Evolutionary Multi-valued Decision Diagrams for Obtaining Motion Representation of Humanoid Robots

Masashi Sakai, Masayoshi Kanoh, *Member, IEEE* and Tsuyoshi Nakamura, *Member, IEEE*

***Abstract*—In this paper, we propose a method using multivalued decision diagrams (MDDs), to obtain motion representation of humanoid robots. Kanoh et al. have proposed a method, which uses multi-terminal binary decision diagrams (MTBDDs), to acquire robot controller. However, non-terminal vertices of MTBDDs can only treat values of 0 or 1; multiple variables are needed to represent a single joint angle. This increases the number of the non-terminal vertices and MTBDDs that represent the controller become complex. Therefore, we consider using the MDD, in which its non-terminal vertices can take on multiple output values. To obtain humanoid robot motion representation, we propose evolutionary MDDs (EMDDs) and show experimental results comparing evolutionary MDDs and evolutionary MTBDDs through simulations of acquisition of robot motion in this paper. Moreover, we verify whether the evolution of MDD using a memetic algorithm is effective.**

***Index Terms*—Multi-valued decision diagram (MDD), evolutionary computation, genetic programming, motion representation, robot control.**

I. Introduction

In recent years, domestic robots that clean homes or monitor homes in the owner's absence, or even communicate with humans have been attracting much attention in the field of robotics [1]–[7]. Of such robots, interest is particularly high in humanoid robots that have bodies similar to humans because they can perform various tasks in place of humans. However, control of the complex mechanisms possessed by humanoid robots is difficult and requires complex mechanics computations. As a way to avoid this problem, we consider use of Binary Decision Diagrams (BDDs) to represent robot motion in this paper.

BDDs are a way of representing a logic function using a directed graph that has a single starting point (root), and that consists of binary switches having one input and two outputs, and they are mainly used to design LSIs [8]–[10]. We believe that if a robot controller can be represented optimally by BDDs, then the robot motion can be generalized. If robot controller that output generalized action can be obtained, then we could expect that even robots that have somewhat different forms can be controlled by using the same diagram. However, as with logic functions, the variables of BDDs can only take values of 0 or 1 as input-output values, so they do not have enough representation of a robot controller. Optimal or suboptimal solution of representation of a BDD for a motion of a robot may be not earned. To acquire robot controllers, Kanoh et al. [11] used Multi-Terminal Binary Decision Diagrams (MTBDDs), and obtained robot motion representation of MTBDDs. However, non-terminal vertices of MTBDDs take only binary values, so multiple variables are needed to represent an angle of robot joint, and that causes the problem of complex representation. Therefore, we consider to represent humanoid robot motion using Multi-valued Decision Diagrams (MDDs).

MDDs, which have multi-branch tree structure, are an extension of MTBDDs, and can represent multi-valued logic functions in compact form on a computer. There are many techniques to find optimal variable ordering of decision diagrams [12]–[16]. These techniques optimize diagrams using given Boolean function(s). However, it is difficult to find an optimum MDD for representing robot motion using the above techniques, because robot motion representation via a Boolean function is unknown. In this paper, we propose evolutionary MDDs (EMDDs) to acquire robot motion representation [17].

We show experimental results comparing evolutionary MDDs and evolutionary MTBDDs through simulations of acquisition of robot motion. Moreover, we verify whether the evolution of MDD using a memetic algorithm is effective.

M. Sakai and T. Nakamura are with the Graduate School of Engineering, Nagoya Institute of Technology, Gokiso-cho, Showaku, Nagoya 466-8555, Japan (email: sakai@ai.nitech.ac.jp and tnaka@nitech.ac.jp).

M. Kanoh is with the Dept. of Mechanics and Information Technology, School of Information Science and Technology, Chukyo University, 101 Tokodachi, Kaizu-cho, Toyota 470-0393, Japan (email:mkanoh@sist.chukyo-u.ac.jp).

Manuscript received April 19, 2005; revised January 11, 2007.

II. Decision Diagrams

A. Binary Decision Diagrams

Binary decision diagrams (BDDs) [8]–[10] are data structures that are compact representations of Boolean functions. BDDs are constructed by two types of vertices (non-terminal vertices and terminal vertices) and two types of edges (0-edge and 1-edge). Each non-terminal vertex has exactly two outgoing edges, one 0-edge and one 1-edge. If the value of the variable assigned to the non-terminal vertex is 0, the 0-edge is followed; when the value is 1, the 1-edge is followed. Each edge connects to a lower non-terminal vertex or a terminal vertex. The terminal vertices represent the Boolean values, 0 or 1. The processing of the variables proceeds from the root, which is the top of the diagram, in order until it reaches a terminal vertex.

An example BDD is shown in Fig. 1(a), where non-terminal vertices are represented by circles and terminal vertices are

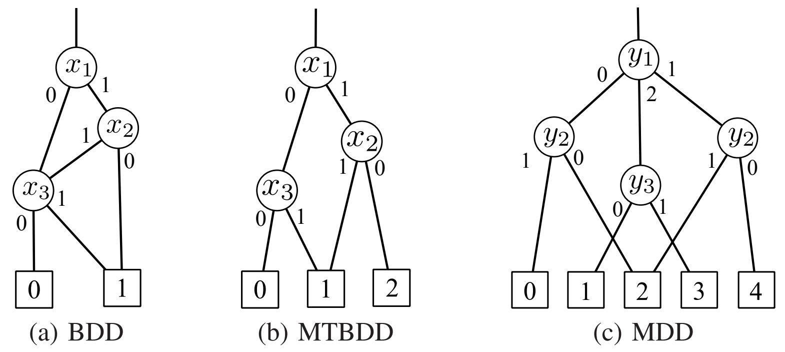


Fig. 1. Decision diagram examples.

represented by squares. In the BDD in Fig. 1(a), three variables (x_1, x_2, x_3) can be processed. For example, when $(x_1, x_2, x_3) = (0, 1, 1)$ is input, first, the 0-edge of the non-terminal vertex for x_1 located at depth 0 is followed. Second, the 1-edge of the non-terminal vertex for x_3 at depth 2 is followed, then the terminal vertex of 1 is reached. Considering all combinations of input, you can know that the BDD depicts the function, $f = x_1\bar{x_2} + x_3$.

B. Multi-terminal Binary Decision Diagrams

Multi-terminal binary decision diagrams (MTBDDs) extend BDDs to have multiple output values. That is to say, MTBDDs allow the constant terminal vertices to have multiple values other than 0 or 1.

In the example MTBDD shown in Fig. 1(b), three variables (x_1, x_2, x_3) can be processed. If $(x_1, x_2, x_3) = (1, 0, 1)$ is input for example, first, the 1-edge is followed from the non-terminal vertex for x_1 located at depth 0. Next, the 0-edge is followed from non-terminal vertex for x_2 at depth 1 to arrive at the output of 2.

C. Multi-valued Decision Diagrams

Multi-valued decision diagrams (MDDs) [18] give multi-branching structured graphs which are improved MTBDDs.

In the example MDD shown in Fig. 1(c), (y_1, y_2, y_3) can be processed. The definition domains of each variable are $y_1 \in \{0, 1, 2\}$, $y_2 \in \{0, 1\}$, and $y_3 \in \{0, 1\}$. When $(y_1, y_2, y_3) = (2, 1, 1)$ is input for example, first, the 2-edge of non-terminal vertex for y_1 at depth 0 is followed, and then the 1-edge of non-terminal vertex for y_3 at depth 2 is followed to reach the output 3.

Such decision diagrams as Fig. 1 in which different variables appear in the same order on all paths from the root are called '*ordered*' and '*reduced*'. An ordered decision diagram is one in which each variable is encountered no more than once in any path and always in the same order along each path. Therefore, the number of variables of an ordered decision diagram limits the maximum depth of one. A reduced decision diagram has two properties: there are no redundant vertices in which both of the two edges leaving the vertex point to the same next vertex present within the diagram; isomorphic sub-diagrams are shared.

