Direct and Explicit Building Blocks Identification and Composition Algorithm

Chalermsub Sangkavichitr and Prabhas Chongstitvatana

Abstract— **This paper proposes a new algorithm to identify and compose building blocks based on minimum mutual information criterion. Building blocks are interpreted as common subsequences between good individuals. The proposed algorithm can extract building blocks in population explicitly. The additively decomposable problems and hierarchical decomposable problems are used to validate the algorithm. The results are compared with Bayesian Optimization Algorithm, Hierarchical Bayesian Optimization Algorithm, and Chi-square Matrix. This proposed algorithm is simple, easy to tune and fast.**

I. Introduction

IN Genetic Algorithms (GAs), the solutions are improved based on the assumption that the good solutions have some common substructures and if they can be combined together correctly, the new solutions will be better [1,2]. The schema theorem tells us about the properties of the good common substructure which is "short, low-order, highly fit" or so-called the building blocks (BBs). This leads to the building block hypothesis (BBH) that is used to explain the mechanism of GAs [1,2]. The crossover operation was claimed that it works as BBs construction more than BBs disruption [3]. Generally, one-point crossover is enough to solve the problems. However there are many crossover methods introduced such as two-point crossover, multiple-point crossover, uniform crossover, etc., they work well for different problems. The mutation operation was claimed that it provides the new source of genetic material [1]. The mutation rate is difficult to determine. We only know that it should be low, conversely the crossover rate should be very high. GAs are quite simple to use but there are many parameters that must be tuned.

There are many ways to identify building blocks; most of them are in the field of estimation distribution algorithms (EDAs) [4,5]. The main concept of EDA is sharing knowledge through a model of distribution of population and sampling the new generation from the model. However most of them need some knowledge to identify relationship between individuals in a population and to build a model.

In this paper, we propose a new and simple method that can identify and compose the BBs explicitly. The schema theorem showed that the good schemata existed in highly fit individuals (fitness above average). We can extract the sharing knowledge using the information theory. If we consider any pair of individuals as two variables, we can identify the degree of dependence through the mutual information [6]. This can be measured with the similarity of the physical structure (genotype). We define the good substructures as common substructures between good individual. This common substructure between any two good chromosomes is the mutual information and we name it "fragment". According to the BBH, fragments are the BBs because they can be recombined together into a higher order [7,8]. The proposed method called “Building Blocks Identification and Composition" algorithm (BBIC) consists of two main parts: the fragment identification and the fragment composition. The fragment identification extracts the good common substructures among good individuals and the fragment composition assembles the fragments to create the new offspring.

The main concept of the proposed method is to capture and to combine the sharing knowledge between good individuals. The main advantage of the algorithm is the simplicity. It requires only one parameter (population-size) for tuning. The hard problems [9] were used as the benchmark problems and the results showed its efficiency comparing with Bayesian Optimization Algorithm (BOA) [10], Hierarchical Bayesian Optimization Algorithm (hBOA) [11] and Chi-square Matrix (CSM) [12].

The paper is organized as follows. The fragment definition is presented in the Section II. Section III describes two main parts of the proposed algorithm: fragment identification and fragment composition. The experiment and benchmark results are shown in Section IV. Finally, the concluding remarks of the proposed method are in Section V.

II. Fragment

The information theory told us about how to measure the information from the data. In GAs, each chromosome holds some information about the solution. We hold the belief that good solutions can guide us to the desired solution because they contain some useful information. The problem is how to extract such information. We can observe the common knowledge between good solutions from the mutual information. In the case of two chromosomes, we consider the similarity of bits in the same position as mutual information between them. Although there are many different genotypes that have the same fitness value but if

C. Sangkavichitr and P. Chongstitvatana are with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Thailand (e-mail: penockio@gmail.com and prabhas@chula.ac.th).

we have enough a number of good solutions, the problem will be solved because there will be some repeat pattern between them. This increases the chance to find common subsequence and maintains diversity of common patterns.

