9th MPS Symposium on Evolutionary Computation, IPSJ, Japan, 2002

Linear and Combinatorial Optimizations by Estimation of Distribution Algorithms

Topon Kumar Paul and Hitoshi Iba
Department of Frontier Informatics
The University of Tokyo
Hongo 7-3-1,Bunkyo-Ku,Tokyo 113-8656
{iba,topon}@miv.t.u-tokyo.ac.jp

Abstract

Estimation of Distribution Algorithms (EDAs) is a new area of Evolutionary Computation. In EDAs there is neither crossover nor mutation operators. New population is generated by sampling the probability distribution, which is estimated from a database containing selected individuals of the previous generation. Different approaches have been proposed for the estimation of probability distribution. In this paper we provide a review of different EDA approaches and show how to apply UMDA with Laplace correction to Subset Sum, OneMax function and n-Queen problems of linear and combinatorial optimizations. The experimental results of the three problems comparing the performance of UMDA with that of Genetic Algorithm(GA) are provided. In our experiment UMDA outperforms GA for linear problems.

Key Words: EDA (Estimation of Distribution Algorithm), Laplace correction, Subset Sum problem, OneMax function, n-Queen problem

1 Introduction

Genetic Algorithms (GAs) are optimization techniques based on selection and recombination of promising solutions. The collection of candidate solutions is called populations of Genetic Algorithms whereas candidate solutions are sometimes named as Individuals, Chromosomes etc. Each individual is an encoded representation of variables of the problems at hand. Each component (variable) in an individual is termed as Gene. Sometimes the components (genes) are independent of one another and sometimes they correlated. But always a communication and information exchange among individuals in a population is maintained through the selection and recombination operator of Genetic Algorithms. This kind of exchange helps to combine partial solutions (individuals) to generate high quality partial solutions—Building Blocks (BBs)[11][12]. The behavior of the GAs depends on the choice of the genetic operators: selection, crossover, mutation, probabilities of crossover and mutation, population size, rate of generational reproduction, number of generations etc. But seldom the problem specific interactions among the variables are considered. As a result, the fixed two parents recombination and evolution sometimes provide inferior quality of solution converging to a local optimum. To avoid the disruption of partial solutions, the two parents recombination processes can been replaced by generating new solutions according to the probability distribution of all promising solutions of the previous generation. This new approach is called Estimation of Distribution Algorithm (EDA). EDAs were introduced in the field of Evolutionary Computation for the first time by [19].

2 Estimation of Distribution Algorithm(EDA)

In EDAs the problem specific interactions among the variables of individuals are taken into consideration. In Evolutionary Computations the interactions are kept implicitly in mind whereas in EDAs the interrelations are expressed explicitly through the joint probability distribution associated with the individuals of variables selected at each generation. The probability distribution is calculated from a database of selected individuals of previous generation.Then sampling this probability distribution generates offspring. Neither crossover nor mutation has been applied in EDAs. But the estimation of the joint probability distribution associated with the database containing

the selected individuals is not an easy task.The flow chart of EDA is show in the figure 1.

