International Journal of the Physical Sciences Vol. 6(14), pp. 3419–3425, 18 July, 2011
Available online at http://www.academicjournals.org/IJPS
DOI: 10.5897/IJPS11.798
ISSN 1992 - 1950 ©2011 Academic Journals

Full Length Research Paper

Linkage learning based on differences in local optimums of building blocks with one optima

Hamid Parvin[1], Hoda Helmi[1], Behrouz Minaei[1], Hamid Alinejad Rokny[2] and Hossein Shirgahi[3]*

[1]School of Computer Engineering, Iran University of Science and Technology (IUST), Tehran, Iran.
[2]Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran.
[3]Young Researchers Club, Jouybar Branch, Islamic Azad University, Jouybar, Iran.

Accepted 07 June, 2011

Genetic algorithms (GA) are categorized as search heuristics and have been broadly applied to optimization problems. These algorithms have been used for solving problems in many applications. Nevertheless, it has been shown that simple GA is not able to effectively solve complex real world problems. For proper solving of such problems, knowing the relationships between decision variables which is referred to as linkage learning is necessary. In this paper, a linkage learning approach is proposed that utilizes the special features of decomposable problems to solve them. The proposed approach is called Local optimums based linkage learner (LOLL). The LOLL algorithm is capable of identifying the groups of variables which are related to each other (known as linkage groups), not minding if these groups are overlapped or different in size. The proposed algorithm, unlike other linkage learning techniques, is not done along with optimization algorithm, but it is done in a whole separated phase from optimization search. After finding linkage group information by LOLL, the optimization search can use this information to solve the problem. LOLL is tested on some benchmarked decomposable functions. The results show that the algorithm is an efficient alternative to other linkage learning techniques.

Key words: Linkage learning, optimization problems, decomposable functions.

INTRODUCTION

A competitive market is a market in which a great number of buyers and sellers are busy trading independently. As a result, none of the market members can influence the price greatly, because the announced prices are almost fixed and also low. In this market, because of the competition, sellers use decision making process under emergency conditions to sell products more. They use their past experiences to present suggestions close to the interests of their customers and attract them more quickly.

Genetic algorithms are the most popular algorithms in the category of evolutionary algorithms. These algorithms are widely used for solving real-world problems. However when it comes to solving difficult problems, GA has deficiencies. One of the main problems of simple GAs is their blindness and oblivion about the linkage between variables. The importance of linkage learning was recognized in success of the optimization search after a long time. There are a lot of linkage learning techniques. Some are based on perturbation methodology, some are categorized in the class of probabilistic model building approaches and some are the

*Corresponding author. E-mail: hossein.shirgahi@gmail.com.

Abbreviations: BOA, Bayesian optimization algorithm; **BB,** building blocks; **GA,** genetic algorithms; **LOLL,** local optimums based linkage learner; **LEGO,** linkage evolving genetic operator; **EDA,** estimation of distribution algorithm; **DSMGA,** dependency structure matrix genetic algorithm; **DHC,** deterministic hill climbers; **HGPA**, hypergraph-partitioning algorithm; **HBOA,** hierarchical bayesian optimization algorithm.

techniques that adapt the linkages along with the evolutionary process by employing special operators or representations.

There are lots of approaches in the class of linkage adaptation techniques. Linkage learning genetic algorithm (Newman, 1991) uses a special probabilistic expression mechanism and a unique combination of the (gene number, allele) coding scheme and an exchange crossover operator to create an evolvable genotypic structure. Punctuation marks are added to the chromosome representation (Miller, 1984). These bits indicate if any position on the chromosome is a crossover point or in another words, a linkage group boundary. Linkage evolving genetic operator (LEGO) (Pelikan, 2005) is another linkage adaptation strategy that in order to achieve the linkages, each gene has associated with it two Boolean flags. These two flags determine whether the gene will link to the genes to its left and right. The two adjacent genes are assumed to be linked if the appropriate flags are both set to true. Therefore building blocks are consecutive linked genes on the chromosome.

