Cytoskeleton Simulation Applied in Equity Portfolio Management

Bo Zhang

Supervisor: Chris Clack

September 2005

Introduction

1.1 Aim

The main aim of this project is to build a system that can be used by other people to formulate hypotheses. It represents the way a fund manager thinks and makes investment decisions in terms of the internal biochemistry of a cell. If the system is successfully built up, it can provide a different way of viewing investment decisions.

In addition to the main aim, there are two secondary aims of this project: 1) the constructed system must allow for contradictory rules to coexist; 2) the system must allow complex behaviours to emerge from simple rules.

From a user's perspective, success may not be viewed as an ability to make money, but rather what the process of designing a rule set does to increase our understanding of how fund managers make investment decisions.

1.2 Objectives

Biologically, cells sense signals from their environment and behave accordingly. But such reaction to the environment has intrinsic delay on account of proteins' reaction chains. As a result, cellular behaviours may be smoothed[1] unless it is placed in a constantly changing environment. However, the environment should not change too quickly; otherwise the cellular behaviour may be unable to reflect the change. In order to clearly observe the cellular adaptation, I choose to model an equity investment procedure with terms of biochemistry in this project[2], as the equity market is usually more volatile than the bonds market but less than others, e.g. the foreign change market. More concretely, I am going to model the investment procedure in a scenario of equity portfolio management.

In order to achieve the project aims, I have set up a set of project objectives as following:

1 Construct a system which can maintain an equity portfolio.

2 The managing procedure is actually conducted in the same way as

[1] By term "smoothed", I mean that signals from environment may fade away during internal protein reactions and thereby cause no cellular behaviour.

[2] The reason that I choose to model only equity investment is based on consideration of time limit of this project.

biological cells react to their environment.

3 The portfolio must be managed by using different technical strategies[3] set by the user. The fundamental strategies are outside of the scope of this project.

4 The system must act dynamically according to real market data.

1.3 Report Structure

I will organize remaining parts of this report into following chapters:

Chapter 2: provides background information of biological cytoskeleton and financial investment and the discussion of related work.

Chapter 3: discusses about designing issue of the system. The contents are divided into environment design and artificial cytoskeleton design.

Chapter 4: provides how the system is implemented in detail. Data structure and algorithm used to construct the system will be given in this chapter.

Chapter 5: evaluates the system. The evaluation is performed by testing system behaviour. A critical evaluation is also given at the end of this chapter.

Chapter 6: concludes the project and reviews the project aims and objectives.

Chapter 7: outlines further work that could be taken to extend the system.

[3] An introduction of strategies will be given in section 2.1.3.

Background

In this chapter, a quick scan of background information is given. There are three sections in this chapter. The first section will briefly introduce the equity portfolio management. The second section focuses on an introduction of cytoskeleton. And I will give a concise review of related work achieved by others in the last section.

1.4 Equity Portfolio Management

Portfolio management is “the art and science of making decisions about investment mix and policy, matching investments to objectives, asset allocation for individuals and institutions, and balancing risk vs. performance” [5]. In this section, I will start with a brief introduction about portfolio management process, and then move on to compare two styles of equity portfolio management. After that, I will shift the focus on a comparison between two main technical strategies. And finally, I will end this section with a quick scan of some popular technical indicators.

1.1.1. Portfolio Management Process

The process of managing a portfolio never stops, and it mainly consists of three steps and three loops [9], as shown in Figure 2-1.

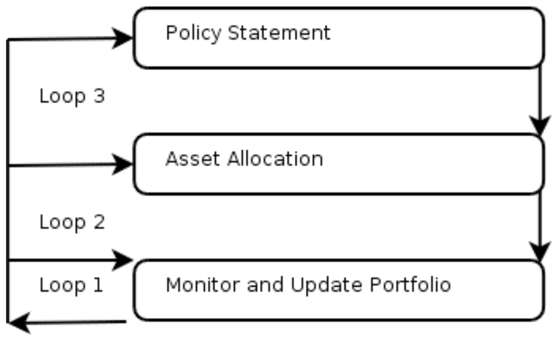


Figure 2-1 Portfolio Management Process

Step one focuses on an investor’s needs and goals. The output of this step is a policy statement. Step two concentrates on the investor’s risk tolerance, and outputs an asset allocation. Fundamental analyses on overall economy and different industries are required at this step. The third step monitors the portfolio performance and triggers modifications of the policy statement and the asset allocation.

