Linkage Learning using the Maximum Spanning Tree of the Dependency Graph

B. Hoda Helmi, Martin Pelikan and Adel Rahmani

MEDAL Report No. 2012005

April 2012

Abstract

The goal of linkage learning in genetic and evolutionary algorithms is to identify the interactions between variables of a problem. Knowing the linkage information helps search algorithms to find the optimum solution efficiently and reliably in hard problems. This paper presents a simple approach for linkage learning based on the graph theory. A graph is used as the structure to keep the the pairwise dependencies between variables of the problem. We call this graph 'the underlying dependency graph of the problem'. Then maximum spanning tree (MST) of the dependency graph is found. It is shown that MST contains all the necessary linkage if the dependency graph is built upon enough population. In this approach, pairwise dependencies calculated based on a perturbation based identification method, are used as the variable dependencies. The proposed approach has the advantage of being capable of learning the linkage without the need for the costly fit-to-data evaluations for model search. It is parameter-less and the algorithm description is simple and straight forward. The proposed technique is tested on several benchmark problems and it is shown to be able to compete with similar approaches. Based on the experimental results it can successfully find the linkage groups in a polynomial number of fitness evaluations.

Keywords

Linkage learning, graph theory, maximum spanning tree, estimation of distribution algorithms, optimization problems, decomposable functions, nonlinearity detection.

Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Department of Mathematics and Computer Science
University of Missouri–St. Louis
One University Blvd., St. Louis, MO 63121
E-mail: `medal@medal-lab.org`
WWW: `http://medal-lab.org`

Linkage Learning using the Maximum Spanning Tree of the Dependency Graph

B. Hoda Helmi
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Dept. of Math and Computer Science, 321 ESH
University of Missouri at St. Louis
One University Blvd., St. Louis, MO 63121
helmib@umsl.edu

Martin Pelikan
Missouri Estimation of Distribution Algorithms Laboratory (MEDAL)
Dept. of Math and Computer Science, 320 ESH
University of Missouri at St. Louis
One University Blvd., St. Louis, MO 63121
pelikan@cs.umsl.edu

Adel T. Rahmani
SCOMAS LAB
Dept. of Computer Science
Iran University of Science and Technology
Tehran, Iran 13114-16846
rahmani@iust.ac.ir

Abstract

The goal of linkage learning in genetic and evolutionary algorithms is to identify the interactions between variables of a problem. Knowing the linkage information helps search algorithms to find the optimum solution efficiently and reliably in hard problems. This paper presents a simple approach for linkage learning based on the graph theory. A graph is used as the structure to keep the the pairwise dependencies between variables of the problem. We call this graph 'the underlying dependency graph of the problem. Then maximum spanning tree (MST) of the dependency graph is found. It is shown that MST contains all the necessary linkage if the dependency graph is built upon enough population. In this approach, pairwise dependencies calculated based on a perturbation based identification method, are used as the variable dependencies. The proposed approach has the advantage of being capable of learning the linkage without the need for the costly fit-to-data evaluations for model search. It is parameter-less and the algorithm description is simple and straight forward. The proposed technique is tested on several benchmark problems and it is shown to be able to compete with similar approaches. Based on the experimental results it can successfully find the linkage groups in a polynomial number of fitness evaluations.

Keywords: Linkage learning, graph theory, maximum spanning tree, estimation of distribution algorithms, optimization problems, decomposable functions, nonlinearity detection.

1 Introduction

Linkage learning is among the most studied topics in evolutionary computation. In optimization, linkage corresponds to interactions between variables of the problem. Knowing the linkage information, searching for the optimum solution can often be performed efficiently (Thierens & Goldberg, 1993). In black box optimization problems, linkage information is not available available on input and the linkages must be identified using samples of candidate solutions and their evaluation. The goal of linkage learning is to find important linkages from such samples. The two priorities in learning linkage are (1) accuracy of the identified linkages, and (2) efficiency of the learning. Several approaches were proposed for learning linkage along with the optimization search (Sastry & Goldberg, 2000; Pelikan, 2005; Yu, Sastry, Goldberg, Lima, & Pelikan, 2009; Larranaga & Lozano, 2002). There are also approaches that do the linkage learning separately (Yu & Goldberg, 2004; Nikanjam, Sharifi, Helmi, & Rahmani, 2010), this class of approaches are called *offline* throughout this paper. The output of this class of linkage learning approaches can then be used by evolutionary algorithms like the simple genetic algorithm with building-block-wise crossover (Sastry & Goldberg, 2000) or local search based on advanced, linkage-based neighbourhood operators (Posik, 2011) to find the optimum solution of the problem. As it is said above, the only source of data in black-box optimization is an evaluated population of potential solutions, so the population size is an important factor for learning of the linkage. Algorithms need certain minimal population size to have enough data to be able to find all the linkage groups correctly. It is important to find the minimal needed population size, because first evaluation of the population may be a time consuming task especially in real world problems, and second too large population may cause falsely discovering of the linkages (Santana, Larranaga, & Lozano, 2007). So unnecessary evaluations are avoided in good linkage learning solutions and size of the population and number of function evaluations are metrics for evaluating the linkage learning algorithms. In this paper, a linkage identification approach is introduced that is built upon the simple concept of maximum spanning tree. The proposed algorithm uses perturbation based pairwise dependencies to construct the underlying dependency graph of the problem. Maximum spanning tree of the graph is then found.