Here, all decision diagrams treated in this paper are ordered and reduced. These properties allow a decision diagram representation to be canonical for a given variable ordering.

III. Evolutionary MDD via Genetic Programming

This section describes the evolutionary MDD (EMDD) [17] via genetic programming [19].

The more compact representation of robot motion is better, so it is desirable that the number of variables constructing a MDD is smaller. However, the problem of finding the best variable ordering of BDDs is NP-hard [20]. Moreover, truth table for appropriate robot motion is unknown, that is, its representation via a Boolean function cannot be provided in advance. In this paper, we consider to acquire quasi-optimum solution by using evolutionary computation.

A. Algorithm Overview

Fig. 2 shows the overview of EMDD. The procedure for the evolution of MDDs is as follows.

1) Initialize generation $g \leftarrow 0$.
2) Generate the initial population $U = \{u_1, u_2, ..., u_n\}$, where n is the number of individuals.
3) Compute the fitness of the individuals in U.
4) Select the top k individuals from U, and let them be the set $P = \{p_1, p_2, ..., p_k\}$ (elitist strategy).
5) To $Q = U \setminus P$, and perform genetic operations to Q.
6) Generate the next-generation set $U = P \cup Q$.
7) Increment $g \leftarrow g + 1$, and return to step 3.

We describe genetic operations for evolutionary MDDs in the next section.

B. Genetic Operations

The genetic operations of EMDDs include five operations (insertion, mutation, deletion, replacement, and inversion). These operations were proposed by Kanoh et al. [11] and Moriwaki et al. [21], [22] for evolutionary BDD and MTBDD. We apply these operations to evolution of MDDs. The operations are explained below.

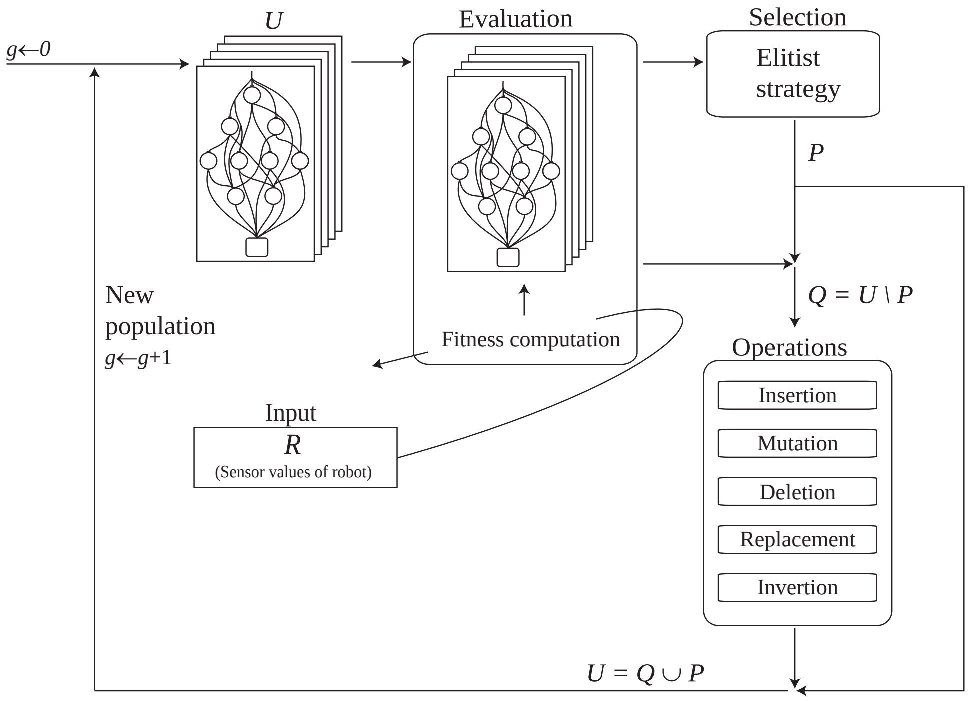


Fig. 2. Evolutionary MDD.

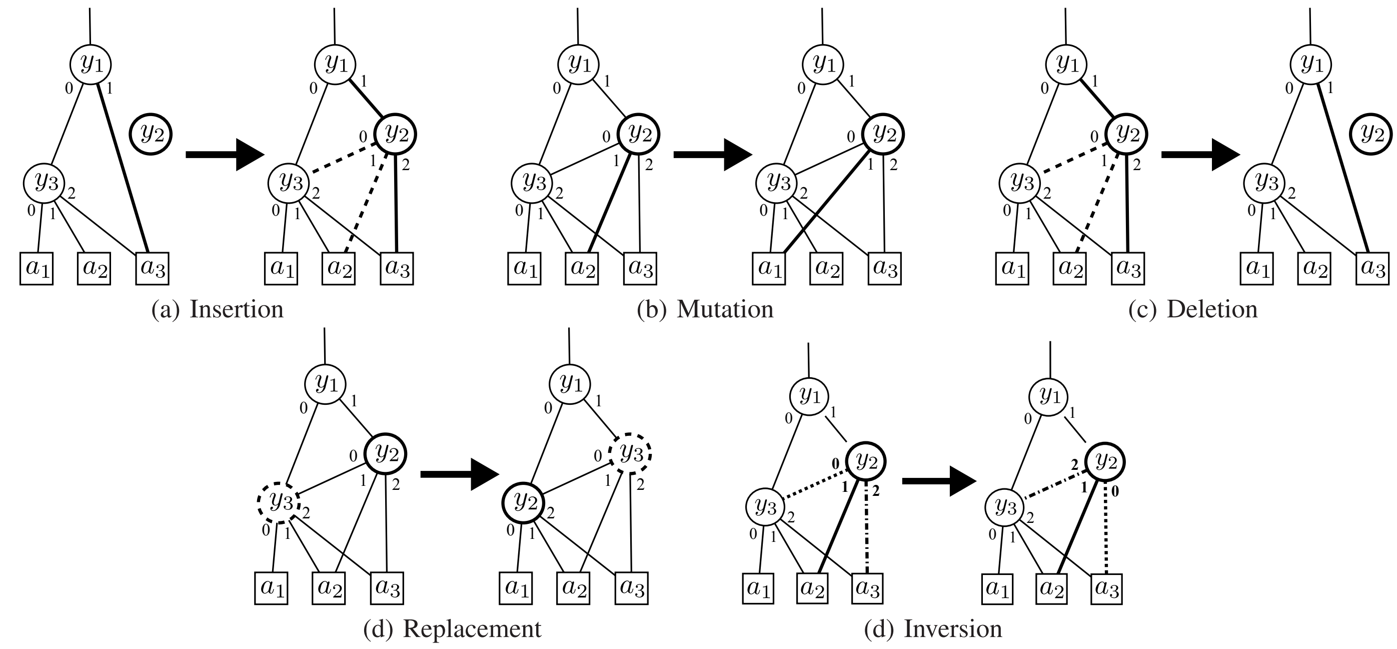


Fig. 3. Genetic operations for MDD.

1) Insertion (Fig. 3(a)): Insertion operation adds a new non-terminal vertex above a randomly selected edge. One of the edges of the added non-terminal vertex remains connected to the vertex to which it was connected prior to the addition. The other edges connect to any lower non-terminal or terminal vertices.

If the variable ordering of MDDs is fixed, it cannot be applied to complex problems, so robot motion may be not obtained. In this paper, we introduce a genetic insert operation, in which a vertex is allocated dynamically on a MDD. We consider that generalized robot motion can be represented through the operation optimally or suboptimally.

Figure 4 shows the insertion operation algorithm.

