

# Context-Dependent DNA Coding With Redundancy and Introns

Peng Xiao, Prahlad Vadakkepat, and Tong Heng Lee

**Abstract**—Deoxyribonucleic acid (DNA) coding methods determine the meaning of a certain character in individual chromosomes by the characters surrounding it. The meaning of each character is context dependent, not position dependent. Although position-dependent coding is most commonly used in genetic algorithms (GAs), a context-dependent coding formation is in fact more closer to the natural DNA chromosome. With the context dependency, the DNA coding methods allow intron parts, redundancy, and variable string length in encoded strings while remaining compatible with the standard genetic operations. This paper tries to explicitly explore the influence of those special features of the DNA coding scheme. Two fundamental DNA coding methods (with and without the use of introns) are constructed and compared with the integer coding method, which lacks the features of interest. The performance of the proposed DNA coding methods is analyzed through the robot soccer role assignment problem. The context-dependent coding exhibits the advantages in handling the negative effect of epistasis. The redundancy and intron parts are helpful in preventing useful schemata from disruption and in increasing the population diversity. The variable length of the individual string enables GAs to evolve both the size and the structure of the fuzzy rule base.

**Index Terms**—Behavior-based architecture, deoxyribonucleic acid (DNA) coding, fuzzy control, genetic algorithm (GA), robotic soccer.

## I. INTRODUCTION

**H**UMANS have always looked toward nature for solutions to everyday problems. Evolutionary computation is a collection of computational algorithms derived from observing nature, in particular the evolution of population through natural selection [1]–[3]. Nature, through constraints in the environment, is able to select the offspring from individuals with the most suitable attributes. In the same way, an optimum solution to computational problems can be selected by applying constraints to a population over many generations.

While the studies and theories of C. Darwin and G. Mendel had been the basis in the science of selection, a great breakthrough in genetics came in 1953 when J. Watson and F. Crick unveiled the secret of life, i.e., deoxyribonucleic acid (DNA)

Manuscript received April 6, 2005; revised August 19, 2005, February 1, 2007, and July 17, 2007. This work was supported by the University Faculty Research Fund under Grant R-263-000-292-112. This paper was recommended by Associate Editor C.-T. Lin.

P. Xiao is with the School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan 430074, China.

P. Vadakkepat and T. H. Lee are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117576 (e-mail: prahlad@nus.edu.sg; eleleeth@nus.edu.sg).

Color versions of one or more of the figures in this paper are available online at <http://ieeexplore.ieee.org>.

Digital Object Identifier 10.1109/TSMCB.2007.912741

[4]. Henceforth, researchers have begun to understand how genetic information is coded and passed on from one generation to the next.

Motivated by the gene expression, which involves the translation of nucleotide sequences of DNA into amino acid sequences, the DNA-like coding method has been proposed for evolutionary computing [5]–[8]. DNA-coded strings are used to represent the fuzzy “if-then” rule bases [5], [6], [8]–[10] or the production rules of the L-system [7] as the individuals of the genetic algorithm (GA). The DNA coding method allows flexible representation, overlapped and redundant coding, variable string length, and no restrictions on the crossover point. The simulation results suggested that the redundancy and overlapping in DNA coding worked well for fuzzy rule discovery [5]. However, the reported works lack in explicit explanation.

This paper aims to project the DNA coding in a more general scheme, which is characterized by the following specific features: context dependency, intron parts, redundancy, and variable coding string length. It aims to explore and explain the influence of these features on the evolutionary algorithms’ performance. With and without the features of interest, two fundamental DNA coding methods and an integer coding method are designed for the purpose of comparison, which is performed via a simulation study in the context of role assignment in a robot soccer system.

In Section II, the general introduction of the normal coding methods for GA is provided. Section III summarizes the background knowledge on DNA and the basics of DNA coding. Section IV introduces the robot soccer system with the fuzzy behavior-based architecture. Section V describes the three coding methods that are compared in the role assignment task. The simulation is presented in Section VI, and the results are analyzed. Section VII concludes this paper.

## II. CODING METHODS FOR GA

For GAs, different coding methods are developed for different problems. The most popular coding methods include binary coding, real-value coding, and permutation coding.

The binary coding is simple and most widely used. An alternative of binary coding is gray coding, which is designed to eliminate the Hamming cliff problem [11]. Binary coding is well suited for problems where the solutions are pseudo-Boolean. Integer or real-valued variables can also be represented by binary strings. However, a bit change in the binary string may cause a large change in the represented integer or real number. Furthermore, both binary and gray coding methods

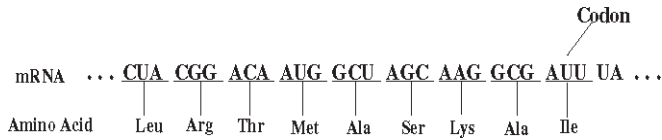


Fig. 1. Codons in mRNA and corresponding amino acids.

may make the representation space much more complicated than the searching space.

The real-value coding method is characterized by the direct operation on a real-valued solution vector. For real-valued problems, there has been a trend away from binary coding to real-value coding in GA research [12]–[14]. The real-value coding outperforms the binary coding in real-valued numerical optimization problems because it is more convenient, consistent, and concise in representation [15].

The permutation-based coding method is usually used to represent logical solutions for scheduling problems and classic combination problems. GAs that use permutation coding are referred to as ordering GAs [16]. With permutation coding, classic genetic operators such as simple crossover and mutation may fail to generate valid offsprings. Specialized recombination operators have been proposed, including order crossover 1 [17], [18], order crossover 2 [19], position crossover [19], partially map crossover [18], and maximal preservative crossover [20].

There are other coding methods, such as mixed-integer coding [21], intron coding [22], [23], and parse tree coding [24]. These coding methods are specialized for catering to different kinds of problems. One class of interesting coding method that is discussed in depth in this paper is the DNA-like coding method, which is motivated by the transcription of DNA to messenger ribonucleic acid (mRNA) and the translation of mRNA to proteins [5]–[8].

### III. DNA-LIKE CODING METHOD

#### A. DNA and Messenger RNA

Before discussing the DNA-like coding method for GA, it is worthwhile to introduce the natural DNA coding mechanism. The DNA is a double-stranded sequence of four bases, which are adenine (A), guanine (G), cytosine (C), and thymine (T). A portion of the DNA sequence (a gene) encodes the information that determines the sequence of amino acids of the protein, i.e., the engines of life. It is for this reason that scientists use the expression *genetic code*.

In the synthesis of proteins, the pre-mRNA is first constructed from the DNA. It contains the same genetic code as the DNA except that all the T bases are replaced by U (uracil) bases. After the first synthesization, splicing is performed on the pre-mRNA to remove some unused parts to create the final mRNA.

In the mRNA, three successive bases called codons are sequentially allocated, as shown in Fig. 1. Most of the codons, e.g., Leu and Arg, are abbreviations for the amino acids (Table I), except for one special *Terminator* codon (Ter). The process of synthesizing the mRNA based on DNA is terminated by the presence of a Ter.