The proposed model has the advantage of being capable of learning the linkage without the need for the costly fit-to-data evaluations in every generation. It only use a population for computing the pairwise dependency by computing the fitness value of every order two schemata. It is parameterless and simple in terms of the algorithm description and algorithm complexity.

In next section, some related works are reviewed. In section 3, the advantages and disadvantages of offline linkage learning is briefly discussed. Section 4 is all about the proposed linkage identification algorithm and the background information is needed to explain it. Section 5 presents the complexity analysis of the algorithm. In section 6, reference algorithms are introduced. Section 7 presents the experimental results of algorithm. To show the performance and efficiency of the proposed approach, it is tested on some benchmark problems and its effectiveness is discussed. Finally in section 8, some conclusion remarks, highlights of the algorithm and future directions are presented.

2 Related Works

Several linkage learning techniques were presented in the literature and these techniques were classified from different perspectives (Chen, Yu, Sastry, & Goldberg, 2007).

Linkage adaptation techniques are the first attempts for linkage learning. They are in fact evolutionary processes which employ special operators or representations for avoiding the disturbance of building blocks by adapting the linkage along with the evolutionary process. the linkage evolves

along with the individual solutions. Several such approaches are reported in literature (ping Chen & Goldberg, 2006; Chen, Yu, Sastry, & Goldberg, 2007).
Estimation of distribution algorithms (EDAs) (Larranaga & Lozano, 2002) replace conventional variation operators such as crossover and mutation by building and sampling a probabilistic model of selected solutions. Many EDAs are able to learn linkage between problem variables, often encoded by graphical models such as Bayesian networks. In most EDAs, linkages are identified in each generation. The first EDAs, such as univariate marginal distribution algorithm (UMDA) (Muhlenbein, 1997) did not use dependencies between variables. The second generation of EDAs used bivariate dependencies and the last generation could discover and use multivariate dependencies. The learning factorized distribution algorithm (LFDA) (Mahnig & Muhlenbein, 1999), the extended compact genetic algorithm (ECGA) (Sastry & Goldberg, 2000) and the Bayesian optimization algorithm (BOA) (Pelikan, 2005) are some of the advanced EDA approaches capable of linkage learning.
Perturbation-based methods, detect linkage groups by perturbing individuals in the population and inspecting the fitness changes caused by the perturbations. The non-monotonicity/non-linearity which can be detected by perturbation is considered as linkage. Gene expression messy genetic algorithm (gemGA) (Kargupta, 1996), which uses transcription operator for identifying linkage groups can be classified in this category. Linkage identification by nonlinearity check (LINC) and Linkage identification by non-monotonicity detection (LIMD) (Munetomo & Goldberg, 1999) are other perturbation-based approaches which detect linkage by examining nonlinearity and non-monotonicity of fitness change by perturbations in pairs of variables. The idea of LINC and LIMD was also generalized for functions with overlapping linkage groups by introducing the linkage identification with epistasis measures (LIEM) (Munetomo, 2001). The perturbation analysis in LIEM can be done for any k-tuple bits. Perturbation-based linkage learning are usually used as part of the genetic algorithms.
The offline utility of the dependency structure matrix genetic algorithm (Yu & Goldberg, 2004) uses a perturbation based linkage identification like LINC. The dependency structure matrix is used for calculating the pairwise dependencies and a DSM clustering approach is introduced for turning pairwise dependencies into linkage groups. The linkage identification phase is completely separated from the genetic search. The DSM clustering method in the offline dependency structure matrix genetic algorithm is based on the evolutionary strategy. DSMC (Nikanjam, Sharifi, Helmi, & Rahmani, 2010) is another offline linkage learning technique that has introduced a density based iterative clustering algorithm for DSM clustering. In this approach, DSM is constructed using a perturbation based metric like LINC. Both of the off-line linkage learning approaches mentioned above use a binary value as the variable interactions, they use a binary matrix as the DSM and two variables are considered either completely dependent or completely independent.
The proposed linkage learning technique can be also classified in the category of perturbation based methods. Our approach is performed offline, similar to offline utility of the dependency structure matrix genetic algorithm and DSMC. These two approaches are used as reference algorithms and are explained more in section 6.