This algorithm is one in which a non-terminal vertex is inserted while variable ordering is decided dynamically. First, a suffix i of variable is selected at random (*l.* 2 in Fig. 4). If the variable x_i is used as the variable of the root, a suffix is selected again because the root is single vertex (*ll.* 4 and 5 in Fig. 4). Second, checking whether x_i is used is performed. When x_i is used, the non-terminal vertex $V(x_i)$ should be

```
Insert an vertex
Input $S$: a set of suffixes of variables already used in MDD.
 1. begin
 2.   select a suffix $i \in \{1, ..., m\}$ of variable at random;
 3.   create a new non-terminal vertex $V(x_i)$
                                     which variable is $x_i$;
 4.   if $x_i$ is used as the variable of the root
 5.   then go to 2.;
 6.   if $i \in S$
 7.   then begin
 8.     $O = \{e | e$ is an edge
                      on which $V(x_i)$ can be inserted$\}$;
 9.     if $O \neq \phi$
10.     then begin
11.       select an edge $e \in O$ at random;
12.       insert $V(x_i)$ on $e$;
13.     end;
14.     else go to 2.;
15.   end;
16.   else insert $V(x_i)$ on an edge selected at random;
17. end.
```

Fig. 4. New insertion operation.

inserted on the same depth, so a set O, which elements are edges on which $V(x_i)$ can be inserted, is created (*l.* 8 in Fig. 4). If $O \neq \phi$, an edge $e \in R$ is selected at random, then $V(x_i)$ is inserted on it. Otherwise if $R = \phi$, an variable is selected again because it cannot be inserted (*l.* 14 in Fig. 4). When x_i is not used, $V(x_i)$ on an edge selected at random is inserted (*l.* 16 in Fig. 4).

This algorithm can construct MDDs with dynamic variable ordering.

2) Mutation (Fig. 3(b)): Mutation operation is a change in the destination of one randomly selected edge of an non-terminal vertex to a randomly selected subordinate non-terminal or terminal vertices.

3) Deletion (Fig. 3(c)): Deletion operation deletes a randomly selected non-terminal vertex. The edges connected to the deleted vertex are set to connect to the vertices to which the edges of the deleted vertex pointed.

4) Replacement (Fig. 3(d)): Replacement operation randomly selects two variables (x_i, x_j) that are to be processed and swaps them. That is to say, after the replacement operation, the vertices that originally processed x_i treat x_j, and the vertices that originally processed x_j treat x_i.

5) Inversion (Fig. 3(e)): Inversion operation exchanges the destinations of the edges of a randomly selected non-terminal vertex. If the vertex has three or more edges, the edge destinations are reassigned randomly.

IV. Evolution Simulation Experiments

A. Experiment Overview

The evolutionary MTBDDs showed better performance than the evolution of finite state automata and classifier system [21], [22]. In this paper, to confirm the effectiveness of EMDDs, we conducted simulation experiments on obtaining motion representation for a humanoid robot standing up from the position of being seated in a chair (Fig. 5), and compared the results of EMDD and the evolutionary MTBDD (EMTBDD) [11]. The simulation was done using the virtual body of the humanoid robot HOAP-1 (Humanoid for Open Architecture Platform) made by Fujitsu Automation Ltd. Figure 6 shows HOAP-1. The robot is 48 centimeters tall, weighs 6 kilograms, has 20 DOFs, and has 4 pressure sensors each on the soles of its feet. Additionally, angular rate and acceleration sensors are mounted in its chest. To simulate evolution of robot motion, we used the Open Dynamics Engine [23].

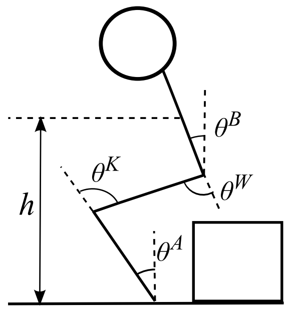


Fig. 5. Standing up from a chair.

Fig. 6. Overview of HOAP-1 (Humanoid for Open Architecture Platform).

TABLE I
Correspondence between angle data and variable values.

Sensed degree	EMDD	EMTBDD
	y	(x_0, x_1)
$\theta_{\min} \leq \theta < \frac{1}{4}\theta_{\max} + \frac{3}{4}\theta_{\min}$	0	$(0, 0)$
$\frac{1}{4}\theta_{\max} + \frac{3}{4}\theta_{\min} \leq \theta < \frac{1}{2}\theta_{\max} + \frac{1}{2}\theta_{\min}$	1	$(0, 1)$
$\frac{1}{2}\theta_{\max} + \frac{1}{2}\theta_{\min} \leq \theta < \frac{3}{4}\theta_{\max} + \frac{1}{4}\theta_{\min}$	2	$(1, 0)$
$\frac{3}{4}\theta_{\max} + \frac{1}{4}\theta_{\min} \leq \theta < \theta_{\max}$	3	$(1, 1)$

The robot control time interval was taken as $\Delta t = 0.01$ [s]. One trial was ended when the robot fell down or time exceeded 10.00 [s]. The fitness was the sum of the values of the robot's chest position $h(t)$ [m] at all times t during a trial ($Fitness = \sum_t h(t)$)), and the standing-up motion was defined as the decision diagram that outputs action whose fitness was 320.0 or higher.

The population size in an EMDD and an EMTBDD was $|U| = 50$ and $|P| = 3$.

B. Robot Control with EMDD

The robot using an EMDD obtains the following data from sensors.

$$y(t) = (y^W, y^K, y^A, y^B), \tag{1}$$

where, $y \in \{0, 1, 2, 3\}$, and y^W, y^K, and y^A are variables that represent the angles of the waist, knee, and ankle, respectively; y^B represents the pitch of the body. These variable values are determined from the current joint angle values as shown in second column of Table I.

In that table, $\theta_{\min}$ and $\theta_{\max}$ are limiting values of the motor range of motion, and they are defined as $\theta^W_{\min} = -80.0$ [deg], $\theta^W_{\max} = 70.0$ [deg], $\theta^K_{\min} = 0.0$ [deg], $\theta^K_{\max} = 120.0$ [deg], $\theta^A_{\min} = -60.0$ [deg], $\theta^A_{\max} = 60.0$ [deg], $\theta^B_{\min} = -60.0$ [deg],

TABLE II
TYPES OF BEHAVIOUR.

Terminal vertex	Motor output		
	Waist	Knee	Ankle
a_0	$+$	$+$	$+$
a_1	$+$	$+$	$-$
a_2	$+$	$+$	0
a_3	$+$	$-$	$+$
a_4	$+$	$-$	$-$
a_5	$+$	$-$	0
a_6	$+$	0	$+$
a_7	$+$	0	$-$
a_8	$+$	0	0
a_9	$-$	$+$	$+$
a_{10}	$-$	$+$	$-$
a_{11}	$-$	$+$	0
a_{12}	$-$	$-$	$+$
a_{13}	$-$	$-$	$-$
a_{14}	$-$	$-$	0
a_{15}	$-$	0	$+$
a_{16}	$-$	0	$-$
a_{17}	$-$	0	0
a_{18}	0	$+$	$+$
a_{19}	0	$+$	$-$
a_{20}	0	$+$	0
a_{21}	0	$-$	$+$
a_{22}	0	$-$	$-$
a_{23}	0	$-$	0
a_{24}	0	0	$+$
a_{25}	0	0	$-$
a_{26}	0	0	0

and $\theta^B_{\max} = 60.0$ [deg], respectively. The variables of the EMDDs are to be placed from depth 0 to depth 3 because the total number of variables is 4.

The robot action output is shown in Table II, where the plus signs ('$+$') signify that the joint is moved at 20.0 [deg/s] and the minus signs ('$-$') signify that the joint is moved at -20.0 [deg/s].

C. Robot Control with EMTBDD

The robot using an EMTBDD obtains the following data from sensors.

$$x(t) = (x^W_0, x^W_1, x^K_0, x^K_1, x^A_0, x^A_1, x^B_0, x^B_1), \qquad (2)$$

where, $x_i \in \{0, 1\}$, and x^W_i, x^K_i, and x^A_i are variables that represent the angles of the waist, knee, and ankle, respectively; x^B_i represents the pitch of the body. These variable values are determined from the current joint angle values as shown in third column of Table I.