1.1.2. Passive and Active Management

There are two styles of equity portfolio management: passive and active. Passive management styles are actually long-term buy-and-hold strategies. The goal of a passive management is normally tracking a chosen index over a period of time. Its purpose is not to beat the target index but rather to match the index. On the contrary, active management style is an attempt to outperform a chosen index[4] within certain risk. An active manager generally modifies the managed portfolio more frequently. [9]

1.1.3. Technical Strategies

Strategies that are used to manage the equity portfolio can be classified as fundamental strategies and technical strategies. As stated in Chapter 1, this project only concerns about technical strategies which are further categorized into momentum strategies and contrarian strategies.

1.1.3.1. Momentum Strategy

Portfolio managers who are using momentum strategies tend to believe that recent price trends will continue. In this case, they usually analyze past market data to discover the momentum[5] of the price movement. If the momentum pushes the price upwards, the portfolio manager assumes such trends will continue and therefore buys the stock and waits to sell it at a relative higher price. The underlying belief is that investors periodically underreact to the arrival of new information.

1.1.3.2. Contrarian Strategy

A contrarian portfolio manager bases his decisions on the hypothesis that the market tends to overreact to new information. A contrarian manager prefers to buy a stock when the majority of other investors are bearish[6] and sell the stock when others are bullish[7]. The overall behaviour of contrarian managers is that they attempt to purchase a stock when it is near its lowest price and sell it when

[4] It always called benchmark index in the scenario of active equity portfolio management.

[5] At this position, momentum means the driving force of price movement. It is different from momentum indicator that will be discussed later.

[6] Bearish investors believe the market will fall in a short time.

[7] Bullish investors believe the market will rise in the short future.

it is near its highest price [9].

1.1.4. Technical Indicators

Technical indicators are mathematical calculations based on market data and used to predict future price trends [5]. Most rational traders in the market use these indicators to support their investment decisions. In the following sections, I will quickly walk through five indicators that are widely used.

1.1.4.1. FTSE

It is a stock market index and best known for the FTSE 100, an index of blue chip stocks on the London Stock Exchange. It can be used as a benchmark to evaluate the portfolio performance and also can be used to predict future market movement.

1.1.4.2. Moving Average (MA)

It is the arithmetic average[8] of a stock price over a certain period. The purpose of using average is to smooth out price fluctuations and emphasize the direction of trend [5]. Normally used MAs are 20-day, 30-day, 50-day and 100-day averages. Interpretations of those MAs are different: 20-day MA keeps some volatility and varies quickly according to the price; 100-day MA is much less volatile but shows a long term trend. For example, when a stock price is below its 100-day average, it means that the stock is possibly oversold.

Moving average can be used in many ways. Investors can employ MA as filters. For example, they may stop selling the stock when its price drops 10% below the MA. Another way of investigating MA is to concentrate on crossovers of different types of MAs. For instance, if a stock's 10-day average is lower than its 20-day average but higher than its 50-day average, we may conclude that this stock has hit its resistance level and tends to fall down.

All MAs I mentioned above are averages of stock prices. However, moving average does not limit to the stock price, and can be used on other indicators, e.g. volume.

[8] There are ways of calculating the moving average other than arithmetic average. For example, Exponential Moving Average (EMA) is calculated in the way that recent prices have more weight.

1.1.4.3. Volume

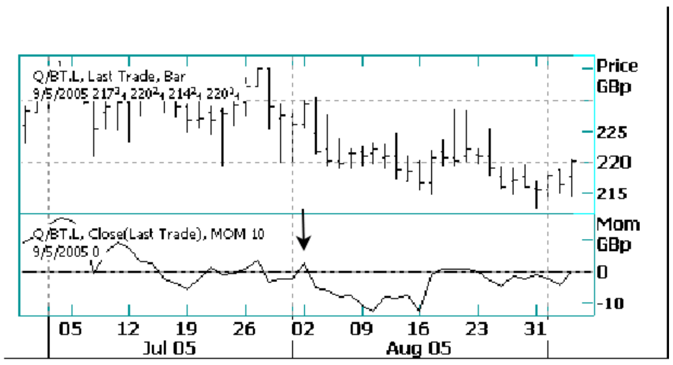
It is an indicator often used along with MA to analyze supply and demand [9]. Volume changes can be used as a confirmative indicator to price movement. For example, when a technical trader detect an upwards price trend, he may then turn to look at volume. If volume is relatively heavier to its normal volume, the bullish movement found in price is probably ensured. Using volume indicators, we can also discovery support and resistance levels of a stock price. Reilly and Brown [9] give a definition of these levels as “support level is the price range at which the technician[9] would expect a substantial increase in the demand of the stock; resistance level is the price range at which the technician would expect an increase in supply of stock and a price reversal”. Support and resistance levels can be further used to indicate whether the stock price is at overbought or oversold[10] position.