Linkage learning is necessary when there are epistatic linkages between variables. Estimation of distribution algorithms (EDAs) are among the most powerful genetic algorithms which try to find these epistatic linkages through building probabilistic models that summarize the information of promising solutions in the current population. In another words, by using probabilistic models these algorithms seek to find linkage between variables of the problem. In each generation they find as much information as possible about the variable dependencies from the promising solutions of the population. Knowing this information, the population of the next generation is created. There are numbers of estimation of distribution algorithms which differ in the model building part. Bayesian networks and marginal product models are examples of the probabilistic models that have been used by BOA (Audebert and Hapiot, 1993) and eCGA (Hillman, 1987). Although EDAs scale polynomial in terms of number of fitness evaluations, the probabilistic model building phase is usually computationally expensive. Perturbation-based method, detect linkage group by injecting perturbations in the population of individuals and inspecting the fitness change caused by the perturbation. Gene expression messy genetic algorithm (gemGA) which uses transcription operator for identifying linkage group is classified in this category.

Dependency structure matrix genetic algorithm (DSMGA) is another approach which models the relationship among variables using dependency structure matrix (DSM) (Strehl and Ghosh, 2002). In DSMGA a clustering method is used for identifying the linkage groups. In spite of these efforts, none of the algorithms have been claimed to be stronger than that HBOA which itself in spite of it's polynomially scale in terms of number of fitness evaluations, is computationally expensive.

In this paper a new linkage learning approach, which is called "Local Optimum based Linkage Learner" (LOLL), is proposed. The proposed algorithm as its title implies, does not fall in the above mentioned categories, but it is a linkage group identification approach which tries to identify multivariate dependencies of complex problems in acceptable amount of time and with admissible computational complexity.

In this section, deterministic hill climbers (DHC) which will be used later in our algorithm and challenging problems which are used to explain and test the proposed algorithm are described. Firstly, some terms should be defined. A partial solution denotes specific bits on a subset of string positions. For example, if we consider 100-bit binary strings, a 1 in the second position and a 0 in the seventh position is a partial solution. A building block is a partial solution which is contained in an optimum and is superior to its competitors. Each additively separable problem is composed of number of partitions each of which is called a "linkage group".

Deterministic hill climber

In this study, deterministic hill climbers (DHC) are used for searching for local optimums. In each step, the deterministic hill climber flips the bit in the string that will produce the maximum improvement in fitness value. This process can be allowed to iterate until no single bit flip produces additional movement. Deterministic hill climber starts with a random string.

MATERIALS AND METHODS

Challenging problems

Deficiencies of genetic algorithms were first demonstrated with simple fitness functions called deceptive functions of order k. Deception functions of order k are defined as a sum of more elementary deceptive functions of k variables. In a deceptive function the global optimum (1, 1) is isolated, whereas the neighbours of the second best fitness solution (0, 0) have large fitness values. Because of this special shape of the landscape, genetic algorithms are deceived by the fitness distribution and most GAs converge to (0, 0). This class of functions is of great theoretical and practical importance. Below the reader will find an additively separable function, Trap 5, where u is the number of ones in the input block of 5 bits. An n-bit Trap 5 function has one global optimum in the string where the value of all the bits is 1, and it has (2n/5) - 1 local optimums. The local optimums are those individuals that the values of the variables in a linkage group are either 1 or 0 (they are all 1, or they are all 0).

$$trap_5(u) = \begin{cases} 5 & if\ (u = 5) \\ 4 - u & otherwise \end{cases}$$
$$f_{trap_5}(X) = \sum_{i=1}^{n/5} trap_5(X_{(i-1)*5+1:(i-1)*5+5}) \quad (1)$$

Also another additively separable function called Deceptive 3 defined as below where u is the number of ones in the input block of 3 bits. An n-bit Deceptive 3 function like an n-bit Trap 3 function has one global optimum in the string where the value of all the bits is 1, and it has (2n/3) - 1 local optimums.

$$\text{deceptive}_3(u) = \begin{cases} 1 & if\,(u=3) \\ 1-u*0.1 & otherwise \end{cases} \qquad (2)$$

$$f_{\text{deceptive}_3}(X) = \sum_{i=1}^{n/3} \text{deceptive}_3(X_{(i-1)*3+1:(i-1)*3+3})$$