There are 20 types of codons for the amino acids and the terminator, which are represented by 64 three-letter triplets

TABLE I  
GENETIC CODE OF AMINO ACIDS

	U	C	A	G		
U	Phenylalanine (Phe)	Serine (Ser)	Tyrosine (Tyr)	Cysteine (Cys)	U	
	Leucine (Leu)		Terminator (Ter)	Ter	C	
C	Leu	Proline (Pro)	Histidine (His)	Arginine (Arg)	A	
			Glutamine (Gln)		G	
			Asparagine (Asn)		Ser	U
A	Isoleucine (Ile)	Threonine (Thr)	Lysine (Lys)	Arg	C	
	Methionine (Met)		Aspartic acid (Asp)		Glycine (Gly)	A
G	Valine (Val)	Alanine (Ala)	Glutamic acid (Glu)	Glycine (Gly)	G	
						U
						C

\*U (uracil) replaces T (thymines) in mRNA

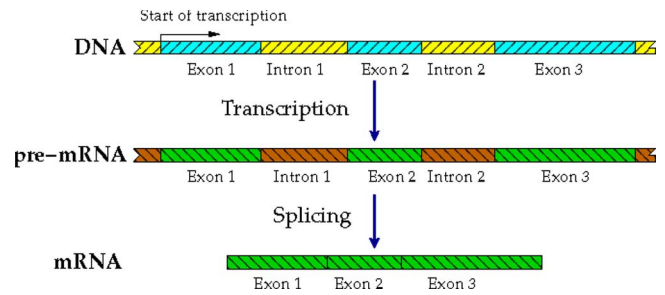


Fig. 2. Exon and intron.

(Table I). That results in a certain extent of redundancy, which helps to relax the accuracy requirement and to increase the flexibility. For instance, a point mutation on the last letter is unlikely to produce a malfunctioning protein. This property is important and can be exploited in the DNA coding method.

Another aspect worthy of mention is the existence of many noncoding sequences in the pre-mRNA after the first synthesization. These noncoding sequences are usually referred to as introns while the coding sequences are named as exons. All the introns are precisely spliced to produce the final mRNA (Fig. 2). The reason for the existence of introns is still being debated. It is possible that introns contain sequences that control the gene activity in the splicing process [25], [26]. The different ways of splicing the introns increase the varieties in genes. Coding methods that utilize introns have been proposed in [22], [23], and [27]. In this paper, the effects of introns on coding methods are explored.

#### B. Basics of Coding

The basic idea of DNA coding for GA can be illustrated by the DNA representation of the fuzzy if–then rules [5], [6]. The parameters and variables of the fuzzy system values are mapped to codons in the DNA chromosomes. Decoding starts with the reading of the DNA string from the beginning or a predefined start point, such as an “ATG” codon (Fig. 3). Every triplet codon is translated into a parameter or logic operation according to the predefined coding table (Table II). The whole procedure is depicted in Fig. 4.

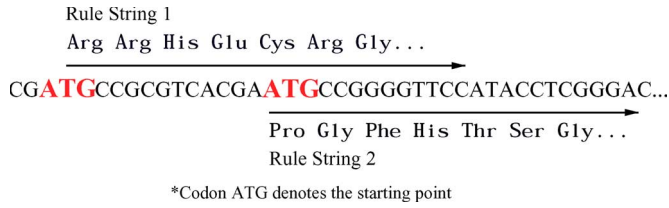


Fig. 3. Reading and translation of genes.

TABLE II  
POSSIBLE SAMPLE TRANSLATION TABLE

	Input	Membership Function		And/Or/Then	Output	Weight	
		Center	Discourse				
Phe	Input 1	Position 1	Width 1	AND	Output 1	0.1	
Leu						0.2	
Ile		Position 2				0.3	
Val						0.4	
Ser	Input 2	Position 3	Width 2	OR	Output 2	0.5	
Pro						Position 4	0.6
Thr							0.7
Tyr	Input 3	Position 5	Width 3	THEN	Output 3	0.8	
His						Position 6	0.9
Gln		1.0					
Asn							
Lys							
Asp							
Glu							
Cys							
Trp							
Arg							
Gly							

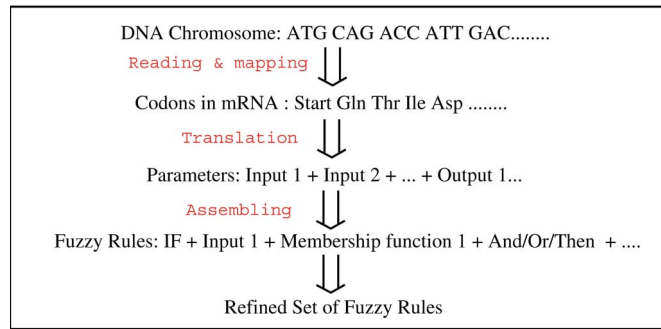


Fig. 4. Translation from DNA to fuzzy rules.

In Table II, every codon has different possible meanings. The exact meaning of a codon depends on the meaning of the codon just before it. For instance, the codon “His” represents the central position of the membership function if it follows a codon representing an input, and the codon after “His” will be translated as the discourse spread of the membership function. On the other hand, “His” can be decoded as another input while its preceding codon represents the fuzzy relation “AND.” In general, the meaning of a codon is context dependent.

In this paper, a more general DNA coding mechanism is explored, which is characterized by the features of context dependency, intron parts, redundancy, and variable string length. Under such a scheme, total flexibility in the codons’ definitions and translation rules are allowed. Even the codons themselves can be redefined as two-letter couplets or any *n*-letter combinations from the alphabet {A, C, G, T}, other than the natural amino acid triplets. Some codons may share the same meaning, which results in redundancies as in natural chromosomes. Some others may stand for nothing in particular, just like introns in the

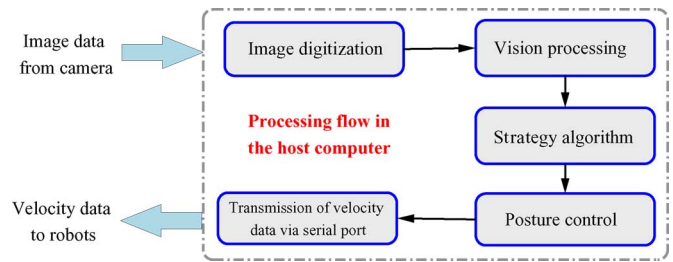


Fig. 5. Overview of the robot soccer system process.

mRNA. The encoded individual string can be of variable length. Since there is no restriction on the formation of the individual string, the DNA-coded strings are compatible with the simple GA (SGA) operators. The features of the DNA-like coding are further analyzed with a case study.

#### IV. ROBOT SOCCER SYSTEM WITH FUZZY BEHAVIOR-BASED ARCHITECTURE

The proposed DNA coding methods are utilized to evolve the fuzzy behaviors for robot role assignment within a robotic soccer environment. The hardware setting for robot soccer consists of three parts: the vision system, the host computer, and the robots. The vision system consists of a CCD camera, a frame grabber card, and drivers. The camera is mounted above the robot soccer field. The image of the entire playground is captured every 20 ms and sent to the host computer. As the “brain” of the system, the host computer manipulates the vision data with its vision-processing module, decides the robots’ subsequent actions, and computes the relative velocity settings of the robots. The resultant velocity commands are sent to the robots via radio-frequency communication. The structure of the control software in the host computer is summarized in Fig. 5. The soccer robots move on the playground following the host computer’s commands and transfer the team strategy into reality.

Based on [28], a fuzzy behavior architecture has been set up for the robot soccer system. Fig. 6 shows the behavior architecture with examples of behaviors at each level of the hierarchy. At the top level, the team behavior represents the team strategy, which can be aggressive or defensive depending on the match situation. At the second layer, the team is decomposed into four robot roles: attacker, midfielder, defender, and goalie. Which role a robot has to perform is decided using fuzzy logic. The four roles are broken into 12 modular behaviors at the third level, such as “shoot,” “chase,” “pass,” “block,” etc. Each role is fulfilled by a set of behaviors. Some of the behaviors (“avoid wall” and “frustration”) are utilized in most of the roles, while the others (“shoot”/“guard”) belong to specific roles (“attacker”/“defender”). The bottom level consists of 14 primitive actions, including “go-position,” “go-angle,” and “spin.”

