A Crossover for Complex Building Blocks Overlapping

Miwako Tsuji
Hokkaido University, JSPS Research Fellow
N11W5, Sapporo
060–0811,Japan
m_tsuji@cims.hokudai.ac.jp

Masaharu Munetomo
Hokkaido University
N11W5, Sapporo
060–0811,Japan
munetomo@iic.hokudai.ac.jp

Kiyoshi Akama
Hokkaido University
N11W5, Sapporo
060–0811,Japan
akama@iic.hokudai.ac.jp

ABSTRACT

We propose a crossover method to combine complexly overlapping building blocks (BBs). Although there have been several techniques to identify linkage sets of loci o form a BB [4, 6, 7, 10, 11], the way to to realize effective crossover from the linkage information from such techniques has not been studied enough. Especially for problems with overlapping BBs, a crossover method proposed by Yu et al. [13] is the first and only known research, however it cannot perform well for problems with complexly overlapping BBs due to insufficient variety of crossover sites. In this paper, we propose a crossover method which examines values of given parental strings minutely and defines which variables are exchanged to produce new and different strings without increasing BB disruptions as much as possible. The method is combined with a scalable linkage identification technique to construct an efficient algorithm for problems with overlapping BBs. We design test functions with controllable complexity of overlap and test the method with the functions.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

Crossover, Linkage, Building Blocks

1. INTRODUCTION

In order to design efficient crossovers, there have been several techniques to identify loci that are tightly linked to construct a building block (BB) [4, 6, 7, 10, 11]. However, the method to exploit the information they provide has not been studied enough especially for the problems with overlapping BBs. The LIMD-TD [8] and the DSMGA [12] divide all loci into non-overlapping sets by deleting weak interactions or detecting strong interactions. A crossover method by Yu et al. [13] is the first and only known method that can deal with problems with strong overlaps, however, when BBs overlap complexly, it gives insufficient variety of crossover sites i.e. it exchanges limited sets of loci.

In this paper, we improve the existing crossover method to decrease BB disruptions and ensure the variety of crossover. The proposed crossover method exchanges various combinations of BBs even when they overlap intricately. We combine the proposed crossover with the D^5 [11], which is a scalable technique to detect dependency between loci. Moreover, we design test functions with controllable complexity of overlap and test the proposed crossover with the functions.

In the next section, we describe our notation and problem definition. Then, we discuss issues in the existing crossover method. In section 4, we propose a new crossover method for problems with overlapping BBs. In section 5, we Wdesign the D^5-GA with the proposed crossover to optimize problems with overlapping unknown BBs. In section 6, we design test functions and perform experiments. The conclusion is in section 7.

2. NOTATION AND PROBLEM DEFINITION

Schema theorem says that short, low-order, and highly fit schemata increase their share to be combined [5]. It is considered that such combination of small parts is consistent with human innovation and important for GAs [2]. These lead to a problem model called an additively decomposable function, which is written as a sum of low-order sub-functions.

In real world problems, which loci belong to a sub-function is unknown and a locus could belong to several sub-functions. Therefore, in this paper, we define problems with overlapping sub-functions as follows: A string $\boldsymbol{s}$ of length l is sometimes described as a series of loci, $\boldsymbol{s} = s_1 s_2 \cdots s_i \cdots s_l$, where the subscripts are identification numbers assigned to loci. In the following discussion, we assume $\boldsymbol{s} = s_1 s_2 \cdots s_l$ is a random permutation of problem variables $\boldsymbol{x} = x_1 x_2 \cdots x_l$ to represent random encoding. The fitness of $\boldsymbol{s}$ is defined as:

$$f(\boldsymbol{s}) = \sum_{j=1}^{m} f_j(\boldsymbol{s}_{\boldsymbol{v}_j}), \tag{1}$$

where m is the number of sub-functions, f_j is a j-th sub-

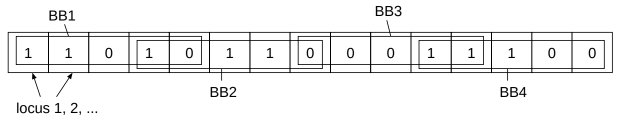


Figure 1: An example of the overlapping sub-functions

function and $\boldsymbol{s}_{\boldsymbol{v}_j}$ is its sub-solution. The $\boldsymbol{v}_j$ is a vector of id numbers of loci and defines $\boldsymbol{s}_{\boldsymbol{v}_j}$. For example, if $\boldsymbol{v}_j = (1,3,7,5)$, $\boldsymbol{s}_{\boldsymbol{v}_j} = s_1 s_3 s_7 s_5$. Two vectors specify identical sub-strings if and only if all of their corresponding components are equal. Let V_j is a *set* of loci which consist $\boldsymbol{v}_j$. The V_j specifies a set of loci which interdepend to construct a sub-solution. Note the following: if $V_j = \{1,2,3,4\}$ and $V_{j'} = \{4,3,2,1\}$ then $V_j = V_{j'}$, while if $\boldsymbol{v}_j = (1,2,3,4)$ and $\boldsymbol{v}_{j'} = (4,3,2,1)$ then $\boldsymbol{v}_j \neq \boldsymbol{v}_{j'}$ and $\boldsymbol{s}_{\boldsymbol{v}_j} \neq \boldsymbol{s}_{\boldsymbol{v}_{j'}}$. This equation can express any function by using $(1,2,\cdots,l)$ as $\boldsymbol{s}_1$.

