Evolutionary Models for Agent-Based Complex Behavior Modeling

Zengchang Qin[†], Yingsai Dong[†] and Tao Wan[‡]

[†] Intelligent Computing and Machine Learning Lab
School of Automation Science and Electrical Engineering
Beihang University (BUAA), Beijing, China
[‡] School of Medicine, Boston University, Boston, USA
zcqin@buaa.edu.cn, dysalbert@gmail.com, taowan@bu.edu

Abstract

In this chapter, the essentials of genetic algorithm (GA) following the footsteps of Turing are introduced. We introduce the connection between Turing's early ideas of organized machines and modern evolutionary computation. We mainly discuss the GA applications to adaptive complex system modeling. We study the agent-based market where collective behaviors are regarded as aggregations of individual behaviors. A complex collective behavior can be decomposed into aggregations of several groups agents following different game theoretic strategies. Complexity emerges from the collaboration and competition of these agents. The parameters governing agent behaviors can be optimized by GA to predict future collective behaviors based on history data. GA can also be used in designing market mechanisms by optimizing agent behavior parameters to obtain the most efficient market. Experimental results show the effectiveness of both models. Using evolutionary models may help us to gain some more insights in understanding the complex adaptive systems.

1 Introduction

Alan Turing (1912-1954) is a legend. He is a profound mathematician, logician and esteemed as the father of computer science. He is also a patriotic wartime codebreaker and, tragically, a victim of prejudice - being prosecuted by the police because of his "illegal" homosexuality, that directly leads his suicide at the age of 41. This has been remembered by us and also recorded in his memorial statue plaque, situated in the Sackville Park in Manchester, England [42]. His legendary contributions founded the modern computing and the indirectly create the machine I am using to type and compile this chapter - a MacBook with 2.4 GHz Intel Core 2 Duo and 3 GB 1067 MHz DDR3. These terms can remind us the path of computing revolutions and those ingenious minds following his footsteps.

Like other geniuses in history, his contributions are not limited to one or two fields. He conceived of the modern computer by introducing *Turing machines*

in 1935, pioneered the field later called *Artificial Intelligence (A.I.)* by proposing the famous *Turing test* [38] as a way of determining whether a machine can think [1]. During World War II, Turing's work in code-breaking was regarded by historians short-ended the war in two years. His 1950 paper *Computing Machinery and Intelligence* [38] gave a fresh approach to the traditional mind-body problem, by relating it to the mathematical concept of computability he himself had introduced in his paper *On computable numbers, with an application to the Entscheidungsproblem*. It has a deep influence not only in mathematics and computer science, but also becomes one of the most frequently cited work in modern philosophical literature [44]. In this paper, Turing considers the question "Can machines think?" Since both the terms "think" and "machine" can't be defined in clear and satisfying way, Turing suggests we "replace the question by another, which is closely related to it and is expressed in relatively unambiguous words." Under this scenario, a hypothetical computing prototype called *Turing machine* is proposed.

A Turing machine is a device that manipulates symbols on a strip of infinite tape according to a table of rules. In modern terms, the table of behavior of a Turing machine is equivalent to a computer program. It goes beyond Charles Babbage's unprogrammable mechanical computer [45]. The Turing machine is not intended as a practical computing technology, but rather as a conceptual device representing a computing machine. Despite its simplicity, a Turing machine can be adapted to simulate the logic of any computer algorithm, and is particularly useful in explaining the functions of a CPU inside a modern computer. It helps computer scientists understand the limits of mechanical computation [44]. His work on Turing machines and *Computing Machinery and Intelligence* can be regarded as the foundation of computer science and of the artificial intelligence. But It is not widely realized that Turing was probably the first person to consider to construct systems which could modify their own programs. By the expression of 'genetical or evolutionary search', he also anticipated the 'genetic algorithms' which since the late 1980s have been developed as a less closely structured approach to self-modifying programs [8].

In this chapter, we will revisit Turing's idea on unorganized machines and how it will contribute to the current connectionism. Following his footsteps, we introduce the modern genetic algorithms and their applications in studying collective behaviors in complex adaptive systems. During the last years of his life, Turing also pioneered the field of *artificial life*. He was trying to simulate a chemical mechanism by which the genes of a fertilized egg cell may determine the anatomical structure of the resulting animal or plant [7]. In this chapter, we are studying a similar but much simpler process of how observable collective behaviors are influenced by consisting individual deterministic behaviors. We

[1] Whether a machine can think has been a controversial topic. For example, John Searle proposed a thought experiment called the "Chinese room", which holds that a program cannot give a computer a "mind" or "understanding", regardless of how intelligently it may make it behave [46].

are also hoping to find the answer of how patterns emerge from the complex adaptive systems like financial markets.

This chapter is structured as follows: Section 2 gives a historical introduction on Turing's idea on unorganized machines, which is related to the modern genetic algorithms. Section 3 gives a general introduction on the genetic algorithms. Two novel applications of GAs to the complex adaptive systems (agent-based virtual market models) with detailed empirical evaluation results are introduced in Section 4 and 5, respectively. At the end, we summarize this field of research and discuss its research potentials worthing further investigations.

2 Turing's Unorganized Machines

Throughout his remarkable career in his short life, Turing had no great interest in publicizing his ideas (possibly because of his Bletchley Park[2] experience). Consequently, important aspects of his work have been neglected or forgotten over the years. In an unpublished report in 1948, he first gave a prophetic manifesto of the field of artificial intelligence. This work is unpublished until 1968, 14 years after Turings death, for which we learn that Turing not only set out the fundamentals of connectionism but also brilliantly introduced many of the concepts that were later to become central to AI, these ideas have been rediscovered or reinvented to develop into the fields of neural networks, evolutionary computation and artificial life [6–8].

Table 1. The NAND operation given two inputs and one output.

Input 1	Input 2	Output ← (Input 1 NAND Input 2)
0	0	1
0	1	1
1	0	1
1	1	0

2.1 Turing's Idea of Neural Computation

In this unpublished report, Turing proposed so-called *unorganized machines (u-machines)*. Two types of u-machines are discussed. The first were A-type machines, which are essentially randomly connected networks of logic gates. Specifically, every node (or neuron in Turing's conceptual cortex model) has two inputs and any number of outputs with two states representing by 0 or 1. The output

[2] Bletchley Park is located in Buckinghamshire, England, currently houses the National Museum of Computing. During World War II, Bletchley Park was the site of the United Kingdom's main decryption base, where Turing was working in secret before moving to Hub 8.

of a neuron is a simple logical function of its two inputs. Every neuron in the network executes the same logical operation of "not and" (or NAND): the output is 1 if either of the inputs is 0. If both inputs are 1, then the output is 0 (Table 1). Fig. 1 illustrates a network of A-type machines. The state transition matrix given node assignment at the time T is shown in Table 2.

The second type u-machines were called the B-type machines, which could be created by taking an A-type machine and replacing every inter-node connection with a structure called a connection modifier -which itself is made from A-type nodes. The purpose of the connection modifiers were to allow the B-type machine to undergo "appropriate interference, mimicking education" in order to organize the behavior of the network to perform useful work. Turing took his inspiration from how human cortex works and its self-adaptive ability [9].

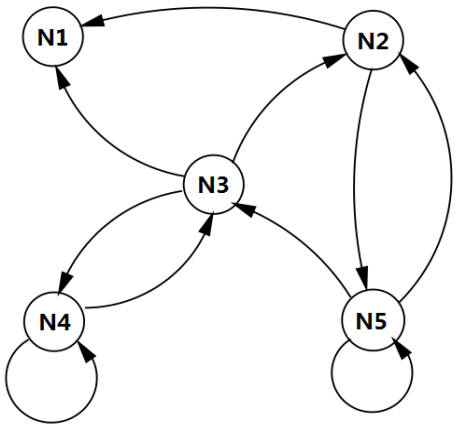

Fig. 1. An example of A-type machine with 5 inter-connected nodes. The graph is modified from [37]. State transition matrix is shown in Table 2.

Table 2. State transition of the network of A-type machines shown in Fig. 1 from time T to $T+7$. For example, $N_1(T+1) = N_2(T)$ NAND $N_3(T) = 1$ NAND $0 = 1$.

Node	T	$T+1$	$T+2$	$T+3$	$T+4$	$T+5$	$T+6$	$T+7$	...
N_1	1	1	0	0	1	0	1	0	...
N_2	1	1	1	0	1	0	1	0	...
N_3	0	1	1	1	1	1	1	1	...
N_4	0	1	0	1	0	1	0	1	...
N_5	1	0	1	0	1	0	1	0	...

Actually, Turing theorized that "the cortex of an infant is an unorganized machine, which can be organized by suitable interfering training." Initially a network that is to be trained contains random inter-neural connections, and the modifiers on these connections are also set randomly. Unwanted connections are destroyed by switching their attached modifiers to interrupt mode. The output of the neuron immediately upstream of the modifier no longer finds its way along the connection to the neuron on the downstream end. Conversely, switching the setting of the modifier on an initially interrupted connection to the other mode to create a new connection [6]. From the modern A.I. point of view, Turing's unorganized machines were in fact very early examples of randomly-connected, binary neural networks, and Turing claimed that these were the simplest possible model of the nervous system.