Fig. 6 also depicts the utilized behavior coordination mechanisms. The fuzzy rule-base coordination method is widely used in each level. The activity and action contribution method is used only for reactive behaviors, such as “avoid wall” and “shun robot.” The goalie adopts a straightforward “if-else-if logic” coordination method for simple but quick reactive motions.

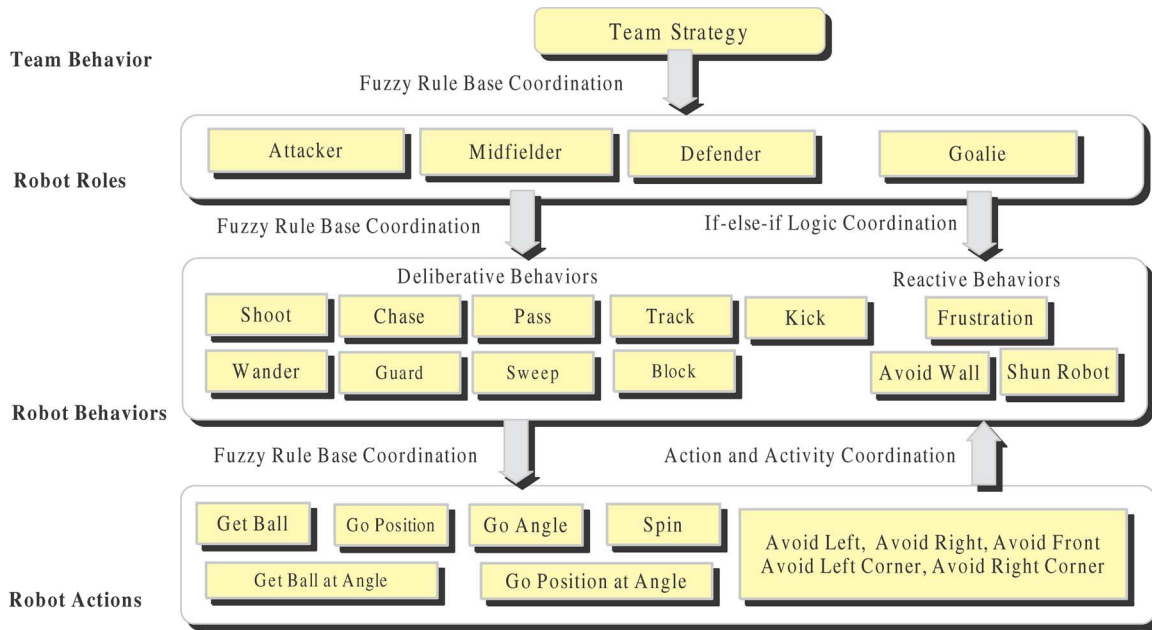


Fig. 6. Behavior architecture of the team of soccer robots.

The above fuzzy behavioral architecture has been realized on a real-world robot soccer system [28]. A satisfying performance is obtained from most of the primitive actions and behaviors. However, some of the behaviors at the higher level are not fully optimized, especially for the role assignment behavior, which adopts a rather simple mechanism. In this paper, the role assignment task, which belongs to the topmost team behavior, is chosen as the benchmark problem to evaluate the performance of the proposed DNA coding methods.

It is noticed that the evolution of the fuzzy behavior for role assignment cannot be performed in a real-world setup. It is not feasible as the system hardware is designed to continuously run for several hours, while the evolution may require several days or even weeks. A simulator platform for robot soccer system is carefully developed to provide a virtual environment as close to reality as possible (Fig. 7).

The simulator for the robot soccer system is programmed with Microsoft Visual C++ using the OpenGL library for visualization. The simulator models many of the environmental conditions of a real robot soccer game, such as the interactions between the robots, the ball, and the boundaries of the soccer field. The robots are modeled as block masses with two-wheel differential drive, and the ball as a circular sliding mass, subject to rolling friction. The collisions between robots and the ball comply with the relative orientations of their colliding surfaces. The effects of collisions are calculated by solving the vector equations for the conservation of momentum, which gives rise to accurate and visually realistic characteristics. In addition, uncertainties in the positions of objects due to noise in the visual perception are also taken into account. To achieve the desirable verisimilitude, all these characteristics are adjustable by the parameter settings in the simulator. Such features are especially useful when systems with different configurations (such as ball and robot masses) are to be simulated. Furthermore, the proposed fuzzy behavior-based architecture



Fig. 7. Robot soccer simulator.

(Fig. 6) is integrated into the simulator to control the virtual robots.

## V. DNA-CODED GA FOR FUZZY ROLE ASSIGNMENT FOR ROBOT SOCCER SYSTEM

To test out the DNA-coding method, the fuzzy behavior for role assignment in the robot soccer system is evolved by the DNA-coded GA. In the original system [28], a quite simple role assign mechanism is adopted. Firstly, the goalie role is fixed with one robot. As the only role equipped with offensive behaviors, the attacker is a must, and it is assigned to the robot closer to the ball. To avoid possible conflicts, each role is restricted to only one robot at any moment. The fuzzy

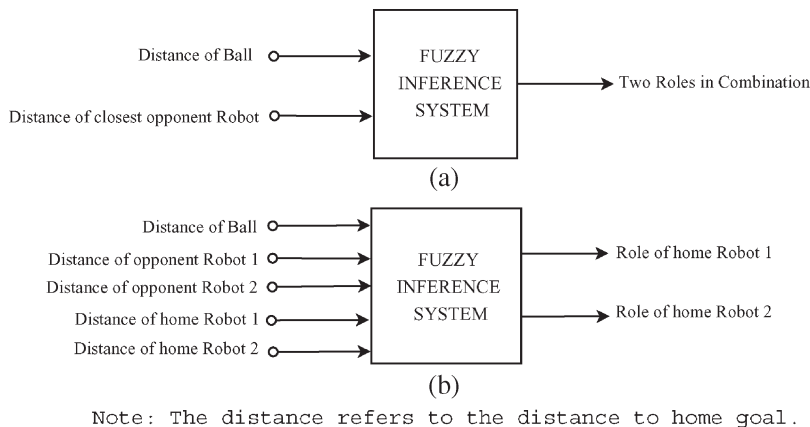


Fig. 8. Two-input and five-input system.

system only needs to select a role between the defender and the midfielder for the last robot based on the distance of the closest opponent robot to the home goal and the distance of the ball to the home goal [Fig. 8(a)]. As a whole, this fuzzy system is too simple to handle the complicated and dynamic system activities. Its inability often causes inappropriate robot behaviors. For instance, it is often found in competitions [29], [30] that, when an attacker is stuck with opponent robots, its role is not taken up by another robot in idle status.

A more complex fuzzy rule base is developed and evolved for role assignment in this paper. Compared to the original two-input system [28] [Fig. 8(a)], five input variables are considered for a new fuzzy system [Fig. 8(b)]. The five inputs are the distance of the ball to the home goal, the positions of the two opponent robots, and the positions of the two home robots. Each variable has three linguistic values: near, medium, and far. The two output variables are the roles of the two robots other than the goalie, which can be the attacker, midfielder, and defender. The limit of “one role for one robot” in the original system is then relaxed. The two robots other than the goalie are allowed to perform the same role at the same time. Consequently, there are  $3^5 = 243$  input states and totally  $3^7 = 2187$  possible rules. The evolution is focused on the rule base of the fuzzy system, while the membership functions are kept intact.

Three coding methods are used for comparison. The first one is the standard integer coding. The others are two DNA coding methods, named as DNA coding 1 and 2. The difference between the two DNA codings is that DNA coding 1 contains no intron while DNA coding 2 does. The four characters representing the DNA base—A, C, G, and T—are employed to build the chromosome string. The biological term “codon” is borrowed to indicate the three-letter triplets and/or two-letter couplets in the DNA coding method.