Where u is the number of ones in the input block of 3 bits. For yet another more challenging problem here we present an additively separable function, one bit Overlapping-Trap 5, where u is the number of ones in the input block of 5 bits. An n-bit Overlapping-Trap 5 function has one global optimum in the string where the value of all the bits is 1 just similar to Trap 5 function, and it has (2(n-1)/4) - 1 local optimums. The local optimums are those individuals that the values of the variables in a linkage group are either 1 or 0 (they are all 1, or they are all 0) again similar to Trap 5 function.

$$overlapping\_trap_5(u) = \begin{cases} 5 & if\,(u=5) \\ 4-u & otherwise \end{cases} \qquad (3)$$

$$f_{overlapping\_trap_5}(X) = \sum_{i=1}^{n/4} overlapping\_trap_5(X_{(i-1)*4+1:(i-1)*4+5})$$

where u is the number of ones in the input block of 5 bits.

Local optimum based linkage learner (LOLL)

The main idea in the proposed approach for identifying the multivariate dependencies is using local optimums. But how can the local optimums lead us to identification of the linkage groups?

Local optimums of "additively separable problems" have some unique features. As it is obvious, the global optimum of an additively separable problem is the one that all of its building blocks are identified. In another words, in the global optimum, all of the linkage groups of the problem have the highest possible contribution to the overall fitness. But in local optimum solutions, not all of the building blocks are found and those partitions or sub-problems of the problem whose optimum values are not found are occupied by the best competitors of the superior partial solution.

In additively separable problems there are lots of these local optimum solutions. Actually, number of such solutions is directly dependant on length of the problem and number of partitions (or sub-problems or linkage groups) of the problem. It can be said that each local solution contains at least one building block (except the one with all 0s) and therefore comparison of the optimum solutions can lead us to identification of the linkage groups.

The following example can reveal this concept more clearly. Consider a 12 bit Trap 3 function. This function has one global optimum 11111111111 and (212/3) - 1 = (15) local optimums. The strings are local optimum if the bits corresponding to each trap partition are equal, but the value of all the bits in at least one trap partition is 0. Some of local optimums are shown in Table 1. A simple comparison between first local solution and fifth local solution helps us find the second linkage group and comparison between third local solution and fourth local solution helps us find the first linkage group.

Now, the algorithm can be explained and the example is continued later. In an overall view, there are two phases of search and analysis. In search phase some local optimums are found and in analysis phase the comparisons between these local solutions are done. If number of local solutions is not enough to discover all the linkage groups of the problem, the local solutions for the remained bits of the problem not assigned to a linkage group yet are to be found by the comparison of the newly found local optimums. This process repeats until remained undiscovered linkage groups of the problem are identified. The process will end if all the variables of the problem are assigned to a linkage group. In the search phase, K deterministic hill climbers are initialized randomly and set to search the landscape (with length (Xs) number of variables). When each DHC finds a peak in the landscape and no movements are possible, that solution which is a local optimum will be saved in a set named "HighModals".

After the search phase, analysis phase starts. In the analysis phase, linkage groups should be identified by comparing different local optimum solutions.

A comparison method is needed for the analysis phase. The comparison method should be able to segregate the BBs of the local solutions and yet it should be simple and uncomplicated. XOR operation is a good candidate for this purpose. This is due to the fact that the local and global solutions of a decomposable function are the two strings with the most differences in their appearance and binary strings are used to code the individuals.

Therefore in the analysis phase, each two local optimum solutions in the "HighModals" set are XORed with each other and the results are stored in "XORed" set. Therefore "XORed" is an array of arrays. The strings with least number of ones are found. Number of 1s (r) in these strings is considered the length of linkage group. And these strings (string with length r) are put in the set "DiscoveredBBs" which is an array of arrays and contains the ultimate results (all the identified linkage groups). All of the other members of "XORed" set with more than r 1s are put in the set "XsArray" which is again array of arrays. After identifying some of the linkage groups, the algorithm is recursively called for finding linkage groups in other parts of the string which are not identified yet. The undiscovered parts are the XORed strings which have length more than r or those variables of the problem which are not in the "XORed" set. Therefore those bits in the Xs which are not in the"XORed" set are added as a separate member to the "XsArray" (step A.4 in the algorithm).