One thing that makes the field so exciting is the way people studying the human brain work with people who are trying to build artificial intelligence. On the one hand, brainlike structures such as artificial neural networks having the ability to change their responses according to their success or failure (that is, to "learn") are surprisingly good at some tasks, ranging from face recognition to flood prediction. Such learning mechanism of "tuning parameters" or "tuning structures of networks" brought a revolutionary technology of *machine learning*, which has becomes arguably the most successful branch of A.I.

2.2 Turing's Idea of Genetic Algorithms

Changing the settings of the connection modifiers in a B-type network changes its topological structure and functions. Turing had realized that the B-type machines could be very complex when the number of nodes in the network was large. In any moderately sized network there will be a very large number of possible patterns of modifier settings, only a tiny fraction of which will be useful. Any attempt to find best setting patterns of the network by exhaustively searching all possibilities, becomes intractable as the number of nodes increases. Turing himself mentioned a method which is believed to be the most promising for solving the B-type training problem; that of a genetic algorithm (GA), or as Turing called it before the term GA was coined, a genetical search. Based on the original idea of Turing, Webster and Fleming replicate the network designing by GAs [40].

To illustrate this idea, we use the following example. The network structure can be coded into a table of 0s and 1s by considering the input-output relations. Table 3 shows a 5×5 matrix used to represent the network shown in Fig. 1. Modifier for direct connection between nodes is represented by 1, otherwise, it is 0. E.g.:

$$N_1 \leftarrow Input(N_2 = 1, N_3 = 1)$$

indicate that the input nodes for N_1 are N_2 and N_3. Given a network of 5 nodes, any possbile structure of the network can be uniquely defined by a 5×5 matrix S. For each predefined S, we have a corresponding state transition matrix given the initial condition. If we know the state transition matrix A and a set of possible

Table 3. Structure of a network can be coded by 0 and 1 to represent the connections of nodes. The following table represents the network in Fig. 1.

Input	N_1	N_2	N_3	N_4	N_5
Node					
N_1	0	1	1	0	0
N_2	0	0	1	0	1
N_3	0	0	0	1	1
N_4	0	0	1	1	0
N_5	0	1	0	0	1

structure $\mathbf{S} = \{S_1, S_2, \ldots, S_K\}$. Which is the most likely structure $S_i \in \mathbf{S}$ given A? Since the number of possible network structure grows exponentially with the number nodes. How can we adaptively learn such a structure, is the problem we can solve by genetic algorithms today.

For example, we would create a population of randomly connected B-type networks and test each in turn to calculate a score based on the given transition matrix. For each node, if the generated state value is identical to the given training data, we will add one to the score. The final score for the network would the sum of scores across the whole networks in T steps. These scores would become the fitness measures of the individual networks and dictate their number of offspring through biased genetic selection. The most fit networks would be disproportionately over-represented in the next generation, while those poorer scoring networks would be under-represented or drop out of the population altogether. If this test-and-reproduce cycle is repeated for many generations individual networks will become better to fit the training data until eventually a network will be created which gains a perfect score. This idea of genetic search of Turing is one of the earliest ideas in the field of evolutionary computation [19].

3 Genetic Algorithms

A genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. It belongs to a general field of *metaheuristics* for designing computational methods to optimize a problem by iteratively trying to improve a candidate solution regard to a given measure of quality [24]. A GA can be used to generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, crossover and mutation.

3.1 Brief History of Genetic Algorithm

The development of genetic algorithms has its roots in work done in the 1950s by biologists using computers to simulate natural genetic systems [21]. John Holland created the genetic algorithm field. In the cybernetics writing of the 1940s and 1950s there are several, usually fairly vague, mentions of the use of artificial

evolution. In an interview, Holland claimed that he has been focus his attention on adaptive systems. Fisher's book *On the Genetical Theory of Natural Selection* had a great influence on him as his starting point for the genetic algorithm [19]. He claimed that:

> "Computer programming was already second nature to me by that time, so once I saw his mathematical work it was pretty clear immediately that it was programmable ... ... I began to think of selection in relation to solving problems as well as the straight biological side of it. In fact, by the time I was doing the final writing up of my thesis I had already gone heavily in the direction of thinking about genetics and adaptive systems. "

Holland's interest was in machine intelligence, and he and his students developed and applied the capabilities of genetic algorithms to artificial systems. He laid the groundwork for applications to artificial systems with his publications on adaptive systems theory [17]. Holland's systems were self-adaptive in that they could make adjustments based on their interaction with the environment over time.

Beginning in the 1960s Holland's students routinely used selection, crossover, and mutation in their applications. Several of Holland's students made significant contributions to the genetic algorithm field. The term "genetic algorithm" was used first by Bagley in [2], which utilized genetic algorithms to find parameter sets in evaluation functions for playing the game of Hexapawn, that is a chess game played on a 3×3 chessboard in which each player starts with three pawns. In 1975, K.A. DeJong finished his Ph.D. dissertation under Holland's supervision [10]. In his work, a few classical complex function optimization problems were studied by using GA, in which two import metrics for GAs were devised, one to measure the convergence of the algorithm, the other to measure the ongoing performance. David E. Goldberg, another of Hollands students, has concentrated on engineering applications of genetic algorithms. His volume published in 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, is one of the most influential books on genetic algorithms [15]. It has been widely used as a textbook of GAs across all over the world. A more comprehensive history note of the genetic algorithm can be found in [21].

3.2 Essentials of Genetic Algorithm

There are a few good textbooks and tutorials for introducing the genetic algorithm [15, 27]. In this chapter, we are not going to talk about the technical details and the variants of GA. Instead, we give short introduction on the basic ideas of GA. By solving a problem using a genetic algorithm, you must represent a solution to your problem as a *chromosome* (or *genome*). Each chromosome can be interpreted into a particular assignment of variables. For example, if the values for the variable x was a number in range of $0 \sim 256$; then an eight-bit binary number was thus an obvious way of representing it. In this example, suppose

the fitness function $f(x)$ of the problem is the sine function, because the nature of the sine function places the optimal value of $x = 128$, where $f(x) = 1$. The binary representation of 128 is 10000000; the representation of 127 is 01111111. Thus, the smallest change in fitness value can require a change of every bit in the representation. Binary encoding of chromosome is the most common type of coding, mainly because first works in GA used this type of encoding [10].

The genetic algorithm then creates a population of solutions and applies genetic operators such as mutation and crossover to evolve the solutions in order to find the best one(s). In using a GA, usually we need to consider the following three most important aspects. (1) definition of the *fitness function*, (2) definition and implementation of the genetic representation for constructing the *search space* and (3) definition and implementation of the *genetic operators*. Once these three have been well defined, the generic genetic algorithm should work fairly well. Beyond that you can try many different variations to improve performance or computational efficiency (e.g., parallel GAs).

Fitness Function A fitness function is a particular type of objective function that is used to measure how close a given design solution is to achieving the set aims. The fitness function basically determines which possible solutions get passed on into the next generation of solutions (after genetic operations). This is usually done by analyzing the chromosome, which encodes a particular candidate solution to the problem you are trying to solve. The fitness function will look at a pool of chromosomes and make some qualitative assessment, returning a fitness value for that solution. The rest of the genetic algorithm will discard any solutions with a "poor" fitness value and accept any with a "good" fitness value. Two main classes of fitness functions exist: one where the fitness function does not change, as in optimizing a fixed function or testing with a fixed set of test cases; and one where the fitness function is mutable, as in niche differentiation or co-evolving the set of test cases [21].

Search Space If we are solving some problems, we are usually looking for some solutions, which will be the best among others. The space of all feasible solutions (it means objects among those the desired solution is) is called search space, also state space. Each point in the search space (in chromosome coding) represent one feasible solution. Genetic algorithms are about search in this space to find the best chromosome(s) guided by the heuristics of maximizing the fitness function. The chromosome with highest fitness has the highest probability to be selected for genetic operations or directly pass into the next generation.

Genetic Operations The most important operator in GA is *crossover*, based on the metaphor of sexual combination and reproduction inspired by the real biological life which are extremely widespread throughout both the animal and plant kingdoms. Crossover is a term for the recombination of genetic information during sexual reproduction. In practice, after we have decided what encoding we will use, crossover selects genes from parent chromosomes and creates a new offspring. The offsprings have equal probabilities of receiving any gene from either parent, as the parents chromosomes are combined randomly. The simplest way is to choose randomly some crossover point and everything

before this point copy from a first parent and then everything after this point copy from the second parent.

In GAs, mutation is the stochastic flipping of bits in chromosome that occurs in each generation. It is always with a very low mutation rate (e.g., with a probability of something like $0.001 \sim 0.05$). This is to prevent falling all solutions in population into a local optimum of solved problem. Mutation changes randomly to generate new offspring. For binary encoding we can switch a few randomly chosen bits from 1 to 0 or from 0 to 1. As a matter of fact, mutation is not an especially important operator in GA. It is usually set at a very low rate, and sometimes can be omitted.

4 Evolutionary Collective Behavior Decomposition

Collective intelligence is a shared or group intelligence that emerges from the collaboration and competition of many individuals and appears in consensus decision making of agents. Collective behaviors can be modeled by agent-based games where each individual agent follows its own local rules. Agent-based models (ABM) [41] of complex adaptive systems (CAS) provide invaluable insight into the highly non-trivial collective behavior of a population of competing agents. These systems are universal and researchers aim to model the systems where involving agents are with similar capability competing for a limited resource. Agents may share global information and learn from past experience.