In this paper, we call a set V_j *a linkage* set and interaction between loci in a same linkage set *linkage*. Candidate solutions for sub-functions are called *building blocks* (BBs). If two different sub-functions are defined over sets of loci which share a locus (or loci), i.e. $V_j \cap V_{j'} \neq \emptyset$, $(j \neq j')$ then, it is said that these sub-functions *overlap*.

$\overline{V}_j$ $(j = 1,2,\cdots)$ denote sets of loci which are estimated to be linked. The aim of the linkage identification techniques such as [4, 6, 7, 10, 11] is to construct $\{\overline{V}_1, \overline{V}_2, \cdots\} \approx \{V_1, V_2, \cdots\}$.

General crossover operators divide all loci into two sets and exchange one of them. For example, the one-point crossover divides $\{1,2,\cdots,l\}$ into $\{1,\cdots,i\}, \{i+1,\cdots,l\}$. In following, we call the set of loci to be exchanged *crossover set*, $\mathcal{V}$.

3. BACKGROUND

3.1 Linkage identification and crossover techniques

As mentioned in the earlier section, there have been several efforts to identify linkage sets. In contrast, how to use the information of the mutual dependency relations between loci has not been studied enough. Most of linkage identification techniques assume that sub-functions do not overlap. For problems with non-overlapping sub-functions, it should be easy to estimate sub-functions and mix BBs using the information of relationships.

To combine BBs effectively for problems with complex interactions is more difficult and less studied. For such problems, the tightness detection [8] removes weak interactions and the dependency structure matrix [12] extracts strong interactions. These methods aim to construct non-overlapping linkage sets when resulted linkages between loci are not trivial.

However, sub-functions sometimes overlap innately and a single locus belongs to two or more linkage sets. For problems with overlapping sub-functions, it is not enough to divide loci into non-overlapping clusters. An example of such problem is shown in Figure 1. In the figure, loci 4 and 5 belong to BB_1 and BB_2. If one of them is exchanged, the other one is disrupted. If both of them are exchanged, no information is exchanged. For such problems, only a few researches such as a crossover method by Yu et al. [13] are known. In the next sub-section, we show their crossover method and its issues.

1. Construct a graph $G = (N, E)$, where the nodes are linkage sets $\overline{V}_j$ and the edges are overlapping relations between two nodes.
2. For each crossover operator: Choose two nodes n_1, n_2 randomly. Then partition the graph G into two sub-graphs $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$ which satisfy conditions: $n_1 \in N_1$, $n_2 \in N_2$ and $|E| - |E_1| - |E_2|$ is minimal.
3. Let $\mathcal{V} = \bigcup_{\overline{V}_j \in N_1} \overline{V}_j$ and exchange loci in $\mathcal{V}$.

Figure 2: The existing crossover algorithm

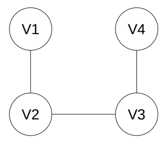


Figure 3: A graph in the existing method. This represents the string in figure 1.

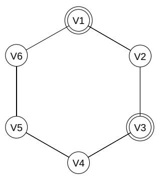


Figure 4: An example of simple overlap (The ring structure)

3.2 The existing crossover method for overlapping problems

Yu et al. [13] investigated the relationship between an inaccurate linkage and the convergence time of GA to developed a graphical crossover method for problems with overlapping sub-functions. The crossover algorithm in Figure 2 is designed from the observation that the prevention of the detection failure error (the linkage model does not link those genes which are linked in reality) is critical to successful recombination. It constructs a graph $G = (N, E)$, where the nodes are linkage sets and the edges are overlapping relations between two nodes. Figure 3 shows the graph of the problem shown in Figure 1. Then it divides G into two graphs at random to disrupt minimal number of BBs.

It works well if BBs overlap simply. For example, let us consider a ring structure shown in Figure 4. This is employed as a test function for the existing method. In the function, each sub-function overlaps with only its neighbor sub-functions. Figure 5 shows crossovers which minimize BB disruptions when double circles (V_1, V_3) are chosen as (n_1, n_2). Either white nodes or black nodes are exchanged. The divisions which result in equivalent strings by the crossovers are removed. When BBs overlap simply, there are several crossover sets even for a same pair of (n_1, n_2). Moreover, the pair (n_1, n_2) can be chosen at random.

However, when overlap becomes more complex, graph partition with minimum BB disruptions is restricted and the partition tends to divide graphs into a small graph and the remaining large one.