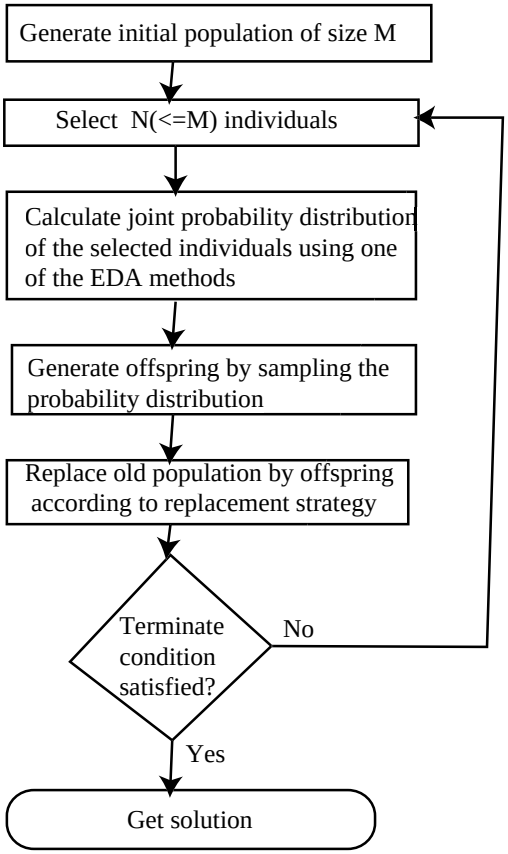

Figure 1: EDA flowchart

3 Different EDA approaches

3.1 Independent Variables

The easiest way to calculate the estimation of probability distribution is to consider all the variables in a problem as univariate. Then the joint probability distribution becomes the product of the marginal probabilities of n variables,i.e., $p_l(x) = \prod_{i=1}^{n} p(x_i)$. Univariate Marginal Distribution Algorithm (UMDA) [16], Population Based Incremental Learning (PBIL) [1][2] and Compact Genetic Algorithm (CGA) [10] consider no interaction among variables.

In UMDAs the joint probability distribution is factorized as a product of independent univariate marginal distribution, which is estimated from marginal frequencies:

$$p_l(x_i) = \frac{\sum_{j=1}^{N} \delta_j(X_i = x_i | D_{i-1}^{Se})}{N} \qquad (1)$$

where $\delta_j(X_i = x_i | D_{l-1}^{Se}) = 1$ if in j^{th} individual X_i has its i^{th} value; otherwise it is zero.There is theoretical evidence that UMDA approximates the behavior of the Simple Genetic Algorithm (SGA) with uniform crossover [17].

In PBILs the population of individuals is represented by a vector of probabilities: $p_l(x) = (p_l(x_1), \ldots, p_l(x_i), \ldots, p_l(x_n))$ where $p_l(x_i)$ refers to the probability of obtaining a 1 in the i^{th} component of D_l, the population of individuals in the l^{th} generation. At each generation M individuals are generated by sampling $p_l(x)$ and the best N individuals are selected. The selected individuals are used to update the probability vector by a Hebbian inspired rule:

$$p_{l+1}(x_i) = (1 - \alpha) p_l(x_i) + \alpha \frac{1}{N} \sum_{k=1}^{N} x_{i,k:M}^{l} \qquad (2)$$

where $\alpha \in (0, 1]$ and $x_{i,k:M}^{l}$ represents the value of x_i at k^{th} selected individual.The update rule shifts the vector towards the best of generated individuals.

In CGAs the vector of probabilities is initialized with probability of each variable 0.5. Then two individuals are generated randomly by using this vector of probabilities and rank them by evaluating. Then the probability vector $p_l(x)$ is updated towards the best one.This process of adaptation continues until the vector of probabilities converges.

All the above mentioned algorithms provide better results for problems of having no significant interaction among variables [16][10][21] but for higher order interaction it cannot provide better results.

3.2 Bivariate Dependencies

To solve the problem of pairwise interaction among variables population based Mutual Information Maximization for Input Clustering (MIMIC) Algorithm [7], Combining Optimizers with Mutual Information Tress (COMIT) [3], Bivariate Marginal Distribution Algorithm (BMDA)[21] were introduced. Where there is at most two-order dependency among variables these provide better result that is far away from the real world where multiple interactions occur.

3.3 Multiple Dependencies

Factorized Distribution Algorithm (FDA)[18], Extended Compact Genetic Algorithm (ECGA)[9], Bayesian Optimization Algorithm (BOA)[20], Estimation of Bayesian Networks Algorithm

(EBNA)[15] can capture the multiple dependencies among variables.