The robot action output, which is the same as for EMDD, is presented in Table II.

D. Probabilities of Genetic Operations

We do not suppose that the method using the genetic operations described above generates a huge variety of individuals if there are a few vertices in the decision diagram. Therefore, the probability of insertion operation, which increases vertices in decision diagrams, is high in early stage of the experiment, when decision diagrams have a few vertex. On the other hand, the probability of deletion operation, which decreases vertices,

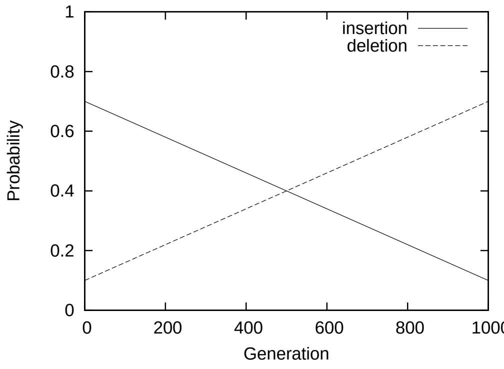


Fig. 7. Probabilities of insertion and deletion operations.

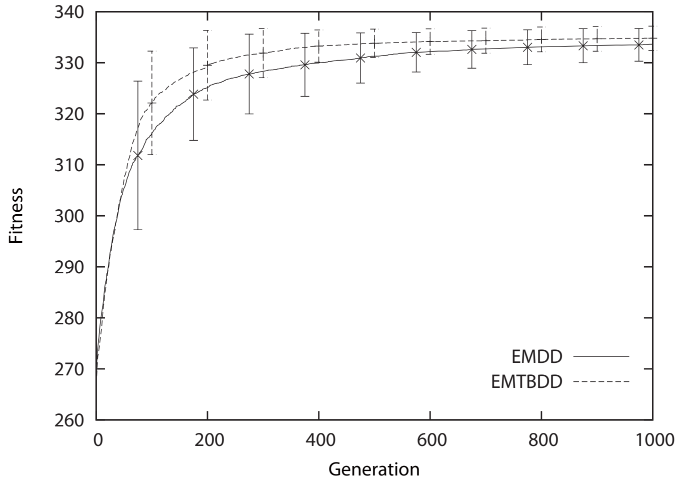


Fig. 8. Relation of generations and fitness (average for 150 repetitions).

is high to generate decision diagrams which have fewer vertices. However, the decrease of vertices decelerates evolution. Therefore, the probability of deletion is high in the final stage, when decision diagrams finish evolving. In the experiment, the probabilities of insertion and deletion vary with the number of generations as shown in Fig. 7. The probabilities of mutation, inversion, and replacement operations are 0.10, 0.05, and 0.05, respectively.

V. Experimental Results

This section presents the results of evolution simulation experiments using EMDDs and EMTBDDs. The evolution simulation experiments were performed 150 times for EMDDs and EMTBDDs.

A. EMDD Experimental Results

All experiments using EMDDs acquired robot motion representation. The standing-up motion acquisition required 155.1 ± 145.0 (mean $\pm$ S.D.) generations. The total number of non-terminal vertices at the 1000th generation was 8.5 ± 3.6 (mean $\pm$ S.D.). The solid line in Fig. 8 represents the relation between number of generations and fitness for EMDDs. The numbers of occurrences of each variable at each depth for experiments are given in Table III. The motion of the best

TABLE III
VARIABLES AND DEPTHS OF OCCURRENCE USING EMDDS (AVERAGE FOR 150 REPETITIONS).

Variables	Depth					Mean	S.D.
	0	1	2	3	Not used		
y^W	20	31	55	41	3	1.80	1.00
y^K	21	47	26	46	10	1.69	1.09
y^A	41	38	41	29	1	1.39	1.09
y^B	68	34	28	19	1	0.99	1.08

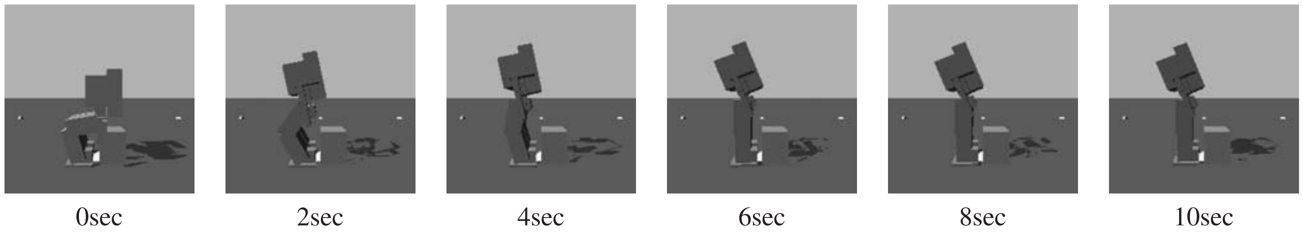


Fig. 9. Evolution results with EMDD (1000th generation).

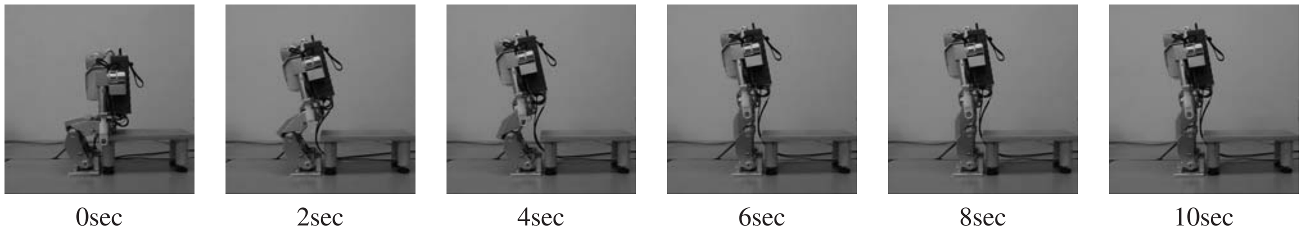


Fig. 10. Motion of HOAP-1 robot using EMDD of Fig. 11.

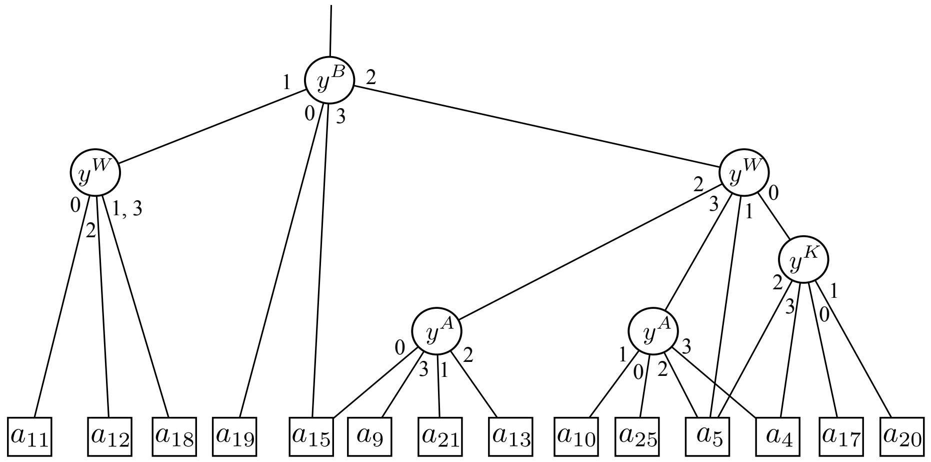


Fig. 11. Acquired EMDD for standing-up motion.

individuals at the 1000th generation is presented in Figs. 9 and 10, where we can see that the robot maintains balance while standing up. The EMDD that outputs this motion is shown in Fig. 11.