1.1.4.4. Momentum

Momentum measures the velocity of price change. John, the author of “Technical Analysis of the Financial Market” [5], defines the formula for momentum as

$$M = V - Vx$$

where M stands for momentum, V is the latest price and Vx is the closing price x number of days ago. In practice, x is often chosen as 10, and calculated momentum of a stock is plotted in a zero line graph. Positive momentum will be plotted above the zero line and negative momentum is plotted below the zero line. And a momentum plot line is thereby drawn (see Figure 2-2). A rising line above zero means an uptrend of price is developing. And a falling line below zero means a downtrend of price (as an arrow pointed in Figure 2-2).



[9] The term “technician” is equivalent to technical trader in this report.

[10] It is widely believed that stock prices at those positions will reverse their trends.

Figure 2-2 10-day Momentum of BT stock price

1.1.4.5. Covariance

Covariance is one of the best-known measures of risk [9]. Reilly and Brown define it as "a measure of the degree to which two variables 'move together' relative to their individual mean values over time" [9]. Covariance can be used differently at different stage of portfolio management. At the stage of asset allocation, covariance is often examined to avoid portfolio risk. For example, three stocks (A, B and C) with same expected return and volatility are under consideration. Stock A and stock B are covariant, and stock C is negatively covariant with others. The portfolio will be constructed with either stock A and C or stock B and C, because these combinations have less overall risk. However, at the stage of monitoring portfolio, the usage of covariance is quite different[11]. Using the same example as above and suppose the portfolio consists of B and C, if a momentum trader has discovered an uptrend in stock B, he might modify the portfolio by buying stock B and selling stock C as the price of stock C is possibly moving downwards.

Listed indicators above are those hired by most technical indicators. There are also many other indicators being employed in real market such as spread, on balance volume and MACD[12]. A technical trader may prefer some indicators over others.

1.5 Cytoskeleton

In "Molecular Biology of The Cell", Bruce [1] defines the cytoskeleton as following:

"The ability of eukaryotic cell to adopt a variety of shapes and to carry out coordinated and directed movements depends on a complex network of protein filaments that extends throughout the cytoplasm. This network is called the *cytoskeleton*."

Mainly there are three kinds of protein filaments formed the cytoskeleton. They are actin filaments, microtubules and intermediate filaments. Actin filaments are long and thin fibers. Intermediate filaments are fibers that have diameters that are between actin filaments' and microtubules'. Microtubules are straight and hollow cylinders. They are usually much longer and thicker than other types of

[11] In reality, covariance is seldom used at this stage. However, in this project, covariance with this meaning is important as it will be discussed in Chapter 3.

[12] MACD is short for Moving Average Convergence Divergence

filaments.

Actin filaments are particular essential to the motility and growth of the cell. According to observations of biological cells, actin filaments usually form a network at the edge of the cell. The network dynamically assembles and dissembles based on environmental signals and consequently alters the shape of the cell. Precisely, when an actin filament is growing towards the cell membrane, it produces a protrusive force against the membrane and push it outwards. However, when most actin filaments shrink away from the cellular membrane, they cause the cell to contract as the membrane is lack of mechanical support. The details of how actin filaments grow and shrink depend on cooperative and competitive interactions of many different kinds of proteins (e.g. ABPs that will be introduced in the next section).

Although we have already gained a lot of knowledge about cytoskeleton, many areas such as proteins functionality, distribution and force of cell motility, are still open and uncertain.

In the remaining parts of this section, I will briefly introduce some actin associated proteins and give a rather detailed description of actin dynamics.

1.1.5. Actin Associated Proteins

Actin is one of the most ubiquitous proteins on the earth. Its related system is so complex that involves about 60 different classes of actin-binding proteins (ABP). Kreis and Vale provide a complete table of all those classes of proteins in [7]. Cofilin, profilin and Arp2/3 complex are three kinds of those ABPs which participate in regulating actin dynamics.

1.1.5.1. Cofilin

Cofilin is relatively small and plays an important role in severing actin filament into monomeric actins[13]. The result of cofilin activities is the acceleration of actin filament turnover rate. Its features can be summarized into following:

1. Cofilin is widely distributed within cytoplasm.
2. They are concentrated in regions containing dynamic actin pools.
3. It helps deactivating Arp2/3 complex and so that debranching actin filament.
4. They also serve as companion to transport actin into the cell nucleus in times of cell stress.

[13] Monomeric actin can be simply referred as actin that is not in a filament.

5. Although it is not clear that how cofilin is balanced in the cell, it is discovered that cofilin concentration decreases by pretreatment with PIP2[14] [7].