3 Why Offline Linkage Learning

Pros and cons of offline linkage learning are discussed in detail in ref. (Yu & Goldberg, 2004); here we discuss the most important ones briefly.
One of the most important benefits of offline linkage learning is the reduction of computational resources. The time savings are mainly due to the fact that the offline linkage learning is per-

formed only once but online linkage learning is performed in every generation. In ref. (Yu & Goldberg, 2004) it is also argued the offline linkage learning saves some number of function evaluations. $O(l \log l)$ is reported as a loose lower bound for number of function evaluations for a GA with off-line linkage learning. This lower bound is estimated by a time to convergence model which assumes an infinite population size and perfect mixing.
Offline linkage learning also enables us to use the linkage information obtained for a problem, when solving similar problems as well.
There are also few disadvantages for using offline linkage learning. The most important one is that the offline linkage learning techniques should be accurate because the wrong identified linkage groups make the convergence difficult for the search algorithms. On the other hand, with online linkage learning if a linkage group is misidentified in one generation, it may be identified correctly in the next generation. Another disadvantage of offline linkage learning is that it can only be used for problems for which one division of variables into linkage groups suffices; for many problems, the model can change over time to cover different sets of dependencies at different times. This is useful especially for problems that are not additively separable of bounded order.

4 Proposed Approach

The proposed algorithm consists of three steps: (1) Calculate pairwise dependencies and construct the pairwise dependency graph of the problem, (2) find the maximum spanning tree of the graph and (3) cut off the low-weight edges (false linkages). The linkage information in the algorithm is in the form of groups of independent variables that should be treated together. The pseudo-code is depicted as Alg 2.
The approach as will be explained here is capable of learning disjoint linkage groups even for problems with linkage groups of varying size. Extensions of the proposed approach to problems with linkage groups that interact with each other (subproblems overlap) is an important topic for future work. The remainder of this section describes details of the proposed approach.

Step 1: Constructing The Dependency Graph

The dependency graph is a weighted, undirected graph where each vertex corresponds to one decision variable and each edge of the graph has a weight e_{ij} representing the strength of dependency between vertex i and vertex j. Edge weights $\{e_{ij}\}$ are real numbers. The larger the e_{ij} is, the stronger the dependency is between vertex i and vertex j.
To construct the dependency graph, a pairwise dependency metric is calculated for each pair of variables. Mutual information, simultaneity metric (Sangkavichitr & Chongstitvatana, 2009) and some other metrics can be used. Since the linkage learning is going to take place offline, in order to maximize information extracted from each evaluated solution, we decided to use perturbation methods as the basis for creating the underlying dependency graph. The main advantage of perturbation methods is that they are capable of measuring strength of interactions even with a uniformly distributed population. However, it is straightforward to adopt other linkage identification mechanisms instead. Because our algorithm uses pairwise dependencies, among the perturbation methods, a linkage identification metric like linkage identification by non-linearity check (LINC) and linkage identification by non-monotonicity detection (LIMD) would be suitable. We use a metric similar to LINC which is used in ref. (Yu & Goldberg, 2004) and ref. (Nikanjam, Sharifi, Helmi, & Rahmani, 2010).

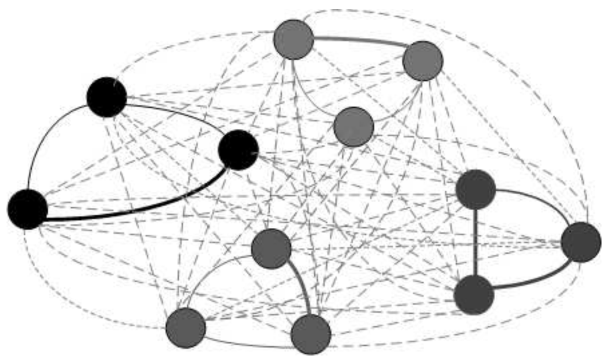

Figure 1: The underlying graph of the maxTrap3-4 problem

Figure 1 shows an example of an underlying dependency graph of a problem with 12 variables (a concatenated function with four 3-bit traps (see Sec.7)). The thickness of edges shows their weight magnitude. Lines are the correct linkages and dashed lined are false linkages.
If the dependency graph is correctly constructed, the edges between dependent variables would be stronger than the edges between non-dependant variables.