Here, we try to apply the EMDD for HOAP-1 to another robot, the Kondo KHR-2HV (Fig. 12) [24]. The robot is 35.3 centimeters tall, weighs 1.27 kilograms, and has 17 DOFs. Figure 13 shows the motion result using the EMDD in Fig. 11. You can see that the Kondo KHR-2HV stood up from a chair. This result indicates that motion representation by EMDDs outputs generalized actions.

B. EMTBDD Experimental Results

All experiments using EMTBDD acquired the standing-up motion. The standing-up motion acquisition required 99.9 ± 68.7 (mean $\pm$ S.D.) generations. At the 1000th generation, the total number of variable vertices was 12.8 ± 5.3 (mean $\pm$ S.D.). The broken line in Fig. 8 represents the relation of the number of generations and fitness when EMTBDD is used. The numbers of occurrences of each variable at each depth for experiments are given in Table IV. The motion of the best individuals at the 1000th generation is presented in Fig 14. You can see that Figs. 9 and 14 have similar look because all EMDDs and EMTBDDs acquired the motion representation in the experiment. The EMTBDD that outputs this motion is

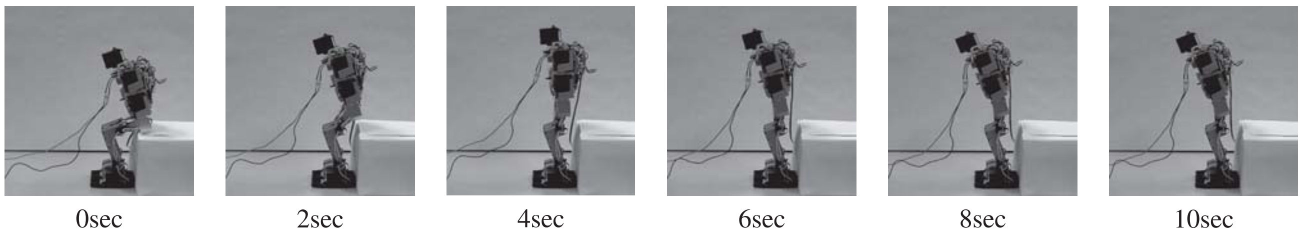


Fig. 13. Motion of KHR-2HV robot using EMDD of Fig. 11.

TABLE IV
VARIABLES AND DEPTHS OF OCCURRENCE USING EMTBDDS (AVERAGE FOR 150 REPETITIONS).

Variables	Depth									Mean	S.D.
	0	1	2	3	4	5	6	7	Not used		
x_0^W	12	15	21	17	21	18	17	14	15	3.57	2.14
x_1^W	9	25	16	21	20	19	14	7	19	3.27	2.00
x_0^K	22	20	19	11	27	18	16	3	14	2.99	2.09
x_1^K	12	15	18	31	17	24	14	8	11	3.40	1.96
x_0^A	10	17	19	15	16	13	14	18	28	3.60	2.26
x_1^A	27	23	18	22	23	16	7	5	9	2.65	2.01
x_0^B	22	17	20	10	16	12	19	4	30	2.94	2.22
x_1^B	36	18	19	22	7	12	11	5	20	2.39	2.16

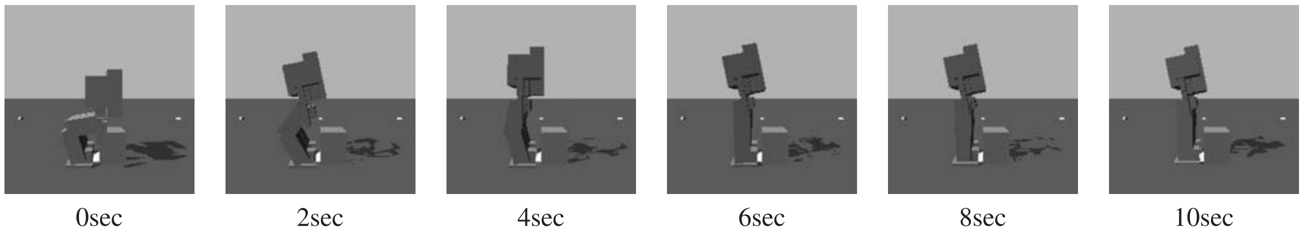


Fig. 14. Evolution results with EMTBDD (1000th generation).

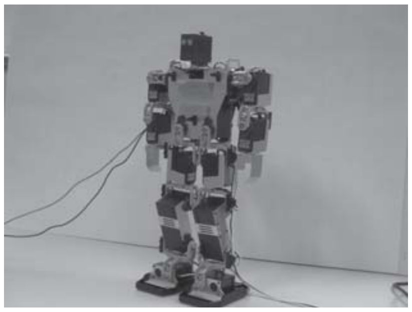


Fig. 12. Overview of Kondo KHR-2HV.

shown in Fig. 15.

C. Discussion

In the assigned position of variables of decision diagrams, there are two tendencies; variables that exert strong control on output of decision diagrams are positioned higher (closer to the root), and variables having local computation are positioned at similar depth [25]–[28]. These tendencies also appear in the assigned position of EMDD variables. Therefore, the assigned variable ordering is useful for humans to understand the importance of variables.

Table III and Table IV show the means and standard deviations of depth of each variable. To evaluate importance of each variable, we did the Friedman test, a nonparametric one-way analysis of variance, and the Scheffe test, a test of statistical significance, because of ordinal scales. Tables V and VI show these test results. From Table V, the importance of each variable of experiments using EMDDs is considered as

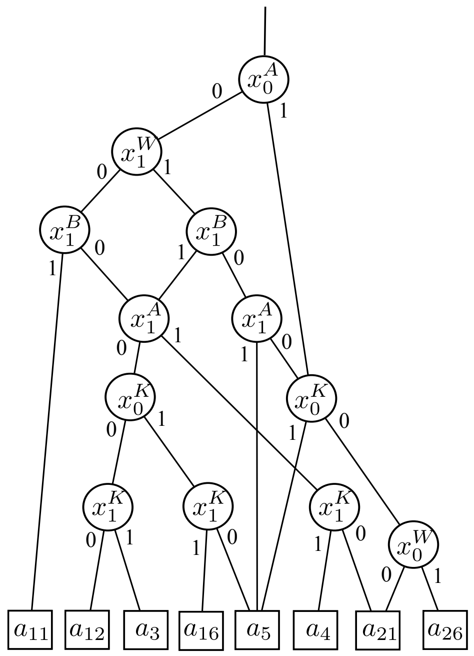


Fig. 15. Acquired EMTBDD for standing-up motion.

TABLE V
SCHEFFE TEST RESULT (EMDD). THE MARK '*' AND '**' MEAN SIGNIFICANT DIFFERENCE AT $p < 0.05$ AND $p < 0.01$, RESPECTIVELY.

	y^W	y^K	y^A	y^B
y^W	–	0.8794	0.0133	0.0000
y^K		–	0.1187	0.0000
y^A	*		–	0.0141
y^B	**	**	*	–

TABLE VI
SCHEFFE TEST RESULT (EMTBDD). THE MARK '*' AND '**' MEAN SIGNIFICANT DIFFERENCE AT $p < 0.05$ AND $p < 0.01$, RESPECTIVELY.

	x_0^W	x_1^W	x_0^K	x_1^K	x_0^A	x_1^A	x_0^B	x_1^B
x_0^W	–	0.04	0.05	0.04	0.05	1.00	0.88	0.40
x_1^W	*	–	0.86	0.01	0.06	0.02	0.06	0.74
x_0^K			–	0.42	0.69	0.16	0.00	0.01
x_1^K	*	*		–	0.01	0.60	0.37	0.05
x_0^A				**	–	0.07	0.03	0.66
x_1^A		*				–	0.72	0.00
x_0^B			**		*		–	0.00
x_1^B			*			**	**	–

follows:

$$y^B > \{y^A, y^K, y^W\}. \quad (3)$$

In the result of EMDD, the joint angle is represented by a single variable, so it is not difficult to understand the importance of each joint. From the EMDD variable importance, the body pitch is the most important than all of the other joints, and the ankle joint is more important than waist joint. Similarly, considering the importance of each variable in EMTBDDs, there was no variable making significant difference to all other variables. In the test result of EMTBDDs, you can not see what is the most important joint or control variable at first glance. This indicates that it is difficult to understand the importance of each joint through motion representation by EMTBDDs. From the above discussion, EMTBDDs do not work well to understand the importance of joints, on the other side EMDDs work well. Therefore, we can conclude that the EMDD method is proper to discover important joints in comparison with EMTBDD.