As was mentioned before we need a mechanism to balance the time spent in the search phase. For this reason a parameter, sp is contrived which determines when to leave the search phase. Leaving the search phase takes place with the probability sp. If sp is small, the set HighModals will become bigger because the search phase takes longer and as such more local solutions are found. Analysis of huge number of solutions is difficult and unnecessary. On the other hand by comparison of too few local solutions there is a little chance of identifying all the linkage groups of the problem. So sp parameter should be determined wisely considering the length of the problem. If the length of the problem is more, the number of local solutions needed to identify the linkage groups is more. If each variable of the problem is assigned to at least one linkage group, the LOLL algorithm terminates. The pseudo code of LOLL algorithm is shown in Algorithm 1.

Xs is an array with length n containing the indexes of the problem variables. DiscoveredBBs is an array of arrays, containing the discovered linkage groups. Each linkage group is shown with an array containing the indexes of the variables in the linkage group. HighModals is an array containing the local optimums of the problem. XORed is an array of arrays containing the result of XOR operation on local solutions. Each XOR result is shown with an array containing the indexes of bits with values 1 after doing XOR operation. DeterminedBits is an array which contains the indexes of the variables which their corresponding linkage group is identified. XsArray is an array of arrays containing those parts which should be searched again for identification of the remaining linkage groups.

As it is obvious, the only parameter which should be set wisely is sp. In the future work, we address solutions to adjust this parameter automatically. Complexity of the algorithm will be discussed later. Now, we go back to our simple example:

Xs is here the array Xs = {1, 2, …, 12} HighModals set is in Table 1.
XORed set of our simple example: [1,2,3,4,5,6] , [1,2,3,7,8,9],

Table 1. Some of the local optimums for Trap 3 size 12.

1	1	1	0	0	0	0	0	0	1	1	1
0	0	0	1	1	1	1	1	1	0	0	0
0	0	0	1	1	1	0	0	0	1	1	1
1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	1	1	1

[7,8,9,10,11,12] , [4,5,6] , [1,2,3,7,8,9,10,11,12] , [1,2,3]

DiscoveredBBs set so far: [4, 5, and 6], [1, 2, and 3]

DeterminedBits set: [1, 2, 3, 4, 5, and 6]

XsArray: [1,2,3,7,8,9] , [7,8,9,10,11,12] , [1,2,3,7,8,9,10,11,12]

LOLL algorithm is again called for three sub-problems in XsArray. The algorithm could be simplified but it is developed in a general form so that it can also identify the overlapping linkage groups. The experimental results regarding overlapping linkage groups and complexity analysis of them are kept to be reported in our future work.

If we have two or more optimums in each building block, the method has some drawbacks in finding the final DiscoveredBBs. For handling this drawback we use each of these DiscoveredBBs as one cluster. Then as cluster ensemble problem is solved by cutting a minimal number of clusters (hyperedges) using the hypergraph-partitioning algorithm (HGPA) (Strehl and Ghosh 2002).

Complexity analysis of the LOLL algorithm

To analyse the complexity of LOLL algorithm, computational cost of each step of the algorithm is calculated. In the first phase there are two main tasks that should be analyzed. The first one is finding the local solutions, which is done in S.1 step in the LOLL algorithm (Figure 1). The second task is deciding to continue the search or leave the search phase (step S.3).

The local optimum solutions are found by deterministic hill climbers. Each DHC starts from random point in the landscape. In each step, it flips the bit which its flip will cause maximum improvement in the fitness of the string. So maximum number of flips for finding the right bit to flip, to get the maximum improvement is equal to length of the string (which in the first pass is equal to n). The maximum number of steps that take for each DHC to find a local optimum in the worst case is less than number of bits of the string (Again in the first pass is equal to n). Therefore in the worst case it takes n2 tasks for each DHC to reach to a local optimum in the first pass of the algorithm. So the complexity of the hill climber is O (n2).

Expected number of repetition of step S.3 is equal to value of parameter sp. As it is stated before the value of sp is directly related to number of variables of the problem (for example sp = 1/5 * number of bits). Therefore number of repeats of the S.3 loop in the first pass of executing the algorithm is equal to length of the problem n and the search phase complexity is of order n3. Number of XOR operations tasks in the first pass of the algorithm is equal to (number of HighModals). Expected number of high modals (local solutions found in search phase) is proportional to the length of the problem n, so, it can be said that complexity of this step is equal to n3.