We will define the fragment as common substructures of any two chromosomes. Given sequence of chromosomes length l

$$C^k = c_1^k c_2^k \ldots c_l^k : k \in \mathrm{I}^+, c_n^k \in \{0,1\}, 1 \le n \le l \qquad (1)$$

Given set of position indices in chromosome length l from position f to position t

$$(f_i, t_i) = \left\{ z_i : 1 \le f_i \le z_i \le t_i \le l,\ f_i, t_i, z_i \in \mathrm{I}^+ \right\} \quad \text{and } \forall i, j \in \mathrm{I}^+ : i \ne j, (f_i, t_i) \cap (f_j, t_j) = \phi \qquad (2)$$

Given common subsequence between two chromosomes C^{k_1} and C^{k_2}

$$F_i^{k_1,k_2} = c_{f_i}^{k_1} c_{f_i+1}^{k_1} \ldots c_{t_i}^{k_1} \text{ such that } c_{z_i}^{k_1} = c_{z_i}^{k_2} \qquad (3)$$

The set of contiguous common subsequences between two chromosomes or the fragments are defined as:

$$S^{k_1,k_2} = \left\{ F_i^{k_1,k_2} : i \le \left\lceil \frac{l}{2} \right\rceil \right\} \qquad (4)$$

The definition above will be illustrated by an example in Fig.1. Note that the string is indexed from left to right started with the index 1. Given two 10-bit chromosome sequences C^1= (1,0,1,1,0,1,0,1,0,1) and C^2= (1,1,1,1,0,0,0,0,0,1), then the common index ranges of C^1 and C^2 are (f_1, t_1)= (1), (f_2, t_2)= (3,5), (f_3, t_3)= (7), and (f_4, t_4)= (9,10) and then the fragments of the template $S^{1,2}$ are $F_1^{1,2} = (c_1^1)$= (1), $F_2^{1,2} = (c_3^1, c_4^1, c_5^1)$= (1,1,0), $F_3^{1,2} = (c_7^1)$= (0), and $F_4^{1,2} = (c_9^1, c_{10}^1)$= (0,1) consecutively.

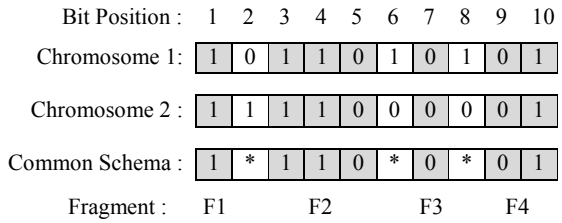

Fig. 1. The common subsequences between two chromosomes (fragments).

The common schema defines as similarity between two chromosomes. The fragment can be interpreted as contiguous common subsequences or common subschema between two chromosomes. The length of each fragment must be longer or equal to one bit. The order and length of the fragment are defined similar to GAs. The order of schema is the number of fixed bit position and the fragment composed of all fixed bit position, thus the size of fragment is the order e.g. in the Fig.1 F2 = (1,1,0) so order(F2) = size(F2) = 3. The length of schema is the distance between the first fixed bit position to the last fixed bit position e.g. in Fig.1 F2 beginning at position 3 and ending at position 5 so length(F2) = 5-3 = 2. The definition of fragment shares the BBs properties that the short and low-order schemata have high potential to survive and then will be recombined to create better solutions. The fragments can be considered as independent contiguous building blocks that have tightly linkage on each fragment.