Second, we discuss the evolution of EMDDs. The evolution process before the EMDD in Figure 11 is shown in Figure 16. You can see that the structure of EMDD becomes more complex at the beginning of the evolution, but the final MDD in Figure 11 is very compact. Figure 17 shows the number of non-terminal vertices of the best individual in each generation. The evolution increases the number of non-terminal vertices until about 600th generation, and then decreases it. This result arises from the setting of probabilities of insertion and deletion operations (see Figure 7). By this setting, we can consider that the compact motion representation by MDD was acquired.

Third, we verify and discuss the effectiveness of proposed insertion operation. We consider that appropriate motion representation is acquired because variable ordering in MDD is changed dynamically by using the dynamic insertion operation. Figure 18 shows the results when variable ordering is fixed as 'Best order' ($y^B - y^A - y^K - y^W$), and 'Worst order' ($y^W - y^K - y^A - y^B$). These ordering were decided from the

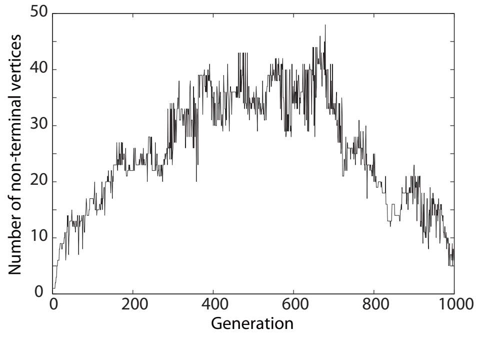


Fig. 17. The number of non-terminal vertices.

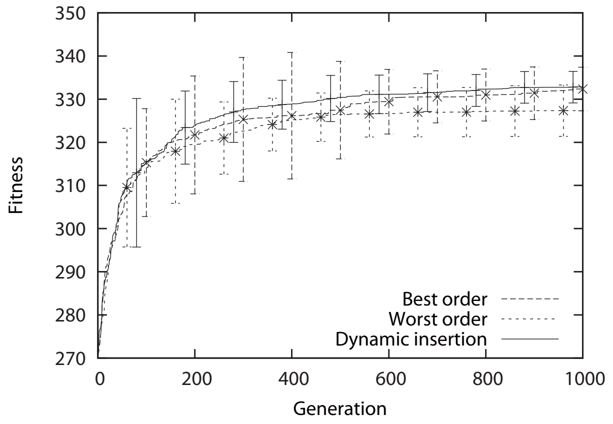
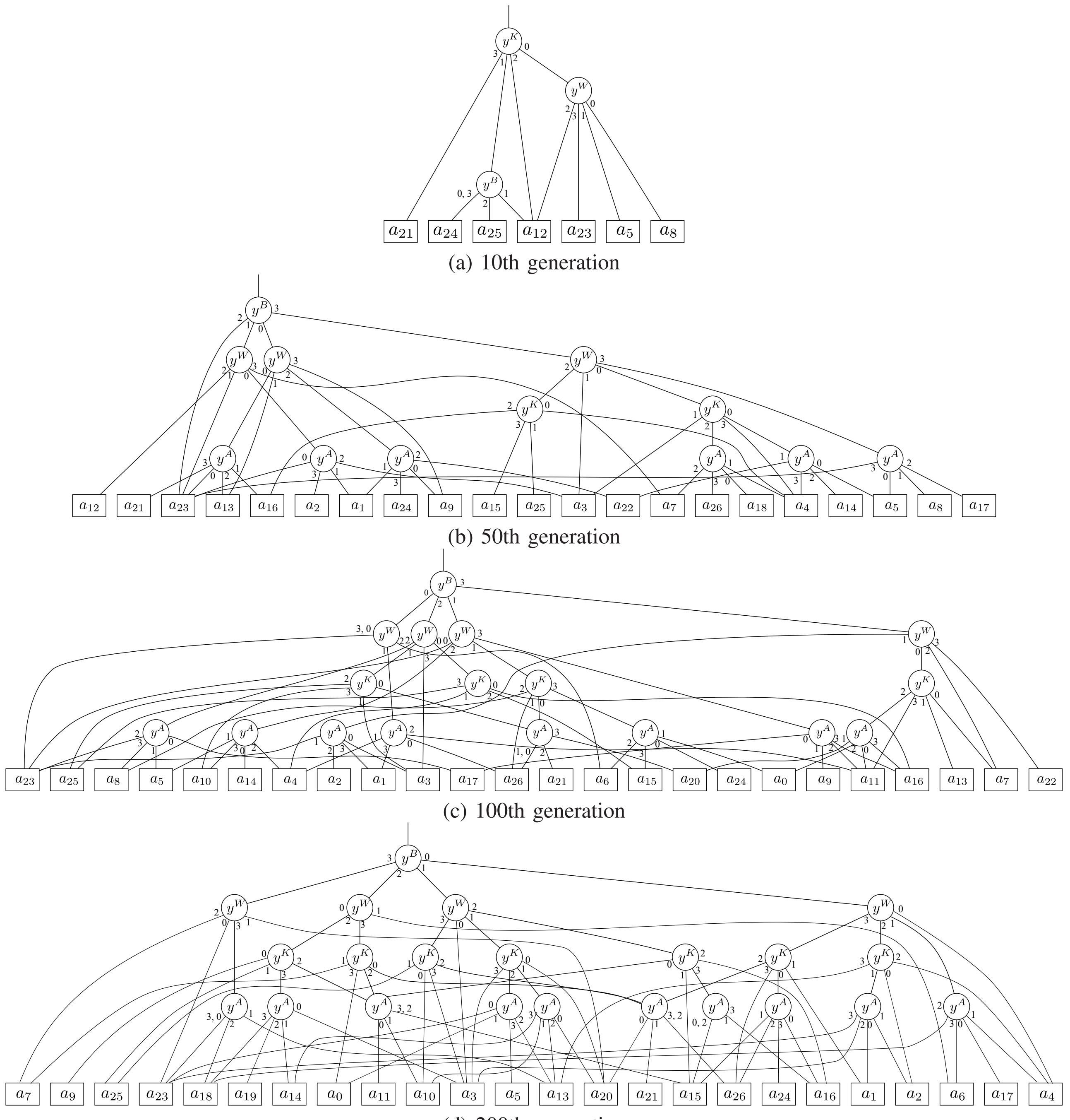


Fig. 18. Comparison with EMDDs via fixed variable ordering (average for 20 repetations).

mean of depth (see Table III). The solid line in the figure is the same as the line of EMDD in Figure 8. Note that if variable ordering is fixed, the replacement operation cannot use. In this experiment, we divided the probability of replacement up evenly between other four operations. This figure shows that the fitness of the dynamic insertion is better than other two fixed methods. In the line of worst order, variables that exert strong control on output of MDDs are positioned lower, so this approach did not work well. On the other hand, the best order earned the fitness nearly equal to dynamic insertion in the 1000th generation. However, around 150 to 500 generation, its fitness is lower than dynamic insertion. We consider that the best final solution can be found by the best order, but local solution (e.g. the motion of half rise from the chair) may require another order. These results indicate that our method can use effectively to the optimization of the problem that not only truth table but also variable ordering is unknown.