So, in the worst case, which is determination of one linkage group in each pass, number of calls to LOLL algorithm is equal to n(n/k) (Equation 1) but worst case never happens. Simply, the worst case is determination of all the local optimums of the problem in one pass which will be of order 2 (n/k) because knowing all the local optimums, all the linkage groups can be discovered in the first pass. We will present the amortized complexity in our future work.

For non-overlapping linkage groups, the algorithm can be simplified and its complexity (worst case) in simplified form would be of order (n3).

T(n) = (n - 1)T(n - k) + n3
= (n - 1)((n - k - 1)T(n - 2k) + (n - k)3) + n3
= (n - 1)((n - k - 1)(n - 2k - 1)T(n - 3k) + (n - 2k)3+
(n - k)3 + n3
…
= (n - 1)(n - k - 1)(n - 2k - 1)…(n – n/k*k) + n3
+ (n - k)3 + (n - 2k)3 + … + (n – n/k*k)3
≈Σn3 + n4 + n5 + … + n(i+3)(i =n/k)
→O(ni)

Complexity analysis by number of fitness evaluations

In different calls of the function LOLL, DHC is executed on different length strings. In the first call of the LOLL, the length of the string is the maximum size which is n. In the following calls of the LOLL the string length decreases at least by k bits.

In the following paragraphs, number of fitness evaluations done by DHC in the first call of the LOLL function is calculated. Fitness evaluations are only done in search phase which is finding the local optimums by a DHC. DHC starts from random point in the landscape. In each step, it flips the bit which its flip will cause maximum improvement in the fitness of the string. So number of flips for finding the right bit to flip, to get the maximum improvement is equal to length of the string (which in the first pass is equal to n). The maximum number of steps that take for each DHC to find a local optimum is different in different cases. Three different cases are explained here.

First case (worst case)

In this case the maximum number of flips should be done to reach to a local optimum. In trap functions based on their fitness function, it happens when the sub-traps tend to reach to the local optimum of all 0s while it has the possible maximum number of 1s which is k - 2 (If a trap has k - 1 or k ones it tends to reach to the global optimum of all 1s). Therefore in this case, having n/k sub-traps in the problem, each sub-trap should go under (k - 1) flips and therefore (k - 2)*n/k = n - 2^n/k flips is necessary to reach to the local optimum of all 0s.

Second case

This case happens when only one flip is needed to reach to a local optimum. It happens when the sub-traps tend to reach to the global optimum of all 1s and have (k - 1) 1s already, or when they tend to reach to the local optimum all 0s and they have (k - 1) 0s already. In this case each sub-trap needs only one flip to reach to the local or global optimum and the total flips for the whole problem to reach to a global or local optimum is n/k.

```
$X_s = (1..n)$;
$r = 0$;
$DiscoveredBBs = LOLL(X_s)$
  Search Phase:
    S.1. Run DHCs;
    S.2. Save local solutions to HighModals set.
    S.3. If $p - exit - search < (sp)$ goto Analysis phase.
    Else goto S.1.
  Analysis Phase:
    A.1. Perform XOR between each two members of
    HighModals set.
    Put the XORed into XORed set.
    A.2. Find the strings with least number of 1s
    in XORed set.
    Set its number of $1s = r$ as length of BB, and put
    those XORed members to set DiscoveredBBs.
    A.3. Delete those XORed members which their length
    is equal to length $X_s$.
    A.4. Put all the indexes of bits of each member of
    XORed set which its length is equal to $r$
    into a set DeterminedBits.
    A.5. for each member of XORed set $i$,
    if $((length(i) > r) \wedge$ all of of bits of $i$ are not
    in DeterminedBits)
          Put that in the set XsArray;
    A.6. Put the ($X_s$-(indexes of all the bits with value
    1 in the XORed set)) into XsArray;
    A.7. If all of the variables of the problem $\in$ DeterminedBits
    (all the variables are assigned to linkage group
    (at least)) terminates.
    Else
      for i:1 to length XsArray do
        $DiscoveredBBs = LOLL(XsArray[i])$;
```

Figure 1. Local optimum based linkage learning algotithm.

Third case (best case)

In this case, the random start point of the DHC is it-self a local optimum, so no flip is need. As it was stated in the above paragraph, DHC in each step evaluate the fitness of each candidate solution exactly n times to decide which bit is to flip.