In this section, we will introduce an evolutionary approach to study the relationship between micro-level behaviors and macro-level behaviors. A complex collective behavior is assumed to be generated by aggregation of several groups of agents following different strategies. The strategy of agents is modeled by some simple games because of limited information available for the agents. Genetic algorithms are used to obtain the optimal collective behavior decomposition model based on history data. The trained model will be used for collective behavior prediction.

4.1 Complex Adaptive Systems and Pattern Formation

Extensive research in econophysics [26] has been done on agent-based experimental games from the perspective of interdisciplinary disciplines such as physics, mathematics and complexity science. For example, Sysi-Aho proposed a genetic algorithm based adaptation mechanisms within the framework of the minority game, and found that the adaptation mechanism leads the market system fastest and nearest to maximum utility or efficiency [30]. Gou [16] studied how the change of mixture of agents in the mixed-game model can affect the change of average winnings of agents and local volatilities of the artificial stock market.

Unfortunately, fewer research focus on exploring macro-level collective behavior prediction by understanding the emergent properties of macro-level behavior from micro-level behaviors. We can rarely see that agent-based models were put into practice of real market predictions, e.g. predicting fluctuation of the stock

prices. In this chapter, we assume that the collective data are generated from the combination of micro-behaviors of variant groups of agents employing different strategies. We then model and estimate the resource-constrained environment parameters to maximize the approximation of the system outputs to the real-world test data.

In his last years, Turing has focus his interests on patter formation, to understand the orderly outcomes of self-organization. Especially in biology, pattern formation refers to the generation of complex organizations of cell fates in space and time. However, our problem of collective behavior decomposition is sort of reverse version of the pattern formation. The patterns of individual agents are lost through behavior aggregation. We hope to rediscover these lost patterns by studying the micro-level and macro-level relations. Fig. 2 gives an example of

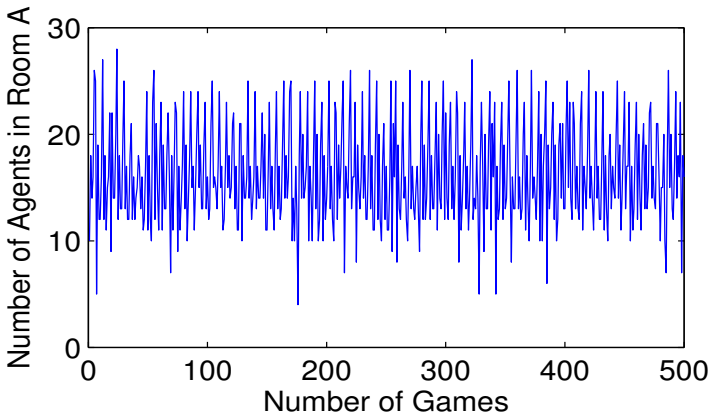

Fig. 2. A sample of random collective behavior, which is generated by a group of agents playing the minority game with fixed strategies.

aggregated collective behavior generated by a group of agents playing the *minority game* [3]. It is obvious to see that the observable collective behaviors are random and no patterns can be directly detected. However, this messy behavior is mostly generated by deterministic individual behaviors. More details are available in the next section.

4.2 Agent Behavior Modeling with Minority Game

Agent-based experimental games have attracted much attention in different research areas, such as psychology [35], economics [13, 36] and financial market modeling [12, 20, 32]. Among these agent-based models, minority game (MG) [3] is an important model in which an odd number N of agents successively compete to be in the minority side. This model can be regarded as a simplified version of

EI Farol Bar Problem [1], in which a number of people decide weekly whether go to the EI Farol bar to enjoy live music in the risk of staying in a crowd place or stay at home. As a new tool for learning complex adaptive systems, the minority game has been applied to variety areas especially in financial market modeling [12, 20, 32]. In real-life scenarios, some agents make random decisions and some groups employ similar strategies. The complexity of marketing world is embodied in existence of varieties types of agents using strategies based on their own rules.

Formally, the minority game consists of N (an odd number) agents, at time t ($t = 1, \ldots, T$), each agent need to take an action $a_i(t)$ for $i = 1, \cdots, N$, to attend room A or B.

$$a_i(t) = \begin{cases} A & \text{Agent } i \text{ choose room A} \\ B & \text{Agent } i \text{ choose room B} \end{cases} \tag{1}$$

At each round t, agents belonging to the minority group win. The winning outcome can be represented by a binary function $w(t)$. If A is the minority side, i.e. the number of agents choosing Room A is no greater than $(N-1)/2$, we define the winning outcome $w(t) = 0$; otherwise, $w(t) = 1$.

$$w(t) = \begin{cases} 0 & \text{if: } \sum_{i=1}^{N} \Delta(a_i(t) = A) \leq \frac{N-1}{2} \\ 1 & \text{otherwise} \end{cases} \tag{2}$$

where $\Delta(\alpha)$ is the truth function:

$$\Delta(\alpha) = \begin{cases} 0 & \alpha \textit{ is} \text{ false} \\ 1 & \alpha \textit{ is} \text{ true} \end{cases} \tag{3}$$

We assume that agents make choices based on the most recent m winning outcomes $h(t)$, which is called *memory* and m is called the *length of memory*.

$$h(t) = [w(t-m), \ldots, w(t-2), w(t-1)] \tag{4}$$

Given the outcome $w(t)$ at the moment t, agent i may keep a record $r_i(t)$ that tells whether it has won the game or not.

$$r_i(t) = \begin{cases} Win & \text{Agent } i \text{ wins at time } t \\ Loss & \text{Agent } i \text{ loses at time } t \end{cases} \tag{5}$$

Table 4. One sample strategy for an agent in the minority game with $m = 4$.

$h(t)$	0000	0001	0010	0011	0100	0101	0110	0111
$S(h(t))$	A	A	B	B	B	A	B	B
$h(t)$	1000	1001	1010	1011	1100	1101	1110	1111
$S(h(t))$	B	A	A	A	A	B	B	B

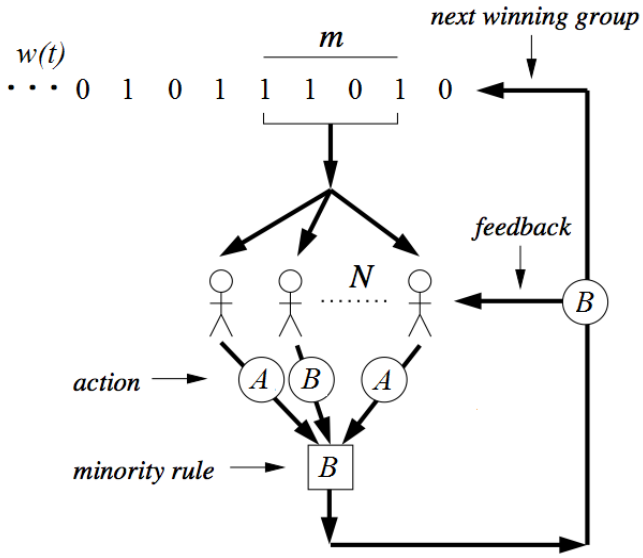

Fig. 3. For a given time step: the strategy maps the last four winning groups ($m = 4$) into the agent decision. Solid thick lines mimic how the information flows in the system: the N agents take the last m numbers (1101 in this case) from the sequence of winning groups and perform an action accordingly. The N actions (A or B) are transformed into the next winning group (0 in this case) through the minority rule. This information is shared with the agents for their own feedback and becomes the next number in the sequence of winning outcomes. This figure is modified from a similar one in [28].

In minority game, we usually assume that each agent's reaction based on the previous data is governed by a "strategy" [3]. Each strategy is based on the past m-bit memory which are described as a binary sequence. Every possible m-bit memory are mapped in correspond to a prediction of choosing room A or B in the next round. Therefore, there are 2^{2^m} possible strategies in the strategy space. Agents employing the same strategy will be categorized as one *strategy group*. Given the memory $h(t)$, the choice for the agent i guided by the strategy S is denoted by $S(h(t))$. The value of m is usually set by a number less than 6 in practical experiments as people tend to use short-term memory rather than a long-term memory in making a 0-1 decision.

Table 4 shows one possible strategy with $m = 4$. For example, $h(t) = [0010]$ represents that if the agent who choose B in the latest three time steps win, the next round (at time t) choice for this agent will be $S([0010]) = B$. A strategy can be regarded as a particular set of decisions on the permutations of previous winning outcomes. The decision process of minority game can be schematically illustrated in the Fig. 3. We assume each agent has its own strategy, at each time step, the agent will take action based on previous m outcomes of the system.

The winning of this round by applying the minority rule will be broadcast to the system.

4.3 Behavior Learning with Genetic Algorithms

In the previous research, Li *et al.* [23] designed an intelligent agent that uses machine learning method to learn the patterns of other agents with complete information, i.e. the information who went to which room in which round of the game is available to the public (i.e. $r_i(t)$ and $w(t)$ for $t = 0, \ldots, T$ and $i = 1, \ldots, n$). Fig. 4 is the performance of the intelligent agent using a probabilistic estimation tree [33]. As we can see from the figure, the predictive power of this agent is significantly better than the random guessing which means that it can capture the patterns very well from a seemingly random and messy collective information shown in Fig. 2.