III. Fragment Identification and Fragment Composition

In this section, we will describe in details of the proposed algorithm. The BBIC consists of two main parts: fragment identification and fragment composition. The fragment identification extracts the fragments from the selected individuals (good solutions) and the fragment composition recombines the fragments into new individuals (new offspring). The algorithm is illustrated in Fig.2 and Fig.5. Firstly, the good solutions are selected from the population. Related to the schema theorem and BBH, the opportunity of schemata survival will increase in chromosome that has fitness above average. Thus the upper half (above average) individuals should be selected. Secondly (Fig.3), the fragments are extracted using the minimum mutual information criterion mentioned previously. In this step, every individuals will be collated together to explore every possible fragments. Thirdly, all fragments will be labeled by their position in the original individual for example in Fig.1 the fragment F2 begin at position 3 in the chromosome. Fourthly (Fig.4), the fragments are selected to build new offspring one by one. There are many ways to compose the fragments however a simple method suffices. The fragments are combined in sequence from the first position to the last position. There are many fragments that begin with the same position. If there is no guideline to tell that which one is better, a random selection seems to be a useful solution because it avoids deceptive biases. When the first fragment is selected from the first position the next fragment is concatenated to the prior fragment and the process repeats until complete a chromosome. If there is no fragment starting at any position, bit value will be drawn from a selected individual because this value comes from a good chromosome. This may occur if the diversity in the population is too low due to lacking of mutual information or the population is too small.

The time complexity of fragment identification and fragment composition are $O(n^2 \cdot l)$ and $O(n)$ respectively where l is the chromosome length and n is the number of selected chromosomes. The proposed algorithm uses only

D denotes a population.
F denotes common subsequence (fragments).
T denotes an archive of common subsequences.

Algorithm Fragmentation

D_0 ← Generate *R* individuals (the initial population) at random

Repeat for $g = 1, 2, \ldots$, until the stopping criterion is met

D_{g-1}^N ← Select *N* from D_{g-1} using truncation selection (Upper half good solutions)

F ← Find the common substructures (fragments) among D_{g-1}^N

T ← Label *F* and record

D_g^N ← Sample R_g individuals (the new population) from *T*

EndRepeat
End.

Fig. 2. Pseudocode for BBIC algorithm

R denotes an individual (chromosome).
F denotes common subquences (fragments).

Algorithm Fragment_Identification
{The comparison between individual R_i to $R_1, \ldots, R_{i-1}$}

For *i* = 2 to (population_size / 2) **do**
For *j* = 1 to (***i***-1) **do**
F ← Compare R_i to R_j (search for common substructure)
EndFor
EndFor
End.

Fig. 3. Pseudocode for Fragment Identification

R denotes an individual (chromosome).
F_i denotes set of common subsequences (fragments) begin with bit at position *i*.
C denotes a fragment

Algorithm Fragment_Composition
{The composition of new individual *R*}

i = 1 {first position}
While *i* < individual_size **do**
if $F_i \neq \emptyset$ then {if has a fragment or more}
C ← *random_select*(F_i)
R ← *R* + *C* {concatenation}
i ← *i* + *size*(*C*) {assign next position}
else {if there is no member}
R ← *random*(0,1) {random from 0 or 1}
i ← *i* + 1
endif
EndWhile
End.