So in the above three cases, number of fitness evaluations are respectively n * (n - 2n)/k, n^2/k and n. The above three explained cases have different probabilities of occurrence. In the first case the probability of event is equal to (C (k, k - 2)/2^k)$^{n/k}$ = ((k2 - k)/$^{2k+1}$)$^{n/k}$. The probability of second case is (2*C (k, k - 1)2^k)$^{n/k}$ = (2k/2^k)$^{n/k}$, probability of the third case is $2^{n/k}$/2^n. But how many local optimums are needed to find all the linkage groups? The best case which fortunately is the most probable one, happens when the local optimum with all 0s (the most probable one) and the local optimums which all of their sub-traps are all 0s but one of the sub-traps is all 1s (the second most probable optimum), are found in the first call of the LOLL. In this case n/(k + 1) strings are needed to be found by DHC. Number of fitness evaluations is at most (n/k*(n^2 - 2n^2/k - kn + 3n)) + (n - 2n/k). This case is the most probable one, because the local optimums seen in this case are the ones with the most probability of happening among all the other possible events. The probability of the local optimum of all 0s is ((2^k – k)/2^k)$^{n/k}$ and the probability of the local optimums with all but one the sub-traps of 0s and the other one of all 1s is (k/2^k)((2^k – k)/2^k)$^{n/k-1}$. So in overall, number of fitness evaluations (at most) is of order O (n^3).

RESULTS

For all tested problems, 30 independent runs are performed and our approach is required to find all the linkage groups accurately in all the 30 runs. The performance of LOLL is measured by the average number of fitness evaluations until it terminates. The results are summarized in the Figures 2 to 4. All these results are obtained without applying the HMETIS. As it is claimed, the finding of building blocks is subquadratic either in non-overlapping challenging problems, or in overlapping ones. It is worthy to note that the time order of the algorithm in the challenging problems increases as the size of building blocks increases no matter it is over-lapping functions. This is very important result, because as the size of building blocks in the bayesian optimization

