Dealing with constraints in Estimation Distribution Algorithms: The univariate case

Roberto Santana, Alberto Ochoa
Center for Artificial Intelligence.
Institute of Cybernetics, Mathematics and Physics.
rsantana@cidet.icmf.inf.cu, ochoa@cidet.icmf.inf.cu

Abstract

This paper proposes a new optimization algorithm to deal with binary constraint problems. The algorithm is based in the Univariate Marginal Distribution Algorithm. We apply our approach to the optimization of functions with different characteristics. For the test functions considered we show the superiority of our algorithm to traditional population search methods that have been used to solve these kind of problems. We report some particular features exhibited by the algorithm and discuss extensions that could make of it a still more powerful optimization tool.

Keywords: Constraint optimization, population based search, genetic algorithm

1. Introduction

This paper deals with the optimization of constraint functions defined over binary vectors. Given a n-dimensional vector x, we want to enforce the following constraint:
$1 \leq a < u(x) \leq b < n$, $a, b, n \in N$ and $u(x)$ is the function of unitation.

Such a class of problem can be frequently found in discrete optimization. However, the main goal of this paper is to investigate it in the context of a simple and efficient evolutionary probabilistic algorithm, namely the Univariate Marginal Distribution Algorithm (UMDA) [Mühlenbein, 1998].

There are two basic approaches to the optimization of constraint problems. The first considers the search over a wider space than that formed by the feasible solutions. A penalty function is generally defined which punishes the non feasible solution points. This approach faces then the problem of how to combine the penalty function with the own objective function.

The other approach searches only in the space defined by the feasible solutions. To do

that the applied optimization operator assures that no illegal solutions will be generated during the search. Depending on the type of optimization method applied these operators can become complex and costly in terms of time.

Population based optimization methods like genetic algorithms have been used for constraint problems. Both penalty functions and new genetic operators have been investigated. However, in a number of papers [Mühlenbein and Paabs,1996] [Mühlenbein, 1997] [Mühlenbein, 1998] have been studied the performance of the simple genetic algorithm. Mühlenbein has shown that the UMDA behavior is at least as well as the simple genetic algorithm for the same kind of problems, but it is simpler in their expression and less costly in computational terms. UMDA exploits the additive genetic variance mainly. Its suitability for solving optimization problems with strongly interacting genes at different loci seems to be limited.

2. A modification of the Univariate Marginal Distributed Algorithm

The UMDA [Mühlenbein, 1997] can be formulated as follows:

UMDA

1. Set T $\leftarrow$1 Generate N>>0 points randomly
2. Select k $\leq$ N points according to a selection method. Compute the marginal frequencies $p^s_i(x_i,t)$ of the selected set.
3. Generate N new points according to the distribution $p(x,t+1) = \prod_{i=1}^{n} p^{s}{}_{i}(x_i,t)$. Set t $\leftarrow$t +1
4. If the termination is not met, go to step 2.

In our constraint problems we want to generate new individuals with a fixed (or within an interval) amount of variables set to 1. The simplest way to do that is sampling without replacement the most frequently set to 1 variables in the selected set. Let us assume that we need to generate exactly r variables set to 1, where $1 \leq a < r \leq b < n$. Then the following modification to the UMDA will do the job:

Step 3.

3.1) Given $S=\sum_{i=1}^{n} p^s(x_i = 1,t)$, set $q_i = p^s(x_i = 1,t)/S$

3.2) For each individual set r of the n variables sampled without replacement with probabilities qi. The remainder variables are set to 0.

We have to choose carefully which random generator to use. For example the choice of a roulette wheel could be inadequated due its high variance. In our experiments we have implemented a generator following [Baker 87]. This scheme called Stochastic Universal Selection has much a smaller variance and is unbiased. The SUS algorithm can be seen as an optimal sampling algorithm. It has zero bias, i.e. no deviation exists between the expected qi value and the algorithm sampling frequency. Furthermore, SUS has a minimal spread, i.e. the range of the possible values for number of times a particular variable is set, setone(xi) is : setone(xi) $\in \{ \lfloor qi \rfloor, \lceil qi \rceil \}$

The resulting algorithm, which we have called Constraint UMDA, (CUMDA) to set the difference, is a minor modification of the UMDA. Nevertheless as long as the class of problems they are used to solve are different, so they are. Also the kind of behavior showed by the CUMDA has some differences with the UMDA. We will consider in extension these aspects in the following section.