The start/stop codon is not used in the DNA coding methods proposed here, which marks an apparent departure from the DNA coding methods in [5]–[7]. The use of the start/stop codon separates the individual strings into segments of useful genes and meaningless introns. Each segment of gene is mapped to a fuzzy rule. If the structure of rule is fixed, whose number of parameters is already known (as in the robot soccer system), the length of the gene segment needs to match the rule. A gene that is too short results in an invalid rule, while a gene

1	2	3	4	5	6	7	8	...	483	484	485	486
1- Output role for robot number 1 under input state 1 2- Output role for robot number 2 under input state 1 3- Output role for robot number 1 under input state 2 4- Output role for robot number 2 under input state 2 5- Output role for robot number 1 under input state 3 6- Output role for robot number 2 under input state 3 7- Output role for robot number 1 under input state 4 8- Output role for robot number 2 under input state 4 ..... ..... 483- Output role for robot number 1 under input state 242 484- Output role for robot number 2 under input state 242 485- Output role for robot number 1 under input state 243												

Fig. 9. Structure of the chromosome encoded with the integer coding method.

that is too long wastes many codons. Both of the cases result in some kind of inefficiency in decoding useful rules from individual chromosomes. However, it is difficult to control the number or the length of useful genes as the start/stop codons are randomly distributed. Another effect of the start/stop codon is that it allows the overlapped reading of genes (Fig. 3), which means that the gene segments can share some portions. There is no empirical evidence to show the advantages of such overlapping. Furthermore, if necessary, the overlapping can be easily realized by rereading the chromosome from another starting point chosen in a random or heuristic way without the definition of a start/stop codon. As a result, the start/stop codon is not defined in this paper.

### A. GA Using Integer Coding Method

Using the integer coding method to encode the fuzzy rule base is quite straightforward. Each individual string represents a complete fuzzy rule base, which includes all the input states indexed from 1 to 243. The string has a fixed length of  $243 \times 2 = 486$  integers. The integer set  $\{1, 2, 3\}$  is mapped to the three output linguistic values: attacker, midfielder, and defender. As there are two robot roles to decide, every two consecutive integers stand for the output resulting from one indexed input state (Fig. 9). The coding method is a typical position-dependent one, as the meaning of every integer is merely determined by its absolute position. For the  $n$ th integer of the individual string, if  $n$  is odd, the integer stands for the

TABLE III  
INDEX OF THE TWO-LETTER CODONS WITH DNA CODING METHOD 1

	A	C	G	T
A	Codon 1	Codon 1	Codon 2	Codon 2
C	Codon 1	Codon 1	Codon 2	Codon 2
G	Codon 1	Codon 3	Codon 3	Codon 2
T	Codon 3	Codon 3	Codon 3	*

TABLE IV  
TRANSLATION FROM CODONS TO FUZZY RULES

Input/Output Variables	Codon 1	Codon 2	Codon 3
Distance: Ball to own goal	Near	Medium	Far
Distance: Opponent 1 to own goal	Near	Medium	Far
Distance: Opponent 2 to own goal	Near	Medium	Far
Distance: Robot 1 to own goal	Near	Medium	Far
Distance: Robot 2 to own goal	Near	Medium	Far
Role: Robot 1	Attacker	Midfielder	Defender
Role: Robot 2	Attacker	Midfielder	Defender

output role of robot number 1 under the input state indexed as  $\lfloor n/2 \rfloor + 1$ ; if  $n$  is even, it indicates the role of robot number 2 for the  $\lfloor n/2 \rfloor$ th input state.

The GA population size is set to 200. The stochastic universal sampling (SUS) [31] with elitism selection is used, while the number of elite individuals is set to 3. Uniform crossover [32] and random multiple points mutation are adopted. The crossover probability  $P_c$  and the mutation probability  $P_m$  are set as 0.6 and 0.05, respectively.

B. GA Using DNA Coding Method 1

In DNA coding method 1, the four DNA bases form the coding alphabet. Each individual is a string of characters randomly selected from the alphabet. Different from the natural three-letter protein codons, 16 two-letter couplets in total are employed and mapped onto the three index codons (Table III). Fifteen two-letter couplets are randomly grouped to represent codons 1, 2, and 3, which are in turn used to encode the three linguistic values for both the five inputs and the two outputs. The sixteenth couplet is used as a wildcard codon, which is randomly mapped to codon 1, 2, or 3. That is to make sure that the three codons have equal chances to appear in individual strings. In this way, all the 16 couplets have useful meanings.

Following the translation (Table IV), the genes are decoded to fuzzy rules, which form the final rule base. Among the resultant fuzzy rules, some of them may have the same antecedent parts standing for same input states. Only the first rule among them is included in the rule base. As all the couplets are meaningful, and there is no start/stop point, the individual chromosomes contain no meaningless parts (introns). The decoding process from DNA strings to fuzzy rules is depicted in Fig. 10. The meaning of codons is partially related to their absolute positions. For instance, the codons in the third, 11th, and 19th positions are always translated to the third input variable, i.e., the distance of the opponent robot 2 to the home goal. However, to which input state the input variable belongs is undecided. This codon needs to be combined with the two codons ahead, which represent the other two input variables' values, to depict a definitive antecedent of a fuzzy rule. In this way, the DNA

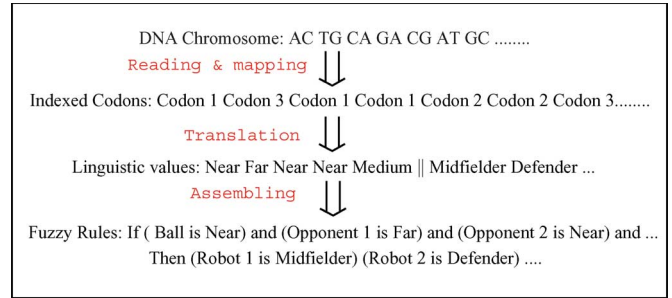


Fig. 10. Decoding process for DNA coding method 1.

TABLE V  
INDEX OF THE TWO-LETTER CODONS WITH DNA CODING METHOD 2

	A	C	G	T
A	-	-	Codon 1	Codon 1
C	Codon 3	-	Codon 2	-
G	-	-	-	Codon 2
T	-	Codon 3	-	-

coding method 1 is context dependent. On the other hand, each codon is represented by at least five two-letter couplets, which results in some kind of redundancy. The individual strings can be of variable length. The SGA operators are used while the validity of individual strings remains unaffected.

Similar to the GA with integer coding, the population size is set to 200. The individual chromosome's initial length is set to 500 characters. The genetic operators and the parameter settings ( $P_c, P_m$ ) are almost the same as that of GA with integer coding, except that the random two-point crossover is used. During the crossover, random crossover points are selected for each parent string, which results in offspring strings of different lengths. In this way, string lengths are varied throughout the GA process.

C. GA With DNA Coding Method 2

DNA coding method 2 differs from method 1 in the mapping mechanism between couplets and codons. Among the 16 two-letter couplets, only six of them are meaningful, while the others are meaningless (Table V). The selection of these six meaningful couplets is rather arbitrary. The only requirement is that the characters in the alphabet {A, C, G, T} have equal number of instances in these six couplets. In other words, these six couplets should have the same chance of appearance in chromosomes. The existence of meaningless couplets introduces intron parts into individual chromosomes. Intron parts are just bypassed while reading the chromosome. The meaningful parts are translated to fuzzy rules using the same translation table used in DNA coding method 1 (Table IV and Fig. 11). The existence of introns makes the coding method a context-dependent one. The other features of DNA coding method 1, e.g., redundancy and variable string length, are also existing in DNA coding 2.