1.1.5.2. Profilin

Profilin is also a small actin that is found globular in the cytoplasm. Its features are:

1. Profilin binds to actin monomers.
2. It also binds to Arp2/3 complex.
3. It links transmembrane signaling to the actin cytoskeleton. Section 2.2.2 will describe how profilin works in transmembrane signaling in detail [7].

1.1.5.3. Arp2/3 Complex

Arp2/3 complex consists of seven actin-related proteins and therefore is larger than cofilin and profilin. Arp2/3 complex is also found ubiquitous in cytoplasm. As will be shown in section 2.2.2, the major role of Arp2/3 complex is to facilitate assembling the actin network.

1.1.6. Actin Dynamics

Actins and some ABPs form an actin network on the edge of a cell. The actin network mechanically supports the cellular membrane and it assembles and disassembles based on environmental signals. The assembly of actin network may protrude the cellular membrane and disassembly may cause the membrane contraction. Figure 2-3 shows an overall life cycle of assembly and disassembly actin network.

---

[14] PIP2 is another kind of membrane protein that takes part in transforming environment signals.

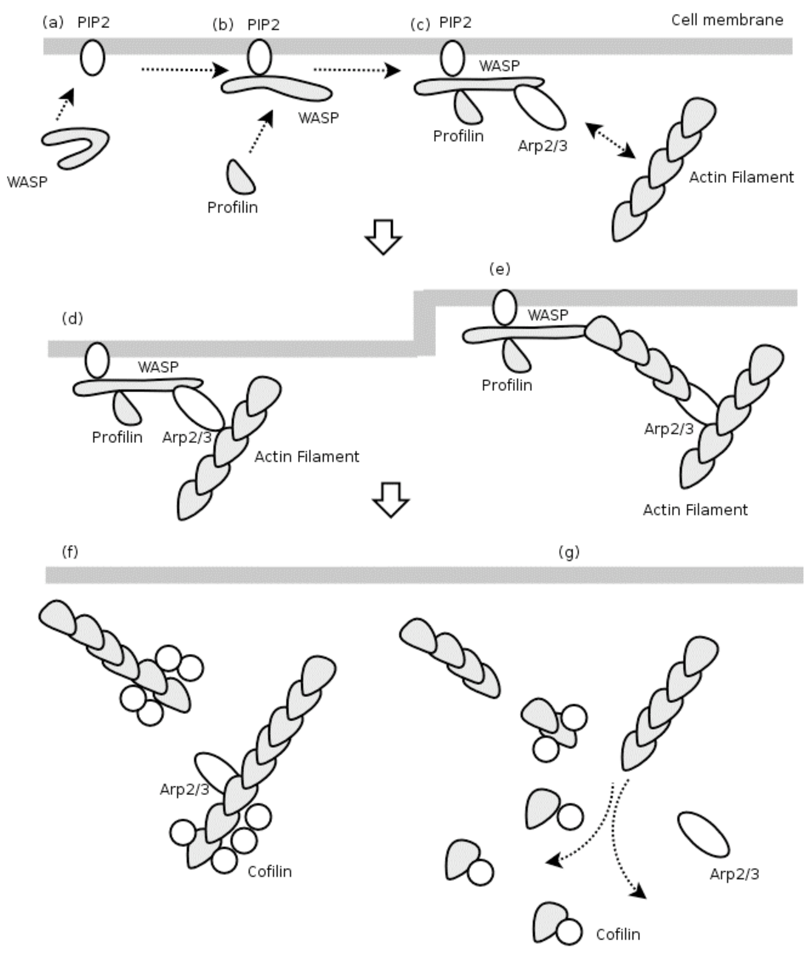


Figure 2-3 Actin Dynamic

1.1.6.1. Assembly of Actin Network

As it is shown in Figure 2-3, when a receptor on the membrane is triggered by its neighbour environment, a nearby PIP2 is also activated. After that, a sequence of protein reactions starts. The first step is that an activated PIP2 binds a WASP protein. The bound WASP protein has an ability to bind other two kinds of proteins: Arp2/3 protein and actin monomers. An Arp2/3 complex is activated when it has been bound to the WASP and the activated Arp2/3 complex has the ability to start a new branch of actin filament in its neighbourhood.

Currently there are two models of branching filaments. One of them believes that the Arp2/3 complex only binds to one side of the filament and starts a new branch of filament at an angle of 70 degree. The branched filament grows slower than the original. In the other model, the branching starts at the barbed end[15] of the filament. In this case, two branches of the filament grow at the same speed [11] (see Figure 2-3).