3. The test functions

In our experiments we have investigated the differences between the performances of CUMDA and of the other two traditional approaches utilized in genetic algorithms to deal with constraint problems. We also pointed out some particular features of the behavior exhibited by CUMDA when applied to the optimization of different functions. For every experiment, and without loss of generality, the number of variables set to 1 has been fixed to a fixed value r. On the other hand the selection method used by CUMDA has been the Truncation selection of parameter T.

31) The function of positions Posfunction

The first function we investigated was a simple 1-linear function, the Posfunction.:

$$\text{Posfunction}(x) = \sum_{i=1}^{n} Pos(x_i)$$

, where $Pos(x_i)=i$ if $x_i=1$, 0 otherwise.

This function reaches its optimum when the last r variables of the vectors are set to 1. $Pos(x_i)$ could be any other function mapping a different value to each variable. In this case Posfunction would be maximized for the r variables with maximum mapped values.

We compare the performance of a genetic algorithm which uses a heuristic crossover operator [Ponce de Leon, Santana and Ochoa, 1997], roulette wheel selection [Golberg, 1989] and constraint mutation[Ponce de Leon and Santana, 1997] with CUMDA. The heuristic crossover and constraint mutation guarantee that non unfeasible solutions

would be created during the evolution.

Figure 1 shows the average results of 40 runs using the genetic algorithm and CUMDA for the Posfunction being r equal to 5,10,15,20 and 25. Figure 2 shows the distribution of results for the Posfunction when the value of r is 5. The results show that CUMDA outperforms the genetic algorithm in both, times that the optimum is reached and average of best values.

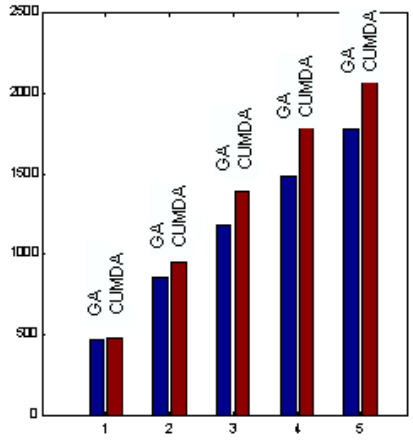

Figure 1: Average results of 40 runs using a genetic algorithm and CUMDA for the Posfunction when r has values 5,10,15,20 and 25

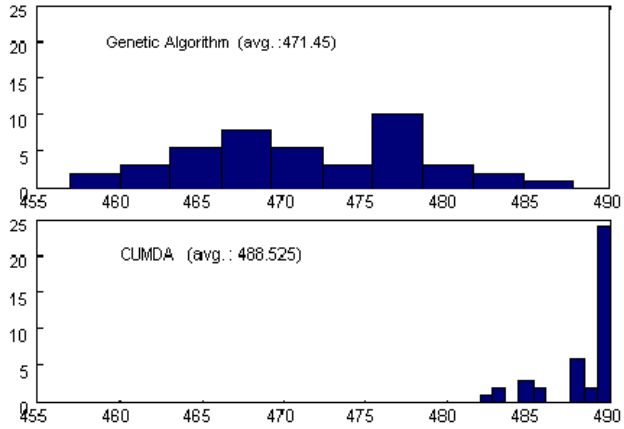

Figure 2: Distribution of results for the Posfunction when the value of r is 5.

Parameters for the used optimization algorithms were:
GA: crossover rate:0.95; mutation rate:0,01;
CUMDA: Truncation coefficient: 0.3
In all the experiments the population size was 100 and the maximum number of

generation was set to 250.

3.2) The Onemaxblock function

The Onemaxblock function associates to each vector X the length of the maximum 1-valued string in X. For the kind of vectors we are working with the maximum value the function can reach is r and it corresponds to a vector having a single block of contiguous bits set to 1. This function reaches its maximum in |X|- r different points, so r also influences the multimodality of the function.

