Expert Systems with Applications 40 (2013) 6311–6320

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Enhanced decision making mechanism of rule-based genetic network programming for creating stock trading signals

Shingo Mabu [a,*], Kotaro Hirasawa [b], Masanao Obayashi [a], Takashi Kuremoto [a]

[a] Graduate School of Science and Engineering, Yamaguchi University, Tokiwadai 2-16-1, Ube, Yamaguchi 755-8611, Japan
[b] Information, Production and Systems Research Center, Waseda University, Hibikino 2-2, Kitakyushu, Fukuoka 808-0135, Japan

ARTICLE INFO

Keywords:
Evolutionary computation
Genetic network programming
Rule extraction
Stock trading
Technical analysis

ABSTRACT

Evolutionary computation generally aims to create the optimal individual which represents optimal action rules when it is applied to agent systems. Genetic Network Programming (GNP) has been proposed as one of the graph-based evolutionary computations in order to create optimal individuals. GNP with rule accumulation is an extended algorithm of GNP, which extracts a large number of rules throughout the generations and stores them in rule pools, which is different from general evolutionary computations. Concretely, the individuals of GNP with rule accumulation are regarded as evolving rule generators in the training phase and the generated rules in the rule pools are actually used for decision making. In this paper, GNP with rule accumulation is enhanced in terms of its rule extraction and classification abilities for generating stock trading signals considering up and down trends and occurrence frequency of specific buying/selling timing. A large number of buying and selling rules are extracted by the individuals evolved in the training period. Then, a unique classification mechanism is used to appropriately determine whether to buy or sell stocks based on the extracted rules. In the testing simulations, the stock trading is carried out using the extracted rules and it is confirmed that the rule-based trading model shows higher profits than the conventional individual-based trading model.

1. Introduction

A graph-based evolutionary algorithm called Genetic Network Programming (GNP) (Mabu et al., 2007) has been proposed for creating action-rules in agent systems. GNP has been successfully applied to decision making problems (Hirasawa et al., 2008) due to its distinguished representation abilities such as successive node transitions and reusability of nodes. Evolutionary Programming (EP) (Fogel, 1994) also evolves graph structure (Finite State Machine), but it must consider all the combinations of available inputs from environments to determine the output and next state. Since GNP uses only the necessary information by connecting nodes flexibly, it can deal with Non-Markovian processes with compact program structures.

Evolutionary computation generally evolves individuals in order to increase their fitness, and it obtains the optimal or near optimal individuals which represent action-rules when they are used in agent control systems. Then, the evolved individuals are actually used in the testing. GNP with rule accumulation (GNP-RA) (Wang et al., 2009) also belongs to evolutionary computation field; however, the concept of evolution is different from general evolutionary computation. Although GNP-RA also evolves many individuals by crossover and mutation until the last generation, the important point is that the action rules (combinations of judgments and actions) obtained by the individuals during the training are stored in rule pools, that is, action rules are accumulated in the rule pools every generation. Therefore, GNP-RA stores many rules throughout the generations, while, in the original GNP, the best individual in the last generation represents action rules. In other words, GNP-RA can store many experiences obtained by the individuals in the training as rules.

Stock market analysis has been one of the most active research fields, where many Machine Learning techniques are adopted. Research on stock price prediction and trading models such as Neural Network (Chang et al., 2009), Support Vector Machines (Huang et al., 2005), rough set theory (Lee et al., 2010) and statistical analysis (Jensen et al., 2004) has been done. In recent years, evolutionary algorithms have been applied to many financial problems such as portfolio optimization (Torrubiano and Suárez, 2010), bankruptcy prediction (Cid et al., 2008), time-series prediction (Iba and Sasaki, 1999). Expert system based approaches for stock trading have been also proposed recently (Dymova et al., 2010; Dymova et al., 2012; Sevastianov and Dymova, 2009).

Generally speaking, methods for predicting stock prices and determining the timing of buying and selling stocks are divided into two groups; one is fundamental analysis which analyzes stock

* Corresponding author. Tel./fax : +81 836 85 9519.
E-mail address: mabu@yamaguchi-u.ac.jp (S. Mabu).

0957-4174/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.05.037

prices using the financial statement of each company, the economic trend and so on; the other is technical analysis which numerically analyzes the past movement of stock prices. Generally, the research on stock price prediction and trading models using soft computing belongs to technical analysis, so it determines the timing of buying and selling stocks based on the technical indexes such as rate of deviation, relative strength index, golden cross and so on. This paper also deals with technical analysis.

Stock trading models based on GNP have been proposed as applications (Chen et al., 2008; Mabu et al., 2007). GNP judges the current situation/trend in the stock market based on the technical indexes, then determines the timing of buying and selling stocks. In these models, the best individual in the last generation is used in the testing simulations, so all the stock trading rules are represented by one best individual. However, it is difficult for one individual to adapt to various kinds of stock market situations due to the following reasons. (1) GNP sequentially executes judgment and processing nodes one by one according to the node transition, so all the experiences/rules obtained in the training period cannot be directly used to decide the current action. (2) The number of action rules which one individual can contain is limited due to the limited size of the program structure, while the large size of the structure even causes much computational cost.