Fig. 4. Pseudocode for Fragment Composition

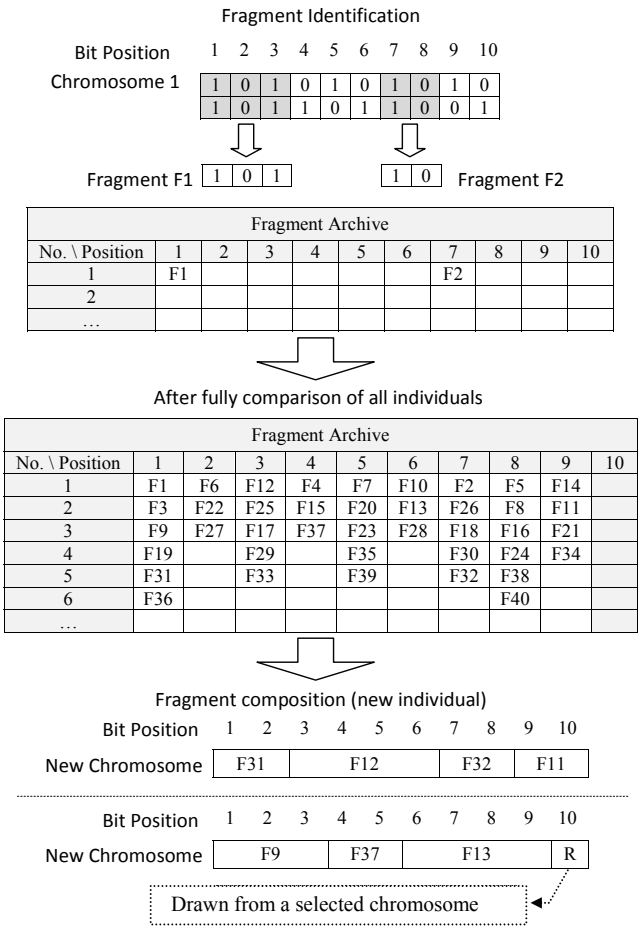

Fig. 5. The BBIC algorithm

one parameter that is population size. So it is very easy to tune compared to other GAs or EDAs algorithms.

The mechanism behind the BBIC can be related to exploitation and exploration [13]. In GAs, it is important to balance them to achieve good results. In canonical GAs, selection is interpreted as exploitation, crossover and mutation are regarded as exploration. For the BBIC, fragment identification captures the mutual information (common substructure) between selected chromosomes as fragments. This part provides exploitation power to extract the good fragments that may be parts of a desired solution. The fragment composition is considered as exploration because it recombines the fragments into all possible patterns without bias. In Fig.5, we can notice that the number of fragment of each position in the archive is different. Normally, the first position has a lot of members because the fragment identification proceeds from left to right. Therefore the next positions from the first have less members. The number of members is increasing in the middle and decreasing in the end. In the early generation, the fragments will be short because it is hard to exploit common information in a highly diverse population. The proposed algorithm keeps mutual information between every chromosome. Moreover it needs to observe enough good

solutions to capture good common substructures. The required population size depends on the problem. During evolution process, the solution distribution is converged gradually to an optimal point. The fragment will be longer because the mutual information is growing.

IV. Benchmarks and Performance

In this section, we explain the test problems and show the experimental results. The test problems can be separated into two classes: additively decomposable problems (ADFs) and hierarchical decomposable problems (HDFs). In general the hierarchical structure is harder to solve than additive structure however it also depends on a particular algorithm. It is hard to claim that what algorithm is suitable for a particular class of problem. Nevertheless the experimental results can be conducted to support the statement.

There are four test problems in this experiment as follows:

OneMax problem is the general widely-use to measure the based performance of GAs to achieve the optimal solution that composed of all one. This problem is the representative of simple problems.

The well-known trap functions are designed for studying the BBs and linkage problems in GAs [14]. The general k-bit trap functions are defined as:

$$F_k(b_1,\ldots,b_k)=\begin{cases} f_{high} & ;\text{ if } u=k \\ f_{low}-\left((u\times f_{low})/(k-1)\right) & ;\text{ otherwise}\end{cases} \quad (4)$$

Where b_i is in $\{0,1\}$, $u=\sum_{i=1}^{k} b_i$ and $f_{high} > f_{low}$. Usually, f_{high} is set at k and f_{low} is set at k-1. The additively decomposable functions, denoted by $F_{m\times k}$ are defined as:

$$F_{m\times k}(k_1\ldots k_m)=\sum_{i=1}^{m} F_k(k_i), k_i\in\{0,1\}^k \quad (5)$$

The m and k are varied to produce a number of test functions. The Trap functions fool the gradient-based optimizers to favor zeros, but the optimal solution is composed of all ones.

The commonly used hierarchical decomposable functions are hierarchical if-and-only-if (HIFF) [15] and hierarchical trap-1 (HTrap-1) [16] functions. The HIFF function is represented as a binary tree. An example is shown in Figure 6 (left). There are three possible values "0", "1" and "-" for each node in the tree and they are interpreted according to the definition as follows:

$$c_i=\begin{cases} 2^i & ;\text{ if node } i \text{ is "0" or "1"} \\ 0 & ;\text{ if node } i \text{ is "-"}\end{cases} \quad (6)$$

Where l is the level (height) of node i and c_i is the fitness value of node i.