Pairwise dependencies:

The pairwise dependency metric defined in Ref. (Yu & Goldberg, 2004) is used for constructing the dependency graph. Define $f_{a_i=x,a_j=y}$ as the average fitness of all the schemata where the i^{th} variable is x, the j^{th} variable is y, and the rest are $*$ (don't care symbol). For example, for $i = 1$ and $j = 4$ in a 5-bit problem, $f_{a_i=0,a_j=1} = f(0**1*)$. Values of $f_{a_i=0,a_j=1} - f_{a_i=0,a_j=0}$ and $f_{a_i=1,a_j=1} - f_{a_i=1,a_j=0}$ should be the same if the i^{th} variable and the j^{th} variable are independent. Therefore, the interaction between the i^{th} variable and the the j^{th} variable is defined as $|f_{a_i=0,a_j=1} - f_{a_i=0,a_j=0} - f_{a_i=1,a_j=1} + f_{a_i=1,a_j=0}|$.
However, the fitness value of schemata cannot be computed unless possible combination space is visited completely. In practice, the actual fitness of the schemata is computationally too expensive to be calculated so the observed fitness of the individuals seen in a large enough population is calculated instead. The interaction between the i^{th} variable and the the j^{th} variable calculated from the seen population by this method is used as the weight of edges between variable i and j in the dependency graph.

Step 2: Finding the Maximum Spanning Tree of the Underlying Graph of the Problem

Once the graph is constructed, a maximum spanning tree of the graph is constructed by one of the classic algorithms like Prime's or Kruskal's one.

Maximum spanning tree of the dependency graph

In graph theory, a spanning tree T of a connected, undirected graph G is a tree composed of all the vertices and some (or perhaps all) of the edges of G. That is, every vertex lies in the tree, but no cycles (or loops) are formed. A maximum spanning tree (MST) or maximum weight spanning tree is the spanning tree with weight greater than or equal to the weight of every other spanning tree. If pairwise dependencies are calculated on enough population, the MST contains the strongest path

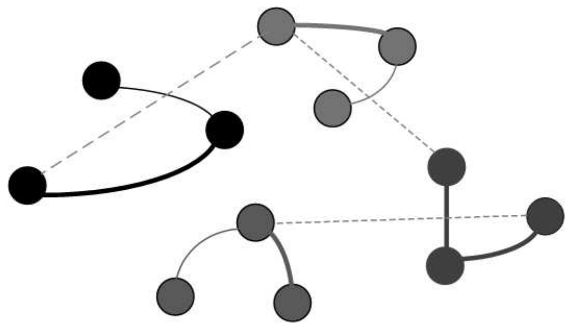

Figure 2: The maximum spanning tree of the maxTrap3-4 problem

between variables in a linkage group. In a problem with m linkage groups of size k, The path has exactly $k-1$ strongest edges between variables in a linkage group of size k. MST also contains $m-1$ weak edges (false linkages) between m linkage groups. These linkages are added to the MST to keep it connected.

In a dependency graph, weights are non-linearities between variables. Ideally, a non-linearity between two independent variables should be zero. In such situations, the underlying graph of the problem would contain edges of weight 0 and the maximum spanning tree would consist of linkage groups that are connected with edges of 0 weight. Even with incomplete information from a sample of limited size, N, Non-linearity between variables that belong to different linkage groups is still expected to be greater than non-linearity between independent variables. A maximum spanning tree contains the minimum required maximum weight edges (linkages) between vertices (variables) of a linkage group and an edge (false linkage) between linkage groups as well. In Fig 2 a maximum spanning tree of the graph shown in Fig1 is demonstrated. In the proposed method Prime's algorithm is used to find the MST. The Prime algorithms is as follow:

Finding the MST is done in $O(n^2)$.