In Figure 3 are shown the results achieved by CUMDA and the same heuristic genetic algorithm used for the Posfunction when both were applied to the optimization of the OneMaxblock function. A third algorithm was tested for the OneMaxblock with penalties (OneMaxblockP) . In this scheme the existence of non feasible solution points ($u(x) \neq r$) is allowed, but these points are penalized by the function.

$$OneMaxblockP(X) = OneMaxblock(X) / (| u(x)-r | + 1)$$

Like OneMaxblock this function reaches its optimum when the length of the maximum 1-valued string in X is r. We optimized it using the Univariate Marginal Distribution Algorithm. In the experiments (Figure 3) we analyzed the performance of the different algorithms considering the number of functions evaluations needed before to reach the best value for each run. For the five functions used CUMDA is clearly superior in four cases to the other two algorithms. For the second function our algorithm is slightly majored by the UMDA.

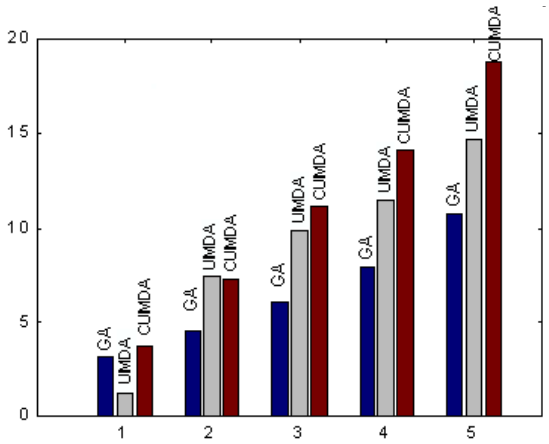

Figure 3: Average results of 40 runs using a heuristic genetic algorithm, the UMDA with a penalty function and the CUMDA for the OneMaxblock function when r taking values 5,10,15,20 and 25.

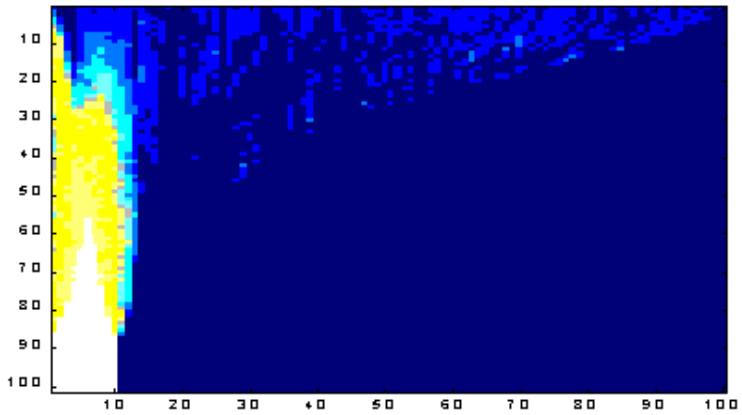

Figure 4: Evolution of the probability vector of the selection set for a typical run of CUMDA for the OneMaxblock function with r equal 10. Clear colors correspond to probabilities values near to 1. When the evolution is close to an end only those variables that are members of the maximum block have clear colors, indicating that in the population they are set to 1with a probability close to 1. Colors corresponding to the probabilities for the rest of the variables are in dark.

Parameters for the used optimization algorithm were:
GA: crossover rate:1; mutation rate:0,02;
CUMDA: Truncation coefficient: 0.25
In all the experiments the population size was 100 and the maximum number of generation was set to 250.

3.3) The dissection function

The other function we have chosen correspond to a real problem defined on graphs. How to find a dissection of a graph ?

A dissection of a graph G is defined to be a set of elementary paths of G such that each vertex of the graph is contained in exactly one path. We will represent one dissection as a graph D= <V',E'> where V'=V and E' $\subseteq$ E. We will call a path closed if it also defines a circuit, otherwise we will call it open. The value of a dissection D is defined to be the number of open paths present in it.

Theorem 1: Given a graph G, the value k of a dissection D of G defines the number of edges the dissection has, |E'|=|V|- k.