The GA with DNA coding method 2 uses the SUS with elitism selection, two-point crossover, and multipoint mutation operators. The parameter settings are also the same as those used in the case of method 1.

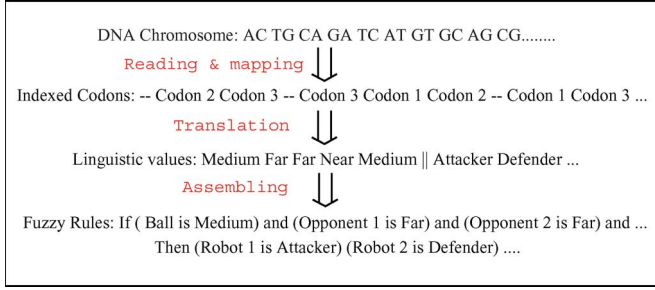


Fig. 11. Decoding process for DNA coding method 2.

There are other considerations for the DNA coding method representing the fuzzy rule bases. In methods 1 and 2, one valid fuzzy rule is represented by seven meaningful couplet codons. The number of valid fuzzy rules represented by an individual is related to the string length. Since the string length may vary during evolution, there could be some extremely long individual strings containing lots of fuzzy rules. However, in total 243 fuzzy rules with different antecedents are enough to set up a complete rule base covering all the input states. It is unnecessary for an individual to contain too many fuzzy rules. Furthermore, as shown in the later part of this paper, one promising purpose of DNA coding is to find a smaller but yet effective rule base that covers only the most important input states. In consequence, an upper limit (denoted as  $R_{\text{num}}$ ) is set to the number of fuzzy rules decoded from one individual. When  $R_{\text{num}}$  is reached, the decoding process stops, and the rest of the strings are ignored.  $R_{\text{num}}$  is usually chosen as 100 in the simulation, while settings for 250 and 25 are also tested for comparison.

Since the size of the rule base represented by an individual is varied with the string length and is limited by  $R_{\text{num}}$ , it is normal to find in the simulation that no fuzzy rule is fired at some time steps. This means that the input states at those instances are not covered by the current rule base. At such moments, a default role setting is chosen as the output. At the beginning of the simulation, the default roles are the attacker for one robot and the defender for the other. After which, the default output is the role assignment of the last time step. Since the input space of the fuzzy controller is continuous, it is reasonable to assume that the input state of the current step is closer to that of the last step. Hence, their outputs should have some similarity.

## VI. SIMULATION AND RESULTS

The GAs with three different coding methods are tested in the simulation environment (Fig. 7). The rule base under evolution is evaluated by competing with a team using the original rule base. Two teams of robots are pitched for a match for a certain period (set as 600 steps). The game immediately ends when the total number of goals scored reaches a limit. The limit is set to 3, which was found to be the maximum number of goals that can be scored by both teams in 600 steps. Similar to the real-world human soccer game, the performance of a team is summarized by some statistical data. The first datum is of course the match score. Second, the time duration for which the ball is within which side of the field is recorded. The duration for which the

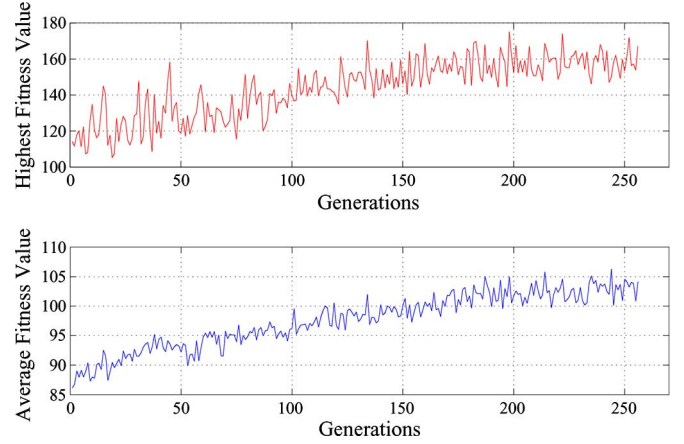


Fig. 12. Fitness curve for the integer coding method.

ball is in one side of the field implies that the team on that side is under attack. The third factor considered is the possession of the ball. Except for the two goalies, the robot that is nearest to the ball is regarded as the possessor of the ball. The fitness function  $F_{\text{role}}$  is constructed as follows:

$$F_{\text{role}} = u + v \cdot (\text{Goal} - \text{Lost}) + w_1 \cdot \frac{T_{\text{ball}}}{T_{\text{total}}} + w_2 \cdot \frac{T_{\text{opp}}}{T_{\text{total}}} \quad (1)$$

where  $u$ ,  $v$ ,  $w_1$ , and  $w_2$  are integer constants, Goal and Lost are the scores of the home and opponent teams,  $T_{\text{ball}}$  is the time for which the ball is under the control of the home team,  $T_{\text{opp}}$  is the time for which the ball is in the opponent half of the field, and  $T_{\text{total}}$  is the total match time. With a weight  $w_1$  of 100, the ratio of  $T_{\text{ball}}$  to  $T_{\text{total}}$  implies which team has possession in the game. The ratio of  $T_{\text{opp}}$  to  $T_{\text{total}}$  is an indication of which team takes the upper hand in the game, and the associated weight ( $w_2$ ) is set to 80. It is observed that the score is indeed not a stable performance index. Furthermore, the two teams seldom score in the match due to the rather short period of 600 steps. As a result, the weight attached to the score difference ( $v$ ) is set to a relative small value of 10. Since a team may lose the game, the second part of  $F_{\text{role}}$  can be negative.  $u$  is set to 30 and is utilized to keep  $F_{\text{role}}$  nonnegative.

In the simulation, the random nature of the soccer competitions brings in some degree of variation in the fitness evaluations. The fitness value of the same chromosome may not be constant. Different rule bases, although similar, are evolved from different GA runs with the same coding method. The average values across 50 different runs for each coding method are depicted as fitness value curves (Figs. 12–15), in which fluctuations are noticed. The fluctuation is especially serious in the “highest fitness value” curves. However, the trend and property of the evolution process are still observable.

With the integer coding method, the fitness values for each generation are summarized in Fig. 12. Both the average fitness and the highest fitness ceased to improve after 200 iterations. The average fitness value falls in the range between 100 and 105, while the highest fitness value is stabilized at around 160.

DNA coding method 1 is used with the upper limit on the number of fuzzy rules decoded from an individual set to 100 ( $R_{\text{num}} = 100$ ). The size of the fuzzy rule base is less than half

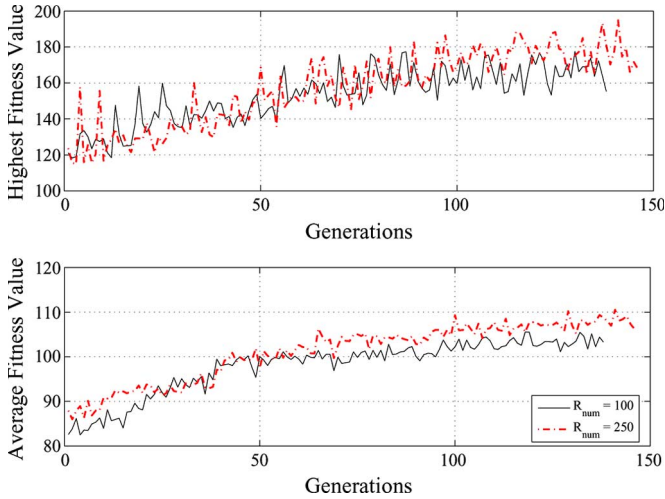


Fig. 13. Fitness curve for DNA coding method 1.

