Linkage Information Processing In Distribution Estimation Algorithms

Peter A.N. Bosman
peterb@cs.uu.nl

Dirk Thierens
Dirk.Thierens@cs.uu.nl

Department of Computer Science, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

Abstract

The last few years there has been an increasing amount of interest in the field of distribution estimation optimization algorithms. As more techniques are introduced, the variety in tested distribution structures increases. In this paper we analyze the implications of the form of such a structure. We show that learning the linkage relations alone and using them directly in a distribution estimation algorithm to generate new samples is not sufficient for building competent evolutionary algorithms. The information needs to be processed to identify and use the building blocks.

1 Introduction

Any reasonable optimization problem has some structure. Knowing and using this structure can aid the search for the optimal solution. It is for this reason that the design of evolutionary algorithms (EAs) has been focusing on learning problem structure since the introduction of the genetic algorithm (GA). Most important in this quest for structure usage is the notion of *linkage*. In binary representation, linkage means the structural cohesion of the bits in the coding string with respect to the search space. Holland [8] recognized the use of adapting the linkage between the genes and proposed the inversion operator to accomplish this. Unfortunately the inversion operator has turned out to be too slow and is not included in most GAs. Nevertheless, ignoring the need for proper building block processing reduces the exploitation of problem structure. This leads to exponential growth in population size requirements for solving deceptive problems [12]. Insights into deceptive problems and linkage by Goldberg have led to new GAs, with the mGA [7] as one of the first. Of more recent time, very good results have been achieved with advanced algorithms that use linkage, such as the GEMGA by Kargupta [3].

Recently there has been an increasing amount of interest in the use of probability density estimators (PDEs). Instead of exchanging information between individuals, these methods regard entire populations at once. From the population statistics are derived and used when generating new individuals. Determining these statistics is done according to some probability density structure. This structure itself or its contents are mostly adapted during the search. The use of such a structure should aid in capturing the problem structure. When the right contents for the structure have been found, the algorithm can sample better at the right locations. The implications that follow from the structure used are the topic of interest in this paper.

Estimation of distributions can be separated into two parts, namely *model selection* and *model fitting*. Selecting the right model regards finding the right network with the right topology (distribution structure). Fitting the model regards the fitting of the data at hand into the model and thus finding the values for the parameters in the network (distribution probabilities). This separation is used commonly because of the difficulty of model selection. This difficulty is found in the amount of possible networks, which is very large and grows exponentially with the amount of variables.

It is the implications of model selection we investigate here. In other words, if we don't have a good description of the required network, we want to know the implications of using some other structure for the network instead. We do not set out to describe a new algorithm for optimization or new techniques for learning distributions. Instead, we show that even though the linkage information might be properly incorporated in some structure, we show that the way in which this information is stored and used is very important.

The remainder of this paper is organized as follows.

In section 2 we provide an introduction to optimization algorithms using PDEs by presenting previous work in this field. These algorithms have come to be called estimation of distribution algorithms (EDAs) by Mühlenbein [9]. More specific representation issues and possibilities are stated in section 3. Also in this section we analyze the implications of the different structures for the probability distributions. In section 4 we present our supporting experiments. We discuss the implications of our findings in section 5 and finally present our conclusions in section 6.

2 Previous work

Many extensions have been made to the classical GA for different purposes. One of the most involved extensions regards the linkage concept. This extension leads to some variation of the selection–recombination GA, whereas the EDAs are a completely different view. Even though we wish to focus on the use of PDEs, we do wish to mention two more GA type of algorithms, namely the GEMGA [3] and the BBF–GA [13]. Both of these algorithms use correlations between bit–values and fitness to find linked bits and process building blocks. This is done by flipping individual bits and observing changes in fitness contribution. Based on this information, linkage sets (GEMGA) or crossover masks (BBF–GA) are constructed that are respected during the evolution process. Most important is the acknowledgement that some bits are linked together and should be processed as blocks. This is attempted to be achieved implicitly in the EDAs described next.