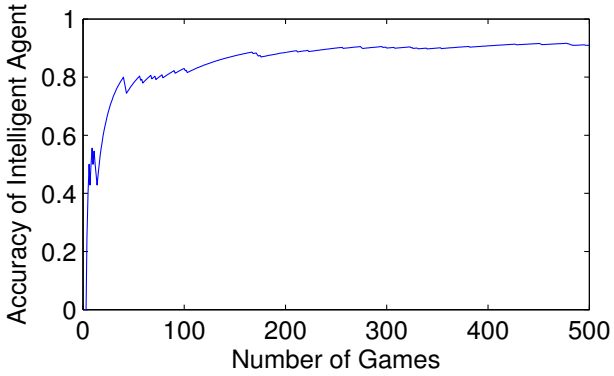

Fig. 4. The performance of the intelligent agent which can learn the behaviors of others with the complete information.

However, the complete information is not a realistic assumption. In most cases, we can only obtain the collective data $w(t)$. Ma *et al.* first proposed a framework that assumes this sort macro-level behavior can be decomposed into the micro-level behaviors of several strategy groups in the minority game. A *Genetic Algorithm* [15] can be used to estimate the parameters of the decomposition. We assume that N agents are divided into a number of strategy groups. One group of agents is random agents, and several groups of agents have fixed strategies of their own. However, we have no idea how many agents in each group and what strategies this group of agents employ. We only know the history of winning outcomes $w(t)$ and an educated guessed maximum group number K. We use a vector of parameters to represent the number of agents in each group and the strategy they use, a GA can be used to optimize these parameters in

order to obtain the most similar history of winning outcome sequence. Since the parameters are independent to each other and the problem is with a large parameter space, using a stochastic search algorithm such as GA is a way for finding the most suitable parameters.

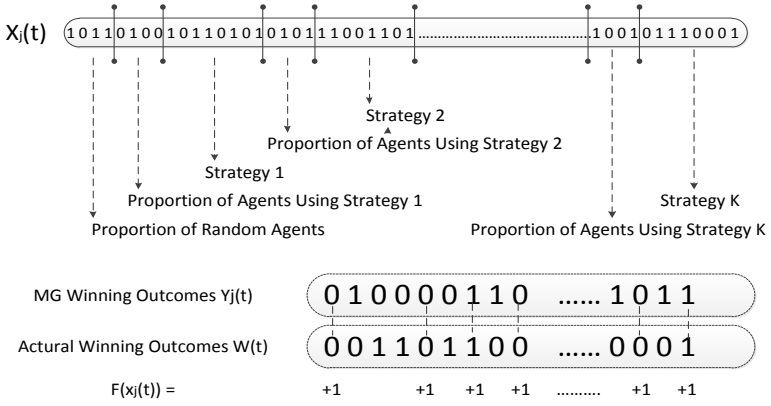

Fig. 5. The process for calculating the fitness function for a chromosome at time t. A chromosome is consisted by numbers of agents in each group and the strategies of this group. For each chromosome $\mathbf{x}_j$, we can obtain a sequence of winning outcomes $y_j(t)$ by running the MGs based on the given parameters. The fitness function is calculated based on the comparisons between $y_j(t)$ and the actual sequence of winning outcomes $w(t)$.

Given the winning outcomes $w(t)$ and a guessed maximum number of groups using fixed strategies K, the agents can be divided into $K+1$ groups:

$$\{G_r, G_1, \ldots, G_K\}$$

where group G_r is the group of random agents and G_k for $k = 1, \ldots, K$ employs the strategy S_k. We use the following parameters to define one MG: the percentage of random agents P_r, percentage of agents with one certain fixed strategy P_{S_k} where S_k is the strategy for the group. Therefore, we can construct a chromosome $\mathbf{x}$ consisting of the following parameters.

$$\mathbf{x} = [P_r, P_{S_1}, S_1, \ldots, P_{S_K}, S_K]$$

The fitness function calculation of $f(\mathbf{x})$ is illustrated in Fig. 5. At time t of the game, in order to evaluate one chromosome $\mathbf{x}_j$ ($j = 1, \ldots, J$ where J is the

population size in the GA), we run the MG with the parameter setting given by $\mathbf{x}_j$ to obtain the history of winning outcomes $y_j(t)$. Comparing $y(t)$ with the actual sequence $w(t)$: for t runs from 1 to a specified time T, once $y_j(t) = w(t)$, we add 1 to $f(\mathbf{x}_j)$. Formally:

$$f(\mathbf{x}_j(t)) \leftarrow \begin{cases} f(\mathbf{x}_j(t)) + 1 & \text{if: } y_j(t) = w(t) \\ f(\mathbf{x}_j(t)) & \text{otherwise} \end{cases} \tag{6}$$

At each time t, the best chromosome $\mathbf{x}^*(t)$ is selected from the pool:

$$\mathbf{x}^*(t) = \arg\max_j f(\mathbf{x}_j(t)) \quad for \quad j = 1, \ldots, J$$

Given the best chromosome $\mathbf{x}^*(t)$, its parameters can give the best possible complete information scenario so that we can use machine learning algorithms to predict each agent's behavior and make final decision based on these predictions [23, 25].

4.4 Modeling with Mixed-Games

The evolutionary collective behavior decomposition is a general framework for studying the micro-level and macro-level relations. In order to obtain a better approximation of the collective behaviors in the real-world market, Gou [16] modifies the MG model and proposes the 'mixed-game model', in which agents are divided into two groups: each group has different memory length, Group G_N plays minority game with the same strategy, while Group G_J plays majority game with the same strategy. Comparing to the MG, the most significant part of mixed-game is that it has an additional group of "trend chasers", therefore be more realistic to simulate a real-world case, e.g., financial market, social networks and etc.

Technically, all agents in G_N choose the best strategy with which they can predict the minority side most correctly, while all agents in G_J choose the best strategy with which they can predict the majority side most correctly. N_1 represents the number of agents in G_N and N_2 represents the number of agents in G_J. We use m_1 and m_2, respectively, to describe the memory length of these two groups of agents. As each agent's reaction is based on a strategy corresponding a response to past memories, there are $2^{2^{(m_1)}}$ and $2^{2^{(m_2)}}$ possible strategies for G_N or G_J, respectively. We assume the completeness of marketing world is embodied in existence of variant groups of agents using their own strategies. Therefore, we improve the mixed-game of Gou [16] by dividing the agents into three diverse types of agents: agents who make random decisions (denoted by G_R), agents of Group G_N (playing the minority game) with different strategies, agents of Group G_J (playing the majority game) with different strategies. Fig. 6 illustrates that the collective behavior is a combination of choices from the above three types of agents. Given history sequence $h(t)$, we can use GA to explore all possible combinations of subgroups or compositions of the market, then use this information to make better choices.

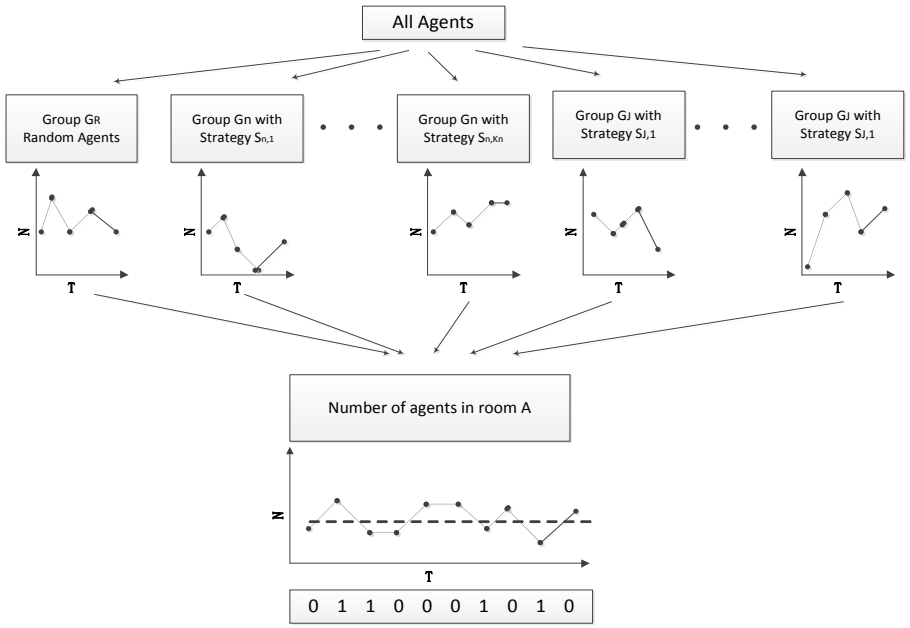

Fig. 6. The generative process for collective data. All agents can be divided into $K_N + K_J + 1$ groups where agents in the same subgroups act identically based on the strategy they follow. The collective data can be regarded as an aggregation of all agents' actions.

Given the history winning outcomes $w(t)$, the expected maximum number of subgroups using fixed strategies in G_N is K_N, and the expected maximum number of subgroups using fixed strategies in G_J is K_J. Thus agents can be divided into $K_N + K_J + 1$ groups:

$$\{G_R, G(S_N^1), G(S_N^2), \ldots, G(S_N^{K_N}), G(S_J^1), G(S_J^2), \ldots, G(S_J^{K_J})\}$$

where G_R represents the group of random agents, $G(S_N^i)$ (for $i = 1, \ldots, K_N$) represents the subgroup agents holding strategy S_N^i in Group G_N (the group playing minority game). $G(S_J^k)$ (for $k = 1, \ldots, K_J$) represents the subgroup agents holding strategy S_J^k in Group G_J.