FDA uses a fixed factorization of the distribution of the uniformly scaled additively decomposed function to generate new individuals. It efficiently optimizes a class of binary functions, which are too difficult for traditional GAs. FDA incrementally computes Boltzmann distributions by using Boltzmann selection. FDA converges in polynomial time if the search distribution can be formed so that the number of parameters used is polynomially bounded in n. But the problem of FDA is the requirement of fixed distributions and additively decomposed functions.

BOA uses the techniques from modeling data by Bayesian Networks to estimate the joint probability distributions of selected individuals. Then new population is generated based on this estimation. It uses the BD(Bayesian Dirichlet) metric to measure the goodness of each structure. This Bayesian metric has the property that the scores of two structures that reflect the same conditional dependency or independency are the same. In BOA prior information about the problem can be incorporated to enhance the estimation and better convergence [20]. In order to reduce the cardinality of search space BOA imposes restriction on the number of parents a node may have. For the problems where a node may have more than 2 parents, the situation is complicated to solve.

In ECGA the factorization of the joint probability is calculated as a product of marginal distribution of variable size. These marginal distributions of variable size are related to the variables that are contained in the same group and to the probability distribution associated with them. The grouping is carried out by using a greedy forward algorithm that obtains a partition among the n variables. Each group of variables is assumed to be independent of the rest. So the factorization of the joint probability on n variables is: $p_l(x) = \prod_{c \in C_l} p_l(x_c)$ where C_l denotes the set of groups in the l^{th} generation and $p_l(x_c)$ represents the marginal distribution of the variables X_c-the variables that belong to the c^{th} group in the l^{th} generation. ECGA uses model complexity and population complexity to measure the quality of the marginal distribution. It can cover any number of interactions among variables but the problem is, it does not consider conditional probabilities which is insufficient for highly overlapping cases.

In EBNA the joint probability distribution encoded by a Bayesian Network is learnt from the database containing the selected individuals in each generation. The factorization can be written as

$$p_l(x) = \prod_{i=1}^{n} p(x_i | pa_i^l)$$

where pa_i^l is the set of parents of the variable X_i. Different algorithms can be obtained by varying the structural search method. Two structural search methods are usually considered: score+search and detecting conditional (in) dependencies ($EBNA_{PC}$). Particularly two scores are used in the score+search approarch: the Bayesian Information Criterion (BIC) score ($EBNA_{BIC}$) and the K2+penalization score ($EBNA_{K2+pen}$). In each case the convergence is only affected by the calculus of the parameters θ_{ijk} , where θ_{ijk} represents the conditional probability of variable X_i being in its k^{th} value while the set of its parent at j^{th} value. The parameter of the local probability distribution can be calculated for every generation by using either:
their expected values as obtained by $K2$ score [5] for their score:

$$E[\theta_{ijk} | D_{l-1}^{Se}] = \frac{N_{ijk} + 1}{N_{ij} + r_i}$$

or maximum likelihood estimates

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}}$$

where N_{ijk} denotes the number of cases in the selected individuals in which the variables X_i takes the k^{th} value and its parents pa_i takes j^{th} value and $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$ [where r_i is the number of different values X_i may take].For the case of expected values when the selection is elitist EBNAs converge to a population that contains the global optimum whereas for maximum likelihood case it is not guaranteed [8]

4 Convergence of UMDA by Laplace correction

[8] has shown that some instances with $p_l(x) \geq \delta > 0$ visits populations of $D*$ which contains global optimum infinitely with probability 1 and if the selection is elitist, then UMDA may converge to a population that contains the global optimum. But the joint probability distribution of UMDA can be zero for some x; for example, when the selected individuals at the previous steps are such that $\exists j, \sum_{j=1}^{N} \delta_j(X_i = x_i | D_{l-1}^{Se}) = 0$. Hence the joint probability distribution $p_l(x) = 0$. So UMDA sometimes may not visit a global optimum [8]. To overcome this problem the way of calculating the probabilities should be changed. One

possible solution is to apply Laplace correction [4]. Now

$$p(X_i = x_i|D_{l-1}^{Se}) = \frac{\sum_{j=1}^{N} \delta_j(X_i = x_i|D_{l-1}^{Se}) + 1}{N + r_i} \quad (3)$$

where r_i is the number of different values X_i may take.

