Applied Soft Computing 12 (2012) 2227–2236

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Immune based fuzzy agent plays checkers game

Seyyed Hadi Cheheltani [a,*], Mohammad Mehdi Ebadzadeh [b]

[a] Department of Computer science, Islamic Azad University of Qazvin, Iran
[b] Computer Engineering and Information Technology Department, Amirkabir University of Technology, Iran

ARTICLE INFO

Article history:
Received 20 July 2010
Received in revised form 15 February 2012
Accepted 4 March 2012
Available online 29 March 2012

Keywords:
Immune network
Clonal selection
Fuzzy inference engine
Checkers game

ABSTRACT

One of the advantages of immune based approaches is the usage of permanent memory cells. These memory cells cause to omit the process of learning for any played strategy and consequently increasing the speed of decision making process. In the proposed method of this article, memory cells represent actions that have the best local payoff for that current state of the game and are generated simultaneously by learning process. These cells help the decision making system to decide better, considering the previous and future state of the game. The decision making system that is used in this method is based on a Mamdani fuzzy inference engine (FIS). The FIS proposes a best action for the current state of the board by extracting memory cells' data. Experiments show that the immune based fuzzy agent which is introduced here has better results among other previous methods. This new method can show proper resistance when confronting a player that uses complete game tree remarkably. Also this method is capable of suggesting an action for each state of the game by generating less number of generations in comparison with other evolutionary based methods.

1. Introduction

One of the greatest ambitions of researchers in the field of artificial intelligence is to create intelligent game program capable of defeating human experts. Many different approaches have been used for different games including neural networks for backgammon [1], special-purpose hardware called Deep Blue for chess [2], and the application of expert knowledge with relatively small computational power for checkers [3] and Othello [4]. Most of these techniques relay on expert knowledge and methods of properly using them like training the evaluation function, relevance factors for the evaluation, the weights of the evaluation factors, opening knowledge, and an endgame database. Acquiring such knowledge requires the help and advice of game experts, computational power for processing the knowledge extracted, and a process of trial and error to find the best overall approach.

Usually in games like checkers, the typical approach uses a computer program to search a game tree to find an optimal move at each play, but there are challenges in overcoming an expert's experience. Sometimes a computer checkers program fails to defeat a human player because it makes a mistake that is not common among human players. Sometimes the fault is discovered by the computer program after searching beyond the predefined depth of the game tree. To defeat the best human players, Chinook uses some fixed predefined strategies for game start and the same features for game ending section. Also, Chinook relies on expert knowledge that is captured in an evaluation function, which is used in the middle of the game. Chinook's success is based largely on traditional game theory mechanics (game tree and alpha-beta search) and expert knowledge.

Recently, evolutionary induction of game strategies has gained popularity because of the success reported in [5] using the game of checkers. This approach does not need additional prior knowledge or expert heuristics for evolving strategies and expert-level strategies have evolved from the process of self-play, variation, and selection. In other games such as Othello [6,7], blackjack [8], Go [9], chess [10], and backgammon [11], the evolutionary approach has been applied to discover better strategies without relying on human experience. However, it might take a long evolution time to create a world-level champion program without a predefined knowledge base. As it is mention in [12], it takes long time to train a checkers player to the level of expert by Fogel and Chellapilla, and it would take even longer time to evolve the world-level champion program.

The important problem for a checkers player after learning is a decision making process. Recently some methods of decision making are used like neural network based approaches, game tree and expert system. All of these methods suffer from need of huge amount of data and consequently slow response time. To decrease the size of data needed to process and to make the response time quicker, utilizing fuzzy rules and fuzzy inference engine is one of methods. The main idea of learning how to play a game like

* Corresponding author.
E-mail address: hadicheheltani@gmail.com (S.H. Cheheltani).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.asoc.2012.03.009

checkers in this article is based on two separated phases of learning and deciding. In the learning phase when an opponent moves a piece, a modified clonal selection algorithm suggests the best move corresponding to current game state. Modification here means that the original version of clonal selection algorithm in [13] changes to represent the checkers state of play. These changes usually include definition of multi-level fitness function, validity of generated cell from mutation in each state of the game and new memory cell definition which is a combination of antigen and antibody. This move is suggested considering previous and possible future of game states. Finally, the cell representing the best local move is saved as memory cell. In decision phase, some fuzzy rules are generated based on those saved memory cells. Then the player can decide what to do after opponent's move using fuzzy decision making interface.

In the following, Section 2 the immune system and clonal selection are described. Section 3 describes fuzzy inference engine. In Section 4 checkers game and its rules are mentioned. The immune based fuzzy algorithm is proposed in Section 5. In Section 6 expert strategies are described and experimental results are presented in Section 7.

