A Crossover Operation for an Evolutionary Method Using Extended Binary Decision Diagrams

Atsuko Mutoh, Satoru Oono*, Kousuke Moriwaki†, Tsuyoshi Nakamura, Nobuhiro Inuzuka, Hidenori Itoh

Department of Intelligence and Computer Science
Nagoya Institute of Technology
Gokiso, Showa-ku, Nagoya 466-8555, Japan
Tel: +81-52-735-5050 Fax: +81-52-735-5477
E-mail: {atsuko, soono, moriw, tnaka, inuzuka, itoh} @ics.nitech.ac.jp

ABSTRACT: We have been proposing an evolutionary method using a gene expression system extending Binary Decision Diagrams (BDDs). The method efficiently solves combinatorial optimization problems. The method extended BDDs to express multi-value functions. The extended BDD is called n-BDD. Three genetic operations have also been introduced. The genetic operations for n-BDDs, however, do not include crossover for a reason of structural problems. In this paper we propose a crossover operation extending Bryant's Apply operation for BDDs, and verify its effect by experiments in a quasi ecosystem. In the experiments the proposed method using crossover has more than 40% high fitness than the conventional method.

KEYWORDS: crossover, binary decision diagrams, evolutionary learning

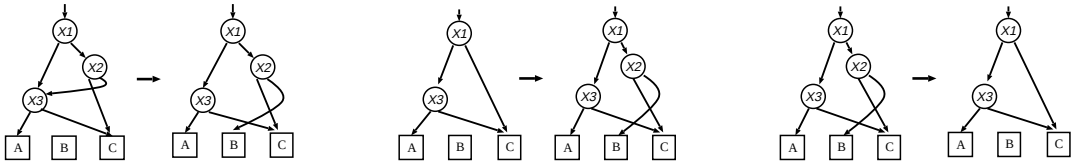
1 Introduction

Binary Decision Diagrams (BDDs), which are originally proposed by Akers in 1978[1], are graph representations of Boolean functions. Methods to operate BDDs are developed in [2]. BDD representation has been applied to various engineering fields, such as CAD systems of LSI design, because of its storage efficiency and processing speed. Moreover, it has been used also for combinatorial optimization problems and has played a large part to solve problems that had not been solved in realistic time until recently. A gene expression system using n-BDDs, which extend BDDs, are proposed in [3] to express multi-value functions. In [3] the n-BDD gene expression system expresses behavior of agents and lets genetic operations work more efficiently in a co-evolutional environment. Genetic operations for n-BDD, however, do not have a crossover operation. After we revisit the n-BDDs gene expression system and genetic operation for n-BDDs in Section 2, we propose a crossover operation for n-BDDs using Bryant's Apply operation[2] in Section 3. Section 4 gives experiments which compare methods using n-BDD with/without the crossover. Section 5 discusses the results of experiments. We also discuss on time and space complexity of genes and operations.

2 A gene expression system with n-BDD and genetic operations

Here we summarise the concept of n-BDD and its genetic operations[3]. A BDD is a graphical notation of a Boolean function. A BDD has terminal nodes, each of which is labeled by only one of two labels true and

*Currently working for Seiko Epson Corp.
†Currently working for NTT Commucationware Corp.



(a) mutation (b) insertion (c) deletion

Figure 1: Genetic operations for n-BDDs.

false, while an n-BDD can have labels from more than two labels and gives a value from any set of values which the labels denote.

In the Figure 1 a circle denotes an input bit and is called a decision node. Each decision node has three directional edges, one of which comes from outside and two of which go outside. An output value is denoted by a square which is called a terminal node. With input bits an output value is calculated as follows: First look at the top decision node $X1$ and take a left edge if $X1$ is 0 or a right edge if $X1$ is 1. We call the left edge and the right edge a 0-edge and a 1-edge, respectively. Iterate this for the decision node indexed by the 0-edge or the 1-edge until a terminal node is indexed. If the edge indexes a terminal node the value in it is the output.

Genetic operations, mutation, insertion and deletion are defined to operate an n-BDD as a gene. Mutation changes a direction of an edge to a randomly selected node. The node must be subordinated by the node which the edge rises. Because of this restriction a loop and a cycle are never caused. Insertion inserts a new decision node on a randomly selected edge. Either of 0-edge or 1-edge of the new decision node is randomly selected to point to the node which pointed before. The other edge becomes to point to a subordinate node randomly selected. Deletion deletes a randomly selected decision node. The edge pointing to the deleted node becomes to point to one of the nodes which pointed by deleted node before.