The value of parent node depend on their children node, if the children node value is all "0" or all "1", the parent node value is "0" or "1" respectively, and if children value are different the parent node value will be "-". The HIFF functions denote by $Hiff(X)$ is defined as

$$Hiff(X)=\sum_{l=0}^{h}\sum_{i=1}^{2^{(h-l)}} c_i^l \quad (7)$$

Where h is the height of the binary tree and c_i^l is the value of node i in level l. The optimal solutions of this problem are all zeroes and all ones.

The hierarchical trap (HTrap-1) function evaluates a solution as a tree that the number of branches is greater than two. An example is show in Figure 6 (right). The left nodes will not be evaluated in the function. The evaluation rules are the same as HIFF function. The fitness value of each node is defined as:

$$c_i=\begin{cases} 3^l\times F_3(b_1,b_2,b_3) & ;\text{ if } b_j \text{ "-" for all } 1\le j\le 3 \\ 0 & ;\text{ otherwise}\end{cases} \quad (8)$$

Where l is the level (height) of node i and b_1, b_2, b_3 are the children of node i from left to right respectively. At the root node, the parameters of trap function are f'_{high} = 1 and f'_{low} = 0.9. The other nodes use f'_{high} = 1 and f'_{low} = 1. The Htrap-1 functions denote by $Htrap(X)$ is defined as

$$Htrap(X)=\sum_{l=1}^{h}\sum_{i=1}^{3^{(h-l)}} c_i^l \quad (9)$$

The optimal solution is composed of all ones.

All five benchmark problems are tested with the sizes (in bit) as follows:

OneMax: 100, 150, 200 and 250
Trap-3: 60, 120, 180 and 240
Trap-5: 100, 150, 200 and 250
HIFF: 32, 64, 128 and 256
Htrap-1: 27, 81 and 243

The benchmark problems are designed to test the performance of algorithm to detect and to compose BBs. There are many powerful algorithms solving BBs identification problems. The BOA is a powerful algorithm to solve ADFs problems that uses Bayesian network to learn and to identify BBs relationship. Later it is developed to hBOA for solving HDFs problems efficiently. The CSM algorithm uses chi-square matrix to find the dependency structure and can solve both ADFs and HDFs problem effectively. The CSM claimed to consume less time and

Fig. 6. An example of HIFF problem is represented as a binary tree (left), and an example of HTrap-1 is represented as a 3-branch tree (right)