A first move toward using PDEs in optimization is found in PBIL by Baluja and Caruana [1]. Here the population is replaced by a single probability vector that describes for each bit the chance that it is set to 1. The bits are regarded independently of each other, so PBIL only regards first order statistics. Higher order statistics, through which linkage can be expressed, were used in the framework known as MIMIC [4] by De Bonet, Isbell and Viola as one of the first. Here, second order statistics are expressed through conditional probabilities. This algorithm first generates a random population. Until the termination condition is met, the algorithm then derives certain empirical probabilities from the population and constructs a new distribution for the binary variables in a solution string. The distribution structure used in MIMIC is no more complex than a chain. Denoting bit i as discrete variable X_i ($X = X_0 X_1 \dots X_{n-1}$), this distributions equals:

$$p(X) = \Big(\prod_{i=0}^{n-2} p(X_{j_i} | X_{j_{i+1}})\Big) p(X_{j_{n-1}})$$

Here, all j_i are different, implying that the density structure is a network in which the nodes are located in a chain of dependencies as can be seen in figure 1. To learn the chain, the j_i have to be determined. In MIMIC an $O(n^2)$ greedy algorithm is employed to minimize the entropy values in the Kullback–Liebler divergence:

$$J(X) = \Big(\sum_{i=0}^{n-2} h(X_{j_i} | X_{j_{i+1}})\Big) + h(X_{j_{n-1}})$$

According to this distribution and the empirical probabilities, new strings are then generated and added to the population, after which selection is applied.

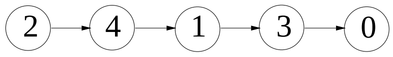

Figure 1: Example of the chain structure used by MIMIC: $p(X_0|X_3)p(X_3|X_1)p(X_1|X_4)p(X_4|X_2)p(X_2)$

The use of the greedy algorithm and the limited structure for the distribution is restrictive in learning the appropriate relations for a problem. Baluja and Davies [2] presented an improvement. In their approach, nodes in the network are allowed to have multiple children, but still a single parent:

$$p(X) = \Big(\prod_{i=0}^{n-2} p(X_{j_i} | X_{e_i})\Big) p(X_{j_{n-1}})$$

Using this structure with $e_i \in \{j_{i+1} \dots j_{n-1}\}$, there is no longer need for a greedy algorithm. Now the divergence can be minimized exactly in $O(n^2)$ time. An example of the structure used can be seen in figure 2.

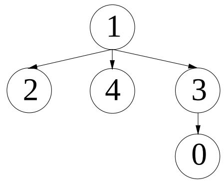

Figure 2: Example of the tree distribution structure: $p(X_0|X_3)p(X_3|X_1)p(X_4|X_1)p(X_2|X_1)p(X_1)$

A similar view on EDA algorithms is found in the UMDA and BMDA [11] by Mühlenbein (*Univariate* and *Bivariate* Marginal Distribution Algorithm respectively). The UMDA uses the most general structure for univariate distributions, which is the variables X_i themselves. Similarly, BMDA uses the most general structure for bivariate distributions, which is a set of disjoint trees. We could thus say that MIMIC and the improvement over it are special versions of BMDA.

Next to these two special EDA algorithms, the FDA [9] by Mühlenbein is of importance. In this algorithm a complete factorization of the distribution is taken as input and used to build the network structure:

$$p(X) = (\prod_i p(\prod_{X \in b_i} X \mid \prod_{X \in c_i} X))p(\prod_{X \in b_0} X)$$

Where $c_i = s_i \cap d_{i-1}$, $b_i = s_i \setminus d_{i-1}$, $d_i = \bigcup_{j=1}^{i} s_j$ and the s_i make up the factorization as input sets with $\bigcup s_i = \{X_0, X_1, \ldots, X_n\}$. Using an actual factorization for the problem at hand, FDA can very efficiently perform optimization. In FDA, the estimation of distributions is clearly separated into model selection and model fitting. Model selection is left to the user as the sets s_i have to be specified that determine the network. Model fitting is then done entirely by FDA. The results on FDA tell us that on the one side given a right distribution structure, the problem can be solved efficiently. However, it also tells us that in that case the UMDA and BMDA algorithms might be too restrictive because of the type of relations allowed in the network. This is obvious since FDA can use joint probabilities directly over multiple variables as depicted in figure 3 instead of something like $p(X_0|X_1)p(X_1|X_2)p(X_2)p(X_4|X_3)p(X_3)$.

