# Modeling Dependencies of Loci with String Classification According to Fitness Differences

Miwako Tsuji, Masaharu Munetomo, and Kiyoshi Akama

Hokkaido University, North 11, West 5, Sapporo, 060-0811 Japan.
{m_tsuji,munetomo,akama}@cims.hokudai.ac.jp

**Abstract.** Genetic Algorithms perform crossovers effectively when we can identify a set of loci tightly linked to form a building block. Several methods have been proposed to detect such linkage. Linkage identification methods investigate fitness differences by perturbations of gene values and EDAs estimate the distribution of promising strings. In this paper, we propose a novel approach combining both of them, which detects dependencies of loci by estimating the distribution of strings classified according to fitness differences. The proposed algorithm called the Dependency Detection for Distribution Derived from *df* (DDDDD or $D^5$) can detect dependencies of a problem which is difficult for EDAs requiring lower computation cost than linkage identifications.

## 1 Introduction

According to the building block hypothesis [6], genetic algorithms (GAs) implicitly decompose a problem into sub-problems by processing building blocks. Therefore proper building block mixing is essential for GAs. However, the exchange operators of traditional GAs such as one- or two- point crossovers depend on the order of variables on a string. If variables of a same sub-problem are encoded loosely on a string, the crossover operators disrupt building blocks — tight linkage is necessary for effective genetic recombination. A set of loci tightly linked to form a building block is called a linkage set. If such set of loci is known, GA can perform mixing more efficiently.

In order to ensure appropriate mixing, several methods have been proposed. Most of them are categorized as:

1. Linkage Identification Methods
2. Estimation of Distribution Algorithms (EDAs)

The Algorithms classified in the first category examine fitness difference by perturbations in loci to detect dependency. For example, the Gene Expression Messy GA (GEMGA) [7] records fitness differences by perturbation of every locus for strings in population and detects relations of genes according to possibilities that the loci construct local optima. The Linkage Identification by Nonlinearity Check (LINC) [11] identifies linkage by detecting the second order nonlinearity. It assumes that nonlinearity must exist within loci to form a building block. If

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 246–257, 2004.
© Springer-Verlag Berlin Heidelberg 2004

a fitness difference by simultaneous perturbations at a pair of loci is equal to the sum of fitness differences by perturbations at each locus in the pair, these loci can be optimized separately; otherwise, they are considered to be linked. Heckendorn et. al. [5] generalized this category through a Walsh analysis.

In these algorithms, the amount of fitness difference depends only on a sub-problem including the perturbed locus and contributions from the other sub-solutions are canceled by subtracting fitness of perturbed string from fitness of original string. Therefore their abilities are not affected by the difference of fitness contribution of each sub-solutions. In addition, they get to know problem structure before they start searching, and so their search procedures themselves are generally efficient. However, because most of them need to evaluate fitness differences by pairwise or more perturbations, they require $O(l^2)$ or more fitness evaluations where $l$ is a string length.

For the second category, EDAs construct a probabilistic model of promising solutions and use that model to generate better solutions. Early EDAs such as the Population Based Incremental Learning Algorithm (PBIL) [14] and the Compact Genetic Algorithm (CGA) [4], evaluate distribution of gene value in every locus independently and assume no dependency of variables. Subsequent works such as the Mutual Information Maximization for Input Clustering (MIMIC) [1], the Factorized Distribution Algorithm (FDA) [8] and the Bayesian Optimization Algorithm (BOA) [13] exploit conditional probabilities to encode dependency of variables in their models.

Because EDAs need no additional fitness evaluation for their modeling processes, the computational cost for fitness evaluations on problems constructed of sub-problems having uniformly scaled fitness contribution is lower than the methods in the first category. However, for problems constructed of sub-problems having various fitness contribution, sub-problems which give small fitness contribution can not be modeled appropriately. This is because such sub-problems can be selected in promising sub-population only if they come with sub-problems with large fitness contribution.