Algorithm 1 Prime algorithm to find the MST

```
Input: A non-empty connected weighted graph with vertices V and edges E.
Output: MST
  Initialize: $V_{new} = x$, where $x$ is an arbitrary node from $V$, $E_{new} = \emptyset$
  repeat
    Choose an edge (u, v) with maximal weight such that u is in $V_{new}$ and v is not.
    Add v to $V_{new}$, and (u, v) to $E_{new}$
  until $V_{new} = V$:
```

Step 3: Cutting Off the False Linkages

The final task consists of cutting off the edges between the linkage groups. If sufficient population is used for constructing the graph, a simple 2-means clustering algorithm (a special case of the k-means algorithm, where $k = 2$) would be able to find the set of false linkages, because with enough population size, the $m * (k-1)$ correct linkages in MST are the strongest ones in the dependency graph and the $m-1$ false linkages in MST are supposed to be weaker than all the correct linkages $(m * k * (k-1)/2)$ in the dependency graph, so there would be a sensible gap between weights of the $m * (k-1)$ correct linkages and $m-1$ false linkages in the MST.

A more intelligent threshold that considers the intra linkage group edges of the graph (that are not

in the MST) may be able to lower the population, but it will definitely complicate the process of pruning the false edges. In this experiment, the 2-means algorithm is simply used to find the false linkages.
Cutting the false linkages will create a maximum spanning forest, consisting of several trees. Each tree represents a linkage group.

Algorithm 2 Algorithm of the proposed approach

```
Output: Linkage groups
 1: Create random population.
 2: Evaluate the population.
    {Computing the pairwise dependencies $e_{ij} \in E$ between variables $i, j \in V$ and constructing the dependency graph $G_d(V, E)$}
 3: for each variable $i$ and variable $j$, do
 4:   for each individual $a$ in the population, do
 5:     Update $f_{a_i=0,a_j=0}||f_{a_i=0,a_j=1}||f_{a_i=1,a_j=0}||f_{a_i=1,a_j=1}$
 6:   end for
 7:   $e_{ij} =$
      $|f_{a_i=0,a_j=1} - f_{a_i=0,a_j=0} - f_{a_i=1,a_j=1} + f_{a_i=1,a_j=0}|$
 8: end for
 9: Find the maximum spanning tree of the graph.
10: Find threshold ($T$) by 2-means clustering on the edges of the MST.
11: Cut the edges weighted less than $T$.
```

5 Compexity Analysis

In this section, the number of function evaluations and the complexity of the proposed linkage learning algorithm is discussed. As discussed above, the algorithm is comprised of three main steps of (1) constructing the underlying graph, (2) finding the MST of the graph and (3) cutting off the false linkages.
Only in the first step of the algorithm, for identifying the non-linearity, evaluation is performed. Each individual in the population is evaluated exactly once. Therefore if population size is N, number of fitness evaluations would be N.
Pair-wise dependencies (f values) are computed by a pass through all the strings in the population and updating four f values for each pair of variables. Computing the f values is done $O(n^2)$ times for each string. Therefore this computation is done in three nested loops and take $O(n^2 \times N)$ iterations, where n is problem size and N is population size.
Finding the MST is done in $O(n^2)$. For cutting the false linkages a simple 2-means clustering algorithm is used. The main loop of the 2-means clustering algorithm is set to a constant number. Maximum number of items for clustering is $n - 1$, so the computation complexity of cutting the false linkages is $O(n)$

6 Reference Algorithms

Two approches are used as a reference. Both are offline linkage learning approaches and are performed once before the search for the optimum solution. Both use a nonlinearity detection method similar to LINC as the dependency metric.

Offline utility of DSMGA (Yu & Goldberg, 2004) uses nonlinearity detection to construct the dependency structure matrix and then cluster the DSM by an evolutionary strategy with a fit-to-data objective function.
The second approach DSMC (Nikanjam, Sharifi, Helmi, & Rahmani, 2010) is also a DSM clustering algorithm. In ref. (Nikanjam, Sharifi, Helmi, & Rahmani, 2010) a density based iterative clustering algorithm is proposed to find linkage groups. This approach does not need to check the model fit with regard to data, but it needs to set some parameters and thresholds to do the clustering.
Both of the offline linkage learning approaches mentioned above use binary values as the variable interactions, they use a binary matrix as the DSM and two variables are considered either completely dependent or completely independent. Deciding the threshold to convert the DSM to a binary DSM, which is done in both approaches, is not an easy task.

7 Experimental Results

In this section, the experimental results are presented. First the experiments and test functions are explained and then the results are shown and analysed.