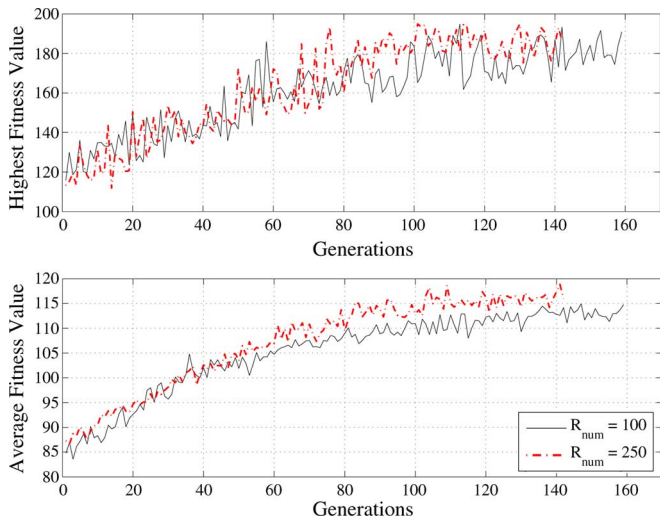


Fig. 14. Fitness curve for DNA coding method 2.

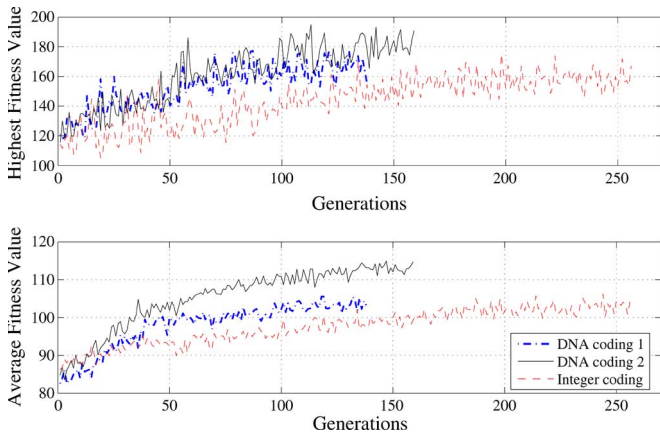


Fig. 15. Fitness curve comparison for the three coding methods.

of the one represented by the integer coding method. In the simulation, the GA process converges at around 125 iterations. The average and the highest fitness values are stabilized at around 103 and 165, respectively (Fig. 13). In the case of  $R_{num} = 250$ , both the average and the highest fitness converge

TABLE VI  
COMPARISON OF SIMULATION MATCH PERFORMANCES

Performance Data	Integer Coding	DNA Coding 1	DNA Coding 2
	Original - Evolved	Original - Evolved	Original - Evolved
Scores	0.3 - 1.6	0.2 - 1.4	0.2 - 2.0
Shots	11.7 - 18.0	12.6 - 17.2	10.4 - 18.8
$\frac{T_{ball}}{T_{Total}}$ %	33.7% - 66.3%	31.6% - 68.4%	23.9% - 76.1%
$\frac{T_{Opp}}{T_{Total}}$ %	36.9% - 63.1%	34.9% - 65.1%	28.2% - 71.8%

to a higher value (Fig. 13). The range of improvement is from 5 to 10.

The same setting of  $R_{num}$  is applied to DNA coding method 2. The evolutionary process with DNA coding method 2 usually converged at around 140 generations. The final average fitness value lingers at around 110, and the highest fitness value fluctuates at slightly around 180 (Fig. 14). If  $R_{num}$  is increased to 250, the average and the highest fitness curves raise to the levels of around 115 and 185, respectively.

The fitness curves of the GAs using the three coding methods are compared in Fig. 15.  $R_{num}$  is set to 100. The starting points of the fitness curves for the three coding methods are almost equal. Despite of the fluctuations, it is observed that the fitness curves with DNA coding 1 reach the same level as that of integer coding (Fig. 15) if not slightly higher. Nevertheless, DNA coding 2 obviously provides the best fitness curves with respect to both the average fitness and the highest fitness.

Performance validation is carried out with the evolved rule bases and the original rule base through simulation soccer matches. The performance data [the scores, number of shots,  $(T_{ball}/T_{total})$ , and  $(T_{opp}/T_{total})$ ] of the rule bases from the same coding method are collected and averaged. The average performance data for each coding method are compared in Table VI.

From the comparisons, it seems that the DNA coding methods outperform the integer coding method in this problem, although the size of rule base represented by a DNA-coded chromosome is only 100 rules at most. One important reason for such a performance is due to the interactions inside the fuzzy rule base. For the fuzzy rule base discussed, the fuzzy rules are in fact not independent from each other. Rules with similar antecedents should also have related consequences. More importantly, the outputs of certain fuzzy rules change the situation in the match field and in turn trigger certain other rules. In the position-dependent integer coding each rule has mapped to a fixed position in the string. The nonlinear causality between rules results in a nonlinear interaction of characters/codons at different positions. The fitness of one character/codon at one position may depend on the value of the other characters/codons. This effect is usually referred to as epistasis [33], [34], which may decrease the performance of GA [35], [36]. According to the schema theorem, GA works well if a string with high fitness can be built from short, low-order, and above-average schemata. Thus, one of the basic requirements of GAs to be successful is low epistasis. To this problem, it is justifiable to assume that a rational schema should contain the characters

interacting with each other. However, if the characters locate at two far-separated positions, the schema will be long and liable for destruction through the GA operators. For the position-dependent coding method, this issue is unavoidable. However, for the context-dependent DNA coding, the rules are not fixed to absolute positions. Those nonlinear-related characters may be moved together during the GA operations, which results in short and reasonable schemata. The position-independent parameters in the individual string in fact alleviate the difficulty caused by epistasis. That is a major advantage of the DNA coding methods.

Meanwhile, DNA coding method 1 and 2 have some degree of redundancy in coding. Referring to Tables III and V, each indexed codon is represented by more than one couplet. The result of such a redundancy is that two individuals displaying the same phenotype (i.e., representing the same fuzzy rule base) may have different genotypes (different strings). A crossover between them may produce offsprings with different phenotypes. This feature helps to increase the population diversity, which is another benefit associated with the DNA coding methods.

Furthermore, DNA coding method 2 introduces introns into individual strings. Researchers have observed that introns may lead to a considerable improvement in performance of GAs [22], [27] and genetic programming [37]. In this paper, DNA coding method 2 demonstrates a better performance than that of DNA coding 1 (Fig. 15). It seems that the meaningless parts (introns) in the strings play a useful role in absorbing the disruption caused by genetic operations. They also provide themselves as building materials that can be transferred to meaningful codons at any time during GA operations. This is some kind of enlargement of the searching space covered by the current population and also an increase in population diversity.

In addition to the fitness values, some other parameters related to the DNA coding's performance are also looked into. The fuzzy rule base decoded from an individual is evaluated in a match lasting for 600 steps. A counter  $C_{\text{fire}}$  is set up for each individual to denote how active the rule base is. In each step, if one rule in the base is triggered, the counter is increased by unity. Thus, the maximum  $C_{\text{fire}}$  is 600. The counter remains unchanged when none of the fuzzy rules is fired, and the output is determined by the default setting. A bigger value of  $C_{\text{fire}}$  indicates a more active rule base. A fuzzy rule base has to be active enough at first before taking any effects. In fact,  $C_{\text{fire}}$  is related to how many input states are covered by the fuzzy rule base, which in turn is associated with the size of the rule base and the length of the individual string. Both the average of  $C_{\text{fire}}$  and the length of the population are plotted in Fig. 16 for DNA coding method 2 with  $R_{\text{num}}$  as 100.