In this paper, we propose a novel approach hinted from both EDAs and linkage identifications in order to detect dependencies. The proposed approach can detect dependencies of problems which are difficult for EDAs with less computational cost than linkage identifications. Although the proposed approach estimates sub-population like EDAs, the sub-population is selected according to fitness differences by perturbations instead of absolute fitness values. Because the fitness difference depend only on a sub-problem including the perturbed locus, distribution of the sub-population has information of the sub-problem. Just as linkage identifications using perturbations are not greatly affected by the scaling of fitness contribution of each sub-function, the proposed approach can detect sub-solutions even if they give a small contribution to the whole fitness value. The proposed approach can detect dependencies accurately by quasi-linear number of fitness evaluations for string length.

We introduce our new method called the Dependency Detection for Distribution Derived from *df* (DDDDD or $D^5$) and its mechanism is explained in

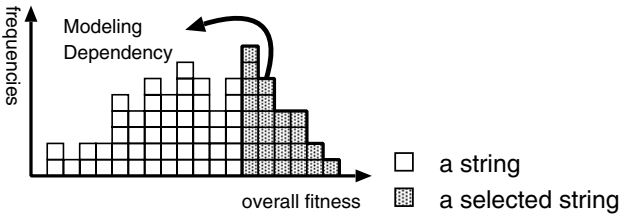


**Fig. 1.** Existing EDAs

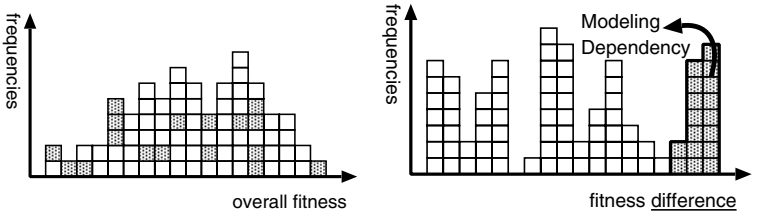


**Fig. 2.** Strings selected for modeling in $D^5$

section 2. The results of experiment are shown in section 3. The conclusion and discussion are mentioned in section 4.

## 2 Modeling Dependencies from Distributions of Strings Classified According to Fitness Differences

Existing EDAs sometimes fail to learn models for sub-problems having little contribution to overall fitness value. This is because they estimate sub-population selected according to the overall fitness value (Fig. 1) and such small contributions can not affect the overall fitness value. If EDAs try to capture such sub-problems, they need to prepar larger number of individuals. For example, the BOA needs $O(l^{1.55})$ if sub-functions are scaled uniformly but it requires $O(l^2)$ if sub-functions are scaled exponentially [12]. On the other hand, linkage identification methods using fitness differences can identify linkages for sub-problems which have small contribution, because the fitness differences for a sub-problem are not affected by the other sub-problems. However, in contrast to EDAs, which need no additional fitness evaluation for modeling, most linkage identification methods need relatively large number of fitness evaluations, $O(l^2)$ where $l$ is string length, to check interdependency of a pair of loci even when a problem is scaled uniformly.

For each locus $i$, the $D^5$ calculates fitness difference by a perturbation at locus $i$, classifies strings according to the difference, and estimates classified strings in order to detect dependency. Because the $D^5$ selects sub-population according to fitness difference (Fig.2 right), even a sub-problem having only a small contribution can be captured. Although it needs additional fitness evaluations

1. initialize population with $n$ strings
2. for each locus $i$
   a) calculate fitness difference $df_i(s^p)$ by a perturbation at locus $i$ in string $s^p$ $(p = 1, 2, \cdots, n)$
   b) classify strings according to their fitness differences into sub-populations (see Fig. 4 for detail)
   c) estimate sub-populations and construct linkage sets (see Fig. 5 for detail)

**Fig. 3.** Overall Algorithm of linkage identification in the $D^5$

to investigate fitness differences, required difference is by a single perturbation. Therefore, the number of evaluations is quasi linear for $l$, which is considerably less than that of linkage identifications $O(l^2)$ especially for large $l$.

In the followings, we show the detail of the algorithm and its mechanism.