5 Some Solutions by UMDA

Here we have applied Univariate Marginal Distribution Algorithm to three well-known problems: Subset Sum problem,OneMax function and n-Queen problem. The Subset Sum and OneMax are linear problems whereas n-Queen is a combinatorial problem.

5.1 Subset Sum Problem

It is the problem of finding what subset of a list of integers has a given sum. The subset sum is an integer relation problem where the relation coefficients are either 0 or 1.If there are n integers, the solution space is 2^n which is the number of subsets for n variables. For small n exact solutions can be found by dynamic programming but for larger n state space tree grows exponentially. It is an NP-Complete problem.

5.1.1 Solving Subset Sum problem by UMDA

In UMDA each individual (solution) is represented by an n-tuple $(x_1, x_2, x_3, \ldots, x_n)$ such that $x_i \in \{0, 1\}$ and $1 \leq i \leq n$. Then $x_i = 0$ if i^{th} integer is not selected and $x_i = 1$ if selected. Each individual is evaluated by finding the difference between the expected sum and the sum of the selected integers in the individual. The smaller difference is the better. Marginal probability distribution is calculated from the best half of the population with Laplace correction and new individuals are generated by sampling this. During replacement we have used elitism. The algorithm terminates when the evaluation of the best individual of a population is zero.

5.1.2 Solving Subset Sum problem by Genetic Algorithm

Using same representation, initialization, evaluation and replacement strategy as those of UMDA we apply GA to the problem. We use simple one point crossover and mutation for generation of offspring. Parents have been selected randomly for crossover.

5.2 OneMax function by UMDA and GA

OneMax function returns the number of ones in an input string, i.e. $f_{OneMax}(x) = \sum_{i=1}^{n} x_i$, where $x = \{x_1, x_2, \ldots, x_n\}$ and $x \in \{0, 1\}$. It is a unimodal function which has optimum in $x_{Opt} = \{1, 1, \ldots, 1\}$. It is a trivial function which is used a test function for the evaluation of performance of Genetic Algorithms and EDAs.With Laplace correction it is guaranteed that UMDA will find optimum value within a small number of generations. In our experiment we have initialized population randomly, selected best half of the population for calculation of probability distribution.And for GA we have used onepoint crossover and mutation.The replacement strategy for both cases is elitism.

5.3 n-Queen problem

n-queen problem is a classic combinatorial optimization problem. The task is to place n-queen on an $n \times n$ chessboard so that no two queens attack each other; that is, no two queens are on the same row, column, or diagonal. Let us number the rows and columns of the chessboard 1 through N and so the queens. Since each queen must be placed on a different row, we can assume that queen i is to be placed on row i. Let the solution be $x = \{x_1, x_2, \ldots, x_n\}$ where x_i represents the column position in row i where queen i can be placed. As all queens must be on different columns, all x_is must be different. So the solution will be a permutation of the numbers 1 through n and the solution space is drastically reduced to $n!$. We can easily fulfill the first two constraints—no two queens on the same row or column by allowing distinct value for each x_i. But how to test whether two queens at positions (i, j) and $(k, l)[i = row, j = column]$ are on the same diagonal or not? If $abs(j-l) = abs(i-k)$, the two queens are on the same diagonal. So the pseudocode for the testing of the constraints for the queens at position x_i and x_j is:

If $((x_i = x_j)$ OR $(abs(i - j) = abs(x_i - x_j))$
    return NOT FEASIBLE;

But this testing requires $\frac{n(n-1)}{2}$ comparisons to calculate the fitness of an individual.

5.3.1 Solving n-Queen by UMDA

With UMDA the individual in a population is represented as $\{x_1, x_2, \ldots, x_n\}$ where each x_i represents the column at row i where i^{th} queen is placed. The fitness of individual is calculated as the number of queens at non-attacking positions.