The demonstration of Theorem 1 could be found in [Santana and Ponce de Leon, 1998]. In that paper we have proposed a function optimization approach for the problem of finding a dissection of a graph with value k.

The codification of solutions is as follows: After the edges belonging to E have been numbered, a binary vector of size |E| is built. We associate the i component of the vector X with the i edge in E. For the component i, value 1 stands for "select the corresponding edge", whereas bit value 0 stands for "do not select the corresponding edge". Then, each vector will represent a subgraph G' of G such that all the edges of G' are those variables of the vector with value 1. Knowing in advance the value of the dissection we are looking for, it is possible to constrain the number of 1's in the vectors to be exactly P(k)= |V|- k. Thus this problem belongs to the class of binary constraint problems we are dealing with.

For evaluating the dissection function of one binary string X, we first count the degree of each vertex of the subgraph of G represented in it. This information will be in the vector W= (w_1,...,$w_{|V|}$). Let the objective function be F(X) = $\sum_{i=1}^{|V|}(W_i - 2)^2$ (i.e. the sum of the squares of differences coordinate to coordinate between the W_i and 2) . F will reach its minimum for all those vectors X which represent dissections of Graph G with value k.

The values this function can reach depends on the graph topology and the parameter k. We have successfully used CUMDA for the search of dissections with different values k. The behavior exhibited by the algorithm also depends on the graph topology and the parameter k. A more detailed analysis of this topic could be found in [Santana and Ponce de Leon, 1998]. Figure 5 shows a simple graph G (|V|=12, |E|=30) in which we have studied the behavior of CUMDA in the search of a dissection of value |V|/2 , corresponding to a perfect matching of G. Figures 6 a), 6 b), 6 c) and 6 d) show the best graph obtained at generations 2, 5, 8, and the optimum reached in generation 10.

For CUMDA we have also considered the advantage of using the theoretical background that supports UMDA. Figure 7 shows the reponse to selection R(t), R(t) = fmean(t+1) - fmean(t), when CUMDA was applied to the optimization of the dissection function. In the previous equation fmean is the mean of the fitness at generation t. The reponse to the selection is a useful estimator of the behavior of the algorithm.

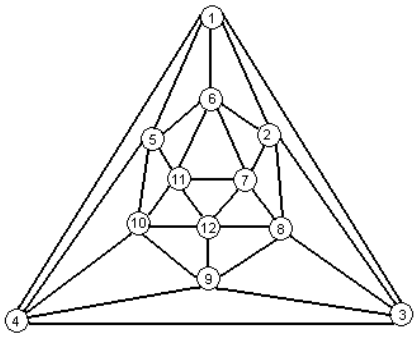

Figure 5: Simple graph G

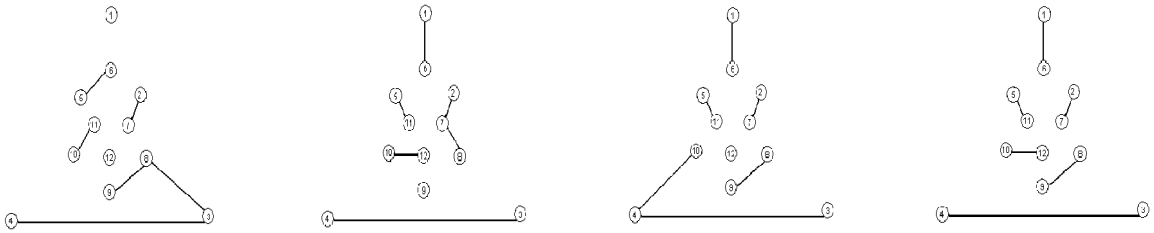

Figure 6: Best graphs found in G at generations 2, 5, 8, and 10 by the Constraint Univariate Marginal Distribution Algorithm.