### 2.1 The Algorithm of the $D^5$

Fig. 3 shows the proposed algorithm. The algorithm consists of three parts : (1) calculation of fitness differences, (2) classification of strings according to the differences and (3) estimation of the classified strings.

After initializing population, following procedures are repeated for each locus $i$ : At first, locus $i$ in each string $s^p$ is perturbed and then fitness difference for the perturbation is calculated as follows:

$$df_i(s^p) = f(s^p) - f(s_i^p) \tag{1}$$

In the above equation, $s_i^p$ is a string perturbed ( $0 \rightarrow 1$ or $1 \rightarrow 0$ ) at locus $i$. Then, strings are classified into sub-populations according to the fitness difference $df_i(s^p)$. The classification method will be mentioned in subsection 2.2. Finally, the sub-populations are estimated in order to detect loci which depend on locus $i$. This stage will be mentioned in subsection 2.3.

### 2.2 Classification According to Fitness Differences

Fig.4 shows a classification method we employed. We employ a centroid linkage method for classification. In this method, the centroid of a cluster is determined by averaging $df_i(s^p)$ of all strings within that cluster. The distance between two clusters is defined as the distance between the centroids of the clusters. The pair of clusters having the smallest distance are merged until a termination criteria — number of clusters becomes smaller than the pre-defined threshold and distance $df_i(c)$ reaches to the above upper bound of distance — is satisfied.

Although this simple classification is enough for quasi-decomposable problem, as future work, other classification methods should be investigated for complex problem.

1. initialize classes
   one class $c^p$ includes one string $s^p$ and fitness difference of the class $df_i(c^p)$ is set to $df_i(s^p)$
2. estimate $|df_i(c^p) - df_i(c^p)|$ for all pair of class $(c^p, c^q)$ and merge the pair with smallest difference.
3. update $df_i(c)$ of new class to average $df_i$ of all strings in the class
4. repeat 2 and 3 until a termination criteria are met.

**Fig. 4.** Classification Algorithm

1. for each sub-population $p$ classified by the Classification Algorithm
   a) initialize set of loci $v_1 = \{1, 2, \cdots, i-1, i+1, \cdots, l\}$ and $v_2 = \{i\}$
   b) while $|v_2| < K$, where $K$ is pre-defined problem complexity
      i. calculate a entropy $E_j = E(v_2 \cup \{j\})$ for all locus $j \in v_1$
      ii. $h = \arg\min_{j \in v_1} E_j$
      iii. update $v_1 = v_1 - \{h\}$ and $v_2 = v_2 \cup \{h\}$
   c) $v_p = v_2$ and $E_p = E(v_2)$
2. select $v_p$ with the smallest $E_p$ as the linkage set for locus $i$

**Fig. 5.** Construct Linkage Set

### 2.3 Construction of Linkage Sets

Fig. 5 is the algorithm to construct a linkage set for locus $i$. First, the set is initialized as $\{i\}$. The locus which gives the smallest entropy joining the linkage set is merged repeatedly until the size of linkage set exceeds pre-defined problem complexity $k$. The entropy measure is used by Harik [3] and defined as

$$E(v_2) = -\sum_{x=1}^{2^{|v_2|}} p_x \log_2 p_x \tag{2}$$

where $n$ is population size $p_x$ is the appearance ratio of each schema $x$ and $2^{|v_2|}$ is the number of all possible schema defined by $v_2$. In our algorithm, if $p_x = 0$, $\log_2 p_x$ is set to 0 for convenience. The reason why we can identify a linkage set by finding $v_2$ which gives the smallest $E(v_2)$ is described in the next subsection.

This procedure is applied for all sub-populations except those including small number of strings. This is because that estimation of distribution from small samples has risk of unreliable result. Finally the linkage set giving the smallest entropy is selected as linkage set $v_i$ of locus $i$ from linkage sets obtained on each sub-population. If there are several sub-populations which give the smallest entropy, then the one with larger sub-population size seems to have more reliability because small sub-populations are likely to have unexpected bias.