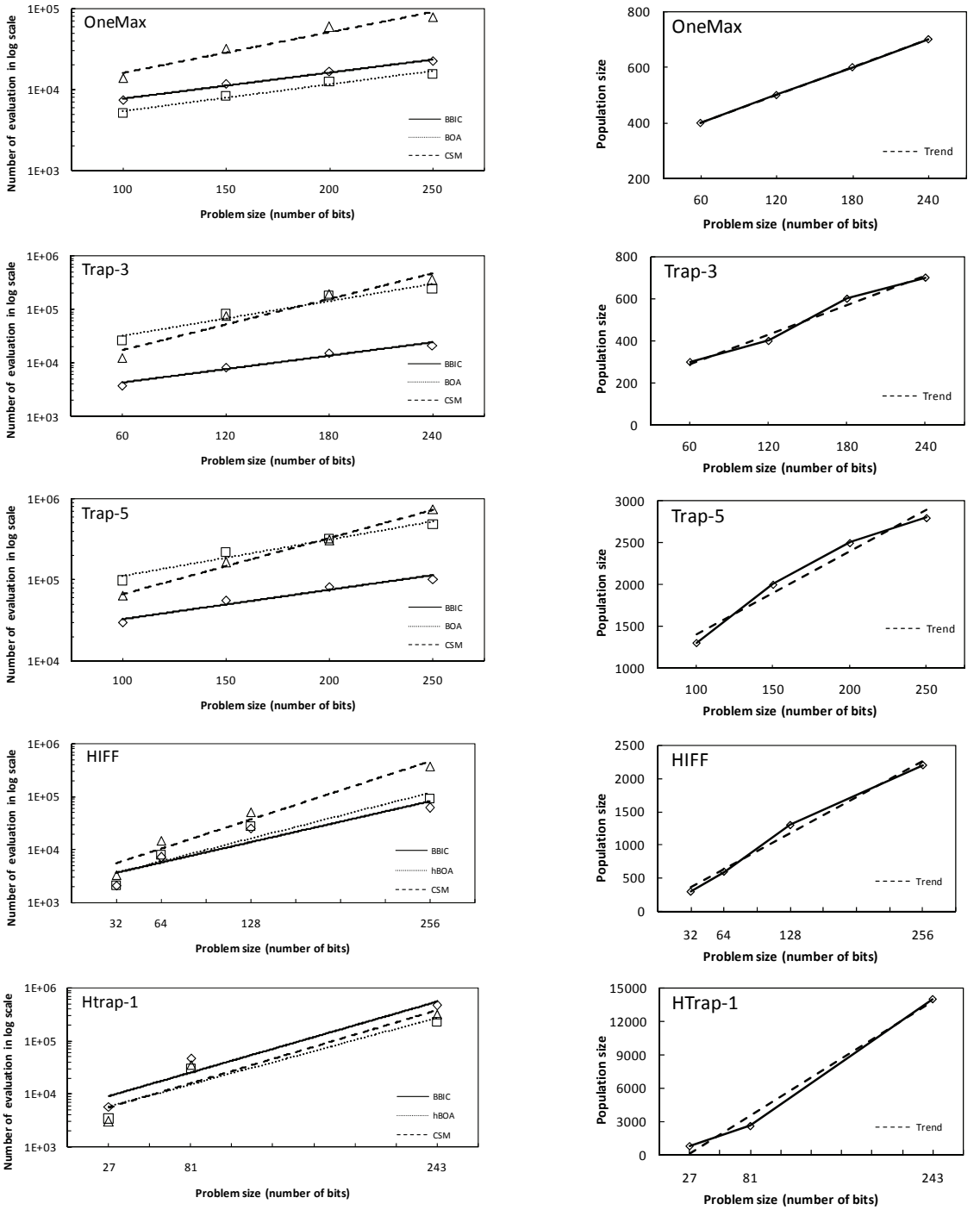

Fig. 7. Performance comparison between BOA, hBOA, CSM and BBIC (left), and Population size using in BBIC method (right)

memory usage than the BOA and the hBOA. However the suitable threshold value for partitioning in CSM varies according to problems. The time complexity of BBIC is lower than CSM and it uses only one parameter for tuning (population size). The performance of the proposed algorithm is shown in Fig. 7

All tested problems are performed with 30 independent runs and it is required to find the optimal solution in all of 30 runs. The results as shown in Figure 7 (left) are the average number of function evaluation of all runs compares with the BOA, hBOA and the CSM. The minimum population size uses to achieve the optimum in 30 runs is shown in Figure 7 (right). The results show that the BBIC outperforms the BOA, the hBOA and the CSM in the Trap-3, the Trap-5 and the HIFF problems. For the OneMax problem, the result is inferior to the BOA but outperforms the CSM. For the Htrap-1 problem, the result is inferior to the hBOA and the CSM. The population size for all problems grows as linear relationship. The efficiency of the BBIC on the tested problems ordered from high to low is as follows: OneMax, Trap-3, HIFF, Trap-5 and HTrap-1.

The fragment is defined as common subsequences between two chromosomes and is interpreted as contiguous building blocks. In Fig.7, we can notice that the performance of BBIC in OneMax problem is close to Trap-3 problem. In the OneMax problems, it is hard to say that it has BBs. If we consider BBs in OneMax problem as one allele, it will have a quantity of BBs equal to the problem size. Although the OneMax is not the deceptive problem, combining the amount of BBs seems to be more complicated in the higher problem dimension. The difficulty of Trap problem depends on the size and the number of BBs. For the hierarchical problem, the HIFF problem is not deceptive; however there are two opposite optimal solutions (all ones and all zeroes). This conflict disrupts each other to achieve an optimal solution. The result of HIFF problem shows that it is easier than Trap-5 problem but harder than Trap-3 problems. The BBIC performs worst in the HTrap-1 problem because the problem combines a Trap problem with a hierarchical problem.