The chromosome for genetic algorithms $\mathbf{x}$ is encoded with the following parameters:

$$\mathbf{x} = [P_R, P(S_N^1), S_N^1, \ldots, P(S_N^{K_N}), S_N^{K_N}, P(S_J^1), S_J^1, \ldots, P(S_J^{K_J}), S_J^{K_J}]$$

- P_R : the percentage of random agents among all agents (i.e. $P_R = \frac{|G_R|}{N}$)
- $P(S_N^i)$: the percentage of the number of agents in the minority game subgroup i ($i \in [1, 2, \ldots, K_N]$) with the fixed strategy S_N^i (i.e. $P(S_N^i) = \frac{|G(S_N^i)|}{N}$).

- S_N^i: Binary coding of the minority game strategy S_N^i.
- $P(S_J^k)$: the percentage of the number of agents in the majority game subgroup k ($k \in [1, 2, \ldots, K_J]$) with the fixed strategy S_J^k (i.e. $P(S_J^k) = \frac{|G(S_J^k)|}{N}$).
- S_J^k: Binary coding of the majority game strategy S_J^k.

This evolutionary mixed-game learning model was first proposed by Du *et al.* [11] and some empirical results to verify its effectiveness in the real-world applications will be given in the next section.

4.5 Experimental Studies

The general framework is referred to *evolutionary game learning* (EGL) and the micro-level behavior of agent can be modeled by either minority game, mixed-game or other game theory models. The model with the mixed game is referred to as *evolutionary mixed-game learning* (EMGL) and the model with the minority game is *evolutionary minority game learning* (ENGL). In the following experiments, We tested these two models on the U.S.Dollar-RMB (Chinese Renminbi) exchange rate[3]. For each trading day t, suppose the opening price is V_b and the closing price is V_f, the fluctuation of price can be transferred to $w(t)$ as follows:

$$w(t) = \begin{cases} 1 & \text{if: } V_b < V_f \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

By correctly predicting $w(t)$ using the learning model, we can capture the ups and downs of the market prices though we are not trying to predict the exact price at this stage.

In the following experiments we set $K_N = K_J = 20$. Since almost all agents play with history memories of 6 or less in a typical MG, and m_N is usually larger than m_J when using mixed-game model to simulate real market [16], we set $m_N = 4, 5, 6$ and $m_J = 3$ to establish three configuration of EMGL models. For example, EMGL(6-3) represents $m_N = 6$, $m_J = 3$. We set $K = 20$ and $m = 3$ for the ENGL model. As for the GA, we set population size $J = 50$, crossover rate $P_c = 0.8$, mutation rate $P_m = 0.05$. We run the whole experiments for 30 times to reduce the influences of randomness in GAs.

From the USD-RMB experiment shown in Figure 7, we can see both EMGL (starred curve) and ENGL (dotted curve) can predict with high accuracy (the mean accuracy is up to 58.6% for ENGL and 63.8% for EMGL (4-3)), indicating a strong existing pattern captured by the new models. In general, almost all results of ENGL and EMGL are statistically better than the random guess (the mean is around 50% with a small variance) plotted at the bottom. Du *et al.* tested the EMGL and ENGL models on 13 Chinese stocks. The experimental results show that both models perform significantly better than the random guess [11].

[3] Data obtained from: http://bbs.jjxj.org/thread-69632-1-7.html

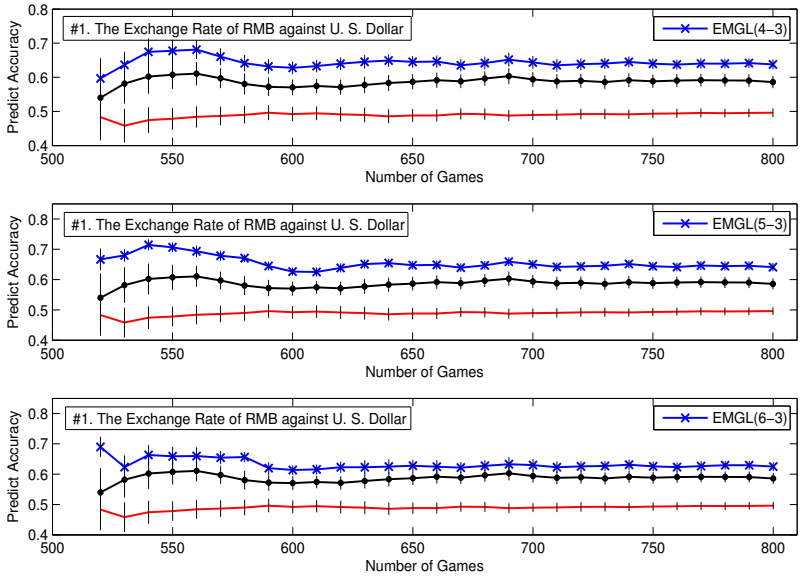

Fig. 7. Performance of the ENGL model and the EMGL model with different memory lengths on the USD-RMB exchange rate. Each curve is plotted on the mean accuracy with plus and minus the standard deviation.

5 Evolutionary Market Mechanism Designs

In the last section, we investigated the collective behavior decomposition in agent-based market and introduced an evolutionary learning framework of modeling agent behavior by game theory models. We use the new model to predict the collective behaviors by learning from the history data. The collective behavior of the market is assumed to be the aggregation of individual behaviors. In the simple minority game and mixed-game modeling, the behavior of agents are governed by a set of parameters and make decisions independently. This is not a realistic assumption, as we know, the interaction between agents are the key issue for why the system is so unpredictable and complex. In this section, we will mainly consider the interactions between agents and how should they operate under the rules of market - the market mechanism.

The market mechanism design is an important topic in *computational economics and finance* for resolving multi-agent allocation problems [22]. In this section, we review relevant background of trading agents, and market designs by evolutionary computing methods. In particular, a genetic algorithm can be used to design auction mechanisms in order to automatically generate a desired market mechanism for markets populated with trading agents.

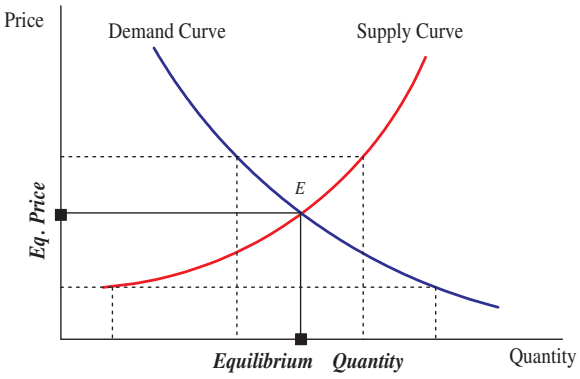

Fig. 8. An schematic illustration of a supply-demand schedule, where the intersection E is the equilibrium.

5.1 Market Mechanism

In every classical economic model, demand and supply always play prominent roles. Supply is used to describe the quantity of a good or service that a household or firm would like to sell at a particular price. Demand is used to describe the quantity of a good or service that a household or firm chooses to buy at a given price. For a buyer, with increasing of quantity of the commodity, he will be inclined to bid a lower price to make a purchase, but with the less quantity of commodity, he has to increase his bid price. Because buyers want to make purchases at lower prices so that the demand curve slopes downward. For sellers, if the commodity is at a higher price, they will be inclined to sell as many as they can, that keeps the supply curve slope upward. The intersection of the supply curve and demand curves is called the equilibrium, and the corresponding price and quantity are called, respectively, the *equilibrium price* and the *equilibrium quantity* (Fig. 8). In case of prices beyond the equilibrium, the market will self-correct them to the equilibrium by an "invisible hand" according to Adam Smith. At an equilibrium price, consumers get precisely the quantity of the good they are willing to buy at that price, and sellers sell out the quantity they are willing to sell at that price. Neither of them has any incentive to change. In a competitive market, the price actually paid and received in the market will tend to the equilibrium price. This is called the law of supply and demand [29].

In economics and game theory, interactions of traders consist of two components: a protocol and a strategy. Protocol defines the valid behavior of traders during the interaction. It is set by the marketplace owner and should be known publicly for all the participants. Strategy is privately designed by each agent to achieve their negotiation objectives within a protocol. In the previous section, the minority game model was used for modeling the agent strategy. In this section, we will put our focus on the protocol. Moreover, the effectiveness of the strategy is very much dependent on the protocol: an optimal strategy for one

protocol may perform very badly for other protocols. In a marketplace, the protocol is an"auction". It is the market mechanism by which buyers and sellers interact in this marketplace. Strategy is the adaptive behavior or "intelligence" of traders such as the ZIP agents' [4] updating rules that will be discussed later.

There are many types of auctions. English Auction (EA), sellers keep silent and buyers quote increasing bid-prices, and the buyer with highest bid is allowed to buy; Dutch Flower Auction (DFA), buyers keep silent and sellers quote decreasing offer-prices and the seller with lowest offer is allowed to sell. In other auctions such as the Vickery or second-price sealed-bid auction, sealed bids are submitted and the highest bidder is allowed to buy, but at the price of the second highest bid. EA and DFA are also called single sided auctions because either buyers or sellers are active but not both. The Continuous Double Auction (CDA), one the most popular of all auctions, allows buyers and sellers to continuously update their bids/offers at any time in the trading period. The bids/offers are quoted simultaneously and asynchronously by buyers/sellers. At any time the sellers/buyers are free to accept the quoted bids/offers [32].