Next, we compare learning efficiency between EMDDs and EMTBDDs. As you can see from Fig. 8, the learning efficiency of EMDDs is lower than EMTBDDs. There is significant difference ($p < 0.000$) between the required generations of EMDDs (155.1±145.0) and EMTBDDs (99.9±68.7) to obtain standing-up motion. Similarly, there is significant difference ($p < 0.000$) between their fitnesses in 1000th generation: EMDD; 333.6 ± 3.15, EMTBDD; 334.8 ± 2.38. However, the numerical difference of fitness is very small, so EMDDs have little direct difference on the learning. If you want to


(a) 10th generation

(b) 50th generation

(c) 100th generation

(d) 200th generation

Fig. 16. EMDDs of best individuals in each generation.

obtain motion quickly and do not need the important joint information, you may use EMTBDDs.

Finally, we discuss to introduce memetic computing to the evolution of MDDs. The earliest and fastest growing area of memetic computing research is memetic algorithm [29]–[31]. The memetic algorithm is a hybrid global-local heuristic search methodology, and is variously used for design of optimal control systems [32], fuzzy rule selection [33], robust design [34], and so on. We evolve MDDs by the following memetic algorithm.

1) Initialize generation $g \leftarrow 0$.
2) Generate the initial population $U' = \{u'_1, u'_2, ..., u'_{n'}\}$, where n' is the number of individuals.
3) For i-th individual, apply local search and create a set $R_i = \{r_1, r_2, ..., r_l\} \cup \{u'_i\}$, where l is the number of neighborhood individuals of u'_i.
4) For each R_i, choose the best individual b_i by computing the fitness of the individuals in R_i.
5) Generate the next-generation set $U' = \{b_1, b_2, ..., b_{n'}\}$.
6) Increment $g \leftarrow g + 1$, and return to step 3.

Here, in the problem of obtaining robot motion representation, not only global search but also local search requires dynamics

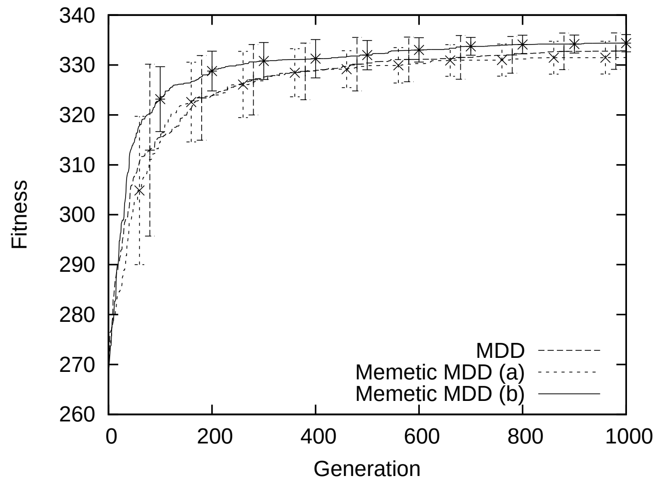


Fig. 19. Comparison between EMDD and memetic EMDDs (average for 20 repetaions).

computation.

For the memetic experiments, we prepared two parameters: (a) $|U'| = 10$ and $|R_i| = 4$; (b) $|U'| = 5$ and $|R_i| = 9$. These parameters were decided to equalize the number of dynamics computations (i.e., for example in memetic EMDD (a), one group which is made by an individual and four local searches needs 5 dynamic computations). Figure 19 shows the experimental results. You can see that memetic EMDD (b) can obtain better performance than other two methods. In this evolutionary experiment, the variable y^B is more important than other three variables, and it should be located at depth 0. In memetic EMDD (b), the parameter of $|U'| = 5$ suffices for locating y^B at depth 0, because the probability of y^B being located at depth 0 is a 1 in 4. Moreover, local search for each individual is done enough, so we consider that memetic MEDD (b) earned higher fitness.

From the above results, effectiveness of memetic algorithm in motion control of humanoid robots using EMDD was confirmed.

VI. Related Works

Many methods for robot control tasks exist; methods using reinforcement learning [35]–[38], genetic programming [39]–[41], decision tree learning [42], [43], and so on. The EMDD has a feature; it obtains simpler motion representation than these methods, because it requires discrete state space and actions. The EMDD can decide the number of output variables arbitrarily, so we consider that it can treat complex motion learning via changing the number of terminal vertices. Kuwayama et al. proposed a motion learning method using the c4.5 decision tree generator [42], [43]. They indicated that motion obtained by decision tree learning is stable. However, to use the method, preparing positive and negative examples and then extracting guidepost for motion generation are required before decision tree learning. Preparing negative examples is done easily, but assembling positive examples is very difficult, especially in learning of complex motion such as locomotion. Moreover, developing a motion generator using guidepost is also needed. On the other hand, the EMDD evolves a tree which outputs appropriate motion, instead of preparing motion before learning. That is, compared with the method by Kuwayama et al., the EMDD has two merits; it does not require assembling motions, and can obtain motion representation dynamically. The EMDD has two tendencies described in Section V-C, and may represent implicit knowledge of motion of human beings or robots.

VII. Conclusion

We proposed a method for obtaining robot motion representation through the evolution of MDDs. To confirm the effectiveness of this method, we performed experiments on motion representation acquisition for the standing-up motion of a humanoid robot. As the result of the experiments, we acquired robot motion in all experiments. We performed the same experiments using the evolutionary MTBDD, and we confirmed that the evolutionary MDD is also more effective than the evolutionary MTBDD in discovering important variables from the decision diagrams. Our future work is acquisition of motion representation for more complex motions than the standing-up motion.

Acknowledgment

A part of this study was carried out by the Grant-in-Aid for Young Scientists (A) of the Ministry of Education, Culture, Sports, Science and Technology (No. 20680014).

References

[1] J. Sung, R. E. Grinter, H. I. Christensen, and L. Guo, “Housewives or Technophiles?: Understanding Domestic Robot Owners,” *ACM/IEEE International Conference on Human Robot Interaction* pp. 129–136, 2008.

[2] J. Forlizzi and C. DiSalvo, “Service Robots in the Domestic Environment: A Study of the Roomba Vacuum in the Home,” *ACM SIGCHI/SIGART Conference on Human-robot Interaction*, pp. 258–265, 2006.

[3] P. Saulnier, E. Sharlin and S. Greenberg, “Using Bio-electrical Signals to Influence the Social Behaviours of Domesticated Robots,” *ACM/IEEE International Conference on Human Robot Interaction*, 2009.

[4] K. Wada and T. Shibata, “Living with Seal Robots. Its Sociopsychological and Physiological Influences on the Elderly at a Care House,” *IEEE Transactions on Robotics*, vol. 23, no. 5, pp. 972–980, 2007.

[5] M. Kanoh, S. Iwata, S. Kato and H. Itoh, “Emotive Facial Expressions of Sensitivity Communication Robot “Ifbot”,” *Kansei Engineering International*, vol. 5, no. 3, pp. 35–42, 2005.

[6] M. Fujita, “On Activating Human communications with Pet-type Robot AIBO,” *Proceedings of IEEE*, vol. 92, no. 11, pp. 1804–1813, 2004.

[7] Y. Fujita, “Personal Robot PaPeRo,” *Journal of Robotics and Mechatronics*, vol. 14, no. 1, pp. 60–63, 2002.

[8] S. B. Akers, “Binary Decision Diagrams,” *IEEE Transactions on Computers*, vol. 27, no. 6, pp. 509–516, 1978.

[9] R. E. Bryant, “Graph-based Algorithms for Boolean Function Manipulation,” *IEEE Transaction on Computers*, vol. 35, no. 8, pp. 677–691, 1986.

[10] D. E. Knuth, “The Art of Computer Programming,” vol. 4, fascicle 1, *Addison-Wesley*, 2009.