**Table 1.** Order 3 sub-problem

| solution | fitness |
|---|---|
| 111 | 30 |
| 110 | 0 |
| 101 | 0 |
| 100 | 14 |
| 011 | 0 |
| 010 | 22 |
| 001 | 26 |
| 000 | 28 |

**Table 2.** $s$, $f(s)$, $f(s_1)$ and $df_1$

| $s$ | $f(s)$ | $f(s_1)$ | $df_1$ |
|---|---|---|---|
| 110001000 | 54 | 81 | 27 |
| 111111101 | 60 | 30 | -30 |
| 101010111 | 52 | 78 | 26 |
| 000111010 | 80 | 66 | -14 |
| 110111000 | 58 | 85 | 27 |
| 001111011 | 56 | 30 | -26 |
| 000110101 | 28 | 14 | -14 |
| 100110111 | 44 | 58 | 14 |
| 011011101 | 0 | 30 | 30 |
| 100011100 | 28 | 42 | 14 |
| 111011100 | 44 | 14 | -30 |
| 111000001 | 84 | 54 | -30 |
| 110101000 | 28 | 55 | 27 |
| 111100100 | 58 | 28 | -30 |
| 000011110 | 28 | 14 | -14 |
| 110111101 | 30 | 57 | 27 |
| 000111010 | 85 | 71 | -14 |
| 000000001 | 82 | 68 | -14 |
| 000010001 | 81 | 67 | -14 |
| | ... | | |
| | ... | | |

**Table 3.** Strings classified according to $df_1$

| $df_1$ | $s$ | schema |
|---|---|---|
| 30 | 011011101 | 011****** |
| 30 | 011111010 | 011****** |
| 30 | 011111011 | 011****** |
| 30 | 011001100 | 011****** |
| 30 | 011010001 | 011****** |
| 30 | 011110011 | 011****** |
| 27 | 110100100 | 110****** |
| 27 | 110111000 | 110****** |
| 27 | 110001000 | 110****** |
| 27 | 110011101 | 110****** |
| 27 | 110101000 | 110****** |
| 27 | 110111101 | 110****** |
| 27 | 110101001 | 110****** |
| 27 | 110101001 | 110****** |
| 26 | 101010111 | 101****** |
| 26 | 101101000 | 101****** |
| 26 | 101110011 | 101****** |
| 26 | 101100011 | 101****** |
| 26 | 101010000 | 101****** |
| | ... | |
| | ... | |

### 2.4 Mechanism of the D$^5$

In EDAs, estimated sub-population must have the information about problem structure to detect dependency of variables. In other words, the estimated sub-population must have a biased and determinate distribution. It is clear that there is not any kind of information in a completely random population i.e. an initial population and that no model can be constructed from such population. Most EDAs select strings with relatively high fitness value to obtain the information from them. However, as mentioned before, the methods have difficulties in capturing sub-problems having a small fitness contribution.

In this part, we describe how the D$^5$ obtains such biased sub-populations by classification according to fitness differences. First, we give an explanation using a concrete problem and then we show a formal exposition.

Table. 1 shows an order 3 deceptive problem which was used as an opponent of the messy GA [2]. Consider a fitness function composed of sum of the 3 sub-functions. For example, we obtain strings with $f(s), f(s_1)$ and $df_1$ for the problem as shown in Table. 2. In this table, $f(s_1)$ is a fitness value of a string perturbed at locus 1 and $df_1 = f(s_1) - f(s)$ is a fitness difference. These strings are classified as in Table. 3. It is clear that loci which depend on locus 1 (locus 2 and 3) have same gene values. In contrast to the case of locus 2 and 3, loci which does not compose a same sub-function with the 1-st locus distribute randomly.

In sub-population having $df_1 = 30$, linkage set $\{1, 2, 3\}$ has only schema 011 and $E(\{1, 2, 3\})$ should be zero. On the other hand, linkage set $\{1, 4, 5\}$ has schemata $010, 001, 010$ and $E(\{1, 2, 3\})$ should be relatively large. Therefore the algorithm evaluate that a relationship between locus 1, 2, and 3 take place more likely than a relationship between locus 1, 4, and 5.

In the followings we give a further explantion using some equations.