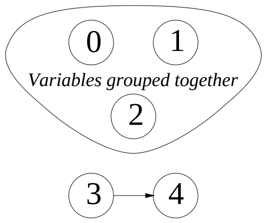

Figure 3: Example of factorized structure used by FDA: $p(X_0, X_1, X_2)p(X_4|X_3)p(X_3)$

The most recent approach, named BOA, was proposed by Pelikan, Goldberg and Cantú–Paz [10]. In BOA, Bayesian networks are used in which each node can have a maximum of k successors, allowing variables now to be dependent on *sets* of variables as illustrated in figure 4. It is now possible to identify complete building blocks through joint probabilities just as in FDA. In contrast to FDA however, BOA *does* have a learning mechanism for the network structure. Learning the network structure, which is model selection, is the only issue left for all presented EDAs except for FDA. Therefore, keeping in mind we wish to investigate the effects of model selection, FDA is more appropriate to use here. This shall become clear in the next section, where we first investigate the PDE structures at hand.

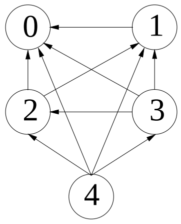

Figure 4: Example of a Bayesian network structure: $p(X_0|X_1, X_2, X_3, X_4)p(X_1|X_2, X_3, X_4)p(X_2|X_3, X_4) \cdot p(X_3|X_4)p(X_4) = p(X_0, X_1, X_2, X_3, X_4)$

3 Representation issues

As we wish to investigate the implications of model selection, we must regard the structure of the probability distribution used in an EDA. In the previous section we already saw a few different possibilities. We can distinguish two types of networks for those structures that differ, namely the Bayesian network of order k and the network in FDA. In the Bayesian network of order k, each node has at most k successors. It is easy to see that the chain used by MIMIC is a special version of a Bayesian network with $k = 1$. The great improvement in using Bayesian networks is that the form of the network is otherwise unrestricted and that the graph does not have to be connected. In structure recognition, this means that it is possible to express the building blocks completely separate from each other. This is impossible when for instance using MIMIC.

The structure in FDA can be different in that probabilities do not have to be expressed conditionally. It is possible to directly group variables together in a joint distribution. This reduces the need for a large k as is obvious from figure 3. How exact and efficient FDA might be however, its practical use is constrained because of the factorization of the problem that is required a priori. On the other hand, having noted how efficient FDA can be, it is desirable to be able to identify building blocks in a joint distribution. In a Bayesian network with large enough k however, it is indeed possible to model this as shown in figure 4. This is possible because $\prod_{i=0}^{n-1} p(X_i|X_{i+1}, X_{i+2}, \ldots, X_{n-1}) = p(X_0, X_1, \ldots, X_{n-1})$. In other words, the structure used in FDA can be written as a Bayesian network. This reduces the amount of arcs required in the network greatly as can be seen in figure 4.

So it seems that using Bayesian networks with a good way of learning the network topology would be a perfect substitute for the exact FDA. The problem lies however in the factor k. Learning the desired network is known as the k–learn problem and has been shown to be NP–complete [5]. In other words, we cannot expect to find the optimal network in polynomial time. Using heuristics is the only practical approach [10]. At this point we can question ourselves if we can also suffice with less complicated and perhaps less competent networks in learning these well instead of approximating higher order networks.

In order to answer this question, we should first realize the trajectory we want the algorithm to follow. First, we want to incorporate the correlations between the bits and the search space in the algorithm. This is of course done by finding the right network as the probability distribution. Through this we wish to express the linkage information for the problem. All EDAs so far implicitly assume that higher order building block problems can then be solved. This is because directly from the probability distribution more samples are generated. However, this does not need to be the case at all. Consider for instance the following trap functions:

$$f(x) = \begin{cases} -\frac{1-d}{l-1}o(x) + 1 - d & \text{if } o(x) < l \\ 1 & \text{if } o(x) = l \end{cases}$$