2. Immune system

The immune system performs several functions. Together with other bodily systems it maintains a stable state of our vital functions, named homeostasis. Its most remarkable roles however, are the protection of the organism against the attack of disease causing agents, called pathogens, and the elimination of malfunctioning cells. Microorganisms like viruses, bacteria, fungi and parasites are classified as pathogens, for they can cause diseases after invading our bodies. The primary problem the immune system is faced with is thus the recognition of these pathogens. The pathogens themselves cannot be directly recognized by the components of the immune system. Some small portions of the pathogens, named antigens, are the molecules that are going to be recognized by the immune system. After recognizing (identifying) a disease causing agent, the immune system is responsible for eliminating it, so as to avoid or block the disease. There are a few other tasks however, that the immune system has to perform so that it can correctly identify and eliminate pathogens. One such task is the recognition of our body's own tissues, which are broadly named self. Like pathogens, the cells and molecules of our body's organisms also present antigens, in this case self-antigens that can be recognized by the immune system. In order to distinguish self-antigens from those presented by pathogens, the latter are named nonself antigens. The process of distinguishing between self and nonself antigens (i.e. what belongs and what does not belong to the body) is termed self/nonself discrimination [13].

2.1. Clonal selection

After successful recognition, the adaptive immune response is elicited. One important immune mechanism of defense is to reproduce those cells capable of recognizing and binding with antigens. The cellular reproduction in the immune system is based on cloning (mitosis), i.e. the creation of offspring cells that are copies of their parent cells subject to mutations. This proliferation will result in the production of a clone of cells of the same type. Due to the mutations, the cells within a clone are all similar but present slight differences and are capable of recognizing the antigen that triggered the immune response. Actually this mutation process occurs on b-cells which carry on the process of maturation. As Clonal selection is a b-cell inspired algorithm, the word cell is meant b-cell in the whole literature. A selective mechanism guarantees that those offspring cells (in the clone) that better recognize the antigen, which elicited the response, are selected to have long life spans; these cells are named memory cells. This is the strategy by which evolution shaped our immune systems so that they became capable of dealing with antigens it has encountered in the past. This is also the principle used for vaccination purposes. The whole process of antigen recognition, cell proliferation and differentiation into memory cells is named clonal selection [14,15].

In [16] authors focus on the clonal selection principle and affinity maturation process of the adaptive immune response in order to develop an algorithm suitable to perform tasks such as machine learning, pattern recognition, and optimization. Their algorithm was evaluated in a simple binary character recognition problem, multimodal optimization tasks and a combinatorial optimization problem; more specifically the traveling salesman problem (TSP). The main immune aspects taken into account to develop the algorithm, named CLONALG, were: selection and cloning of the most stimulated cells proportionally to their antigenic affinity; death of non-stimulated cells; affinity maturation and selection of cells proportionally to their antigenic affinity; and generation and maintenance of diversity. The algorithm CLONALG works as follows [13]:

1. Generate a set of N candidate solutions (antibody repertoire) in a shape–space to be defined by the problem under study.
2. Select n_1 highest affinity cells in relation to the antigen set to be recognized or to the function being optimized.
3. Clone (generate identical copies of) these n_1 selected cells. The number of copies is proportional to their affinities: the higher the affinity, the larger the clone size (number of offspring).
4. Mutate with high rates (hypermutation) these n selected cells with a rate inversely proportional to their affinities: the higher the affinity, the smaller the mutation rate.
5. Re-select n_2 highest affinity mutated clones to compose the new repertoire.
6. Replace some low affinity cells by new ones.
7. Repeat steps 2–6 until a given stopping criterion is met.

3. Fuzzy inference engine

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made, or patterns discerned. The process of fuzzy inference involves following pieces: membership functions, logical operations, and if-then rules. Two types of fuzzy inference systems are applicable in problems: Mamdani-type and Sugeno-type. These two types of inference systems vary somewhat in the way outputs are determined [17–19]. Fuzzy inference systems have been successfully applied in fields such as automatic control, data classification, decision analysis, expert systems, and computer vision. Because of its multidisciplinary nature, fuzzy inference systems are associated with a number of names, such as fuzzy-rule-based systems, fuzzy expert systems, fuzzy modeling, fuzzy associative memory, fuzzy logic controllers, and simply (and ambiguously) fuzzy systems. Mamdani's fuzzy inference method is the most commonly seen fuzzy methodology. Mamdani-type inference, expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification. It is possible, and in many cases much more efficient, to use a single spike as the output membership functions rather than a distributed fuzzy set. This type of output is sometimes known as a singleton output membership function, and it can be thought of as a pre-defuzzified fuzzy set. It enhances the efficiency of the defuzzification process because it greatly simplifies the computation required by the more general Mamdani method, which finds the centroid of a two-dimensional function. Rather than