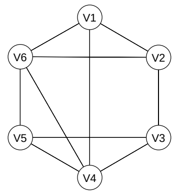


Figure 6: An example of complex overlaps

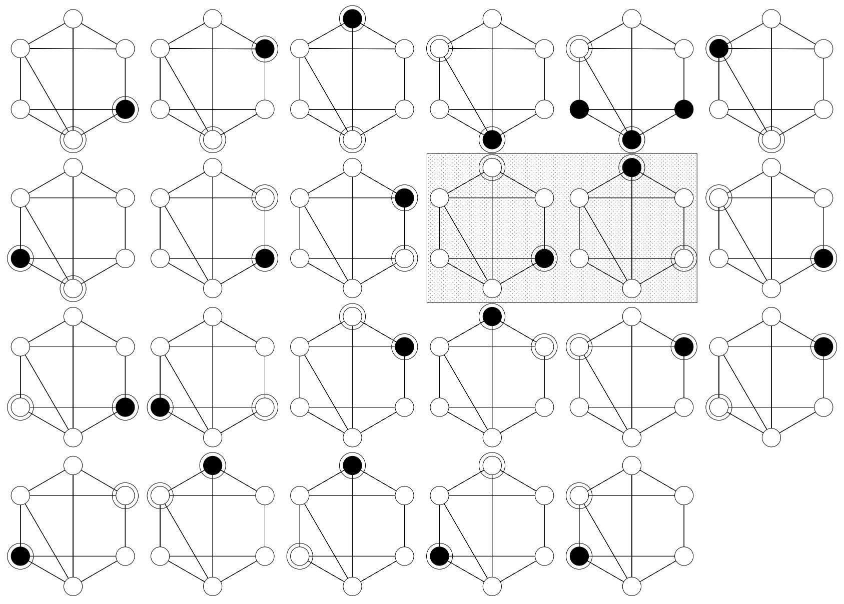


Figure 7: All possible crossovers for the problem in Figure 6. The divisions which show equivalent crossover for a (n_1, n_2) pair is removed. The graph partitions for V_1, V_3 are highlighted.

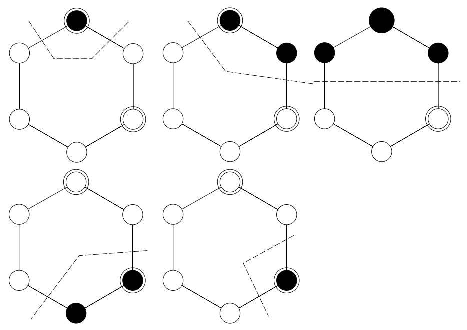


Figure 5: Resulted crossover from the existing method. The divisions which show equivalent crossover are removed.

As an extreme case, we consider a graph where each node has e random edges. Let $|N|$ the number of nodes in the graph $G = (N, E)$. A node in G_1 should have $e\frac{|N_2|}{|N|-1}$ edges to nodes in G_2. Then, the expected number of edges from G_1 to G_2 is

$$\begin{aligned} \sum_{i \text{ s.t. } V_i \in N_1} \frac{e|N_2|}{|N|-1} &= e \sum_{i \text{ s.t. } V_i \in N_1} \frac{|N|-|N_1|}{|N|-1} \\ &= -e\frac{|N_1|(|N_1|-|N|)}{|N|-1}. \qquad (2) \end{aligned}$$

This function is convex upward for $|N_1|$. Of $|N_1| = 1, 2, \cdots, |N|-1$, only 1 and $|N|-1$ give the minimum number of edges between two sub-graphs.

Let us consider a problem with complex overlaps shown in Figure 6. There are ${}_6C_2 = 15$ choices for (n_1, n_2). For most pairs of nodes, the number of minimum disruption crossovers is small. All possible crossovers for the problem in Figure 6 are shown in Figure 7. Again, the double circles show (n_1, n_2) and the divisions which show equivalent crossover are removed. In the figure, there are a few crossover sets for each (n_1, n_2) pair and all except the one example exchange only one BB. For example, for a pair (V_1, V_3), although the five graph partitions shown in Figure 5 are available for the simple problem, only the two graph partitions highlighted in the figure 7 divide the pair with minimum disruptions for the complex problem.

In this paper, we modify the existing method to enrich variety of crossovers without increasing BB disruptions. Moreover, the proposed method sometimes disrupts less BBs than the existing method.

4. CROSSOVER FOR COMPLEX INTERACTIONS AND OVERLAPS

In order to hold BB disruptions to a minimum and preserve diversity of crossover, we propose a crossover method shown in Figure 8. While the existing method searches the best division over a single graph G for all pairs of parents, the proposed method reconstructs the graph for each pair of parent strings $\boldsymbol{s} = s_1 s_2 \cdots s_i \cdots s_l$ and $\boldsymbol{t} = t_1 t_2 \cdots t_i \cdots t_l$.

Figure 10 shows an example of the proposed crossover for parent strings shown in Figure 9. First, the nodes on which sub-solutions (BBs) are identical are removed (Figure 10, left), because whether such BBs are exchanged or not has no effect on the offspring. This operator ensures the obtained offsprings always differ from their parent strings. In our example, because BBs on the V_1 are same, the node for V_1 is removed.

Then, the edges where no BB disruption occurs practically are removed (Figure 10, middle). In our example, the edge between V_3 and V_4 are removed because for parent sub-

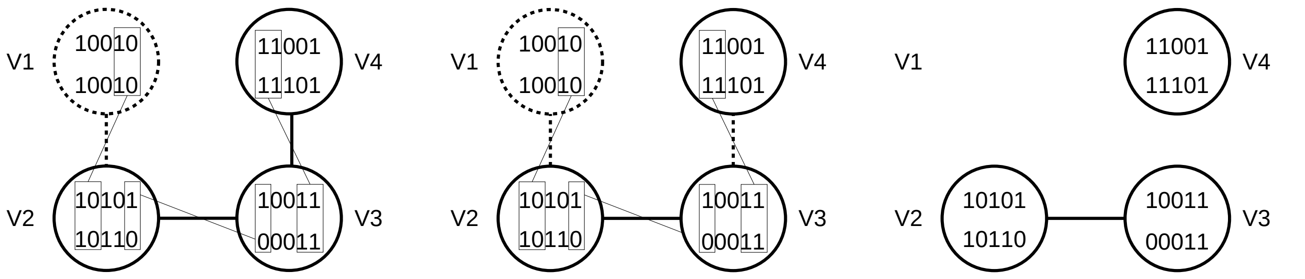


Figure 10: Example of the proposed crossover method. left : remove same BBs, middle : remove edges where BB disruption does not occur, right : resulted graph