Here $o(x)$ stands for the amount of ones in substring x of length l. The maximum value of 1 is achieved for a string with l ones and the suboptimum of $1-d$ is found for a string with l zeros. When d goes to 0, the problem becomes fully deceptive for some value of d. This means that all schemata of an order smaller than l lead to the suboptimum (deceptive attractor) [6]. The fitness function consists of concatenating m such blocks of l bits in length. Clearly, the building blocks for this fitness function are located at positions $(0+il, 1+il, \ldots, l-1+il), i \in \{0, 1, \ldots, m-1\}$. The order of the bits within each block is unimportant, so using for instance the chain on the first block, a distribution such as $p(X_0|X_1)p(X_1|X_2)p(X_2|X_3)p(X_3|X_4)p(X_4)$ would seem to suffice. In this case any ordering of the variables in the chain would be evenly proper. Using multiple building blocks implies a different distribution:

$$p(X) = \prod_{i=0}^{m-1}(\prod_{j=0}^{l-2} p(X_{j+il}|X_{j+il+1}))p(X_{(i+1)l-1})$$

However, when using MIMIC, the independent chains have to be linked together because we can only use a single chain, implying a distribution such as $p(X) = (\prod_{i=0}^{n-2} p(X_i|X_{i+1}))p(X_{n-1})$. Here we have introduced dependencies *across* the building block boundaries. A first observation could therefore be that in order to exactly identify building blocks, a completely connected chain is insufficient. On the other hand, the only difference in the distribution is that $p(X_{il-1})$ is replaced by $p(X_{il-1}|X_{il})$ for $i \in \{1, 2 \ldots, m-2\}$.

The larger problem is in the combination of probabilities. Taking the independent chains that fit the building blocks for instance, any settings of the bits in one block is a combination of l chances. For the problem suggested, the optimum is reached when all bits are set to 1. Even if all chances are 0.9 for instance, the probability that bits $0 \ldots i$ ($i \in \{0, 1, \ldots l-1\}$) in a block are set to 1 equals 0.9^{i+1}. For $i = 4$, this already equals 0.59049. The chances have to be very close to 1 in other words to succeed. Selection will drive the better blocks out of the population when good combinations can't be made fast enough because of such composition of chances.

This problem will also be present when using for instance the better fitting tree, even though the problem can be reduced. A better fitting distribution such as $p(X_0|X_4)p(X_1|X_4)p(X_2|X_4)p(X_3|X_4)p(X_4)$ could be used for each block. Here X_4 is thus the root of a tree with height 1. Given again all chances of 0.9 at setting bits to 1, the root is set to 1 with chance 0.9 and all other bits are then set to 1 with chance $0.9^2 = 0.81$, which is significantly better. Still however, the block is not processed as a whole. Note that the entire block is still set to one with probability 0.9^l for l bits. In general it can be expected that for deceptive problems, selection will drive the good blocks out of the population too fast in order for any correct structure and probabilities to still be learned.

The two commonly accepted remedies for the problem at hand are to increase the population size and to reduce the selection pressure (or some other means to maintain diversity longer). However, increasing the population size will not help when the distribution estimation isn't effective in the first place. On the other hand, when it is effective, increasing the population size *will* help. The larger the population, the less coincedental it will be that there are a few good blocks. In the case of inadequate relation learning, the information will be lost in selection and not be regenerated. When relation learning *is* adequate, the blocks will be regenerated in time, preventing that they are expunged from the population. Furthermore, with a non deceptive problem, the good blocks will not be driven out by selection quickly and there will be enough time to find the right relations. All that is then required is the proper mixing of blocks. Reducing selection pressure will not help either. In fact, increasing the pressure will be better because without selection the initial statistics are based upon a random population and the next population is just another random population because no selection was applied. A too high selection ratio on the other hand might drive the good blocks out of the population all too fast.

The trajectory of finding correlations, placing them in some probability density structure, expressing linkage information through this and then solving higher order building block problems is thus not expected to be followed by using just any density structure. In other words, just as was the case for the variants of the GA and is still the case in research, we need to identify building blocks and use them as blocks. Even when the linkage information for a block is expressed by a chain, not acknowledging that the bits need to be processed as a block will keep the algorithm from solving higher order building block problems. In the next section we will present experiments that justify this viewpoint.