In 1950s, Smith [36] demonstrated that markets consisting of small numbers of traders could still exhibit equilibration to values predictable from classical microeconomic theory. In a given supply-demand schedule with n transactions between 'sellers' and 'buyers', the coefficient of convergence α ($0 \leq \alpha \leq 1$)is introduced to measure the deviation of transaction prices from the theoretical market equilibrium price p_0 [36]. α is calculated at the end based on transaction prices p_i for $i = 1, \cdots, n$. The coefficient of convergence is defined as follows:

$$\alpha = 100 \cdot \delta_0 / p_0 \tag{8}$$

where

$$\delta_0 = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (p_i - p_0)^2} \tag{9}$$

The E-market discussed in this chapter as well as in [5] and [32] is based on Smith's experiment and the α measure is used to evaluate the convergence of the market.

5.2 Agent Strategy Modeling

Zero-intelligence (ZI) agents were initially proposed to explore the relationship between limited rationality, market institutions and the general equilibration of markets to the competitive equilibrium [12]. The fundamental discovery is that within the classical double auction (CDA) market only the weakest elements of rationality is needed to exhibit high allocative efficiency and price convergence in a competitive market. This convergence is later proved as a statistical must but not an emergent behavior. Zero intelligence plus (ZIP) agents, proposed by Cliff [4] as an augmented version of ZI agents use a simple machine learning algorithm to adapt their behavior for maximizing their own utility function.

Each ZIP trader i is given a private secret limit price, λ_i, which for a seller is the price below which it must not sell and for a buyer is the price above which it must not buy (based on Smith's experiment). The pseudo-code of the ZIP agent's strategy is shown in Alg. 1. If a ZIP trader completes a transaction at its λ_i price then it generates zero utility, where utility for traders means the profit for the sellers or saving for the buyers. Each ZIP trader i maintains a time-varying profit margin $\mu_i(t)$ and generates quote-prices $p_i(t)$ at time t according to

$$p_i(t) = \lambda_i(1 + \mu_i(t)) \tag{10}$$

$$p_i(t) = \lambda_i(1 - \mu_i(t)) \tag{11}$$

for sellers and for buyers, respectively. Trader i is given an initial value $\mu_i(0)$ (when $t = 0$) which is subsequently adapted over time using a simple machine learning technique known as the Widrow-Hoff (W-H) rule which is well used in gradient optimization and back-propagation neural networks. The W-H rule has a "learning rate" β_i that governs the speed of convergence between trader i's quote price $p_i(t)$ and the trader's idealized target price $\tau_i(t)$ which is determined by a stochastic function of last quote price with two small random absolute perturbations: $A_i(t)$ and $R_i(t)$. $A_i(t)$ is generated uniformly from the interval $[0, C_a]$ denoted by $\mathcal{U}[0, C_a]$ for sellers and $\mathcal{U}[-C_a, 0]$ for buyers. For sellers, $R_i(t)$ is generated from

$$R_i(t) \sim \mathcal{U}[1, 1 + C_r]$$

and for buyers

$$R_i(t) \sim \mathcal{U}[1 - C_r, 1]$$

C_a and C_r are called system constants. To smooth over noise in the learning, there is an additional "momentum" γ_i for each trader (momentum is also used in back propagation neural networks.

For each ZIP agent i, its adaptation is governed by three real-valued parameters: learning rate β_i, momentum γ_i and initial profit margin $\mu_i(0)$. Because of the randomness and the uncertainty involved in trading, a trader's values for these parameters are assigned at initialization, using uniform distributions: for all traders, β_i, γ_i and $\mu_i(0)$ are sampled from:

$$\beta \sim \mathcal{U}(\beta_{min}, \beta_{min} + \beta_{\Delta})$$

$$\gamma_i \sim \mathcal{U}(\gamma_{min}, \gamma_{min} + \gamma_{\Delta})$$

$$\mu_i(0) \sim \mathcal{U}(\mu_{min}, \mu_{min} + \mu_{\Delta})$$

Hence, to initialize an entire ZIP trader market it is necessary to specify values for the six market-initialization parameters β_{min},β_{Δ}, γ_{min}, γ_{Δ}, μ_{min}, μ_{Δ} plus the other two system constants C_a and C_r. Clearly, any particular choice of values for these eight parameters can be represented as a vector:

$$V = [\beta_{min}, \beta_{\Delta}, \gamma_{min}, \gamma_{\Delta}, \mu_{min}, \mu_{\Delta}, C_a, C_r] \in \mathbf{R^8}$$

which corresponds to a single point in the 8-dimensional space of possible parameter values. A Genetic Algorithm can be used to explore this space for parameter optimization. The degree of price convergence to the equilibrium price can be used as the fitness function.

5.3 Evolutionary Optimization

Market mechanism design addresses the problem of designing an auction in which the agents' interaction generates a desirable macro-scale outcome, by assuming the trading agents are self-interested. A desired market can be simply considered as the one with least transaction price variance to the equilibrium price determined by the market's supply-demand schedule. Therefore, the fitness function for each individual can be calculated by monitoring price convergence in a series of n CDA market experiments, measured by weighting Smith's α measurement on the given supply-demand schedules. If each experiment lasted k "days", the score of experiment number e is:

$$S(V_i, e) = \frac{1}{k} \sum_{d=1}^{k} w_d \alpha(d) \tag{12}$$

where $\alpha(d)$ is the value of α and w_d is the weight on the day d. According to the experiments in [5], all experiments last for 6 days and we place a greater emphasis on the early days of trading. The weights are set as follows: $w_1 = 1.75$, $w_2 = 1.50$, $w_3 = 1.25$ and w_4, w_5 and w_6 are all equal to 1.00. The fitness of the genotype V_i is evaluated by the mean score of n experiments:

$$F(V_i) = \frac{1}{n} \sum_{e=1}^{n} S(V_i, e) \tag{13}$$

Where $n = 50$ the performance of trading experiments are fairly stable based on empirical work in [34]. The lower fitness a market has, the sooner the market approaches to the equilibrium and the smaller price variance the market has. GAs were used for optimizing the parameters for ZIP agents and showed that evolved parameter settings via GAs perform significantly better than "educated guessing" in CDA [4].

Now consider the case when we implement CDA. At time t, either a seller or a buyer will be selected to quote, which means that sellers and buyers have a fifty-fifty chance to quote. We use Q_s to denote the probability of the event that a seller offers. Then in CDA, $Q_s = 0.5$. For English Auction $Q_s = 0$ and Dutch Flower Auction $Q_s = 1$; which means, sellers cannot quote and sellers are always able to quote, respectively. The inventive step introduced in [5] was to consider the Q_s with values of 0.0, 0.5 and 1.0 not as three distinct market mechanisms, but rather as the two endpoints and the midpoint on a continuum referred as a continuous auction space. For other values, e.g., $Q_s = 0.1$, it can be interpreted as follows: on the average, for every ten quotes, there will be only one from sellers while 9 are from buyers. This also means, for a particular significant time

t, the probability of a seller being the quoting trader is 0.1. The fact is, this kind of auction is never found in human-designed markets. However, no one knows whether this kind of *hybrid mechanism* in which $Q_s \neq 0, 0.5$ or 1.0 is preferable to human-designed ones. This motivates us to use a GA to explore with additional dimension Q_s ranging from 0 to 1 giving us the following genotype based on the old one by adding a new dimension Q_s:

$$[\beta_{min}, \beta_{\Delta}, \gamma_{min}, \gamma_{\Delta}, \mu_{min}, \mu_{\Delta}, C_a, C_r, Q_s] \in \mathbf{R^9}$$

According to the experiments in [5], the hybrid mechanisms are found to be the optimal auctions in 2 of the 4 given schedules.

Although the case of $Q_s = 0.5$ is an exact approximation to the CDA in the real-world, the fact that a trader will accept a quote whenever the quoting price satisfies his expected price. For the two single sided extreme cases of $Q_s = 0.0$ and $Q_s = 1.0$, this model is not an exact analogue of the EA and DFA. Qin and Kovacs [34] proposed a more realistic auction space. All the following experiments are conducted in this realistic auction space. More detailed are available in [34].

5.4 Trading with Heterogeneous Agents in CDA

Smith's experiment [36] qualitatively indicated that the relationship of the supply-demand schedule has an impact way in which transaction prices approached the equilibrium, even with a small number of participants. This experiment has been conducted by using ZI [12] and ZIP agents [5], respectively. Here we will consider the case of using a mixture of the same number of ZI and ZIP agents, which are referred to as heterogeneous agents experiments.

For all agents, the distribution of limit price determines the supply and demand curves for the experiment and their intersection indicates the theoretical equilibrium price and quantity. In the simulation of real marketplaces, we assume that each significant event (quoting, making deal or not making deal etc.) always occurs at a unique time. In the CDA market, at time t, an *active* trader (seller or buyer) i is chosen randomly to quote a price $p_i(t)$ to become the "current quote $q(t)$", where the active traders are ones who still have utility (goods or money) for deals. Next, all traders on the contra side (i.e. all buyers j if i is a seller, or all sellers j if i is a buyer) compare $q(t)$ to their current quote price $p_j(t)$ and if the quotes cross (i.e. if $p_j(t) \leq q(t)$ for sellers or $p_j(t) \geq q(t)$ for buyers) then the trader j is able to accept. If no traders are able to accept, the quote is regarded as "ignored". For ZIP traders, either the current quote is accepted or ignored and the traders update their profit margins $\mu(t)$ using the W-H rule. For example, suppose the last quote is an offer and was accepted at price q then any sellers for which their price is less than q should raise their profit margin with learning rate of β_i. The details about the updating rules for ZIP agents can be found in [4] (See Alg. 1). For ZI traders, the previous transaction prices and the status of the last offer do not have cause any influence on their further actions (ZI traders are not intelligent, they only quote prices randomly).