1. Construct a graph $G = (N, E)$, where the nodes are linkage sets $\overline{V}_j$ and the edges are overlapping relations between two nodes.
2. For each crossover operator for parent strings $\boldsymbol{s} = s_1 s_2 \cdots s_i \cdots s_l$ and $\boldsymbol{t} = t_1 t_2 \cdots t_i \cdots t_l$
   (a) Remove nodes $\overline{V}_j$ where $\boldsymbol{s}_{\overline{\boldsymbol{v}}_j} = \boldsymbol{t}_{\overline{\boldsymbol{v}}_j}$.
   (b) Remove edges between $\overline{V}_j$ and $\overline{V}_{j'}$ if the following conditions are hold :
      - The exchange between BBs $\boldsymbol{s}_{\overline{\boldsymbol{v}}_j}$ and $\boldsymbol{t}_{\overline{\boldsymbol{v}}_j}$ does not disrupt BBs $\boldsymbol{s}_{\overline{\boldsymbol{v}}_{j'}}$ and $\boldsymbol{t}_{\overline{\boldsymbol{v}}_{j'}}$.
      - The exchange between BBs $\boldsymbol{s}_{\overline{\boldsymbol{v}}_{j'}}$ and $\boldsymbol{t}_{\overline{\boldsymbol{v}}_{j'}}$ does not disrupt BBs $\boldsymbol{s}_{\overline{\boldsymbol{v}}_j}$ and $\boldsymbol{t}_{\overline{\boldsymbol{v}}_j}$.
   (c) Choose two nodes n_1, n_2 randomly. Then partition the graph G into two sub-graphs $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$ which satisfy conditions: $n_1 \in N_1$, $n_2 \in N_2$ and $|E| - |E_1| - |E_2|$ is minimal.
   (d) Let $\mathcal{V} = \bigcup_{\overline{V}_j \in N_1} \overline{V}_j$ and exchange loci in $\mathcal{V}$.

Figure 8: The proposed crossover algorithm

strings

$\underline{\texttt{10011001}}$ with BBs $\underline{\texttt{10011}}$ and $\underline{\texttt{11001}}$

`00011101` with BBs `00011` and `11101`,

obtained sub-strings are

$\texttt{00011}\underline{\texttt{001}}$ with BBs `00011` and $\texttt{11}\underline{\texttt{011}}$

$\underline{\texttt{10011}}\texttt{101}$ with BBs $\underline{\texttt{10011}}$ and `11101`,

or

$\underline{\texttt{100}}\texttt{11101}$ with BBs $\underline{\texttt{100}}\texttt{11}$ and `11101`

$\texttt{000}\underline{\texttt{11001}}$ with BBs `00011` and $\underline{\texttt{11001}}$,

when V_3 or V_4 is exchanged respectively.

After that, the resulted graph (Figure 10, right) is divided into two sub-graphs to minimize the number of cut edges. The right graph of Figure 10 obtained by the proposed method suggests that there is a graph partitioning without BB disruption. On the other hand, the existing method simply chooses a partitioning from

$\{V_1, V_2, V_3 | V_4\}$,

$\{V_1, V_2 | V_3, V_4\}$ which cuts one edge, and

$\{V_1 | V_2, V_3, V_4\}$ which creates no new string.

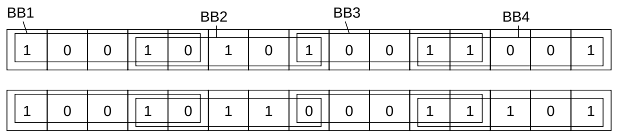


Figure 9: Parents

Moreover, the proposed method can give various crossover sets even for the complex problems like the one shown in Figure 6. The reason is as follows: For every crossover, the proposed method removes nodes and edges to simplify the graph G. The reconstruction of the G depends on the values of parental strings. Therefore, different pairs should give different graphs. The different graphs result in various sub-graphs.

The reduced surrogates crossover proposed by Booker [1] examines non-matching alleles in parental strings to make reduced strings and selects crossover points for the reduced strings. It always produces variants and maintains population diversity to avoid premature convergence. While the proposed method ignores BBs which are identical like the reduced surrogates crossover, it does not perform crossover in allele-wise but in BB-wise.

To sum up the characteristics of the proposed crossover,

- It can reduce BB disruptions.
- It can make various variations of crossover even for problems with complex overlaps and interactions.

Because the diversity of offsprings can be held by making several crossover sets with the least BB disruptions, we can reduce the innovation time, which is the number of generations required to find a new and better solution than all the strings in a population.

5. D^5-GA FOR PROBLEMS WITH OVERLAPPING SUB-FUNCTIONS

In order to work the proposed and existing crossover methods well, linkage sets must be known in advance. There are several techniques to identify the linkage sets and the proposed and existing crossover methods are independent of the way to identify linkage. In this paper, we employ the D^5 (Dependency Detection for Distribution Derived from *df*) [11], which is a scalable technique to detect dependency

```
1. Initialize population. Let the number of linkage sets $j = 0$.
2. For each locus $i$:
   (a) Perturb $s_i$ ($0 \to 1$ or $1 \to 0$) and calculate fitness difference $df_i(\boldsymbol{s})$ for all strings.
   (b) Classify strings according to fitness differences $df_i(\boldsymbol{s})$ into sub-populations.
   (c) For each sub-population:
       i. Find $\overline{V}_j$ which minimizes entropy $E(\overline{V}_j)$ in the sub-population.
       ii. Let $j = j + 1$.
```

Figure 11: Algorithm of the D^5 for problems with overlapping sub-functions

between loci to identify the linkage sets, and modify it for problems with overlapping sub-functions.