We assume that problem is composed as sum of completely separable sub-problems like

$$f(s) = \sum_{v \in V} f_v(s). \tag{3}$$

where $v$ is a set of loci which compose a sub-problem $f_v(s)$ and $V$ is a set of linkage groups (a set of a set of loci). This class of problems are known as additively decomposable functions.

Let $\hat{v}$ the sub-problem including locus $i$ then equation (3) is represented as

$$f(s) = f_{\hat{v}}(s) + \sum_{v \neq \hat{v}, v \in V} f_v(s). \tag{4}$$

It is clear that $f_v(s)$ is calculated independently of locus $i$ if $v$ does not include $i$. Therefore the following equation is true :

$$\sum_{v \neq \hat{v}, v \in V} f_v(s) = \sum_{v \neq \hat{v}, v \in V} f_v(s_i) \tag{5}$$

where $s_i$ is a string having same values as $s$ in all loci except locus $i$. From the above equation, $df_i(s)$ is represented as follows :

$$\begin{aligned} df_i(s) &= f(s) - f(s_i) \\ &= [f_{\hat{v}}(s) + \sum_{v \neq \hat{v}, v \in V} f_v(s)] - [f_{\hat{v}}(s_i) + \sum_{v \neq \hat{v}, v \in V} f_v(s_i)] \\ &= f_{\hat{v}}(s) - f_{\hat{v}}(s_i). \end{aligned} \tag{6}$$

Accordingly, it is said that $df_i(s)$ is defined independently of loci $j \notin \hat{v}$ and depend only on loci $j \in \hat{v}$.

The entropy $E(v_2)$ in equation (2) is a measure of uncertainty of distribution of schemata. Therefore, if schemata defined by $v_2$ distribute randomly, $E(v_2)$ takes large value. Because $df_i(s)$ dose not depend on loci $j \notin \hat{v}$, such loci distribute randomly, such linkage sets give large $E(v_2)$ over sub-population classified according to $df_i(s)$. On the other hand, if the distribution of schemata is biased, $E(v_2)$ becomes small. Since the sub-population is classified by $df_i(s)$ and $df_i(s)$ is depend on loci $j \in \hat{v}$, a correct linkage set $\hat{v}$ can identify by finding $v_2$ which minimize $E(v_2)$.

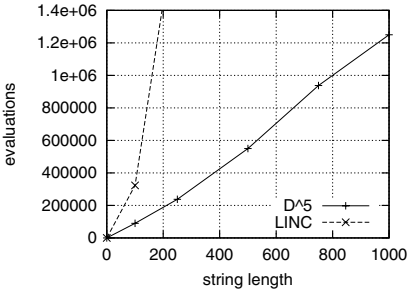


**Fig. 6.** Results of $D^5$ and LINC for 5-bit Trap Function

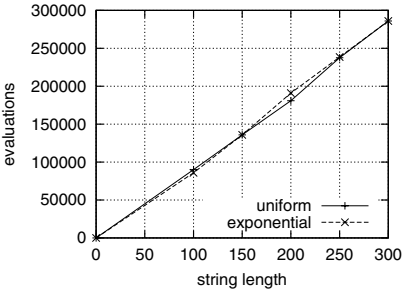


**Fig. 7.** Results of $D^5$ for Trap Function with Uniformly Scaled Sub-Functions and Trap Function with exponentially Scaled Sub-Functions

## 3 Experiments

In this section, we give simulation results of the $D^5$ in comparison to the LINC and its variations. Experiments in subsection 3.1 and 3.2 examine efficiency of the algorithms for a problem composed of uniformly scaled trap sub-problems and a problem composed of exponentially scaled trap sub-problems respectively. An experiment in subsection 3.3 investigates the ability of the algorithms for a problem having not only interdependencies in sub-functions but overall complexity.

### 3.1 Uniformly Scaled 5-Bits Trap Function

First test function is defined as the sum of uniformly scaled 5-bit trap sub-function defined as equations (7) and (8).