4 Experiments

In order to provide supporting results for the claim in the previous section, we test one strategy and use different distribution structures. For a higher order deceptive building block problem, we should then see that optimization can only be achieved when the building blocks can be processed as a whole. Using structures such as the chain or the tree will fail to capture the building block structure and not be able to perform optimization. Furthermore, we can simulate the best case scenario for MIMIC by fixing the distribution at the best possible chain using FDA. The issue is that building blocks cannot be processed, however good the linkage information, so even fixing the distribution at the best possible chain will not help. Instead of moving up from MIMIC toward better algorithms, we can go from the FDA and degrade the algorithm. By slightly perturbing the perfect distribution for the problem, it is stressed again that the most important thing is processing the building blocks if after perturbation the problem is still optimized. At the same time we wish to show the influence of the population size and selection ratio. For this, we present the results with the population size on the horizontal axis and perform tests for different selection ratios. Finally, we also wish to test the problem both as fully deceptive and as non-deceptive. To do this we use the trap functions with $d = 0.2$ and $d = 0.8$ respectively. The experiments in this section were all executed in the general EDA and FDA fashion [9]:

1. Generate a random population of N individuals.
2. Select τN individuals with $0 < \tau < 1$.
3. Update the distribution with selected individuals.
4. Generate a new population of N individuals using the new distribution.
5. Incorporate the best individual from the previous generation (elitism).
6. If termination condition is not met, go to step 2.

Updating the distribution is done by using a strategy as found in MIMIC or Baluja and Davies for instance. In other words, we override the selection framework that is normally used, which is just a variation of the selection framework above. When using the approach by Baluja and Davies, we disable the estimator array by setting the initial value c_{init} and the decay factor α both to 0. Termination occurs when all individuals in the population are equal. All tests were combined over 30 independent runs. Furthermore, the test function used is the trap function as presented in the previous section with $l = 5$, $m = 10$ (string length is thus 50) and we use $d = 0.2$ or $d = 0.8$. All figures show the percentile of building blocks correct upon termination along the vertical axis and the population size, which was increased in steps of 50, along the horizontal axis.

In a first test, we investigate the implications of using the chain and the tree structure. We want to test these structures using learning techniques to find the actual topology of the networks. To be more precise, we test MIMIC and the approach by Baluja and Davies with $\tau = 0.5$. From figure 5 it follows that both approaches indeed cannot succeed in finding the optimum value when the problem is fully deceptive. Furthermore the results show that indeed when distribution estimation is successful, an increasing population size contributes to its success. However, when it is not successful (deceptive problem), an increasing population only stimulates its downfall. For $d = 0.8$, the algorithms show better results, but optimization is still not achieved. Other tests have shown that the 50% truncation ratio is still too strict for these algorithms to solve the non deceptive problem. So in the remainder, we shall employ 20% and 80% truncation ratios. Furthermore the two tested algorithms learned the topology of the network. This is the harder part of the problem, which is *model selection*. To show the implications of selecting the right type of model only, we shall in the remainder fix these distributions. This means that steps that adapt the network structure or topology in the EDAs are simply skipped. Only the empirical probabilities are determined and used to generate new individuals. The network itself is fixed on beforehand. In other words, we shall from now on regard the best case scenario for using a certain network model by choosing the best possible topology of the network on beforehand. Methods that learn this topology can then only provide worse results. We shall use the FDA to investigate such fixed distributions by choosing the right sets s_i.

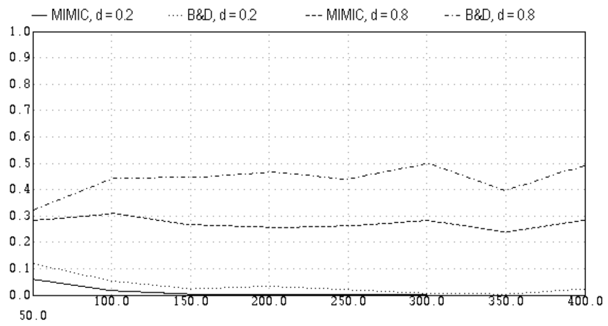

Figure 5: Percentile BBS correct for MIMIC and Baluja & Davies.