Test functions

Trap function is a benchmark problem in the linkage learning community. It is a function with hard-to-detect linkage.

$$f(x) = \begin{cases} k & \text{if } u = k \\ k - 1 - u(x) & \text{else} \end{cases} \tag{1}$$

where $u(x)$ returns number of 1s in string x. It has one global optimum in individual of all 1s and one local optimum in individuals of all 0s. The function is difficult because the local optimum has larger basin of attraction.
A Concatenated Trap function is a sum of Trap subfunctions. In this paper, concatenated Trap functions are used as the test function. A concatenated Trap function with m subfunctions, has one global optimum and $2^m - 1$ local optima.

Setup of experiments

The results are shown for the concatenated Trap functions with linkage groups of sizes 4 (for problem sizes 20 to 160), 5 (for problem sizes 25 to 200) and 6 (for problem sizes 24 to 240).
A concatenated Trap function with different-sized Trap subfunctions (mixedTrap) is also used as test function. The concatenated mixed-Trap function used in the experiments is additively composed of 2-bit, 3-bit, 4-bit, 5-bit, 6-bit trap functions. The concatenated mixed-Trap functions for different problem sizes (20, 40, 60, 80 and 100) are used.

Population size estimation

Population size is an important factor for learning of the linkage. Algorithms need certain minimal population size to have enough data to be able to find all the linkage groups correctly. It is important to find the minimal needed population size because first, evaluation of the population may be a

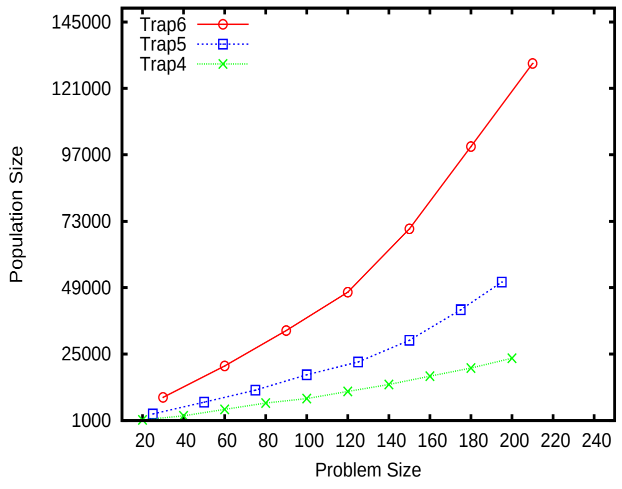

Figure 3: Population size needed for the proposed algorithm to successfully learn all the linkage groups for concatenated Trap function

time consuming task especially in real world problems, and second, too large population may cause falsely discovering of the linkages (Santana, Larranaga, & Lozano, 2007). Two population sizing schemata are introduced in (Yu & Goldberg, 2004) for off-line linkage learning approaches. In this study we didn't use a population sizing estimation and it is left for future works.
To report the enough population size for our approach, bisection method is used. Population size is determined by bisection and the success metric has been 50 successful independent runs of the algorithm in which all the linkage groups are identified correctly. The error rate for linkage learning is zero and all the linkage groups are learned correctly in 50 independent runs with the population size determined by bisection.
In figure 3, population size (number of fitness evaluations) needed for the proposed algorithm to learn linkage groups for Trap4, Trap5 and Trap6 is plotted. In figure 4, population size for MixedTrap problem is shown.

Scalability

We have fitted our results to $N \approx an^b$ (polynomial scaling) and $N \approx an^b \log n$ reported for EDAs. Based on the coefficient of determination (R^2) both models are acceptable. This deduction is based only on the available data.
In figure 5, number of fitness evaluations for Trap4, Trap5 and Trap6 are plotted together with the fit model $N \approx an^b$. b is observed to be in the range (1,1.7) for the proposed algorithm.

Comparison with reference algorithms

The reference algorithms are described in section 6. For offline utility of DSMGA, results on a concatenated Trap function with 10 subfunctions of order 5 is reported, 11712 function evaluations was needed to correctly identifies 99.8 of linkage groups. For the same problem 7625 function evaluations are needed for the proposed algorithm to find all the linkage groups.
The results of both DSMC and the proposed algorithm are depicted in figure 6. Comparing the results visually, the DSMC needs smaller number of fitness evaluations, but it comes at the cost