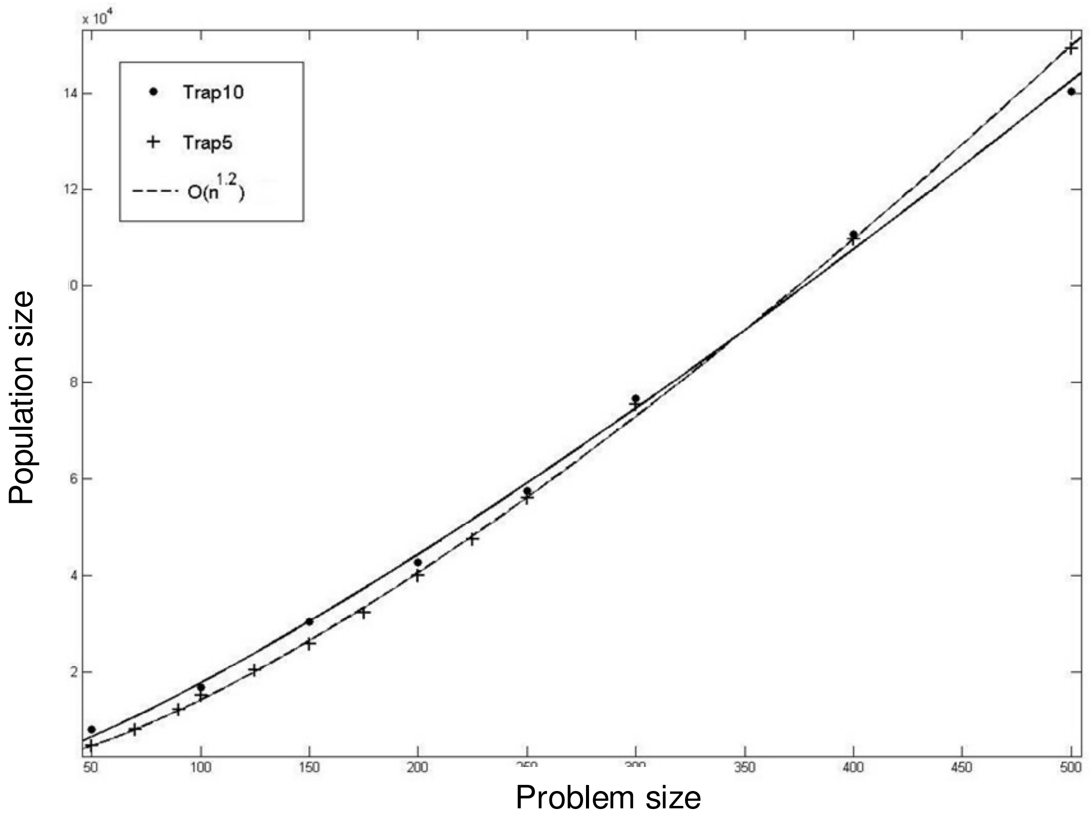

Figure 2. Number of fitness evaluations against problem size for trap5 and trap10.

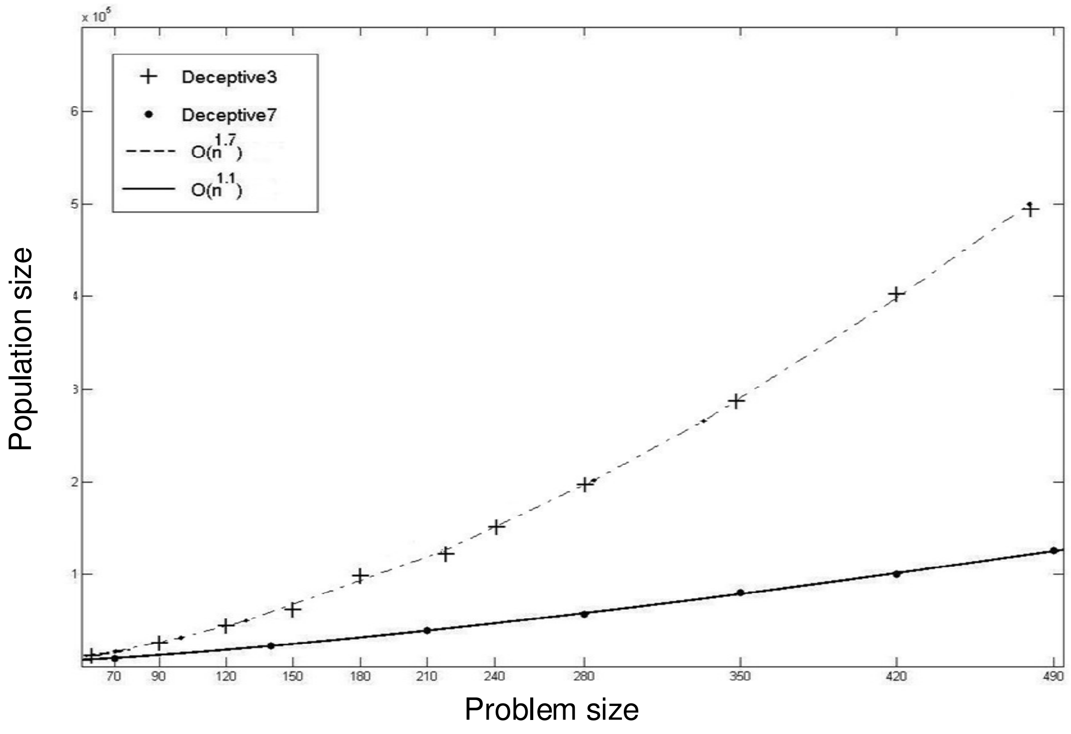

Figure 3. Number of fitness evaluations against problem size for deceptive 3 and deceptive 7