5.1 D^5 for overlapping functions

The D^5 constructs sets of loci which depend on each locus i by estimating sub-populations classified according to fitness differences by perturbations (change gene value $0 \to 1$ or $1 \to 0$) at locus i. The algorithm of the D^5 for problems with overlapping sub-functions is shown in Figure 11. The entropy $E(\overline{V}_j)$ in the figure is calculated as follows:

$$E(\overline{V}_j) = -\sum_x p_x \log p_x, \tag{3}$$

where x is possible sub-strings $\boldsymbol{s}_{\boldsymbol{v}_j}$ and p_x is the appearance ratio of each sub-string x in a sub-population.

The item 2–(c)–i in the figure searches the set of loci whose values distribute unequally measuring entropy. When a problem is decomposable like the equation (1), fitness difference by the perturbation at locus i is as follows:

$$df_i(\boldsymbol{s}) = f(s_1 \cdots \overline{s}_i \cdots s_l) - f(s_1 \cdots s_i \cdots s_l) \tag{4}$$

$$= \sum_{j \text{ s.t. } i \in V_j} \Big(f_j(s_1 \cdots \overline{s}_i \cdots s_{l_{\boldsymbol{v}_j}}) - f_j(s_1 \cdots s_i \cdots s_{l_{\boldsymbol{v}_j}}) \Big) \tag{5}$$

where $\overline{s}_i$ is 0 if $s_i = 1$ or 1 if $s_i = 0$. The equation (5) shows that $df_i(\boldsymbol{s})$ is defined by the sub-functions f_j such that $i \in V_j$. If $i \notin V_j$, substrings $s_1 \cdots s_i \cdots s_{l_{\boldsymbol{v}_j}}$ and $s_1 \cdots \overline{s}_i \cdots s_{l_{\boldsymbol{v}_j}}$ are identical and their fitness difference is zero. Therefore, in each sub-population of strings with same $df_i(\boldsymbol{s})$, loci in V_j such that $i \in V_j$ should distribute unequally and the entropy of them in the strings should be small. The other loci should distribute randomly and the entropy of them in the strings should be large.

In the above process, assuming non-overlapping problems, the original D^5 defines the size of a linkage set in advance. For problems with overlapping sub-functions, it is required to define the size adaptively because the number of loci which interdepend on a locus depends on the number of sub-functions which the locus belongs to. In most implementations, each linkage set is constructed incrementally i.e. a locus which gives the smallest $E(\overline{V}_j \cup \{i\})$ is added one by one starting with $\overline{V}_j = \{i'\}$, where i' is the perturbed locus. We set a threshold E_t $(0 < E_t < 1)$ to decide whether a new locus should be added to the linkage set or not. When a locus i is added to a set $\overline{V}_j$, if $E_s(\overline{V}_j \cup \{i\}) = 0$ or

$$\big(E_o(\overline{V}_j \cup \{i\}) - E_s(\overline{V}_j \cup \{i\})\big) - \big(E_o(\overline{V}_j) - E_s(\overline{V}_j)\big) > E_t$$

then let $\overline{V}_j = \overline{V}_j \cup \{i\}$ and search a next locus. Otherwise, return $\overline{V}_j$. In the above inequality, $E_o(V)$ is entropy of V in the original population and $E_s(V)$ is entropy of V in the sub-population. The inequality means that if the relative bias of $\overline{V}_j \cap \{i\}$ in the sub-population to the original population is larger than that of $\overline{V}_j$, then the locus i is rejected. If i is irrelevant to the perturbed loci, the distributions in the sub-population and original population are not different and $E_o(\overline{V}_j \cup \{i\}) - E_s(\overline{V}_j \cup \{i\})$ should be small.

Moreover, because the original D^5 assumes that a locus construct one sub-function, it chooses a single $\overline{V}_j$ for a single locus from estimations in several sub-populations. We modify this to hold all linkage sets constructed in the all sub-populations, because the source of each $df_i(\boldsymbol{s})$ could sometimes come from different sub-functions where the locus i exists redundantly.

5.2 Constructing linkage sets from dependency information

The D^5 gives sets $\overline{V}_j$ $(j = 1, 2, \cdots)$ of loci which depend on each locus. Some of them should include one or more linkage sets because perturbed loci sometimes belong to several linkage sets $V_x, V_y, \cdots$. In this sub-section, we show a simple method to infer linkage sets from the sets.

First, we should estimate sub-problem order k. This parameter is not algorithm specific but required by general genetic algorithms to decide adaptive population size etc.. Then,

1. Remove duplications of linkage sets.
2. Extract linkage sets from unions.
3. Remove unions of other sets.
4. Remove proper subset of other sets.

Some of $\overline{V}_j$ are unions of V_j s which overlap each other. A linkage set is extracted from the unions in the item 2. Product sets $\overline{V}_j \cap \overline{V}_i$ which satisfy the following conditions are extracted.