1.1.6.2. Disassembly of Actin Network

The actin network disassembles when its actin filaments are depolymerized. Normally, the depolymerization of an actin filament is cause by the hydrolysis of ATP to ADP. The actin filament grows by recruiting ATP-actins at its barbed end. Driven by the hydrolysis of ATP to ADP, the ATP-actin turns into an ADP-actin within the filament. And the ADP-actin is then bound by cofilin and dissociates from the actin filament at its pointed end. However, if an actin filament is branched by an Arp2/3 complex, its pointed end is caped by Arp2/3. In this case, the filament depolymerizes in one of two ways: 1) the filament is severed by cofilin binding to ADP-actins that are not at the pointed end[16]; 2) Arp2/3 loses its affinity to the ADP-actin which is at the pointed end of branched filament and uncaps the branched filament. The Arp2/3 complex dissociated from the branched actin filament because it has a much lower affinity for the pointed ends of ADP-actin filaments than for those of ATP-actin filaments. To sum up, the actin network disassembly is driven by the hydrolysis of ATP to ADP as filaments age and cofilin binding.

1.6 Related Work

1.1.7. Multi-agent Based Portfolio Management System

As the financial market provides plenty of price and volume data together with a great number of unresolved problems, it attracts attention of the academic society. Multi-agent system is one of widely accepted methodologies to study the financial market. In this section, the very two different multi-agent systems used in portfolio management are introduced.

1.1.7.1. WARREN

WARREN system is an application of RETSINA (Reusable Task

[15] In reality, a filament grows and shrinks at both ends. However, there is one end that filament grows faster than filament shrinks. Such end is often called the barbed end of the filament. The other end shrinks faster and is called the point end.

[16] Severing filaments may also trigger polymerizations of actin filaments when the severing happens at the point of an ATP-actin. It can not be achieved by cofilin binding as cofilin only binds to ADP-actin.

Structure-based Intelligent Network Agents) framework[17] in financial portfolio management. Its structure is similar to that of investment houses which have teams of specialists working for different tasks. The system consists of three types of agents: interface agents, task agents and information agents. The user can interact with the system through web pages generated and maintained by interface agents. Behind those interfaces agents, a few kinds of task agents such as fundamental analysis agents, technical analysis agents and breaking news agents are working together to accomplish several component tasks of portfolio management. These agents make requests for information to finish their task. The requests trigger the third type of agents, information agents, to gather information from various sources [10].

1.1.7.2. Simulated Stock Market

Artificial financial market is another approach of using multi-agent systems to deal with financial problems. Most of these models attempt to "create an entire functioning financial market" [6]. For example, Kendall and Su buildup a simulated stock market in [6]. In their model, many agents are acting as traders in the market. Each agent investigates some technical indicators and uses a multi-layer neural network to develop its trading strategy. The evolvement of the system is governed by a genetic algorithm in which each agent performs both individual learning and social learning. According to the experiments with this system, agents have abilities to learn different strategies with different stocks. However, their model can only detect buying and selling signals of one stock and it is currently unable to maintain a portfolio of stocks.

1.1.8. Artificial Cytoskeleton

The cytoskeleton is complex and involves a large number of proteins' interactions. Bentley and Clack [2] model an artificial cytoskeleton with only a few types of important proteins, such as actin, Arp2/3 complex and PIP2. The system is actually a multi-agent system that utilizes cell automata techniques. Although each agent acts by simple rules, the overall model successfully simulates assembly and disassembly activities within the cell and consequently reacts to environmental signals dynamically. More descriptions of their model are explored in the remaining parts of the report.

[17] More information about RETSINA is provided on this website: http://www.cs.cmu.edu/~softagents/retsina.html

Design

Within this chapter, I am going to refine the aim of the project into several objectives, and then I will give an overview of the system design. After that, detailed descriptions about two main subsystems, environment and Cellanimat[18], will be given. And I am going to finish this chapter by a detailed explanation about how to design the investment strategies and code them into Cellanimat.

1.7 Objectives

According to the aim of project mentioned in Chapter 1, I further refine it into following objectives:

- To design a system that consists of large number of agents with simple rules. These agents can cooperate and compete with each other so that their overall behaviour can be regarded as monitoring an equity portfolio (see loop 3 in section 2.1.1). Technical analysis will be used.

- To buildup a subsystem to simulate a cytoskeleton, called the Cellanimat. The Cellanimat must operate only based on simple cellular automata rules and follow theories / hypotheses and facts that had been proved or widely approved in molecular biology society. The system will be an extension of the model developed by Bentley and Clack [2].

- To construct a mapping from portfolio management behaviours to morphogenesis and movement of the Cellanimat. Specifically, the volume of the Cellanimat must reflect the number of stock shares in the portfolio. They style of management and strategies that are used in management can be formulated with the Cellanimat.