In a second test, we use the FDA to test the best possible chain of dependencies by specifying the sets s_i to be $\{i-1, i\}$ for $i > 0$ and $\{0\}$ for $i = 0$. This will then lead to a probability distribution used in FDA equal to $p(X) = p(X_0)(\prod_{i=1}^{n-1} p(X_i|X_{i-1}))$. Note that the linkage information is actually expressed because the bits in the individual blocks are directly linked. The only thing is that bits between blocks are linked in a single position as well. To this end, we can alter the sets s_i to be $\{i-1, i\}$ for $(i \bmod 5) \neq 0$ and $\{i\}$ otherwise, which will give us a probability distribution that models a chain for each building block separately: $p(X) = \prod_{j=0}^{m-1} p(X_{lj})(\prod_{i=1}^{l-1} p(X_{lj+i}|X_{lj+i-1}))$.

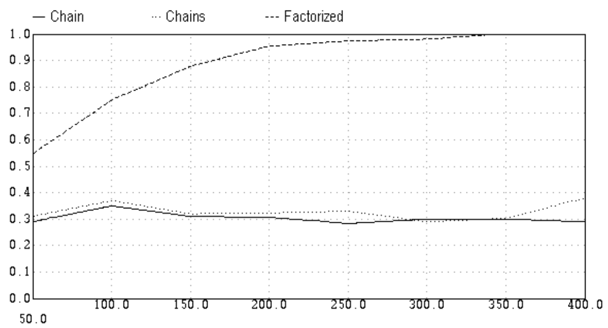

Figure 6: Percentile BBS correct for FDA variants, $d = 0.2, \tau = 0.2$.

All figures showing results for both the complete chain as well as the individual chains (figures 6 through 9) show that there is no significant difference between the two chain approaches. This means that separating the building blocks is not the most important issue. From all figures it also shows that identifying and using as such the individual building blocks *is* important. The tests that did succeed are namely the ones using the sets $s_i = \{5i, 5i+1, \ldots 5i+4\}$ for $i \in \{0, 1, \ldots, m-1\}$. These sets are completely disjoint and the resulting probability distribution

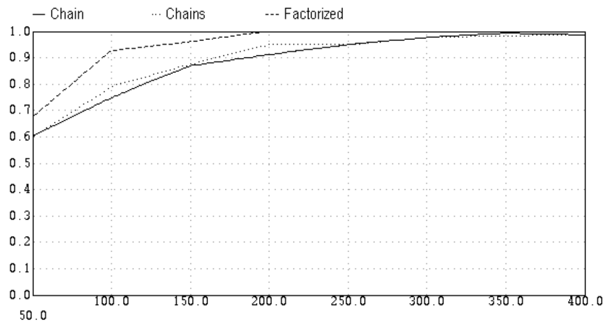

Figure 7: Percentile BBS correct for FDA variants, $d = 0.8, \tau = 0.2$.

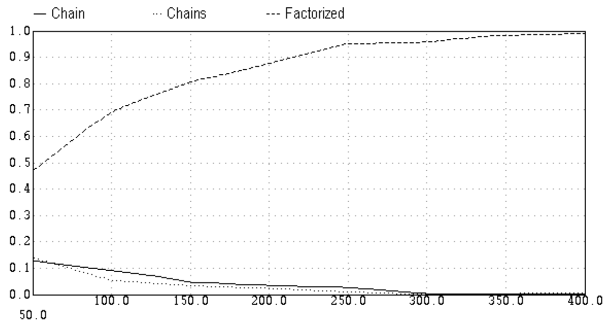

Figure 8: Percentile BBS correct for FDA variants, $d = 0.2, \tau = 0.8$.

is $p(X) = \prod_{i=0}^{m-1} p(X_{5i}, X_{5i+1}, \ldots, X_{5i+4})$. Using this distribution, the building blocks are completely processed as a whole. As a result, as can be seen from the figures, all problems are successfully optimized. Other series of tests have been portrayed with 100 bits and blocks of 5, which have provided the same results.

Third and finally, we have tested FDA with almost the same disjoint sets s_i that have proven so successful. These tests are to emphasize the fact that identifying and processing as a whole the complete building blocks is most important. Instead of directly using these sets however, we have added a perturbing element. The amount of perturbation is determined by a perturbation factor ζ. To show that the chain itself is not the problem, but the fact that the building blocks are not processed as a direct whole, we convert the perfect distribution to a chained distribution. However, we do keep the building blocks themselves intact. This is done by adding to set s_i the first ζ elements of set s_{i-1} for each $i > 0$. This leads to the distribution

$$\begin{aligned} p(X) = \; & p(X_0, X_1, \ldots, X_4) \cdot \\ & \prod_{i=1}^{m-1} p(X_{5i}, X_{5i+1}, \ldots, X_{5i+4} | \\ & \quad X_{5(i-1)}, X_{5(i-1)+1}, \ldots, X_{5(i-1)+\zeta-1}) \end{aligned}$$