If accurate problem structure is known, following optimization will be success. Therefore, we examine how many evaluations of fitness function are needed to obtain correct dependency for all loci. The numbers of evaluations for detecting accurate problem structure are recorded for various string lengths (i.e. problem size). Since all test functions we use are composed of sum of sub-functions having deceptiveness, it is considered that when loci in the same sub-function with locus $i$'s are detected perfectly for all locus $i$ in 10 independent runs, problem structure is obtained accurately.

$$f(s) = \sum_{i=1}^{m} \mathrm{trap}_5(s_{5\cdot i}, \cdots, s_{5\cdot i+4}) \tag{7}$$

$$\mathrm{trap}_5(s_{5\cdot i}, \cdots, s_{5\cdot i+4}) = \begin{cases} 5 & \text{if} \quad u = 5 \\ 4 - u & \text{otherwise} \end{cases} \tag{8}$$

where $u$ is the number of ones in each 5 bit substring $s_{5\cdot i}, \cdots, s_{5\cdot i+4}$ and $m$ is the number of sub-functions. In our experiment, we try various $m$ and change overall string length $l = 5 \times m$

Fig.6 shows the number of evaluations on the 5-bit trap function. It is clear that the number of evaluations required by the $D^5$ is far smaller than the LINC and exceeds $O(l)$ slightly.

### 3.2 Exponentially Scaled 5-Bits Trap Function

Second test function is the sum of 5-bit trap sub-function having exponentially increasing contribution.

$$f(s) = \sum_{i=1}^{m} 2^i \mathrm{trap}_5(s_{5\cdot i}, \cdots, s_{5\cdot i+4}) \tag{9}$$

where $\mathrm{trap}_5(s_{5\cdot i}, \cdots, s_{5\cdot i+4})$ is a same function as (8). Contribution of $i$-th sub-function is controlled by $2^{i-1}$.

In this function, a solution which has high fitness values in $1, 2, \cdots, m-1$ -th sub-functions and low fitness value in $m$-th sub-function can not be modeled due to inequality $\sum_{i=1}^{m-1} 2^i < 2^m$. Therefore, EDAs should model each sub-function one by one from the one having large contribution to the one having small contribution and can not get accurate model for sub-function having small contribution immediately. As mentioned in section 2, the BOA needs $O(l^{1.55})$ if sub-functions are scaled uniformly but it requires $O(l^2)$ if sub-functions are scaled exponentially.

Fig.7 shows the number of fitness evaluations for the uniformly scaled function and the exponentially scaled function by the $D^5$. The solid line shows result for the uniformly scaled function and the dotted line shows result for the exponentially scaled function. It is said that the $D^5$ can solve a problem which has exponentially scaled sub-problems as efficient as a problem which has uniformly scaled sub-problems because two lines overlap each other considerably. This is because fitness differences for perturbations of loci are not affected by nonuniform scaling. Note that also the LINC can perform on the uniformly scaled function as on the exponentially scaled function but it needs $O(l^2)$ fitness evaluations for both functions.

### 3.3 Overlapped Function

In this experiment, we use the sum of trap sub-functions, but having overall nonlinear interdependency as follow:

$$f(s) = [\sum_{i=1}^{10} \mathrm{trap}_5(s_{5\cdot i}, \cdots, s_{5\cdot i+4})]^m \tag{10}$$

**Table 4.** Average and Standard Deviation of Accuracy of linkage identification

| $m$ | $D^5$-800 | | $D^5$-1600 | | LIEM-32 | | LIEM-64 | | $LIEM^2$-32 | | $LIEM^2$-64 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Avg. | S.D. | Avg. | S.D. | Avg. | S.D. | Avg. | S.D. | Avg. | S.D. | Avg. | S.D. |
| 1 | 1.00 | 0.00 | 1.00 | 0.00 | 0.95 | 0.04 | 1.00 | 0.00 | 0.95 | 0.06 | 1.00 | 0.00 |
| 2 | 0.99 | 0.01 | 1.00 | 0.00 | 0.96 | 0.04 | 1.00 | 0.00 | 0.94 | 0.05 | 1.00 | 0.00 |
| 3 | 0.96 | 0.03 | 1.00 | 0.00 | 0.94 | 0.04 | 1.00 | 0.00 | 0.95 | 0.04 | 1.00 | 0.00 |
| 4 | 0.86 | 0.05 | 0.94 | 0.05 | 0.79 | 0.08 | 0.98 | 0.03 | 0.95 | 0.04 | 1.00 | 0.00 |
| 5 | 0.68 | 0.06 | 0.88 | 0.06 | 0.31 | 0.11 | 0.49 | 0.07 | 0.95 | 0.05 | 1.00 | 0.00 |