In addition to above two major objectives, other yet substantial sub-objectives are worth taken into account:

- Cellanimat must have ability to employ hybrid investing strategies.

- Cellanimat must have high extensibility, especially must be able to customize new investing strategies.

[18] Cellanimat is used to refer artificial cytoskeleton that will be developed in this project in order to distinguish from the term of grid cell which will be given in section 3.2.2 [. It is equivalent to terms of “cytoskeleton”, “biological cell” and “cell” within the scope of this report. The term “Cellanimat” is actually borrowed from [4]

1.8 System Overview

1.1.9. General Structure

The system can be divided into two subsystems: financial and cellular subsystem (see Figure 3-1). The financial subsystem is responsible for connecting to financial data source and conducting basic financial analysis, e.g. preparing technical indicators. The cellular subsystem maintains an ecosystem which contains a Cellanimat and its environment. The system updates temporally step by step. In each time step, the financial subsystem feeds the Cellular subsystem with financial data and receives information about the Cellanimat's behaviours (e.g. growing and shrinking) and maps them into trading behaviours (e.g. buying and selling).

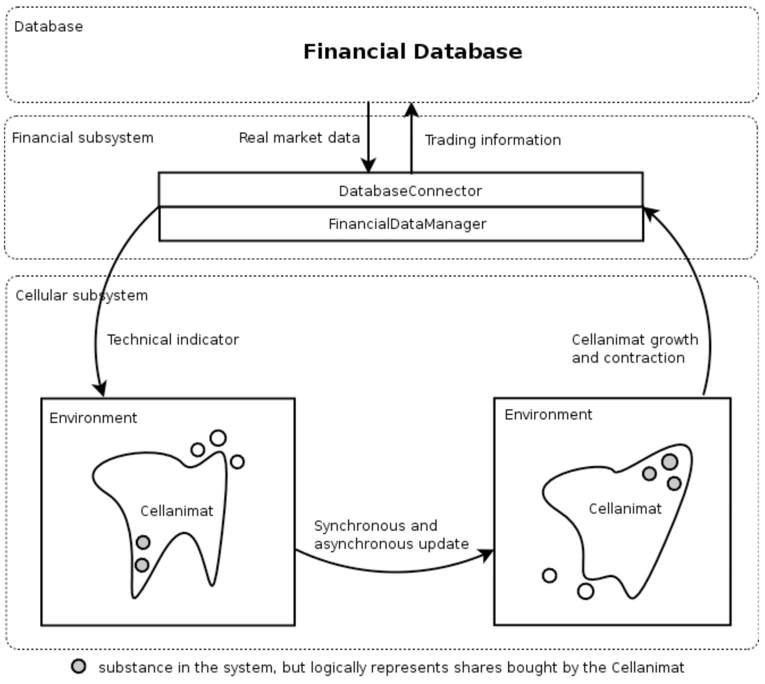


Figure 3-1 Overall view of the system

1.1.9.1. Financial Subsystem

The financial subsystem is responsible for fetching and cleansing financial data, performing basic technical analysis, such as calculating covariance, and feeding those indicators to the environment of biological subsystem. Information does not only flow from financial subsystem to biological system, but also streams back as trading records (see Figure 3-1).

1.1.9.2. Cellular Subsystem

The cellular part of the system maintains an ecosystem which can be further divided into an environment and a ‘biological’ cell. The cell is able to sense the environment, triggers a sequence of protein reactions and so exhibits overall morphological plasticity and locomotion. A further discussion about the Cellanimat is in section 3.4. The environment part lies between the financial and biological subsystems. Further explanation of environment will be given in section 3.3.

1.1.10. Raster World

Partially inspired by artificial chemistry modeling techniques, Bentley and Clack use a 3D voxellated world to model their artificial cytoskeleton [2]. Their data structure provides a convenient platform to model and run Cellular Automata. So I follow the same idea of the fundamental data structure as Bentley and Clack employ in [2], yet use a 2D raster world as basic data structure of biological subsystem, since it is simpler and sufficient to model cellular the Cellanimat.

Possible contents in each grid cell are modeled as agents. They represent different kinds of molecules and proteins in the system. Each agent has some encoded simple rules to follow. The overall performance of the system depends on cooperative and competitive interactions between these agents. Table 3-1 lists all possible contents in a grid cell.

