki/Self-organized_criticality
6. Bak, P., Tang, C., Wiesenfeld, K.: Self-organized Criticality. Physical Review A 38(1) (1988)
7. http://staffweb.cms.gre.ac.uk/~c.walshaw/partition
8. Boettcher, S.: Extremal Optimization of Graph Partitioning at the Percolation Threshold. Physics A 32(28), 5201–5211 (1999)
9. Boettcher, S., Percus, A.G.: Extremal Optimization for Graph Partitioning. Physical Review E 64, 21114 (2001)
10. Switalski, P., Seredynski, F.: Solving multiprocessor scheduling problem with GEO metaheuristic. In: IEEE International Symposium on Parallel&Distributed Processing (2009)
11. http://www.kasahara.elec.waseda.ac.jp
12. Mühlenbein, H., Mahnig, T.: Evolutionary Optimization and the Estimation of Search Distributions with Applications to Graph Bipartitioning. Journal of Approximate Reasoning 31 (2002)