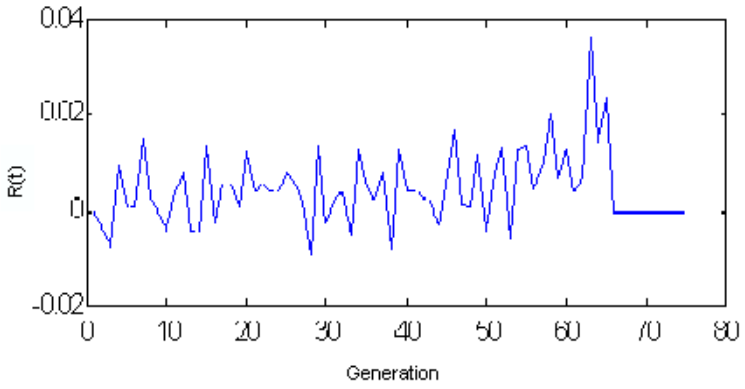

Figure 7: Response to the selection R(t) exhibited by CUMDA during the optimization of the dissection function for a graph with 80 edges and 32 nodes.

Although the results achieved by CUMDA in the search of dissections were encouraging for small graphs, these results degrade in quality as the number of edges and vertices increase. For graphs were the degree of the vertices vary and the edge set relates every vertex to almost all the rest of vertices the algorithm converge to suboptimal solutions.

This fact can be explained by the existence in these problems of a set of dependencies among the variables that the optimization algorithm should capture in order to make an efficient search. This kind of constraint problems are very hard for CUMDA as they are for the Simple Genetic Algorithm. We consider that the Factorized Distribution Algorithm proposed in [Mühlenbein et al., 1998] could lead to better results for the optimization of the dissection function.

Conclusions

In this paper we have introduced a population based search algorithm for the optimization of constraint problems. This algorithm, which is based in the UMDA make explicitly use of the univariate marginal probabilities. For all the functions we analyzed, we have validated with our experiments that CUMDA outperforms the two traditional approaches used for the optimization of constraint functions. Finally we consider that the application of population based search methods to the solution of constraint problems is far from being exhausted. Further work on this trend is ongoing and we think that the use of optimization methods that consider higher interaction between variables may lead to improve the results presented here.

References

Baker, J. E. (1987) : Reducing bias and inefficiency in the selection algorithm. In Proceedings of the

Second International Conference on Genetic Algorithms. Cambridge, MA, Lawrence Erlbaum Associates., pp. 14-21

Baluja, S. and Caruana, R. (1995): Removing the genetics from the standard genetic algorithms., International Conference on Machine Learning -12.

Goldberg, D. E.(1989): Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley.

Mühlenbein, H. (1997): Genetic algorithms, In E. Aarts & J. Lenstra, eds, Local search in combinatorial optimization, John Wiley and Sons, pp. 137 - 171.

Mühlenbein, H. (1998): The equation for response to selection and its use for prediction. Evolutionary Computation, 5, pp 303-346.

Mühlenbein, H. Mahnig T. and Ochoa A. (1998): Schemata, Distributions and Graphical Models in Evolutionary Optimization. To be published in the Journal of Heuristics, also at http://set.gmd.de/AS/ga/publi-neu.html#index

Muehlenbein, H. and Paabs G. (1996): From recombination of genes to the estimation of distributions I. Binary parameters, in Voigt et al., pp. 178-187.

Ponce de León, E.; Santana, R. and Ochoa, A. (1997): A genetic algorithm for a Hamiltonian path problem: Mutation - crossover interaction. Proceedings of the 13th ISPE/IEE International Conference on CAD/CAM Robotics & Factories of the Future'97. Universidad Tecnologica de Pereira, diciembre 15- 17, 1997, Colombia. pp. 1001:1006

Ponce de León, E. and Santana, R. (1997): A genetic algorithm and a local search procedure for a Hamiltonian path problem. Memorias del Encuentro Latino Iberoamericano de Optimizacion, I ELIO. Congreso Chileno de Investigación Operativa OPTIMA 97. Universidad de Concepción, noviembre 3-6, Chile.

Ochoa, A. : (1997) "How to deal with costly fitness functions in evolutionary computation ". Proceedings of the 13th ISPE/IEE International Conference on CAD/CAM Robotics & Factories of the Future'97. Universidad Tecnologica de Pereira, diciembre 15- 17, 1997, Colombia. pp 788:793

Santana, R., Ponce de León, E. (1998): An evolutionary optimization approach for detecting structures on graphs. Research Report ICIMAF (1998).