[11] M. Kanoh and H. Itoh, “A New Dynamic Insertion Operation for n-BDD – Applying to Obtaining Robot Controller –,” *Journal of Japan Society for Fuzzy Theory and Intelligent Informatics*, vol. 20, no. 6, pp. 909–920, 2008 (in Japanese).

[12] P.W. C. Prasad, A. Assi, A. Harb, and V.C. Prasad, “Binary Decision Diagrams: An Improved Variable Ordering using Graph Representation of Boolean Functions,” *International Journal of Computer Science*, vol. 1, no. 1, pp. 1–7, 2006.

[13] S. J. Friedman and K.J. Supowit, “Finding the Optimal Variable Ordering for Binary Decision Diagrams,” *IEEE Transactions on Computers*, vol. 39, pp. 710–713, 1990.

[14] F. Somenzi, "Efficient Manipulation of Decision Diagrams," *International Journal on Software Tools for Technology Transfer*, vol. 3, pp.171-181, 2001.
[15] S. Panda and F. Somenzi, "Who Are the Variables in Your Neighborhood," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 74–77, 1995.
[16] R. Rudell, "Dynamic Variable Ordering for Ordered Binary Decision Diagrams," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 42–47, 1993.
[17] M. Sakai, Y. Tomoto, M. Kanoh, T. Nakamura, and H. Itoh: "Acquisition of Robot Control Rules by Evolving MDDs," *IEEE World Congress on Computational Intelligence*, pp.550-556, 2010.
[18] A. Srinivasan, T. Kam, S. Malik, and R. K. Brayton, "Algorithms for Discrete Function Manipulation," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 92–95, 1990.
[19] J. R. Koza, "Genetic Programming, On the Programming of Computers by Means of Natural Selection," *MIT Press*, 1992.
[20] B. Bollig, and I. Wegener, "Improving the Variable Ordering of OBDDs Is NP-Complete," *IEEE Transactions on Computers*, vol. 45, no. 9, pp. 993–1002, 1996.
[21] K. Moriwaki, N. Inuzuka, M. Yamada, K. Itoh, H. Seki, and H. Itoh, "Self Adaptation of Agent's Behavior using GA with n-BDD," IEEE International Workshop on Robot and Human Communication, pp. 96–101, 1996.
[22] K. Moriwaki, N. Inuzuka, M. Yamada, H. Seki, and H. Itoh, "A Genetic Method for Evolutionary Agents in a Competitive Environment," *Soft Computing in Engineering Design and Manufacturing*, pp. 153–162, 1997.
[23] Russell Smith, "Open Dynamics Engine." Available from: http:// www. ode. org/
[24] Kondo Kagaku Co.Ltd., Available from: http:// kondo-robot. com/ guide/ english.html
[25] M. Fujita, H. Fujisawa, and N. Kawato, "Evaluation and Improvement of Boolean Comparison Method based on Binary Decision Diagrams," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 2–5, 1988.
[26] S. Malik, A. Wang, R. Brayton, and A. Sangiovanni-Vincentelli, "Logic Verification using Binary Decision Diagrams in a Logic Synthesis Environment," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 6–9, 1988.
[27] S. Minato, "Techniques for BDD Manipulation on Computers," *Journal of Information Processing Society of Japan*, vol. 34, no. 5, pp. 593–599, 1993 (in Japanese).
[28] M. Yanagiya, "Combinational Optimization via BDD-based Procedures," *Journal of Information Processing Society of Japan*, vol. 34, no. 5, pp. 617–623, 1993 (in Japanese).
[29] P. Moscato, "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms," *Caltech Concurrent Computation Program*, Report 826, 1989.
[30] X. S. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan, "A Multi-Facet Survey on Memetic Computation," *IEEE Transactions on Evolutionary Computation*, Accepted in 2011 and In Press.
[31] Y. S. Ong, M. H. Lim, and X. S. Chen, "Research Frontier: Memetic Computation - Past, Present & Future," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 24–36, 2010.
[32] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, and M. Sumner, "A Fast Adaptive Memetic Algorithm for Online and Off line Control Design of PMSM Drives," IEEE Transactions on Systems, Man, and Cybernetics, Part B, vol. 37, pp. 28–41, 2007.
[33] H. Ishibuchi, and T. Yamamoto, "Fuzzy Rule Selection by Multi-objective Genetic Local Search Algorithms and Rule Evaluation Measures in Data Mining," Fuzzy Sets and Systems, vol. 141, pp. 59–88, 2004.
[34] Y. S. Ong, P. B. Nair, and K. Y. Lum, "Max-min surrogate-assisted evolutionary algorithm for robust design," IEEE Transactions on Evolutionary Computation, vol. 10, pp. 392–404, 2006.
[35] M. Asada, Y. Katoh, M. Ogino, and K. Hosoda, "A Humanoid Approaches to the Goal – Reinforcement Learning Based on Rhythmic Walking Parameters –", *RoboCup International Symposium*, 2003.
[36] Y. Nakamura, T. Mori, M. Sato, and S. Ishii, "Reinforcement Learning for a Biped Robot Based on a CPG-Actor-Critic Method," *Neural Networks*, vol. 20, no. 6, pp. 723–735, 2007.
[37] J. Morimoto and K. Doya, "Acquisition of Standup Behavior by a Real Robot Using Hierarchical Reinforcement Learning," *Robotics and Autonomous Systems*, vol. 36, no. 1, pp. 37–51, 2001.
[38] J. Morimoto and C. G. Atkeson, "Learning Biped Locomotion: Application of Poincare-map-based Reinforcement Learning," *Robotics and Automation Magazine*, IEEE, vol. 14, no. 2, pp. 41–51, 2007.
[39] T. Yanase and H. Iba, "Evolutionary Motion Design for Humanoid Robots, Frontiers in Evolutionary Robotics," Hitoshi Iba (Ed.), *Frontiers in Evolutionary Robotics*, I-Tech Education and Publishing, 2008. Available from: http://www.intechopen.com/articles/show/title /evolutionary_motion_design_for_humanoid_robots
[40] S. Kamino, H. Mitsuhashi, and H. Iba, "Integration of Genetic Programming and Reinforcement Learning for Real Robots," *Genetic and Evolutionary Computation Conference*, pp. 470–477, 2003.
[41] N. Ogihara and N. Yamasaki, "Generation of Human Bipedal Locomotion by a Bio-Mimetic Neuro-musculoskeletal Model," *Biological Cybernetics*, vol. 84, no. 1, pp. 1–11, 2001.
[42] K. Kuwayama, S. Kato, and H. Itoh, "A Concept Learning Based Approach to Motion Control for humanoid robots," *International Conference on Informatics in Control, Automation and Robotics*, vol. 2, pp. 335–338, 2004.
[43] K. Kuwayama, S. Kato, T. Kunitachi, and H. Itoh, "Motion Control for Humanoid Robots Based on the Motion Phase Decision Tree Learning," *IEEE International Symposium on Micro/Nanomechatronics and Human Science*, pp. 157–162, 2004.

Masashi Sakai received the B.E. degree from the Department of Computer Science, Nagoya Institute of Technology in 2010. He is currently pursuing M.E. degree at the Department of Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology. His research interests include robotics and genetic algorithm.

Masayoshi Kanoh received the B.S., M.S. and Ph.D. degrees in Engineering from Nagoya Institute of Technology, Japan, in 1999, 2001 and 2004, respectively. Between 2004 and 2010, he was an Assistant Professor at Chukyo University, where he is currently an Associate Professor. His research interests include human-robot interaction, intelligent robots and humanoid robots.

Tsuyoshi Nakamura received the Ph.D. degree from Nagoya Institute of Technology in 1998, studying computer graphics based on softcomputing. He joined Nagoya Institute of Technology as a research associate in 1998. In 2003 and now, he is an associate professor. His research interests include CG, CV softcomputing, and HAI(http://ai.web.nitech.ac.jp). He is a member of the IEEE and the ACM.