3 Crossover based on Bryant's Apply operation

Here we give a notation of n-BDDs based on a data structure. A node of an n-BDD is a non-terminal node, represented by a triple (variable name, left-node, right-node), or a terminal node, represented by a value. An n-BDD is a set of nodes and a node that represents a top node. We can refer an n-BDD by its top node. We equate an n-BDD (and a function f represented by it) with its top node $(f.top, f_0, f_1)$.

Bryant developed methods to operate BDDs[2]. One of them is Apply. The procedure Apply provides a basic method for creating a representation of a function that composing two functions according to structure of Boolean expressions of the two functions or logic gate networks. It takes BDDs representing functions f and g, and a binary Boolean operator ($\circ$), and produces a reduced BDD representing the function $f \circ g$ defined as

$$f \circ g(X_1, X_2, \ldots, X_n) = f(X_1, X_2, \ldots, X_n) \circ g(X_1, X_2, \ldots, X_n).$$

Bryant's Apply takes a binary Boolean operator as ($\circ$), while our proposed Apply extended for n-BDD takes a binary operator on a given finite domain. Based on the Shannon expansion[4], we can expand the above expression as follows.

$$f \circ g(X_1, \ldots, X_i, \ldots, X_n) = \begin{cases} f(X_1, \ldots, X_{i-1}, 0, X_{i+1}, \ldots, X_n) \\ \quad \circ g(X_1, \ldots, X_{i-1}, 0, X_{i+1}, \ldots, X_n) & ;\text{if } X_i = 0 \\ f(X_1, \ldots, X_{i-1}, 1, X_{i+1}, \ldots, X_n) \\ \quad \circ g(X_1, \ldots, X_{i-1}, 1, X_{i+1}, \ldots, X_n) & ;\text{if } X_i = 1 \end{cases} \quad (1)$$

If we note facts that $f(X_0, \ldots, X_{i-1}, 0, X_i, \ldots, X_n) = f_0$ and $f(X_0, \ldots, X_{i-1}, 1, X_i, \ldots, X_n) = f_1$, where $X_i = f.top$, we can re-formalize the expression (1) as $f \circ g = (f.top, f_0 \circ g_0, f_1 \circ g_1)$, where we assume that $f.top = g.top$. In the case $f.top \neq g.top$ we can write that $f \circ g = (f.top, f_0 \circ g, f_1 \circ g)$, when $f.top$ has an upper order than $g.top$ or $f.top$ is not appeared in g, or $f \circ g = (g.top, f \circ g_0, f \circ g_1)$, when $g.top$ has an upper order than $f.top$ or $g.top$ is not appeared in f. Based on this idea the algorithm to calculate $f \circ g$ is given in Figure 2. To use this algorithm the order of variables should be fixed. Our genetic operators including Apply do not violate the order.

The algorithm proceeds from the top of the two n-BDDs downward. It creats a top node in the resultant n-BDD at the nodes of the two n-BDDs, and calculates recursively each generated sub-n-BDDs until obvious calculation related to terminal values are appeared.

Bryant's Apply operation is for BDDs and for a logical connectives. If we choose the conjunction for $(\circ)$ Apply procedure calculates a BDD for $f \wedge g$. For our purpose we use a probabilistic operator as $(\circ)$. For two values a and b, $a \circ b$ is a or b in the probability 0.5. If we use this $(\circ)$ operation, $f \circ g$ inherits both behavior of f and g probabilisically, this is an expected behavior of crossover operation.

If f or g is a terminal node or $f = g$ **then**
$h \leftarrow f \circ g$
If $f.top = g.top$ **then**
$h_0 \leftarrow f_0 \circ g_0, h_1 \leftarrow f_1 \circ g_1$
  if $h_0 = h_1$ **then** $h \leftarrow h_0$
  else $h \leftarrow (f.top, h_0, h_1)$
If rank of $f.top$ is higher than that of $g.top$ **then**
$h_0 \leftarrow f_0 \circ g, h_1 \leftarrow f_1 \circ g$
  if $h_0 = h_1$ **then** $h \leftarrow h_0$
  else $h \leftarrow (f.top, h_0, h_1)$
If rank of $g.top$ is higher than that of $f.top$ **then**
$h_0 \leftarrow f \circ g_0, h_1 \leftarrow f \circ g_0$
  if $h_0 = h_1$ **then** $h \leftarrow h_0$
  else $h \leftarrow (g.top, h_0, h_1)$

Figure 2: An algorithm to calculate $h(= f \circ g)$.