- $|\overline{V}_j \cap \overline{V}_i| \geq k$

In the item 3, sets $\overline{V}_j$ which satisfy the following conditions are removed as unions.

- linkage sets $\overline{V}_x, \overline{V}_x$ are exist such that
  - $\overline{V}_x \cap \overline{V}_y \neq \emptyset$, $\overline{V}_j \cap \overline{V}_x \neq \emptyset$, $\overline{V}_j \cap \overline{V}_y \neq \emptyset$ for $x \neq y \neq j$
  - $|\overline{V}_j| > |\overline{V}_x|$ and $|\overline{V}_j| > |\overline{V}_y|$
  - $\overline{V}_x \cup \overline{V}_y \supseteq \overline{V}_j$

5.3 Niching

If there are complex overlaps, it is difficult to find all V_j perfectly since the number of loci which depend on a locus becomes large. To overcome the failure in linkage identification, population diversity must be maintained. Therefore, we introduce niching to GAs for problems with overlapping

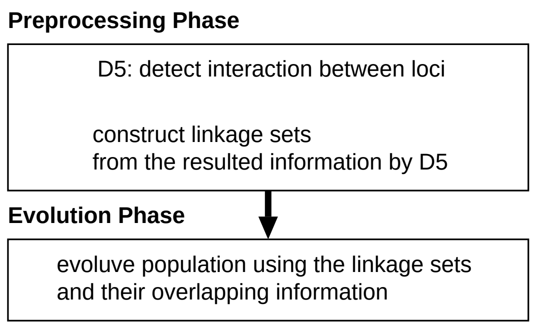

Preprocessing Phase
D5: detect interaction between loci
construct linkage sets
from the resulted information by D5
Evolution Phase
evoluve population using the linkage sets
and their overlapping information

Figure 12: Overall algorithm of the D^5-GA

sub-functions. Niching is also useful when optimal sub-solutions for overlapping sub-functions are contradictory to each other. We employ the restricted tournament selection [3] which compares offsprings to similar strings. Other niching techniques can also be employed.

5.4 Overall algorithm of the D^5-GA for problems with overlapping sub-functions

The overall algorithm of the D^5-GA for problems with overlapping sub-functions is shown in Figure 12. It can be divided into preprocessing phase and evolution phase. The preprocessing phase obtains linkage sets. The evolution phase evolves strings to find optimal solution(s). The both phases use strings but the number of them could differ in each phase.

6. EXPERIMENTS

6.1 Test functions

Yu et al. employ a test function whose sub-functions overlap circulate and each overlapping length is 2. Each sub-function is a 5-bit trap function defined as

$$\mathrm{trap5}(\boldsymbol{s}_{\boldsymbol{v}_j}) = \begin{cases} \frac{4-u}{5}, & u = 0,1,2,3,4 \\ 1, & u = 5 \end{cases} \tag{6}$$

and decision variables of sub-function j are defined as:

$$\begin{aligned} \boldsymbol{v}_j &= (3(j-1)+1 \bmod l, 3(j-1)+2 \bmod l, \\ &\quad \cdots, 3(j-1)+5 \bmod l). \end{aligned}$$

Therefore, the whole function is

$$\begin{aligned} f(\boldsymbol{s}) &= \mathrm{trap5}(s_1 s_2 s_3 s_4 s_5) + \mathrm{trap5}(s_4 s_5 s_6 s_7 s_8) \\ &\quad + \cdots + \mathrm{trap5}(s_{l-2} s_{l-1} s_l s_1 s_2). \end{aligned} \tag{7}$$

We design a function with stochastic circularly overlapping sub-functions based on the function with circularly overlapping sub-functions. It enables us to control the complexity of overlap from systematic (circulate) one to random one. The loci which belong j-th sub-function are defined as follows:

$$\begin{aligned} \boldsymbol{v}_j &= (N(3j,\sigma^2) \bmod l, N(3j,\sigma^2) \bmod l, \\ &\quad \cdots, N(3j,\sigma^2) \bmod l). \end{aligned}$$

where $N(\mu,\sigma^2)$ is a normal distribution with mean μ and variance σ^2. If a locus is already in $\boldsymbol{v}_j$, a new locus-id-number should be sampled from $N(3j,\sigma^2)$. The overlapping length can also be controlled to change the interval of μ. Figure 13 and 14 show a sub-function with small σ^2 and one with large σ^2 respectively. The boxes in the figure show loci and painted boxes show the loci which belong the sub-function. The graphs in Figure 15 show the functions with $l = 60$ and $\sigma^2 = 1, 25, 100^2$. The function with small σ^2 is similar to the original function with circularly overlapping sub-functions. Larger σ^2 gives more complex overlaps.

6.2 Experiments with known linkage sets

In order to compare the proposed crossover with the existing crossover in terms of only crossover performance, we perform experiments assuming that linkage sets are known. We employ the functions with stochastic circularly overlapping sub-functions with $\sigma^2 = 1, 4, 25, 100$ and $l = 60, 90, 120$. For each test function and each crossover method, we perform 30 independent runs and obtain the % of the runs which achieve the global optimal solution and the # of generations required to achieve the global optimal solution if it can be achieved. Population sizes are fixed for each σ^2 even when l is changed. The maximum number of generations set to 200. If there is no optimal solution at that time, the run is considered to be a failure. No mutation is employed to investigate performance of crossover.