The initial population of size M is generated randomly with the constraint that all values in an individual are distinct numbers of the set $1, 2, \ldots, n$. By doing this we have implicitly satisfied the constraint that no two queens are on the same row or column. Then, fitness calculation is just to check whether two queens are on the same diagonal or not. In each generation the best half of the individuals of the previous generation are selected for the calculation of joint probability distribution using marginal frequencies of each x_i. During calculation of marginal distribution of each variable Laplace correction has been used. By applying Laplace correction we have ensured that the probability of any variable will be greater than zero and hence increase the joint probability distribution. Then M new individuals are generated.

During generation of each individual we have applied probabilistic modification to enforce the first constraint of n-Queen. In probabilistic modification when some variables have been selected their probabilities for the next turn have been zero and the probability of non selected variables are increased proportionately. Consider the example in the table 1 . If x_1 is selected for the first position then the probability of x_1 for 2nd and 3rd position should be zero and the probabilities of x_2 and x_3 will increase proportionally. The temporary table should look like the one at the right above. By doing this we can ensure that distinct value will be generated for each component of an individual if Roulette Wheel selection is used.

	Before Selection			After Selection		
Position/ variables	1	2	3	1	2	3
x_1	0.7	0.1	0.5	0.7	0	0
x_2	0.1	0.6	0.1	0.1	0.65	0.35
x_3	0.2	0.3	0.4	0.1	0.35	0.65

Table 1: Example of probabilistic modification

For replacement strategy we have use elitism. The algorithm stops when fitness of the best individual is n; that is, all queens are at non-attacking position or certain number of generations have passed.

Fitness improvement heuristics can be applied to the problem and performs much better than blind search using UMDA. For combinatorial optimization 2-opt [6] algorithm or Partially Matched Crossover (PMX) [13] is widely used. I have used 2-opt algorithm as local heuristic.

5.3.2 n-Queen by Genetic Algorithm

Using same representation, initial population and fitness function we have applied Genetic Algorithm to the problem with Partially Matched Crossover,Swap Mutation and Elitism as replacement strategy.

6 Experimental Results

In this section we have presented some experimental results obtained by applying UMDA to the problems described before.We have run the programs on a computer with 902 MHZ AMD Athlon Processor and 512 MB of RAM and in Borland C++ builder 6.0 environment.

6.1 Subset Sum Problem

Here we have generated positive integers randomly. Then expected sum has been generated by randomly selecting those generated integers so that there is always a solution. For trivial case we have chosen expected sum equal to the sum of all generated integers. In this case the solution is $\{1, 1, \ldots, 1\}$. For random case,

(expected sum) $<$ (sum of all generated integers)

The parameters for the problems are:Total Run=50; Population Size=1000;Crossover Rate=0.7, Mutation rate=0.1; Elite=10% and Truncation Selection for UMDA.The experimental results are shown in fig 2-5.

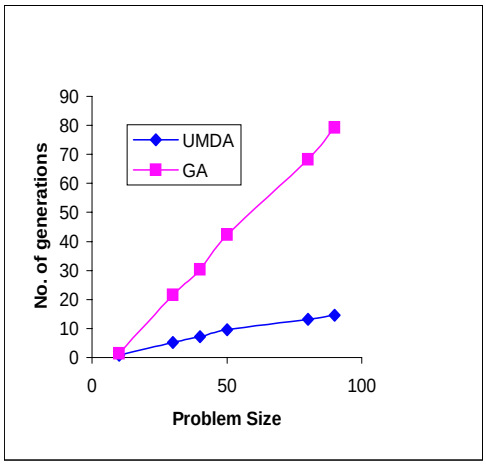

Figure 2: Average no. of generations required for Subset Sum problem (trivial case)