Algorithm 1: Pseudocode for updating rules of ZIP traders

```
For Sellers: ;
if the last shout was accepted at price q then
    any seller s_i for which p_i ≤ q should raise its profit margin;
    else
        if the last shout was a bid then
            any active seller s_i for which p_i ≥ q should lower its margin
        else
            if the last shout was an offer then
                any active seller s_i for which p_i ≥ q should lower its margin
For Buyers: ;
if the last shout was accepted at price q then
    any buyer b_i for which p_i ≥ q should raise its profit margin;
    else
        if (the last shout was an offer then
            (any active buyer b_i for which p_i ≤ q should lower its margin)
        else
            if the last shout was a bid then
                any active buyer b_i for which p_i ≤ q should lower its margin
```

5.5 Experimental Studies

In this section, we conduct a series of experiments of evolutionary designs of market mechanism based on heterogeneous agents where ZI and ZIP agents have the approximately same number. The auction space model is the one proposed in [34]. All experiments are based on four given supply-demand schedules: SD_1, SD_2, SD_3 and SD_4 (see Fig. 9). There are 22 trading agents in the experiments, 11 sellers and 11 buyers, each of them is initialized with one unit of goods and their limit prices are distributed as supply and demand curves show. The vertical axis represents price and the equilibrium price is 2.00 for all these 4 given schedules. Each schedule of supply and demand curves is stepped. This is because the commodity is dealt in indivisible discrete units, and there are only a small number of units available in the market. Thus, supply and demand in this simple market differs appreciably from the smoothly sloping curves of an idealized market. These are the same schedules have also been used in previous studies [4, 5, 31, 34, 32] for the convenience of comparison studies.

Fig. 10 shows the performance of the three groups of agents: ZI only, ZIP only and the heterogenous mixture of of ZI and ZIP. It is obvious that the ZIP only group has the minimum variance (α value) because the learning ability of the ZIP agents. ZI agents are the most naive agents without learning ability, the α value for this group is no doubt the largest of the 3. The right-hand side figure of Fig. 10 shows the performance of the heterogeneous agents under different auctions: EA, DFA and CDA. Though the differences are not statistically different, we

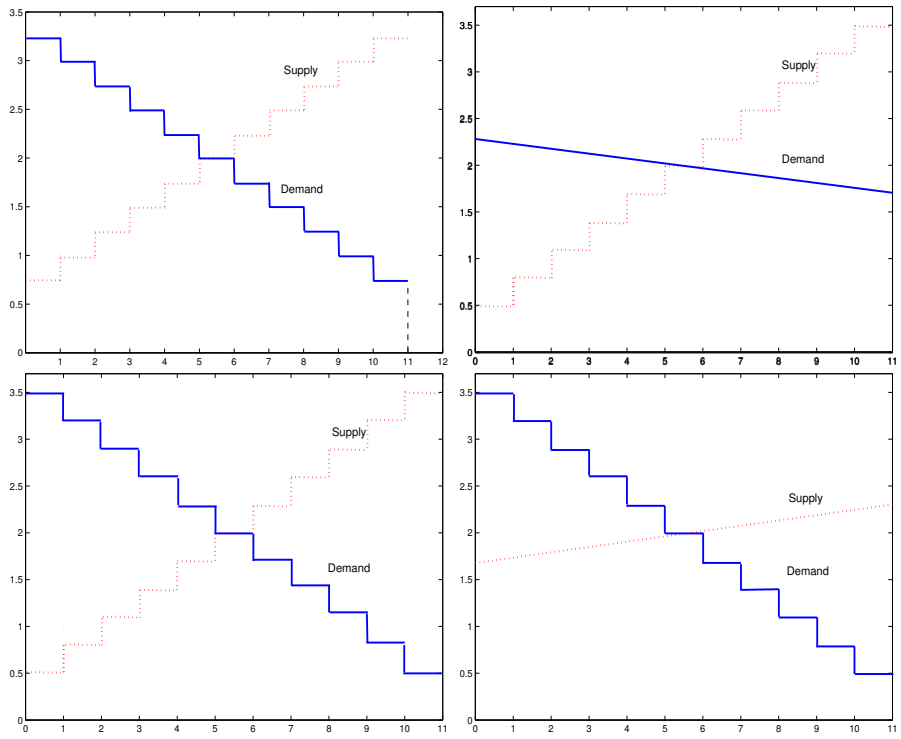

Fig. 9. Supply-demand schedules for experiments: SD1, SD2 (upper) and SD3, SD4 (bottom).

can still see the CDA gives the best performance in these three human-designed auctions.

In the market evolution experiments, a simple GA is used to minimize the fitness value (see equation 13) given 25 independent runs of trading experiments. Population size is 20 and each parameter is coded with 8 bits, crossover rate is a constant with the value of 0.7 and mutation rate is 0.015. Elitism strategy is applied which means that the fittest individual in each generation is logged. We run 600 generations in a single experiment. However, one of the drawbacks of GA is that it cannot be guaranteed the global optimum. Thus we gain formal simplicity at the cost of computation. We run the entire process of evolution many times independently and reduce the effect of randomness as time goes by, to encourage convergence. The results of Q_s represented here are based on 25 independent runs of the GA on the given 4 supply-demand schedules and the average results with standard deviation through generation 600 are shown in Fig. 11.

As we can see from the figures, although Q_s values converges to real-world auctions in 3 of the 4 given schedules, we still found a hybrid auction in SD_4.

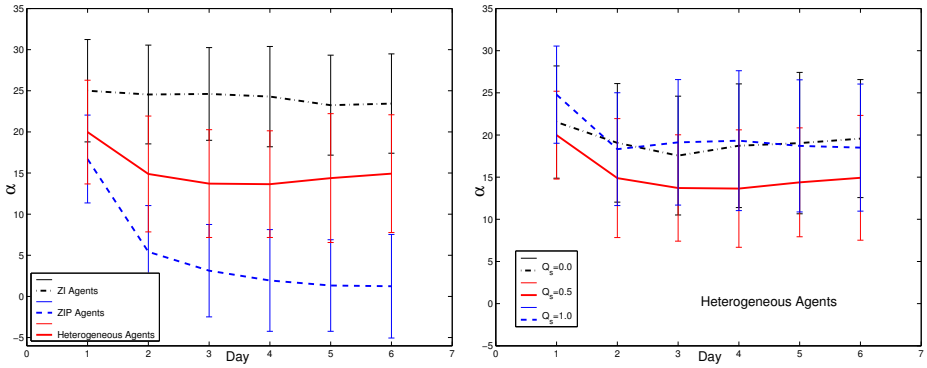

Fig. 10. Left: the average performance of 3 groups of agents: ZI only, ZIP only and the mixture of the same number of ZI and ZIP. Right: given a population of heterogenous agents, the comparisons of α value under different auctions: EA ($Q_s = 0$), DFA($Q_s = 1$) and CDA ($Q_s = 0.5$).

Comparing the ZIP agents in the old auction space and the new auction space, the only difference is SD_3. Both in the old auction [4] and new auction space [34, 32] with ZIP agents, there were hybrid auctions found by GAs. Cliff [5] presented a result of using only ZI agents given SD_3 and the hybrid auction was found. However, the Q_s values for these hybrid auctions are different: $Q_s = 0.39$ for experiments with ZI agents only, $Q_s = 0.16$ for ZIP agents in the old auction space and $Q_s = 0.23$ for ZIP agents in the new auction space [34]. Here in the experiment with heterogeneous agents which are a mixture of ZI and ZIP agents, the optimal auction is CDA but not a hybrid one. We believe that the optimal auction for a market is related to the supply-demand schedule given. So far, we just demonstrated with empirical studies due to the complexity of such problems. The theoretic relations among hybrid auction, supply-demand schedule, the number of agents and other factors are considered as a future work. However, we demonstrated that given a particular supply-demand schedule, we can use some machine learning technology to find the optimal auction for such a market.

6 The End

Turing died from cyanide poisoning, possibly by his own hand. On June 8, 1954, shortly before what would have been his 42nd birthday, he was found dead in his bedroom. The logo of Apple computer is often erroneously referred to as a tribute to Alan Turing, with the bite mark a reference to his method of suicide. It is not true though even Steve Jobs hopes it were [43]. He had left a large pile of handwritten notes. Decades later this fascinating material is still not fully understood.