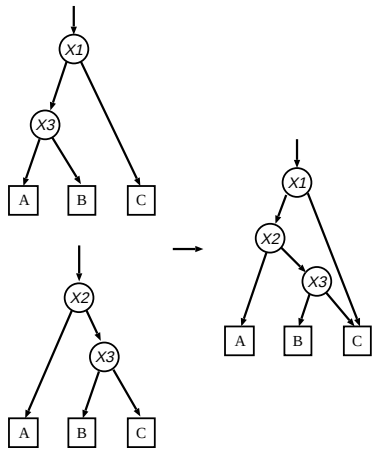


Figure 3: Crossover using Apply operation.

4 Evaluation of the method

To compare ability between two gene expression systems using n-BDDs with crossover and ones without it we use an environment of a quasi ecosystem. The quasi ecosystem is introduced in [3] and is a good test bed for evaluation. This model is used to investigate basic ability of each gene expression. In the following paragraphs, a method of experiments is explained.

4.1 A model of quasi ecosystem

Here we define an environment where two artificial animals act. One of them, carnivore, has a fixed action strategy and tries to capture another one, herbivore, which is evolved using the proposed genetic operations. By observing evolution of herbivor's behavior we examine effect of the two methods.

A gene, n-BDD, takes a bit string as an input, the bit string which gives perceptional information from the environment shown in Table 1 (which is used by both carnivore and herbivore). The order of the variables also obeys this table. The information includes states of an animal, hungry or repletion, and visual perception of a carnivore, a herbivore and plants. A gene outputs a value from values walk, runaway, eat and do-nothing. An animal acts the output value. Table 2 shows the detail of the actions that correspond to the values.

An input bit string is assigned to the decision nodes of an n-BDD. The output values are assigned to terminal nodes of the n-BDD.

The field is a two-dimensional 20×20 array. The herbivore and the the carnivore act according with actions decided by their genes, n-BDDs with/without crossover. Figure 4 shows the fixed action strategy of carnivore.

Thirty plants are distributed in the field randomly. When a plant is eaten by a herbivore, another plant is appeared in a randomly selected position. A stage in our simulator of the quasi ecosystem starts with a herbivore, a carnivore and thity plants, and it terminates when the herbivore dies of hunger or being eaten.

Table 1: Assignments of bit string.

$X0$	hungry
$X1$	repletion
$X2$	carnivore is visible far
$X3$	carnivore is visible near
$X4$	herbivore is visible far
$X5$	herbivore is visible near
$X6$	plant is visible far
$X7$	plant is visible near

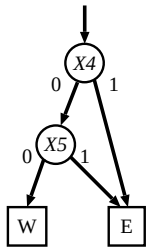


Figure 4: The carnivore's fixed strategy.

Table 2: Actions.

walk(W)	Animal moves to eight neighborhoods at random.
runaway(R)	Animal moves in the opposite direction to the carnivore.
eat(E)	Animal moves in the direction of the food, and eat it if reached.
do-nothing(N)	Animal does not move.

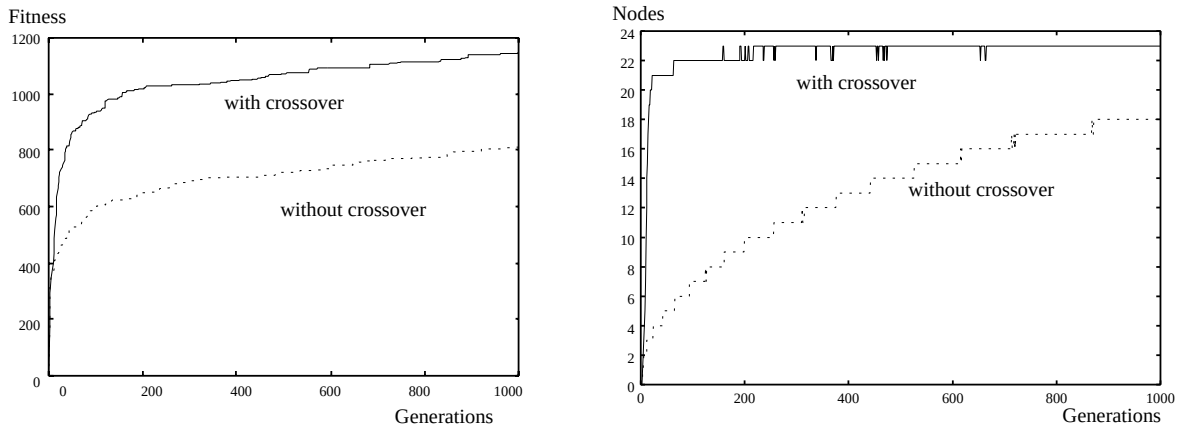
4.2 An experiment with the quasi ecosystem