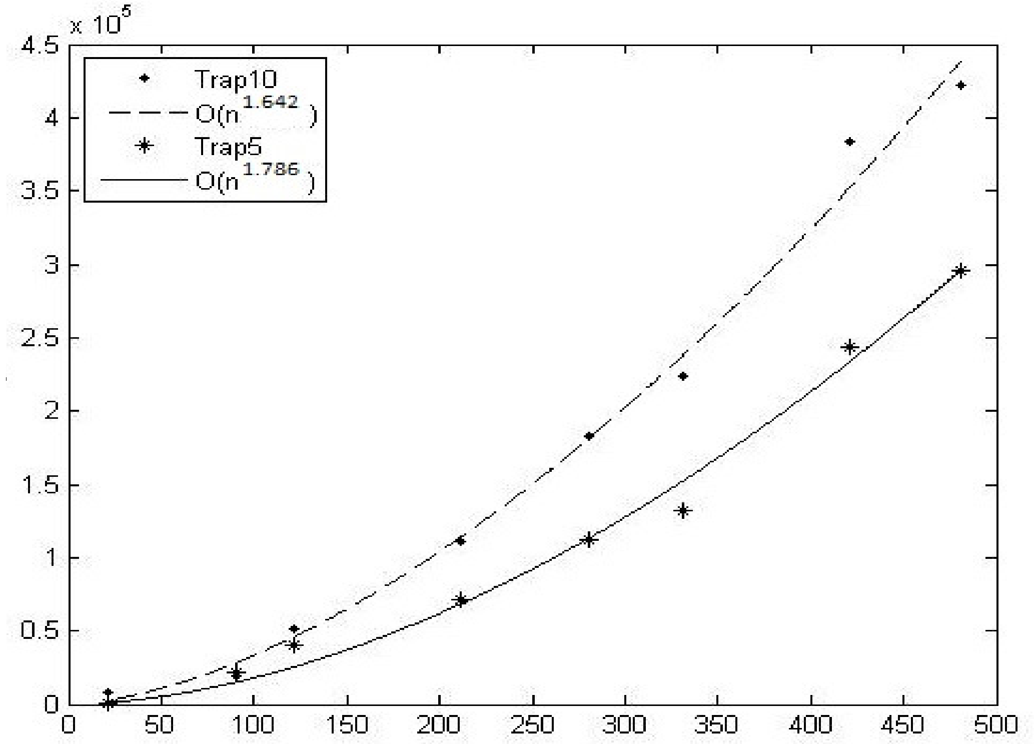

Figure 4. Number of fitness evaluations against problem size one-bit overlap trap5 and trap10

algorithms (BOA) and in the hierarchical bayesian optimization algorithms (HBOA), the times orders of these algorithms increase exponentially (Strehl and Ghosh, 2002).

Conclusion

With the purpose of learning the linkages in the complex problem a novel approach is proposed. There are other approaches that are claimed to be able to solve those challenging problems in tractable polynomial time. But the proposed approach does not classified into the existence categories. This work has looked at the problem from whole different points of view. Our method is based on some properties of additively decomposable problems in order to identify the linkage groups. The amazing property of additively decomposable problems that our method is based on is the special form of their local optimums which a bunch of them would give us lots of information about the linkage groups. The proposed algorithm is called local optimum based linkage learner (LOLL). The algorithm is capable of solving the challenging problems effectively.

LOLL is capable of identifying the linkage groups in a simple and straightforward manner. As it is shown in terms of numbers of fitness evaluation the complexity of LOLL has been O ($n^{1.2}$) in the two test cases over a trap problem and O ($n^{1.7}$) and O ($n^{1.1}$) in deceptive 3 and deceptive 7 problems. Moreover we believe that the proposed algorithm (without any major changes) is capable of finding the overlapping building blocks. The result testing the proposed approach on overlapping problems and more detailed analysis of the algorithm will be represented in our future work. Analyzing the proposed algorithm in the context of optimization problem and along with an optimization search is one of the tasks that can be done as future works. Comparing the results with the other approaches is also left as future work.

REFERENCES

Audebert P, Hapiot P (1993). Electrochemical Evidence of pi-Dimerization with Short Thiophene Oligomers. J. Electroanal., 6(9): 1549–1555.
Hillman AR (1987). In: R.G. Linford (Ed.), Electrochemical Science and Technology of Polymers, vol. 1, Elsevier, Amsterdam, Ch. 5.
Miller B (1984). Proc. 6th Australian Electrochem. Conf., Geelong, Vic., 19-24 Feb., J. Electroanal. Chem., 86- 91.
Newman J (1991). Electrochemical Systems, 2nd ed., Prentice-Hall, Englewood Cliffs, NJ.
Pelikan M (2005). Hierarchical Bayesian optimization algorithm: Toward a new generation of evolutionary algorithms, Springer.
Strehl A, Ghosh J (2002). Cluster Ensembles A Knowledge Reuse Framework for Combining Multiple Partitions, J. Machine Learning Res., 3: 583-617.