The initial length for the DNA coding chromosomes is set to 500 characters. With this length, a chromosome contains at most 35 valid fuzzy rules, which implies a quite small rule base. The average  $C_{\text{fire}}$  is about 200, which means the fuzzy rule base is not active for two-thirds of the time. As the evolution goes on, the average length of all the individuals quickly increases. At the end of the evolution, the average length is around 13 000 characters. The chromosome then contains about 6500

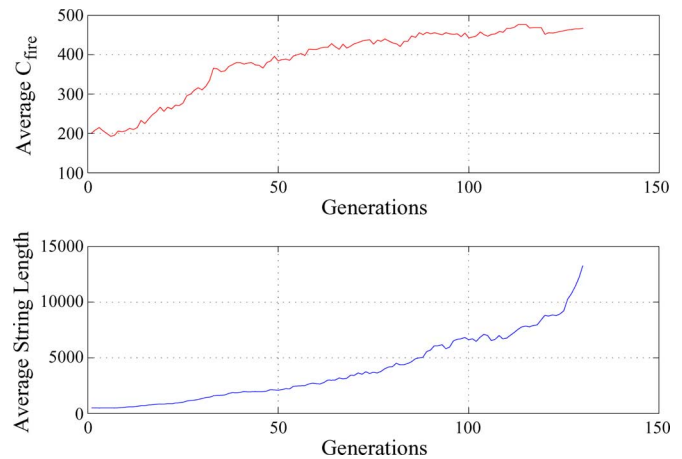


Fig. 16. Change of  $C_{\text{fire}}$  and string length throughout the evolution.

two-letter couplets, in which approximately six-sixteenths of them are valid codons (Table V). As a result, about  $6500 \times (6/16) \times (1/7) \approx 348$  valid rules can be decoded. The rules with duplicated antecedent parts are further filtered. The final rule base should already reach the limited size of 100 rules. Consequently, the average  $C_{\text{fire}}$  steadily increases from 200 to 460. The rule base becomes more and more effective as its size increases. The DNA coding allows individuals to be of variable length. That means the size of the encoded fuzzy rule base is also flexible and can be evolved by the GA.

The benefits of the variable size for the fuzzy rule base seem not prominent at the first glance. It is obvious that the bigger rule base is always preferred with respect to the coverage of the input states. If  $R_{\text{num}}$  is not imposed, the size of the rule base will surely grow to the maximum of 243 during the evolution. However, it seems that the effectiveness of the rule base is not linear to its size. While the size is limited by  $R_{\text{num}} = 100$ , the rule base is already active for 76% (460/600) of the time. To study the influence of the variable  $R_{\text{num}}$  on the GA process, the fitness curves with different settings of  $R_{\text{num}}$  are depicted in Fig. 17. While  $R_{\text{num}}$  is increased from 100 to 250, the size of the resultant rule base is more than doubled. However, the improvements on both the average fitness and the highest fitness are just around 2%–3% (Fig. 17). An even smaller  $R_{\text{num}}$  of 25 is tested. With such a small rule base, the decline of fitness curves is only about 12%–15%.

Such results suggested that not all the rules in the rule base are of the same importance. Some of them are related to those most crucial input states and thus dominate the performance of the fuzzy rule base. A compact rule base containing only those important rules is more computationally efficient than the complete rule base covering all the input states. This feature becomes more and more meaningful as the fuzzy rule base gets more complicated and bigger in size. On the other hand, it is usually hard to tell which input states are more important at the designing stage. With the DNA coding methods, the GA can automatically search for and optimize the crucial rule base with different size settings. As to the integer coding method, the input states are attached to absolute positions on the strings. Under such circumstances, GAs cannot optimize the fixed structure of the rule base.

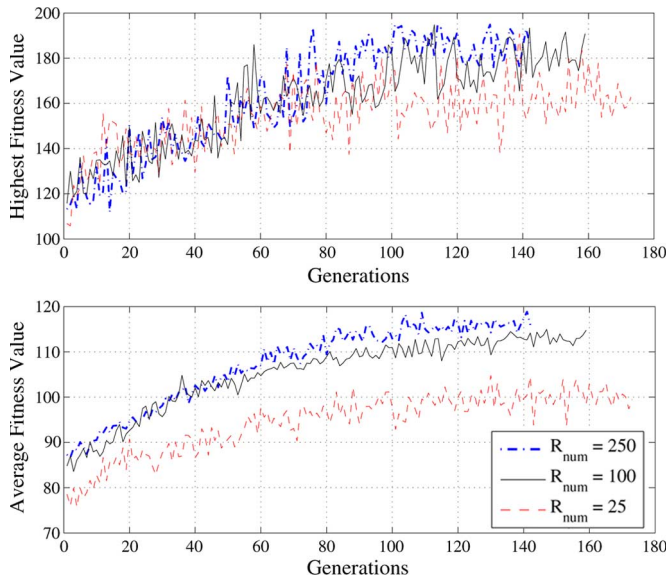


Fig. 17. Fitness curve comparison for different  $R_{\text{num}}$  settings.

## VII. CONCLUSION AND DISCUSSION

In this paper, the context-dependent DNA-like coding methods have been discussed and implemented for the evolution of the fuzzy rule base. The DNA coding methods are flexible and easy to use. They can be directly used with the standard genetic operators. Two types of DNA coding methods (with and without the intron parts) are compared with the traditional integer coding method. The GAs with the three coding methods are applied to the fuzzy rule assignment problem in a robot soccer system.

The results show that the DNA coding methods outperform the integer coding method. The context-dependent coding can handle the negative effect of epistasis, which degrades the performance with the position-dependent coding. The redundancy in DNA coding increases the population diversity. The DNA coding with intron parts displays an even better performance. It seems that the intron parts are helpful in preventing useful schemata from disruption and in increasing the population diversity. The DNA-coded individuals are of variable length. Such a feature provides GA with the possibility to evolve both the size and the structure of the fuzzy rule base. As a result, a compact but efficient rule base can be developed by the DNA-coded GA.

The potential and characteristics of DNA coding methods are investigated in this paper with a case of fuzzy rule base representation. Due to its flexibility, the coding scheme can be adapted for other GA-related applications. Further exploration on the DNA coding with other evolutionary algorithms, e.g., evolutionary programming, seems to be worthy and interesting.