The left figure in Figure 16 shows the % of runs which obtain optimal solutions. Even for small σ^2, the proposed method obtains the optimal solutions with higher probability than the existing method. For large σ^2, the existing method cannot obtain optimal solutions while the proposed method can archive them. The right figure in Figure 16 shows the average number of generations. For each string length, the proposed method requires smaller number of generations than the existing method. While the proposed method requires larger computational cost in each crossover, the number of crossovers is smaller than the existing method because it finds the optimal solution faster the existing method.

6.3 Experiments with unknown linkage sets

In general, linkage sets are not known and must be identified. We perform experiments the D^5-GA for problems with overlapping sub-functions described in the section 5. We investigate the numbers of evaluations required to obtain optimal solutions all 30 runs in the functions with stochastic circularly overlapping sub-functions with $\sigma^2 = 1, 4, 25, 100,$ 10000 and $l = 60, 90, 120$. No mutation is used again. For comparison, not only the D^5-GA with the existing crossover but also the BOA [9] with niching is performed. The BOA does not use crossover but builds a Bayesian network from promising strings and generates new strings from the network. The Bayesian network can represent overlapping relations.

For each algorithm, population size is defined empirically to obtain optimal solution using the smallest computation cost. For the D^5-GAs, the populations for the preprocessing phase and the evolution phase are defined independently.

Figure 17 (a)–(d) show the results. Figure 17 (a) shows the number of evaluations in the whole phase of the D^5-GA with the proposed crossover and figure 17 (b) shows the number of evaluations in the evolution phase of the D^5-GA with the proposed crossover. From these two results, it is shown that when BBs overlap complexly, the computation cost for the evolution phase becomes large. Comparing the

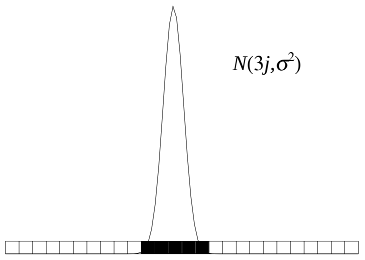


Figure 13: A sub-function with small σ^2

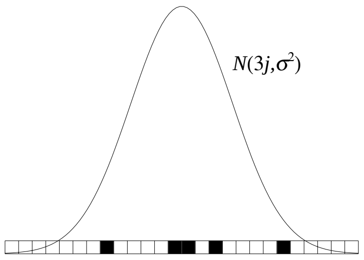


Figure 14: A sub-function with large σ^2

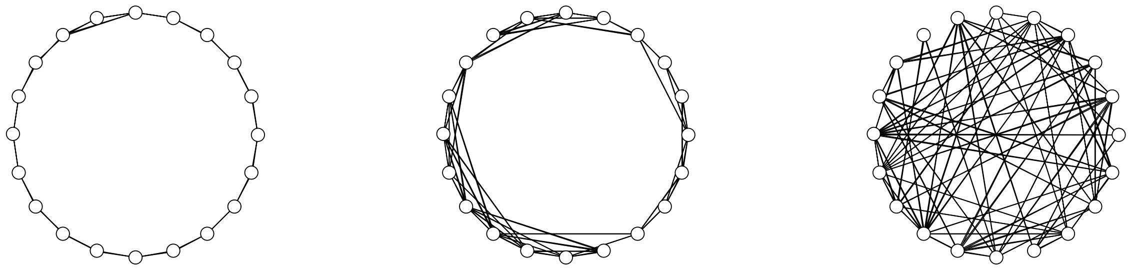


Figure 15: Functions with stochastic circularly overlapping sub-functions. $l = 60, \sigma^2 = 1$ (left), $l = 60, \sigma^2 = 25$ (middle) and $l = 60, \sigma^2 = 100^2$ (right). Nodes are linkage sets and edges are overlapping relations between nodes.

(a) (c) and (d) in the figure, it is clear that the D^5-GA with the proposed crossover gives the best result for large l even when σ^2 is large. Figure 17 (c) shows that the D^5-GA with existing crossover requires enormous numbers of evaluations for problems with large σ^2 and l.

7. CONCLUSION

In this paper, we design an efficient crossover method for problems with overlapping sub-functions. Although the existing method can work well for problems with simple overlaps, for problems with complex overlaps, it cannot perform well due to insufficient variety of crossover sites. The proposed crossover method can ensure the variety because it examines values of given parental strings minutely and defines which variables are exchanged to produce new and different strings without increasing BB disruptions.

We combine the proposed crossover method and the D^5 (one of linkage identification techniques) to construct a comprehensive algorithm for problems with complex and unknown interactions. The test function with controllable complexity of overlap is designed and used to verify the effectiveness of the proposed method.

8. REFERENCES

[1] L. Booker. Improving search in genetic algorithms. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, pp. 61–73. Morgan Kaufmann, 1987.

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

[3] G. R. Harik. Finding multimodal solutions using restricted tournament selection. In L. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 24–31, San Francisco, CA, 1995. Morgan Kaufmann.

[4] R. B. Heckendorn and A. H. Wright. Efficient linkage discovery by limited probing. In *Genetic and Evolutionary Computation - GECCO2003 Part 1, Lecture Notes in Computer Science 2723*, LNCS 2723, pp. 1003–1014. Springer-Verlag, 2003.