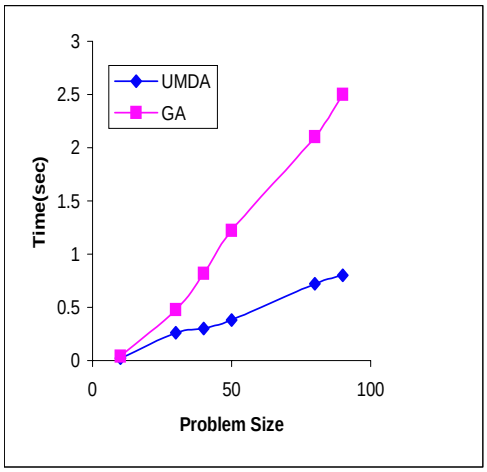

Figure 3: Average time(sec) required for Subset Sum problem (trivial case)

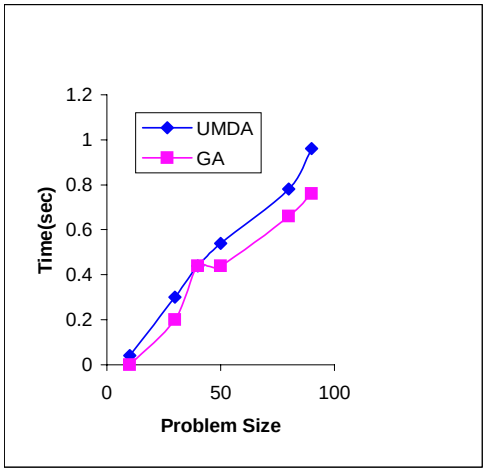

Figure 5: Average time(sec) required for Subset Sum problem (random case)

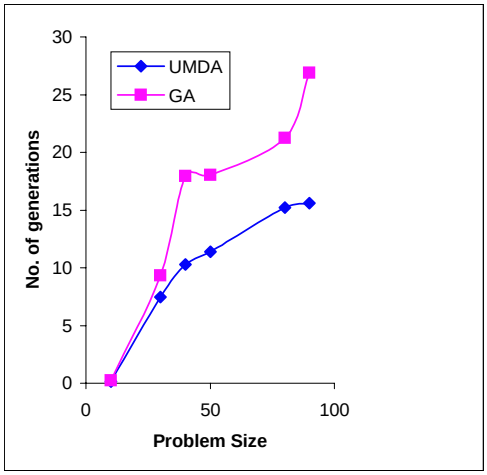

Figure 4: Average no. of generations required for Subset Sum problem (random case)

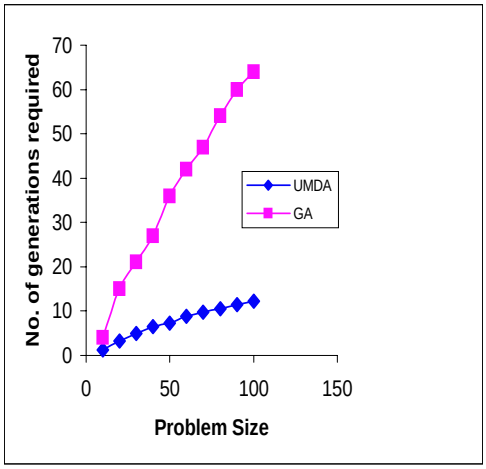

Figure 6: Average no. of generations required for OneMax function

6.2 OneMax Function

OneMax function is a trivial one. It has only one solution of all ones. For this function we have chosen the following parameters: Total run=10; Population size=10* problem size; Crossover rate=0.7; Mutation rate=0.1; Elite=10%; Truncation selection for UMDA. The results are shown in fig 6-7

6.3 n-Queen Problem

For n-Queen problem the solution space is $n!$ and some of them are feasible solution.For small n one possible solution can be found quickly by using dynamic programming. But for larger n this may not be right choice as it may consume a lot of memory for recursive function, neither it would be possible to try all permutations in solution space. So EDA can be a choice for the problem to get solution in a reasonable time limit.