Content	Description
Substance	Substance that moves around the envrionment, may be eaten / vomited by the Cellanimat
Receptor	Receptor on the membrane of the Cellanimat. l
Cytoplasm	Cytoplasm of the Cellanimat that contains various proteins
Actin protein	Actin protein, major components of actin filaments
Arp2/3 complex	Arp2/3 complex that branches actin filaments

Table 3-1 Contents of grid cell

1.9 Environment Design

1.1.11. Previous Design

In the first design, the environment is designed in a straightforward way: 1) the whole raster world is divided into sections; 2) each section radiates from the centre at certain direction; 3) each section is associated with one potential stock that the Cellanimat might invest; 4) when the Cellanimat grows in one section, it means buying the associated asset. Selling procedure happens when it shrinks in that section.

In the second edition, substance agents are introduced into the system and local rules of the receptor are created: 1) shares of stocks are modeled into substance agents; 2) a large number of substance agents are moving around in the environment; 3) types of substance movements include random and forced; 4) forced movement is governed by covariance between stocks that substance agents are associated with, i.e. substance agents with positive covariance attract each other while those with negative covariance repel each other ; 5) Cellanimat eats and vomits substance agents corresponding to buying and selling trades.

1.1.12. Defects of Previous Designs

1.1.12.1.Buying stocks with negative covariance

In the first design, two adjacent sections may unfortunately associate to two stocks with negative covariance. In such case, receptors may behave very different even they are very close spatially. Therefore, the membrane tends to be zigzag. However, the membrane will not really exhibit zigzag appearance as internal filaments' growth can not be precisely directed by receptors. The direct consequence is that purchasing one stock may be accompanied with buying a neighbour (in the context of Cellanimat) stock even they are negative covariance[19]. The overall performance of Cellanimat will be discounted if it happens too frequently.

1.1.12.2.Receptor behaves independently

In the first design, receptors behave according to their attributes which are inherited from their spatial positions. That means receptors behave independently to each other. They base their actions on global information rather than local information i.e. what they detect from their neighbours. Such independent activities are against design objectives[20].

1.1.12.3.Congregation of substance

In the second design, although substance moves around at early stage of the system, it will eventually bump into another substance which is associated with

[19] The reason why buying stocks with negative covariance is bad is given in section 2.1.4.5

[20] Receptors' independent activities do not correspond to what is found in a biological cell.

the same stock and therefore attract to each other and stop moving. Given a certain amount of time, all substance in the environment may conglomerate together to form several blocks of substance. At that time, the system behaves in a similar way as first design or even worse as some stocks may congregate at the place which is unreachable to cellanimat. To solve this problem, the attraction force between substances has to be removed.

1.1.13. Refined Design

To overcome previous defects, I extend the second design into following: 1) chemical substance agents are still moving around in the environment but can only perform random and repelled movement; 2) substances that are eaten by the Cellanimat also performs intracellular movements to mimic the substance and cytoplasm streaming (see Figure 3-2); 3) before the system starts to run, substance agents are randomly chosen to fill up the Cellanimat to form an initial portfolio; 4) the number of substance agents in the system is constant, although the number of substance agents in the environment may vary.

Receptors on the membrane of the Cellanimat are now able to sense local environment signals that are carried by chemical substance agents in their neighbourhood. When the substance agent is on the membrane of the Cellanimat, it does not mean that it is eaten by the Cellanimat and therefore the represented stock share is still in the market.

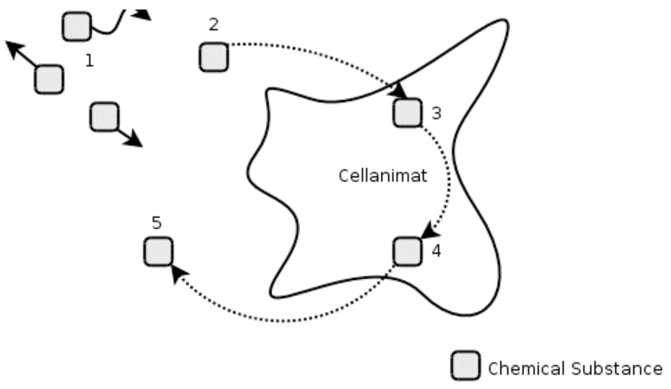


Figure 3-2 Substance movement cycle

Figure 3-2 shows the movement cycle of substance agents: 1) substance agents randomly or forced move in the environment; 2) they can be eaten by the Cellanimat when they are near the cellular membrane; 3) after eaten by the Cellanimat, they start intracellular movements to form a substance streaming within the Cellanimat; 4) they can be vomited by the Cellanimat; 5) the vomited substance agent again starts to randomly or forced move around the environment.

1.1.14. Movement

As stated above, substance agents perform three kinds of movements:

Brownian movement, randomly move into an empty neighbour grid cell.