I. Conclusion

The good substructures can be interpreted as common subsequences between two chromosomes called fragments. We can regard these fragments as BBs because they are short, low-order and come from the highly-fit above average chromosomes. Therefore the building block can be identified explicitly. The results indicate that there are good substructures existed in good individuals and the BBIC can identify and recombine them to create better solutions. This shows the existing of the BBs and supports the understanding of the mechanism in evolutionary process based on BBH. The time complexity of fragment identification and composition are $O(n^2 \cdot l)$ and $O(n)$ respectively where l is the chromosome length and n is the number of selected chromosomes. This is far lower than the BOA, the hBOA and the CSM. The proposed method is very simple for implementation and tuning. It is also efficient to solve hard problems. We want to eliminate the bias in fragments recombination process so the uniform random is chosen. However the extra knowledge can be integrated in the part of fragment composition to tell that which fragment is fitter or how are they related.

References

[1] David E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison Wesley, 1989.

[2] D. E. Goldberg, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, 2002.

[3] M. Mitchell., S. Forrest and J. H. Holland, "The RoyalRoad for genetic algorithms: Fitness landscapes and GA performance", *Proc. of the First European Conf. on Artificial Life*. 1992.

[4] P. Larrañaga, J. A. Lozano, *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.* Kluwer Academic Publishers, 2002.

[5] M. Pelikan, K. Sastry, E. Cantú-Paz, *Scalable Optimization via Probabilistic Modeling*, Springer, 2006.

[6] Dong-il Seo, Yong-Hyuk Kim, and Byung-Ro Moon, "New Entropy-Based Measures of Gene Significance and Epistasis", *Genetic and Evolutionary Computation Conference- Lecture Notes in Computer Science 2724*, 2003, pp. 1345-1356.

[7] C. R. Stephens, H. Waelbroeck, "Schemata Evolution and Building Blocks", *IEEE Congress on Evolutionary Computation*, 1999, pp. 109-124.

[8] C. R. Stephens, H. Waelbroeck, R. Aguirre, "Schemata as Building Blocks: Does Size Matter?" *Foundations of Genetic Algorithms 5 (FOGA)*, June, 1999.

[9] F. Stephanie, M. Mitchell, "What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation." *Machine Learning 13 (2-3)*, 1993, pp. 285-319.

[10] M. Pelikan, D. E. Goldberg & E. Erick Cantu-Paz. "BOA: The Bayesian optimization algorithm." *Proc. of the Genetic and Evolutionary Computation Conf. (GECCO)*, 1999, pp. 525-532.

[11] M. Pelikan, "Bayesian optimization algorithm: From single level to hierarchy." Doctoral dissertation, University of Illinois at Urbana-Champaign, 2002.

[12] C. Aporntewan and P. Chongstitvatana, "Chi-Square matrix: an approach for building-block identification", M.J. *Maher (Ed.): 9th Asian Computing Science Conference*, 2004, pp. 63-77.

[13] A. E. Eiben , C. A. Schippers, "On evolutionary exploration and exploitation", *Fundamenta Informaticae*, 1998, pp. 35-50.

[14] G. R. Harik, "Learning linkage", *Foundation of Genetic Algorithms 4*, 1997, pp. 247–262

[15] R. A. Watson, G. S. Hornby and J. B. Pollack, "Modeling Building Block Interdependency". *Proc. of Parallel Problem Solving from Nature V (PPSN V)*, 1998, pp. 97-106.

[16] R. A. Watson and J. B. Pollack, "Hierarchically consistent test problems for genetic algorithms". *Proc. of Congress on Evolutionary Computation*, 1999.