## REFERENCES

- [1] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithm*. New York: Wiley-Interscience, 1997.
- [2] R. Biesbroek, *The genetic algorithm tutorial webpage*. [Online]. Available: <http://www.estec.esa.nl/outreach/gatutor/Default.htm>
- [3] L. C. Jain, *Evolution of Engineering and Information Systems and Their Applications*. Boca Raton, FL: CRC, Sep. 1999.
- [4] N. A. Campbell, *Biology*. Redwood City, CA: Benjamin Cummings, 1996.
- [5] T. Furuhashi, "Development of IF-THEN rules with the use of DNA coding," in *Fuzzy Evolutionary Computation*, W. Pedrycz, Ed. Boston, MA: Kluwer, 1997.
- [6] T. Yoshikawa and Y. Uchikawa, "Effect of new mechanism of development from artificial DNA and discovery of fuzzy control rules," in *Proc. IIZUKA*, 1996, pp. 498–501.
- [7] K.-Y. Lee, D.-W. Lee, and K.-B. Sim, "Evolutionary neural networks for time series prediction based on L-system and DNA coding method," in *Proc. IEEE Congr. Evol. Comput.*, 2000, vol. 2, pp. 1467–1474.
- [8] T. Yoshikawa, T. Furuhashi, and Y. Uchikawa, "A combination of DNA coding method with pseudo-bacterial GA for acquisition of fuzzy control rules," in *Proc. 1st WSC*, 1996, pp. 107–112.
- [9] L. Ren, Y. Ding, and S. Shao, "DNA genetic algorithms for design of fuzzy systems," in *Proc. 9th IEEE Int. Conf. Fuzzy Syst.*, May 7–10, 2000, vol. 2, pp. 1005–1008.
- [10] L. Ren and Y. Ding, "Parameter optimization for a class of general TS fuzzy controllers via a new DNA-based genetic algorithm intelligent control and automation," in *Proc. 9th World Congr. WCICA*, Jun. 15–19, 2004, vol. 3, pp. 2149–2153.
- [11] J. D. Schaffer, R. A. Caruana, E. Eshelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer Ed., pp. 51–60.
- [12] L. Davis, "A genetic algorithms tutorial," in *Handbook of Genetic Algorithms*, L. Davis, Ed. New York: Van Nostrand Reinhold, 1991, pp. 1–101.
- [13] R. Belew and L. Booker, Eds., *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA: Morgan Kaufmann, 1991.
- [14] S. Forrest Ed., *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo CA: Morgan Kaufmann, 1993.
- [15] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1996.
- [16] H. Kargupta, K. Deb, and D. E. Goldberg, "Ordering genetic algorithms and deception," Univ. Illinois, Urbana, IL, Tech. Rep. IlliGAL Rep. 92006, 1992.
- [17] L. Davis, "Applying adaptive algorithms to epistatic domains," in *Proc. 9th Int. Joint Conf. Artif. Intell.*, A. Joshi, Ed, Los Angeles, CA, Aug. 1985, pp. 162–164.
- [18] D. E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in *Proc. 1st ICGA*, J. J. Grefenstette, Ed., 1985, pp. 154–159.
- [19] G. Syswerda, "Schedule optimization using genetic algorithms," in *Handbook of Genetic Algorithms*, L. Davis, Ed. New York: Van Nostrand Reinhold, 1991, pp. 332–349.
- [20] H. Mühlenbein, "Evolution in time and space—The parallel genetic algorithm," in *Foundations of Genetic Algorithms*, G. J. E. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 316–337.
- [21] T. Bäck and M. Schütz, "Evolution strategies for mixed-integer optimization of optical multilayer systems," in *Proc. 4th Annu. Conf. Evol. Program.*, J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, Eds., 1995, pp. 33–51.
- [22] J. R. Levenick, "Inserting introns improves genetic algorithm success rate: Taking a cue from biology," in *Proc. 4th Int. Conf. Genetic Algorithms*, R. Belew and L. Booker, Eds., 1991, pp. 123–127.
- [23] A. S. Wu and R. K. Lindsay, "Empirical studies of the genetic algorithm with noncoding segments," *Evol. Comput.*, vol. 3, no. 2, pp. 121–147, 1995.
- [24] N. L. Cramer, "A representation for the adaptive generation of simple sequential programs," in *Proc. 1st ICGA*, J. J. Grefenstette, Ed., 1985, pp. 183–187.
- [25] J. H. Rogers, "The role of introns in evolution," *FEBS Lett.*, vol. 268, no. 2, pp. 339–343, Aug. 1990.
- [26] L. Duret, "Why do genes have introns? Recombination might add a new piece to the puzzle," *Trends Genet.*, vol. 17, no. 4, pp. 172–175, Apr. 2001.
- [27] S. Forrest and M. Mitchell, "Relative building-block fitness and the building-block hypothesis," in *Foundations of Genetic Algorithms 2*, L. D. Whitley, Ed. San Mateo, CA: Morgan Kaufmann, 1993, pp. 109–126.
- [28] P. Vadakkepat, O. C. Miin, X. Peng, and T. H. Lee, "Fuzzy behavior-based control of mobile robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, pp. 559–565, Aug. 2004.
- [29] FIRA, *Federation of International Robot-Soccer Association*, 1995. [Online]. Available: <http://www.fira.net>

- [30] SRG, *Singapore Robotic Games*, 2001. [Online]. Available: <http://guppy.mpe.nus.edu.sg/srg/>
- [31] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in *Proc. 2nd Int. Conf. Genetic Algorithms Appl.*, 1987, pp. 14–21.
- [32] G. Syswerda, "Uniform crossover in genetic algorithms," in *Proc. 3rd Int. Conf. Genetic Algorithms*, J. D. Schaffer, Ed., 1989, pp. 2–9.
- [33] C. R. Reeves and C. C. Wright, "Epistasis in genetic algorithms: An experimental design perspective," in *Proc. ICGA*, 1995, pp. 217–224.
- [34] R. Salomon, "Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions: A survey of some theoretical and practical aspects of genetic algorithms," *Biosystems*, vol. 39, no. 3, pp. 263–278, 1996.
- [35] D. Beasley, D. R. Bull, and R. R. Martin, "Reducing epistasis in combinatorial problems by expansive coding," in *Proc. 5th Int. Conf. Genetic Algorithms*, S. Forrest, Ed., 1993, pp. 400–407.
- [36] Y. Davidor, "Epistasis variance: A viewpoint on GA-hardness," in *Foundations of Genetic Algorithms*, G. J. Rawlins, Ed. San Mateo, CA: Morgan Kaufmann, 1991, pp. 23–35.
- [37] P. J. Angeline, "Genetic programming and emergent intelligence," in *Advances in Genetic Programming*, K. E. Kinnear, Jr., Ed. Cambridge, MA: MIT Press, 1994, pp. 75–98.



**Peng Xiao** received the B.Eng. and M.Sc. degrees from Hunan University, Hunan, China, in 1997 and 2000, respectively, and the Ph.D. degree from the National University of Singapore, Singapore, in 2006.

From 2004 to 2006, he was an R&D Engineer with the Advanced R&D Department, STMicroelectronics Asia Pacific. He is currently a Lecturer with School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China. His research interests include robotics, fuzzy logic control, neural networks, evolutionary algorithms, and image processing.



**Prahlad Vadakkepat** received the M.Tech. and Ph.D. degrees from the Indian Institute of Technology Madras, Chennai, in 1989 and 1996, respectively.

From 1991 to 1996, he was a Lecturer with the Regional Engineering College Calicut (now the National Institute of Technology Calicut), Calicut, India. From 1996 to 1998, he was a Postdoctoral Fellow with the Korea Advanced Institute of Science and Technology, Daejeon. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests are in humanoid robotics, distributed robotic systems, evolutionary robotics, neuro-fuzzy controllers, and intelligent control techniques. He has been Associate Editor of the *International Journal of Humanoid Robotics* since 2003.

Dr. Vadakkepat is the Founder Secretary of the Federation of International Robot-Soccer Association (FIRA) and currently the FIRA General Secretary. He was appointed as the Technical Activity Coordinator of IEEE Region 10 from 2001 to 2002.



**Tong Heng Lee** received the B.A. degree (with First Class Honors) in engineering tripos from Cambridge University, Cambridge, U.K., in 1980 and the Ph.D. degree from Yale University, New Haven, CT, in 1987.

He is currently a Professor with the Department of Electrical and Computer Engineering, National University of Singapore (NUS), Singapore. He is also currently Head of the Drives, Power and Control Systems Group in this department, and the Vice-President and Director of the Office of Research, NUS. He is the coauthor of three research monographs, and is the holder of four patents (two of which are in the technology area of adaptive systems, and the other two are in the area of intelligent mechatronics). His research interests are in the areas of adaptive systems, knowledge-based control, intelligent mechatronics, and computational intelligence.

Dr. Lee was the recipient of the Cambridge University Charles Baker Prize in Engineering. He is currently an Associate Editor for *Automatica*, the IEEE TRANSACTIONS IN SYSTEMS, MAN AND CYBERNETICS, *Control Engineering Practice*, the *International Journal of Systems Science*, and *Mechatronics*.