We apply the simplest one UMDA with 2-opt algorithm as local heuristics with the following parameters:Total Run=50; Population Size=10*Variable Size; Crossover Rate=0.7; Mutation Rate=0.1; Elitism=10% and the results are show in fig 8.

7 Discussion

7.1 Subset Sum Problem

From the experimental results we find that for medium size of problem UMDA provide better results in respect of generation when the expected sum is the sum of a proper set of the set of integers. But calculation of probability distribution

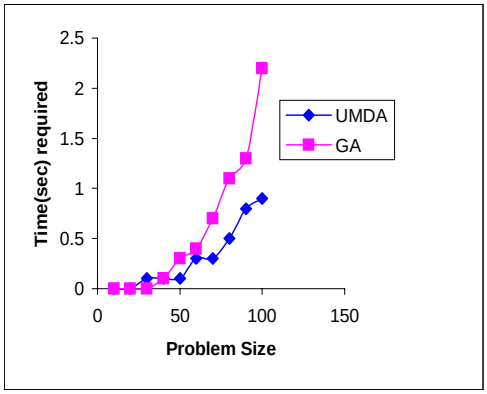

Figure 7: Average time(sec) required for OneMax function

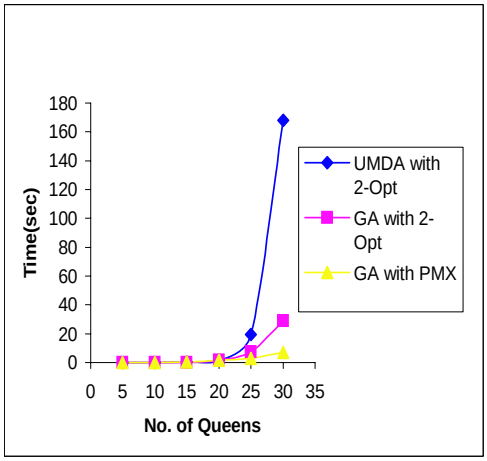

Figure 8: Average time(sec) required for n-Queen problem

takes time. For trivial solution (when the final solution consists of all integer) UMDA outperforms GA in both respects. This is due to the fact that the variables in the problem are less interdependent.

7.2 OneMax Function

The OneMax function is a trivial one and the variables are independent of one another. So UMDA provides better results than those of GA.

7.3 n-Queen problem

From the experimental results we see that UMDA with 2-opt as local heuristics produces solution slower than two other methods, while GA with Partially Matched Crossover produces result very fast for higher variable size. This is due to that fact that the variables, which represent the positions of the queens in a checkerboard, are highly co-related. As we have said, UMDA cannot capture the interdependencies among the variables of a problem. For highly correlated problems we may get the dependencies among the variables by Bayesian networks or other methods, which is our future work.

8 Related Works Using Estimation of Distribution Algorithms

Some optimization problems have been solved using Estimation of Distribution Algorithms in [14]. Of them 0/1 Knapsack problem and Traveling Salesman Problem (TSP) are related to our works. For 0/1 Knapsack problem they have used both binary and permutation based representations in discrete and continuous domains. They have applied UMDA,MIMIC and Estimation of Gaussian Networks Algorithm(EGNA) for the problem. They have shown results by randomly obtaining profits,weights and the capacity of the knapsack.

For TSP they have used 2-opt algorithm as local search optimization and shown the performance of different EDAs for two benchmark problems: Gröstel24 and Gröstel120. Their algorithms found optimal tour length for the Gröstel24 while 1.2 times greater than that of optimal tour for the Gröstel120.

9 Conclusion

In this paper we have discussed Estimation of Distribution Algorithm. It seems reasonable to apply EDA in place of GA. But the estimation of the joint probability distribution associated with the database containing the selected individuals is a bottleneck of this new heuristic. There is no easy method to calculate it. If the distribution is more general we get better result, but calculation of this distribution is time consuming and complicated and sampling of new instances using this distribution is not an easy task.