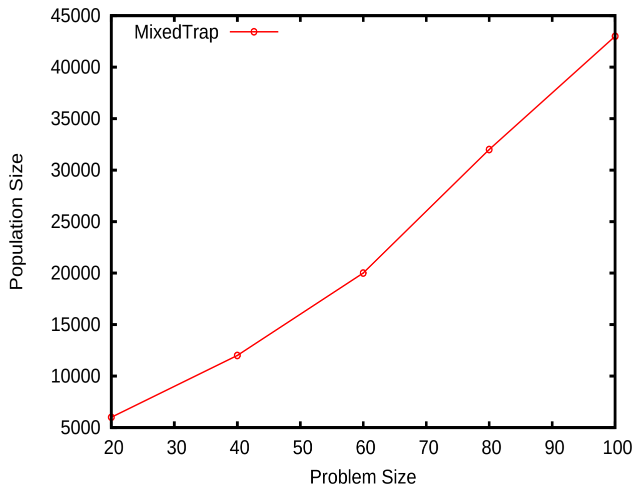

Figure 4: Population size needed for the proposed algorithm to successfully learn all the linkage groups for concatenated MixedTrap function

of setting and defining several metrics and parameters for iteratively refining the initial clusters of DSM and find the linkage groups.
As said before both DSMC and the proposed approach use the same metric as the pairwise dependencies. Therefore, their difference is in the way they find the linkage groups. The computational complexity of the clustering algorithm of DSMC is $O(n^3)$ (Nikanjam, Sharifi, Helmi, & Rahmani, 2010), while the computational complexity of the proposed algorithm for finding the linkage groups is $O(n^2)$ which is the complexity of finding the maximum spanning tree.

8 Summary and Conclusion

This paper introduced a simple offline linkage learning approach. The algorithm consists of three main steps. First, a dependency graph is created using a pairwise metric. Second, a maximum spanning tree is built for the dependency graph. Third, edges corresponding to weakest dependencies in the maximum spanning tree are eliminated and the connected components are used as linkage groups. The proposed approach uses a perturbation based linkage identification method to find the pairwise dependencies between variables, although other pairwise metrics could be adopted in a straightforward manner. The proposed method does not need to do the costly fit-to-data check, it is one of the features of the algorithm. To demonstrate the performance of the proposed approach, the results on different sized Trap functions and mixed-Trap functions are reported. It is shown that the proposed approach can find all the linkage information correctly (at least for the tested problems) with a polynomial number of fitness evaluations. The results are compared to the results of two reference approaches and it is shown that performance of the three methods is comparable in terms of the number of function evaluations. The main advantage of the proposed approach compared to prior work in offline linkage learning and DSMC is that the algorithm contains no parameters that must be tuned. Linkage groups discovered by the proposed algorithm can be used by any search algorithm to find the optimum. The search algorithm can be chosen knowing the number of linkage groups and order of linkage groups, it can be a simple local search or a building-block wise crossover genetic algorithm. The algorithm can successfully find the linkage groups of the additively separable functions with different sized subfunctions. As the future work,

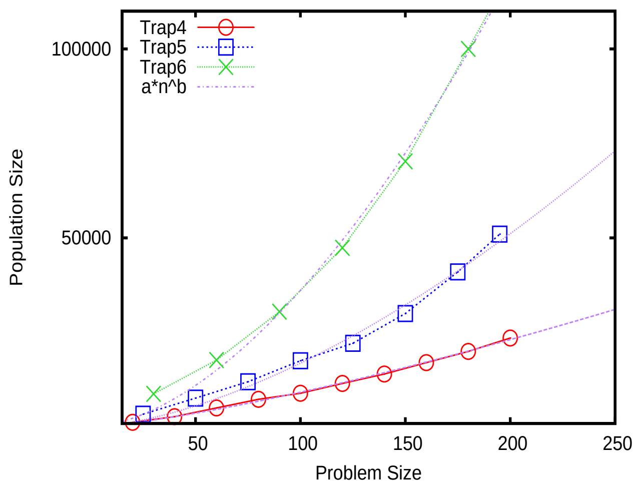

Figure 5: Number if fitness evaluations and the fit $N \approx an^b$ to the results for concatenated Trap function

Performance of the proposed algorithm should be tested on other problems as well, like functions with overlapping linkage groups and exponentially scale problems. A population size estimation is needed for automatic determination of a good population size, two approaches are introduced in ref. (Yu & Goldberg, 2004) which should be tested for the proposed algorithm. The performance of the proposed algorithm as a model building approach along with the optimization search should be tested and discussed in future works.

Acknowledgments

References

Chen, Y.-P., Yu, T.-L., Sastry, K., & Goldberg, D. E. (2007). *A survey of linkage learning techniques in genetic and evolutionary algorithms* (Technical Report). Illinois Genetic Algorithms Laboratory, Urbana, IL: IlliGAL Report No. 2007014, University of Illinois at Urbana-Champaign.