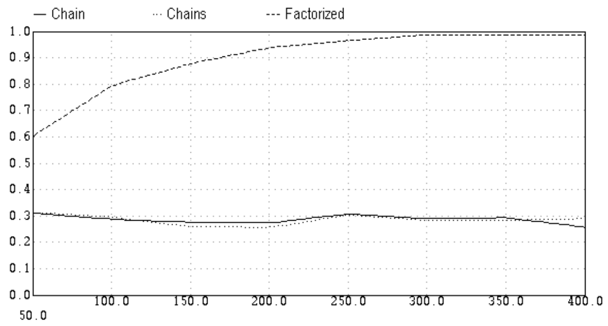

Figure 9: Percentile BBS correct for FDA variants, $d = 0.8, \tau = 0.8$.

In figures 11 and 10 it is shown that the perturbated algorithm even to a factor $\zeta = 4$ still optimizes the problem. Clearly it becomes harder to find the optimum, but most importantly, given a large enough population, the problem will still be solved. This is contrary to what we observed for the chain. Once again, this emphasizes the fact that processing the building blocks as a whole is required to optimize the problem. Our tests indicate that the EDAs are sensitive to unprecise block recognition, but that this is not a crucial factor. What is crucial is the fact that what has to be processed as a whole is the building blocks. In other words, the linkage information found has to be exploited in the form of using the building blocks in order to find competent EDAs.

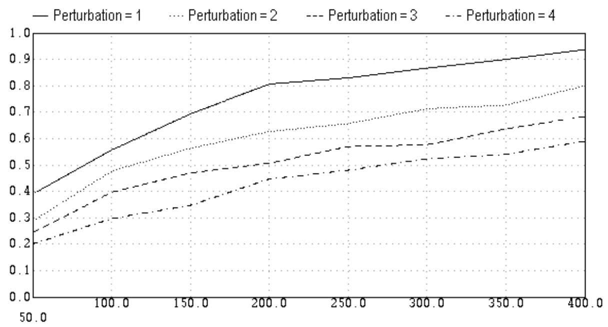

Figure 10: Percentile BBS correct for perturbed FDA, $d = 0.2, \tau = 0.2$.

5 Discussion

Using linkage information has been acknowledged to be of importance. Finding building blocks is the main issue. Once the building blocks are known, they can be searched more efficiently and be mixed together. Such was already the attempt of algorithms such as the mGA [7] and the fmGA. A good approach is therefore

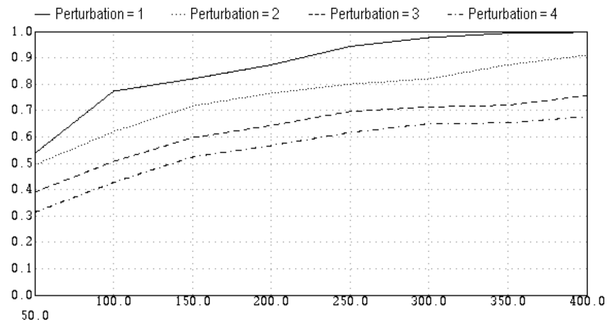

Figure 11: Percentile BBS correct for perturbed FDA, $d = 0.8, \tau = 0.2$.

to first search for the building blocks, filter them and then respect them in a mixing stage. Such an approach has been taken by some algorithms so far [3, 13] with very good results.

In the EDAs, linkage information is expressed through the probability distribution. This distribution is then immediately used to produce new individuals. The filtering and mixing stages are thus combined all together. It has become clear however that unless the building blocks are properly identified, optimization will fail. Most importantly, even though the linkage information is well expressed, not identifying building blocks will trouble optimization. For instance when using the second order chain of dependencies with the best possible distribution, either in a complete chain or in separate chains, optimization of deceptive problems with higher order building blocks fails. Directly using the linkage information degrades the potential of the methods. Note that the more sophisticated networks that *can* model complete building blocks have the same problem, unless the building blocks are identified and used. This means that the approach using Bayesian networks [10] of order k for instance requires a good learning algorithm. Without that, the proper network topology is not found and the building blocks are still not processed correctly.