[5] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

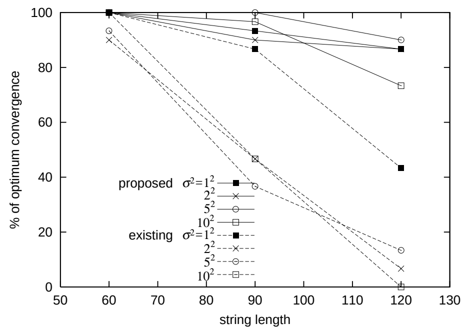
[6] H. Kargupta and B.-H. Park. Gene expression and fast construction of distributed evolutionary representation. *Evolutionary Computation*, 9(1):43–69, 2001.

[7] M. Munetomo and D. E. Goldberg. Identifying linkage groups by nonlinearity/non-monotonicity detection. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, pp. 433–440. Morgan Kaufmann Publishers, 7 1999.

[8] M. Munetomo and D. E. Goldberg. Linkage identification by non-monotonicity detection for overlapping functions. Technical Report IlliGAL Report No.99005, University of Illinois at Urbana-Champaign, 1 1999.

[9] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian optimization algorithm. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, pp. 525–532. Morgan Kaufmann Publishers, 1999. ftp://ftp-illigal.ge.uiuc.edu/pub/src/sBOA/C++/sBOA.tar.Z.

[10] M. J. Streeter. Upper bounds on the time and space complexity of optimizing additively separable functions. In *Genetic and Evolutionary Computation - GECCO2004 Part 2, Lecture Notes in Computer Science 3103*, pp. 186–197. Springer-Verlag, 2004.



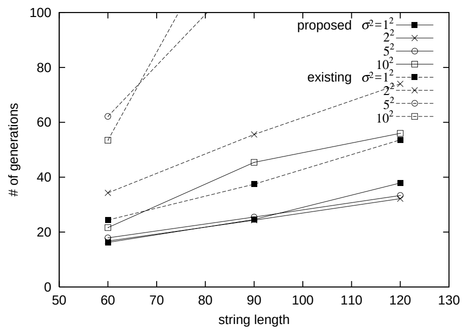


Figure 16: The % of runs which obtain optimal solutions (left) and the average number of generations required to obtain the optimal solutions (right). The solid lines show the results of the proposed method and the dotted lines show the results of the existing methods. Population sizes are 500, 600, 1200 and 2000 for $\sigma^2 = 1^2, 2^2, 5^5$ and 10^2 respectively.

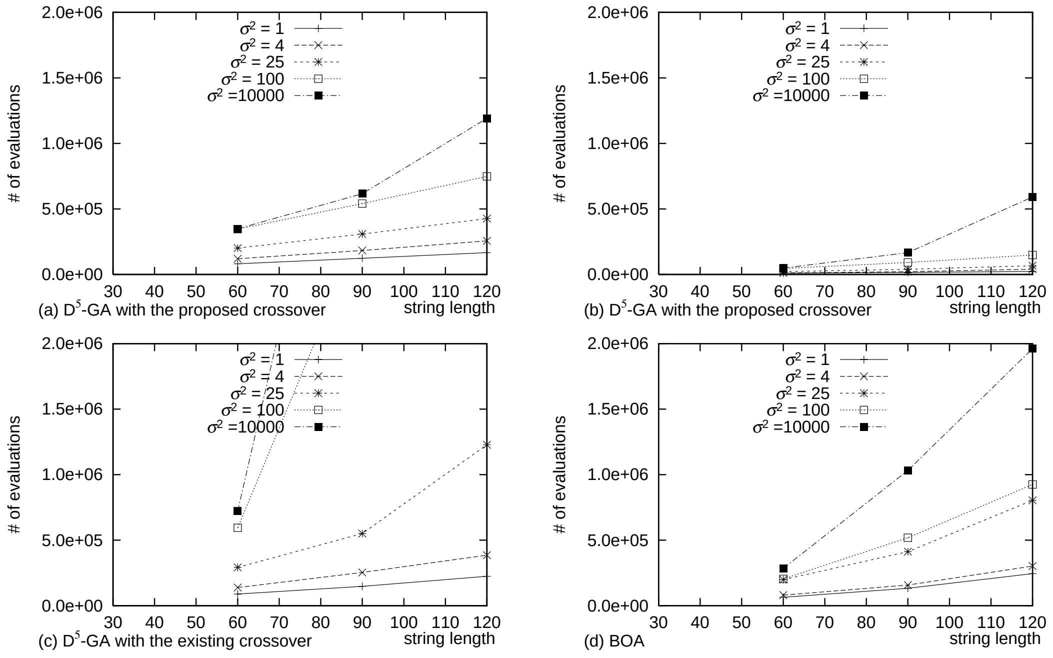


Figure 17: The numbers of evaluations required to obtain optimal solutions in all 30 runs

[11] M. Tsuji, M. Munetomo, and K. Akama. Modeling dependencies of loci with string classification according to fitness differences. In *Genetic and Evolutionary Computation - GECCO2004 Part 2, Lecture Notes in Computer Science 3103*, pp. 246–257. Springer-Verlag, 2004.

[12] T.-L. Yu, D. E. Goldberg, A. Yassine, and Y.-P. Chen. Genetic algorithm design inspired by organizational theory: Pilot study of a dependency structure matrix driven genetic algorithm. In *Proceedings of Artificial Neural Networks in Engineering 2003 (ANNIE 2003)*, pp. 327–332, 6 2003.

[13] T.-L. Yu, K. Sastry, and D. E. Goldberg. Linkage learning, overlapping building blocks, and systematic strategy for scalable recombination. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pp. 1217–1224, 6 2005.