To execute evolution with our simulator of the quasi ecosystem, a group in a generation is composed of thirty individuals of herbivore. Each herbivore of them and the fixed carnivore are let be in the field and are let behave in the ecosystem during a constant time. The number of steps for which the herbivore survived in the field is taken as a fitness, which is used to select the next generation of herbivores. Five herbivores with the best fitness are included in the next generation. Twenty-five herbivores are generated by the genetic operations from the other twenty-five herbivores and joined with the best five. The twenty-five herbivores are generated by using crossover, mutation, insertion, or deletion in the probability of 0.9, 0.05, 0.025 and 0.025, respectively.

4.3 The results

In the experiment we observed the speed of convergence with each gene expression. We conducted 100 times of experiments. Figure 5(a) shows the average of the maximum fitness of herbivores using n-BDDs with/without crossover at each generation. It is understood that the rise of the fitness of n-BDDs with crossover is faster than the other.

Figure 5(b) shows the average of the number of nodes in an n-BDD with/without crossover at each generation for the 100 times. Although it shows that nodes in an n-BDD with crossover outnumber those of the other, the number of nodes does not exceed a saturated number. This is because crossover for n-BDD is reduced in the algorithm.



(a) fitness. (b) the number of nodes.

Figure 5: The results of a experiment.

5 Discussions

In this paper, we defined a crossover operation for n-BDD based on Bryant's Apply operation. We confirmed effectiveness by the experiments. From the experiments, we can find that the proposed crossover has remarkable effect in evolution. It achieved more than 40% high fitness than the conventional method in the saturated generations.

We should also discuss on the time and space cost of the method. The number of nodes in an n-BDD is at most $2^{k+1} - 1$, which is happen if the n-BDD is a complete binary tree, where k is the length of perceptual bit string and this value can be considered a constant value. The number of nodes in an n-BDD operated by crossover is also bounded. Hence, necessary space to store an n-BDD is bounded by $O(2^k)$. The complexity of mutation, insertion and deletion for n-BDD is $O(1)$. On the other hands the complexity of crossover for n-BDD is $O(|F| \cdot |G|)$, where $|F|$ and $|G|$ is the number of nodes in an n-BDD before crossing graph f and graph g. We expect that crossover of n-BDD takes a proportional time to square of the number of nodes. In our experiment, however, the number of nodes did not grow very much, and so the time spent in the two experiments did not have much difference.

References

[1] S. B. Akers, 1978, *Binary Decision Diagrams*, IEEE Trans. Comput., pp. 509-516.

[2] R. E. Bryant, 1986, *Graph-Based Algorithms for Boolean Function Manipulation*, IEEE Trans. Comput., pp. 677-691.

[3] K. Moriwaki, N. Inuzuka, M. Yamada, H. Seki, and H. itoh, 1997, *A Genetic Method for Evolutionary Agents in a Competitive Environment*, Soft Computing in Engineering Design and Manufacturing, pp.153-162, Springer.

[4] C.E.Shannon, 1938, *A Symbolic analysis of relay and switching circuits*, Trans.AIEE, vol.57, pp.713-723.

[5] S.Minto, 1993, *BEM-II; An Arithmetic Boolean Expression Manipulator Using BDDs*, IEICE Trans. Fundamentals, Vol. E76-A, No.10.

[6] T. Takashina and S. Watanabe, 1995, *Simulation model of self adaptive behavior in quasi-ecosystem*, IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E78-A, No. 5.