We do note that in this paper we have only discussed tests over a single problem. Even though the results are encouraging, verification on other problems is desired. Also we should bear in mind the no free lunch theorem for search [14], telling us that there have to be problems on which the described algorithms perform bad. The problem described however has a clear structure. We are thus focusing on such problems, meaning our results remain valid in that domain.

In terms of building block filtering and mixing, it would be interesting to investigate an approach that uses an EDA to filter the building blocks. After this, a GA could be used to mix the blocks. For instance, assuming the relations can be learned through the correlations, a position biased crossover operator (such as one or two point crossover) could be used. The GA with this operator is applied after the genes are positioned next to each other according to the linkage information. For the chain distribution this is straightforward. By doing so, we attempt to process the building blocks as a whole in a better way.

6 Conclusions

Using correlations between alleles such as in EDAs to estimate probability distributions and using these in optimization is a good way to find linkage information. Directly using this linkage information in generating new individuals is however a troublesome issue in EDAs. Even though simple structures are capable of expressing linkage information, using them directly does not imply that higher order building block problems can be solved. Whereas the variables of the building blocks are placed close together in the network, the blocks are not identified as a whole. As a result of this, the block processing is too slow and optimization will not be successful.

The linkage information needs to be processed to identify the building blocks. Using the probability distribution then found, an EDA with this distribution will perform efficient optimization. So even though linkage information is very important for exploiting problem structure, building blocks need to be identified from this information after which they must be used in order to design competent EAs.

References

[1] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. Technical report, Carnegie Mellon University, 1995

[2] S. Baluja and S. Davies. Using optimal dependency–trees for combinatorial optimization: Learning the structure of the search space. In D.H. Fisher, editor, *Proceedings of the 1997 International Conference on Machine Learning.* Morgan Kauffman publishers, 1997. Also available as Technical Report CMU-CS-97-107.

[3] S. Bandyopadhyay, H. Kargupta, and G. Wang. Revisiting the gemga: Scalable evolutionary optimization through linkage learning. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 603–608. IEEE Press, 1998. Also available as Technical Report EECS-97-004, Washington State University, Pullman.

[4] J.S. De Bonet, C. Isbell, and P. Viola. Mimic: Finding optima by estimating probability densities. *Advances in Neural Information Processing*, 9, 1996

[5] D. Chickering, D. Geiger, and D. Heckerman. Learning bayesian networks: Search methods and experimental results. In *Proceedings of the fifth conference on Artificial Intelligence and Statistics*, pages 112–128. IEEE Press, 1995

[6] K. Deb and D.E. Goldberg. Sufficient conditions for deceptive and easy binary functions. *Annals of Mathematics and Artificial Intelligence*, 10:385–408, 1994

[7] D.E. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis and first results. *Complex Systems*, 10:385–408, 1989

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

[9] H. Mühlenbein, T. Mahnig, and O. Rodriguez. Schemata, distributions and graphical models in evolutionary optimization. ftp://borneo.gmd.de/pub/as/ga/gmd_as_ga-98_02.ps, 1998

[10] M. Pelikan, D.E. Goldberg, and E. Cantú-Paz. Linkage problem, distribution estimation and bayesian networks. Also available as IlliGAL Report 98013. ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/98013.ps.Z, 1998

[11] M. Pelikan and H. Mühlenbein. The bivariate marginal distribution algorithm. In *Advances in Soft Computing – Engineering Design and Manufacturing.* Springer–Verlag, 1999

[12] D. Thierens and D.E. Goldberg. Mixing in genetic algorithms. In S. Forrest, editor, *Proceedings of the fifth conference on Genetic Algorithms*, pages 38–45. Morgan Kaufmann, 1993

[13] C.H.M. van Kemenade. Building block filtering and mixing. In *Proceedings of the IEEE conference on Evolutionary Computation.* IEEE Press, 1998

[14] D.H. Wolpert and W.G. Macready. No free lunch theorems for search. Santa Fe Institute Technical Report SFI-TR-95-02-010. ftp://ftp.santafe.edu/pub/wgm/nfl.ps, 1995