where $m = 1, 2, \cdots, 5$ and $\mathrm{trap}_5(s_{5\cdot i}, \cdots, s_{5\cdot i+4})$ is an equation (8). This function was proposed to ensure the performance of the LIEM and the $LIEM^2$, which are expansion of the LINC for overlapped function [9, 10]. Problem length is fixed to $5 \times 10 = 50$ and a strength of overall complexity $m$ is varied from 1 to 5. The accuracy of linkage identification is measured and is compared with the LIEM and the $LIEM^2$.

In the $D^5$, we employ fixed population size $n = 800$ and $n = 1,600$ for all $m = 1, \cdots, 5$ therefore the number of fitness evaluations is $40,800$ and $81,600$ respectively. In the classification phase, sub-populations are merged until the number of clusters becomes 8. In the estimation phase, sub-populations which have 8 or lower strings are ignored due to their unreliability. The population size for the LIEM and the $LIEM^2$ is $n = 32$ and $n = 64$. The numbers of fitness evaluations are $40,832$ for $n = 32$ and $81,664$ for $n = 64$.

Table. 4 shows the ratio of correct linkage groups in 10 independent runs for equation (10). The numbers following the name of the algorithms are their population size. As mentioned before, the number of fitness evaluations of $D^5$ using 800 strings is roughly the same as the number of the LIEM and the $LIEM^2$ using 32 strings. For the problem of $m = 1, 2, 3$, the $D^5$-800 can perform best for the number of fitness evaluations. However, when $m = 4, 5$, the $LIEM^2$-32 gives the best result of the three algorithms. The $D^5$-800 can perform better than the LIEM-32 for these problems.

The number of fitness evaluations of the $D^5$ using 1,600 strings is also roughly same as the number of the LIEM and the $LIEM^2$ using 64 strings. All of the three can identify all linkage sets accurately for the problem of $m = 1, 2, 3$. For the problem of $m = 4$, the $D^5$-1600 and the LIEM-64 also give high score but they can not archive complete identification like the $LIEM^2$-64. For the problem of $m = 5$, the accuracy of the LIEM-64 decreases considerably and it is lower than the $D^5$-1600 and even the $D^5$-800.

Because of relatively small problem size, the number of evaluations of all algorithms is set equivalent. However, it is clear from the experiments in subsections 3.1 and 3.2 that the number of evaluation of the $D^5$ is smaller than the LIEM and the $LIEM^2$ for large problem size.

The horizontal axes of the histograms in Fig. 8 – 10 show normalized fitness difference by the perturbation at a locus and the vertical axes is the number