Due to simplicity we have applied UMDA. It provides better results for linear problem when there is no dependency among variables. n-Queen problem has positional dependencies among variables. Bayesian network may be a possible structure-learning algorithm for estimation of their probability distributions, which is our future work.

References

[1] Baluja, S. (1994). Population based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pennsylvania.

[2] Baluja,S. and Caruana,R. (1995): Removing the genetics from standard genetic algorithm. In A. Prieditis and S. Russell, editors, *Proceedings of the International Conference on Machine Learning*, Morgan Kaufmann, 38-46.

[3] Baluja, S. and Davies, S.(1997). Using optimal dependency trees for combinatorial optimization: Learning the structure of search space. Technical Report No. CMU-CS-97-107, Carnegie Mellon University, Pittsburgh, Pennsylvania.

[4] Cestnik,B.(1990).Estimating probabilities: A crucial task in machine learning .*Proceedings of the European Conference in Artificial Intelligence*, 147-149.

[5] Cooper,G.F. and Herskovits, E.A.(1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9: 309-347.

[6] Croes,G.A.(1992). A method for solving traveling salesman problem. *Operations Research*, 6:791-812.

[7] De Bonet, J.S., Isbell,C.L. and Viola, P.(1997). MIMIC: Finding Optima by estimating probability densities. *Advances in Neural Information Processing Systems*, Vol. 9

[8] González, C., Lozano,J.A. and Larrañaga,P. Mathematical modeling of discrete estimation of distribution algorithms. In P. Larrañaga and J.A. Lozano,editors, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Optimization*. Kluwer Academic Publishers, Boston, 2001.

[9] Harik, G.(1999).Linkage learning via probabilistic modeling in the ECGA. Illigal Report No. 99010,Illinois Genetic Algorithm Laboratory, University of Illinois, Urbana, Illinois.

[10] Harik,G. R., Lobo, F.G. and Goldberg, D.E.(1998).The compact genetic algorithm. *In Proceedings of the IEEE Conference on Evolutionary Computation*, 523-528

[11] Holland, J.H.(1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan.

[12] Goldberg, D.E.(1989). *Genetic Algorithms in search, optimization and machine learning*. Addison-Wesley, Reading, Massachusetts

[13] Goldberg, D.E. and Lingle R. (1987). Alleles, loci, and the traveling salesman problem. In John J. Gerenstette, editor, *Proceedings of the International Conference on Genetic Algorithms and their Applications*, Morgan Kaufmann Pulishers Inc. 1987.

[14] Larrañaga,P. and Lozano,J.A.(2001). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Optimization*. Kluwer Academic Publishers, Boston, 2001.

[15] Larrañaga,P., Etxeberria, R., Lozano,J.A. and Peña, J.M.(2000). Combinatorial Optimization by learning and simulation of Bayesian networks. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, Stanford, 343-352.

[16] Mühlenbein, H. (1998). The equation for response to selection and its use for prediction. *Evolutionary Computation*, 5(3): 303-346.

[17] Mühlenbein,H. and Mahnig,T. Evolutionary Algorithms: From Recombination to Search Distributions . In L. Kallel, B. Naudits and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, Springer publication, 2001.

[18] Mühlenbein,H. and Mahnig,T.(1999). The Factorized Distribution Algorithm for additively decomposed functions. *Proceedings of the 1999 Congress on Evolutionary Computation*, IEEE press,752-759.

[19] Mühlenbein, H. and Paaß,G. (1996).From recombination of genes to the estimation of distributions I. Binary parameters. *In Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature-PPSN IV*,178-187.

[20] Pelikan, M.,Goldberg,D.E. and Cant-paz, E.(2000). LinkageProblem, Distribution Estimation and Bayesian Networks.*Evolutionary Computation* 8(3):311-340.

[21] Pelikan,M. and Mühlenbein, H.(1999).The bivariate marginal distribution algorithm. *Advances in Soft Computing-Engineering Design and Manufacturing*, 521-535