Kargupta, H. (1996). The performance of the gene expression messy genetic algorithm on real test functions. In *1996 IEEE International Conference on Evolutionary Computation* (pp. 631–636). Morgan Kaufmann Publishers, Inc.

Larranaga, P., & Lozano, J. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Kluwer Academic Pub.

Mahnig, T., & Muhlenbein, H. (1999). FDA - a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation,*, *7*(4), 353–376.

Muhlenbein, H. (1997). The equation for response to selection and its use for prediction. *Evolutionary Computation*, *5*(3), 303–346.

Munetomo, M. (2001). Linkage identification based on epistasis measures to realize efficient genetic algorithms. In *World Congress on Computational Intelligence* (pp. 1332–1337). IEEE

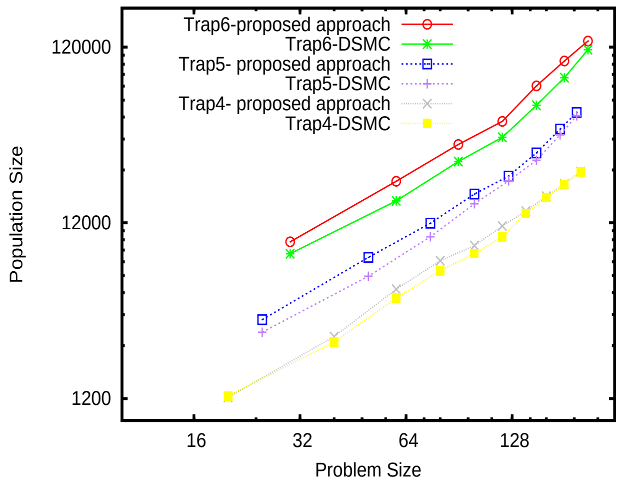

Figure 6: Number of fitness evaluations needed to solve concatenated Trap problem for both DSMC and the proposed approach

Computer Society.

Munetomo, M., & Goldberg, D. E. (1999). Identifying linkage groups by nonlinearity/nonmonotonicity detection. In *Genetic and Evolutionary Computation Conference (GECCO-99)* (pp. 433–440). Morgan Kaufmann Publishers, Inc.

Nikanjam, A., Sharifi, H., Helmi, B. H., & Rahmani, A. T. (2010). A new dsm clustering algorithm for linkage groups identification. In *Genetic and Evolutionary Computation Conference (GECCO-10)* (pp. 367–368). ACM.

Pelikan, M. (2005). *Hierarchical bayesian optimization algorithm.* Springer-Verlag Berlin Heidelberg.

ping Chen, Y., & Goldberg, D. E. (2006, March). Convergence time for the linkage learning genetic algorithm. *Evolutionary Computation*, *13*(3), 279–302.

Posik, S. V. P. (2011). Parameter-less local optimizer with linkage identification for deterministic order-k decomposable problems. In *Genetic and Evolutionary Computation Conference (GECCO-11)* (pp. 577–585). ACM.

Sangkavichitr, C., & Chongstitvatana, P. (2009). Direct and explicit building blocks identification and composition algorithm. In *CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation* (pp. 2460–2465). IEEE Press.

Santana, R., Larranaga, P., & Lozano, J. (2007). *Challenges and open problems in discrete edas.* University of Basque Country, Spain: Technical Report EHU-KZAA-IK-1/07 Department of Computer Science and Artificial Intelligence.

Sastry, K., & Goldberg, D. E. (2000). *On extended compact genetic algorithm.* Illinois Genetic Algorithms Laboratory, Urbana, IL: IlliGAL Report No. 2000026, University of Illinois at Urbana-Champaign.

Thierens, D., & Goldberg, D. E. (1993). Mixing in genetic algorithms. Morgan Kaufmann Publishers, Inc.

Yu, T.-L., & Goldberg, D. E. (2004). Dependency structure matrix analysis: Offline utility of the dependency structure matrix genetic algorithms. In *Genetic and Evolutionary Computation Conference (GECCO-2004), Part II, LNCS 3103* (pp. 355–366). Springer.

Yu, T.-L., Sastry, K., Goldberg, D. E., Lima, C. F., & Pelikan, M. (2009). Dependency structure matrix, genetic algorithms, and effective recombination. *Evolutionary Computation*, *17*(4), 595–626.