In this paper, GNP-RA for stock trading problems is developed, which extracts and stores useful buying and selling rules in the rule pools and the buying and selling actions are determined by a unique matching calculation and decision making techniques. Matching degrees introduced in this paper can tell us the timings of buying/selling. In addition, reinforcement learning algorithm is also used to extract rules. In Mabu et al. (2007), reinforcement learning is applied to GNP in order to enhance the intensified search and online learning abilities. This paper also uses reinforcement learning in order to obtain good action sequences, however, the more important point is to extract a large number of rules by an exploration ability of reinforcement learning with ε-greedy policy.

This paper is organized as follows. In the next section, after the basic structure of GNP is explained, how to represent action rules, how to store the rules in the rule pools and how to decide actions are explained. In Section 4, the stock trading problems treated in this paper is described and the simulation results are analyzed. Section 5 is devoted to conclusions.

2. Review of genetic network programming with reinforcement learning

GNP was originally developed to create programs that work in dynamic environments. In the original GNP, an individual is represented by a directed graph structure, evolved by crossover and mutation, and the node transition rules are learned by reinforcement learning. In this section, the phenotype and genotype representations and the learning and evolution processes are reviewed.

2.1. Phenotype representation

2.1.1. Features of the directed graph structure

The phenotype representation of GNP is a directed graph structure shown in Fig. 1. It consists of a number of judgment nodes and processing nodes and one start node. The number of nodes is determined in advance depending on the complexity of the problems. Each judgment node has a if-then branch decision function and each processing node has an action function. For example, the judgment nodes in Fig. 1 examine technical indexes used in the stock market analysis. Node 1 examines Relative Strength Index (RSI) and selects a branch corresponding to the judgment result. The processing nodes determine buying and selling actions. The role of the start node is to determine the first node to be executed. Therefore, the node transition starts from the start node, and judgment and action sequence is realized by the connections between nodes and judgment results.

The nodes in GNP can be repeatedly used due to the graph structure, so complex programs can be created by compact structures, which contributes to the efficiency of the evolution (Mabu et al., 2007). In addition, only the necessary nodes for solving the target problem can be connected and used, so GNP does not need the complete information of the environment unlike Finite State Machines.

2.1.2. Subnodes in judgment and processing nodes

Extended GNP with reinforcement learning (GNP-RL) (Mabu et al., 2007) has subnodes in each judgment and processing node (Fig. 2). In the original GNP, one function is assigned to each node and executed when the node is visited. On the other hand, in GNP-RL, several functions (two functions in Fig. 2) are assigned as subnodes and one of them is selected and executed by a reinforcement learning algorithm. Therefore, reinforcement learning can select better function/subnode for each node and the route of the node transition can be optimized. Since this paper deals with a stock trading problem, each subnode in a judgment node has a function to examine a technical index and that in a processing node has a function to buy or sell stocks.

2.2. Genotype representation

The graph structure is realized by the combination of gene structures shown in Fig. 3. NT_i shows the node type of node i. $NT_i = 0$ means start node, $NT_i = 1$ means judgment node, and $NT_i = 2$ means processing node. d_i is a time delay which shows the time spent on the execution of node i. In this paper, d_i of judgment node is set at 1 and that of processing node is set at 5. The time delay is useful to fix the maximum number of nodes to be executed in each action step (in this paper, an action step is one trading day). If five time units (time delays) are assigned to one action step, the action step ends when the time units used by the node transition become five or exceed five. That is, GNP can execute "at most five judgment nodes" or "less than five judgments nodes and one processing node" in one step. The time delay is also effective for avoiding the deadlock of the node transition due to the judgment loop. Even if the judgment loop occurs and GNP cannot execute any processing, each action step will ends with five time units and finally GNP ends its trial without taking any actions, which leads to the low fitness and such GNP will not be selected for the next generation.

Q_{i1},Q_{i2}, ... are Q values (Sutton and Barto, 1998) assigned to subnodes in node i. ID_{i1},ID_{i2}, ... are the node functions of the subnodes. A Q value estimates the sum of the discounted rewards obtained in the future. The contents of the functions are described in the node function library. For example, $NT_i = 1$ and $ID_{i2} = 2$ show the function of subnode 2 in node i is J_2. Judgment nodes examine the technical indexes in the stock market, e.g., RSI, ROC and so on, (The technical indexes used in this paper are described later in Table 4). Each technical index is assigned to each judgment function listed in the library. Processing nodes decide buying and selling actions, so P_1 and P_2 in Fig. 3 show buying and selling actions, respectively.

$C^A_{i1}, C^B_{i1}, \ldots$and $C^A_{i2}, C^B_{i2}, \ldots$show the next node numbers connected from node i. For example, subnode 1 in node i is connected to $C^A_{i1}, C^B_{i1}, \ldots$. The superscripts A and B correspond to the judgment results, so if the judgment result is A at subnode 1, the next node becomes C^A_{i1}.