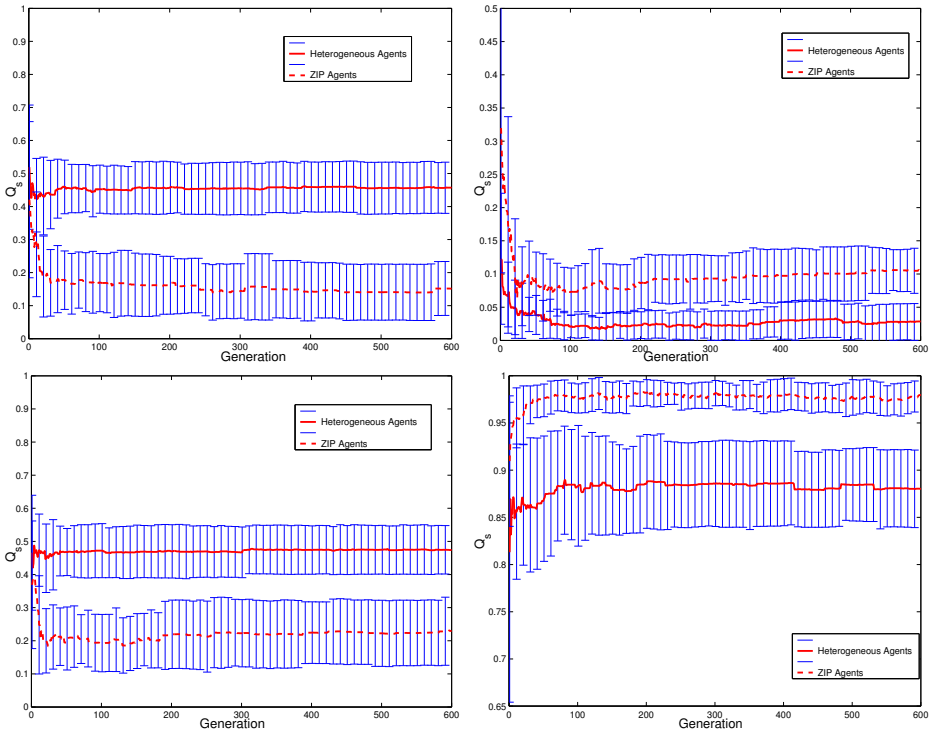

Fig. 11. The comparisons of evolutionary trials of Q_s for ZIP (dot lines) and heterogeneous agents (solid lines) on schedules SD1 to SD4 through 600 generations.

In this chapter, we follow Turing's footstep and recognize his early ideas in neural networks and evolutionary computation thanks to Copeland and his colleagues [6–8]. We interpret his ideas of genetic algorithm by a novel example based on Webster and Fleming's work [40]. The essentials of genetic algorithm are summarized following after a brief history of the GA. We introduced two novel evolutionary models in agent-based computational economics. Both models use GAs to optimize the agent behavior to obtain the wanted market dynamics. The first model studies the collective behavior decomposition which is to estimate individual agent's strategies (or behaviors) from the random macro-level information. The second model uses GAs to optimize both agents' strategies and the protocol between them (market mechanism) in order to obtain the most efficient market. Both models' performance are testified by experimental studies to show the effectiveness.

2012 is the year we celebrate Turing's 100 birthday and his contributions that has led us to the new era of computing and information technology. We have witnessed the development of computer science and its impact on our life. The research presented in this chapter is relatively new. Following Turing's footsteps,

we shall see a bright future of using computation to understand the economics, psychology, sociology and other complex adaptive systems.

Acknowledgment

This work is partially funded by the NCET Program of MOE, China and the SRF for ROCS. ZQ also thanks the China Scholar Council for the 6-month visiting fellowship (No. 2010307502) to CMU.

References

1. Arthur, W.: Bounded rationality and inductive behavior (the El Farol problem). American Economic Review 84. 406 (1994)
2. Bagley, J. D.: The behavior of adaptive systems which employ genetic and correlation algorithms. Ph.D. dissertation, University of Michigan, Ann Arbor (1967)
3. Challet D. and Zhang Y.: Emergence of cooperation in an evolutionary game. Physica A 246, 407 (1997)
4. Cliff D.: Minimal-intelligence agents for bargaining behaviors in market-based environments. Technical Report HPL-97-91, Hewlett-Packard Laboratories (1997)
5. Cliff, D.: Explorations in evolutionary design of online auction market mechanism. Electronic Commerce Research and Applications, Vol. 2: 162-175 (2003)
6. Copeland, B.J. (Ed.): The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life plus the Secrets of Enigma. Oxford Press (2004)
7. Copeland, B.J. and Proudfoot, D.: Alan Turings forgotten ideas in computer science. In Scientific American. April: 99-103 (1999)
8. Copeland, B.J. and Proudfoot, D.: On Alan Turing's anticipation of connectionism. Synthese, 108: 361-377 (1996)
9. Eberbach, E., Goldin, D., and Wegner P.: Turings ideas and models of computation Ch. Teuscher (Ed.), Alan Turing: Life and Legacy of a Great Thinker, Springer-Verlag 159-194 (2004)
10. De Jong, K. A.: An analysis of the behavior of a class of genetic adaptive systems. Doctoral dissertation, University of Michigan, Ann Arbor (1975)
11. Du, Y., Dong, Y., Qin, Z. and Wan, T.: Exploring market behaviors with evolutionary mixed-games learning model. In Proceedings of International Conference on Computational Collective Intelligence (ICCCI), Part I, LNCS 6922, 244-253 (2011)
12. Gode, D. and Sunder, S.: Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality, Journal of Political Economy, Volume 101(1): 119-137 (1993)
13. Farmer, J. D. and Foley, D.: The economy needs agent-based modelling, Nature, 460, 685-686 (2009).
14. Fisher, R. A. : On the Genetical Theory of Natural Selection. Oxford: Clarendon Press (1930)
15. Goldberg, D. E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley (1989)
16. Gou, C.: Agents play mix-game. Econophysics of Stock and Other Markets, LNCS, Part II, 123-132 (2006).

17. Holland, J. H.: Outline for a logical theory of adaptive systems. Journal of the Association for Computing Machinery, 3, 297-314 (1962)
18. Holland, J. H.: . Adaptation in Natural and Artificial Systems. (2nd Ed.). The MIT Press (1992)
19. Husbands, P., Holland, O. and Wheeler M. (Eds.): The Mechanical Mind in History. MIT Press (2008)
20. Johnson, N., Jefferies, P. and Hui, P.: Financial Market Complexity, Oxford University Press, Oxford (2003)
21. Kennedy, J. and Eberhart, R. C., with Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers (2001)
22. LeBaron, B.: Agent-based computational finance: Suggested readings and early research, Journal of Economic Dynamics and Control, Vol. 24: 679-702 (2000)
23. Li, G., Ma, Y., Dong Y. and Qin, Z.: Behavior learning in minority games, In Proceedings of Collaborative Agents - Research and Development (CARE 2009/2010). LNCS 6066, 125-136 (2011)
24. Luke, S.: Essentials of Metaheuristics. (2009)
available at http://cs.gmu.edu/∼sean/book/metaheuristics/
25. Ma, Y., Li G., Dong Y., and Qin Z.: Minority game data mining for stock market predictions In Proceedings of International Workshop on Agents and Data Mining Interaction (ADMI). LNCS 5980, 178-189 (2010)
26. Mantegna R. and Stanley H.: An Introduction to Econophysics: Correlations and Complexity in Finance, Cambridge University Press (1999)
27. Mitchell M.: An Introduction to Genetic Algorithms. MIT Press (1996)
28. Moro, E.: The Minority Game: an introductory guide. Korutcheva Cuerno (Ed.) Advances in Condensed Matter and Statistical Physics. Nova Science Publisher, Inc. (2004)
29. Stiglitz J. E. and Driffill, J.: *Economics*, W. W. Norton & Company, Inc. (2000)
30. Sysi-Aho, M. Chakraborti, A. and Kaski K. : Searching for good strategies in adapative minority games. Physical Review. DOI:10.1103/PhysRevE.69.036125 (2004).
31. Z. Qin, *Evolving Marketplace Designs by Artificial Agents*, MSc Dissertation, Computer Science, University of Bristol, 2002.
32. Qin, Z.: Market mechanism designs with heterogeneous trading agents, Proceedings of Fifth International Conference on Machine Learning and Applications (ICMLA), 69-74 (2006)
33. Qin, Z.: Nave Bayes classification given probability estimation trees, the Proceedings of Fifth International Conference on Machine Learning and Applications (ICMLA), 34-39, (2006)
34. Qin, Z. and Kovacs, T.: Evolution of realistic auctions, M. Withall and C. Hinde Ed. Proceedings of the 2004 UK workshop on Computational Intelligence, 43-50, Loughborough, UK (2004)
35. Rapoport, A., Chammah A. and Orwant C.: Prsoner's Dilemma: A Study in Conflict and Cooperation, University of Michigan Press, Ann Arbor (1965)
36. Smith, V.: An experimental study of competitive market behavior, Journal of Political Economy, 70: 111-137 (1962)
37. Teuscher, C. and Sanchez E.: A revival of Turings forgotten connectionist ideas: exploring unorganized machines. In Connectionist Models of Learning, Development and Evolution, Perspectives in Neural Computing, 153-162 (2001)
38. Turing, A.: Computing machinery and intelligence, Mind, LIX (236): 433-460. doi:10.1093/mind/LIX.236.433
39. Turing, A.: Intelligent machinery. In Collected Words of A. M. Turing: Mechanical Intelligence, D. C. Ince (Ed.), Elsevier Science (1992)

40. Webster, C. and Fleming, W.: Evolved Turing neural networks. Available at: http://compucology.net/evolved
41. Wiesinger, J., Sornette, D., Satinover J.: Reverse engineering financial markets with majority and minority game using genetic algorithms. Swiss Finance Institute Research Paper No. 10-08 (2010).
42. http://en.wikipedia.org/wiki/Alan_Turing_Memorial
43. http://en.wikipedia.org/wiki/Alan_Turing
44. http://plato.stanford.edu/entries/turing/
45. http://en.wikipedia.org/wiki/Charles_Babbage
46. http://en.wikipedia.org/wiki/Chinese_Room