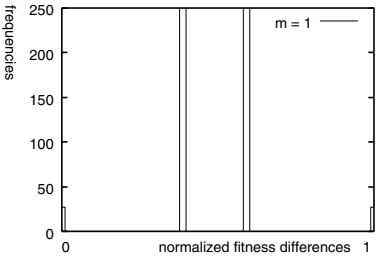


**Fig. 8.** Histogram of $m = 1$ problem

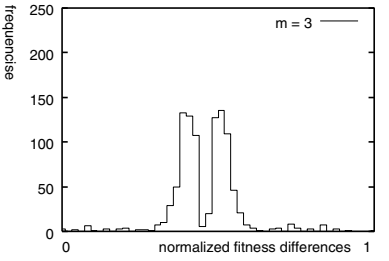


**Fig. 9.** Histogram of $m = 3$ problem

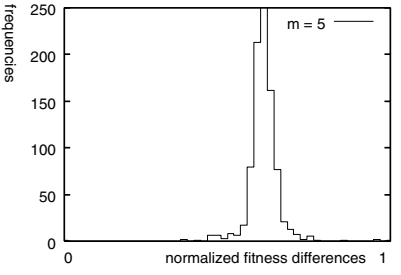


**Fig. 10.** Histogram of $m = 5$ problem

of strings showing the fitness difference. It is clear that the function of $m = 1$ has the character that is easy to be classified. The both sides of strings have sub-solution of (8) of $u = 5$ and sub-solution of $u = 4$ which became $u = 5$ by 1-bit perturbation respectively. On the other hand, it seems that the function of $m = 5$ has the character that is difficult to be classified.

It is said that although the $D^5$ uses only quasi linear fitness evaluations, it can solve problem having weak overall complexity.

## 4 Discussion and Conclusion

In this paper, we propose a novel approach to detect dependency between loci. The proposed $D^5$ estimates dependencies of loci based on sub-populations classified according to fitness differences by perturbation at each locus. The fitness difference depends only on loci constructing a same sub-problem as a perturbed locus. Therefore the classified sub-populations has biased distribution in such loci and and and dependencies of a problem are obtained by estimating the sub-populations.

The $D^5$ is hinted from two existing method (1) Linkage Identification Methods and (2) Estimation of Distribution Algorithms (EDAs). It can detect dependencies with smaller number of fitness evaluations than that of the linkage identifications, because it needs only $O(l)$ perturbations while linkage identifications need $O(l^2)$ perturbations for each pair of loci. In addition, the $D^5$ can solve

problems consisting of exponentially scaled sub-functions, which is difficult for the EDAs and is applicable to problems having weak overall complexity.

The $D^5$ can be located in the place where existing methods can not perform well. For future work, more powerful classification method should be investigated for complex problems.

## References


1. Jeremy S. De Bonet, Jr. Charles L. Isbell, and Paul Viola. MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing Systems*, 9:424–430, 1997.
2. David E. Goldberg, Bradley Korb, and Kalyanmoy Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems*, 3(5):415–444, 1989.
3. Georges Harik. Linkage learning via probabilistic modeling in the ECGA. Technical Report IlliGAL Report No.99010, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.
4. Georges Harik, Fernando G. Lobo, and David E. Goldberg. The compact genetic algorithm. *Proceedings of the 1998 IEEE Conference on Evolutionary Computation*, pages 523–528, 1998.
5. Robert B. Heckendorn and Alden H. Wright. Efficient linkage discovery by limited probing. In *Proceedings of the 2003 Genetic and Evolutionary Computation Conference*, pages 1003–1014. Morgan Kaufmann Publishers, 12–16 July 2003.
6. John H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.
7. Hillol Kargupta. The gene expression messy genetic algorithm. In *International Conference on Evolutionary Computation*, pages 814–819. Springer Verlag, 9 1996.
8. Heinz Mühlenbein and Thilo Mahnig. FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7(4):353–376, 1999.
9. Masaharu Munetomo. Linkage identification based on epistasis measures to realize efficient genetic algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 445–452, 2002.
10. Masaharu Munetomo. Linkage identification with epistasis measure considering monotonicity conditions. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning*, 2002.
11. Masaharu Munetomo and David E. Goldberg. Identifying linkage groups by nonlinearity/non-monotonicity detection. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, 1999.
12. Martin Pelikan. *Bayesian Optimization Algorithm: From Single Level to Hierarchy.* Doctroral dessertation, University of Illionis at Urbana-Champaign, 2002.
13. Martin Pelikan, David E. Goldberg, and Erick Cantú-Paz. BOA: The bayesian optimization algorithm. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, pages 525–532. Morgan Kaufmann Publishers, 1999.
14. Rafal P. Salustowicz and Jürgen Schmidhuber. Probabilistic incremental program evolution: Stochastic search through program space. In M. van Someren and G. Widmer, editors, *Machine Learning: ECML-97*, volume 1224, pages 213–220. Springer-Verlag, 1997.