Forced movement, move into an empty neighbour grid cell with a direction which is decided by covariance between substance agents

Movement within Cellanimat, if two substance agents with negative covariance are adjacent inside the Cellanimat, they force each other to move away if only they are still inside the Cellanimat. Substance agents that are on the membrane can also push negatively covariant substance agents which are inside the Cellanimat away.

Covariance is used to regulate substance movement so that: 1) environmental signals in a small area may be magnified by covariant signals; 2) if the Cellanimat forms growing and shrinking ends, i.e. moves towards one direction, substance agents with negatively covariant stocks will not exist in the same end.

1.10 Cellanimat Design

In this section, I will first give an overview of how the Cellanimat is modeled, and then shift focus on detailed designing issues about actin and actin binding proteins (ABP). And the section will be finished by a discussion about one of the most critical agents: receptor.

1.1.15. Cellanimat Overview

As mentioned in section 2.2, many parts of the internal structure of a biological cytoskeleton still remain uncertain. It is impossible to give a rather complete model of cytoskeleton at present. And therefore carefully choosing proteins to model becomes the first task of the Cellanimat design.

1.1.15.1.Selected Proteins

Bentley and Clack have selected Actin, Arp2/3, profilin and some membrane proteins such as PIP2, WASP and receptor in their artificial cytoskeleton model [2]. I am going to reuse part of their selection for following reasons: 1) as mentioned in section 2.2, actin and actin filament are most common structural proteins in biological cells; 2) Arp2/3 proteins play an important role in regulating assembly and disassembly of actin networks; 3) membrane proteins and profilin together forms a transduction pathway[21] which relays external signals to internal protein reactions. Beside the above proteins, cofilin is also modeled in the system. As explained in section 2.2.2, cofilin is a kind of protein which is essential for the turnover rate of actin filaments, i.e. increases the chance of breaking down filaments. A summary of modeled proteins is given in Table 3-2

Protein	***Description***
Profilin	Accelerate actin filament growth, modeled within CytoplasmicProtein
Cofilin	Accelerate actin filament shrinkage, modeled within CytoplasmicProtein
Receptor	Transform environmental signals into intracellular reactions, modeled as ReceptorAgent
Actin	Structural protein of cytoskeleton, modeled as ActinAgent
Arp2/3 complex	Branches actin filament, modeled as Arp23Agent

Table 3-2 Summary of modeled proteins in the system

As shown in Table 3-2, PIP2 and WASP are not modeled. Instead, receptors are modeled as a combination of receptor, PIP2 and WASP.

1.1.15.2.Collaboration of proteins

In this section, I will describe a scenario of how Cellanimat proteins collaborate with each other and consequently modify the Cellanimat's shape and position. Imagine at a certain time step, somewhere on the Cellanimat membrane, a receptor receives some signals from its neighbour environment. The receptor updates its state to either *active* or *inactive* and diffuses / absorbs small amounts of profilin and cofilin. Proteins (actin and Arp2/3) inside the Cellanimat will assemble or disassemble the actin network according to neighbour profilin, cofilin and receptor's state. At the same time, receptor agents decide to grow or shrink based on the fact of whether there is any actin filament pushing at them and thereby changes the Cellanimat's shape. Locomotion happens if the Cellanimat keeps growing towards one direction while shrinking in the other.

1.1.16. Cytoplasmic Protein

Profilin and Cofilin are small and ubiquitous proteins in cytoplasm. Their distribution can be effectively modeled by diffusion. The detailed diffusion algorithm will be given in Chapter 4

[21] Transduction pathway can be regarded as a signaling channel that transforms external signals.

1.1.16.1.Cofilin

Cofilin is essential to recycle monomer actins[22] by severing actin filament. In the system, when an actin agent is in a filament, at each time step, it sums up all cofilin in its 3×3 neighbourhood. If the sum is over a threshold that is previously set by a user, the actin agent changes its state into sequestered actin (SA, see Figure 3-3).

Other than being regulators in the monomer actin recycle, cofilin also takes part in transporting actin towards the cell nucleus in times of cell stress [7]. This is an important phenomenon as it can be used to model cell stress which may be further mapped into an urgent sell[23]. Therefore, actin agents in the system monitor the cofilin difference in their neighbourhood and act accordingly if the difference is over an initially configured threshold. For example, if an actin agent detects that cofilin in its northern adjacent grid cell is much higher than in southern grid cell, the actin agent will consider move towards south even if it is in a filament.

1.1.16.2.Balance of Cytoplasmic Protein

The Cellanimat is initialized with no profilin and cofilin. As it ages, receptor agents diffuse profilin and cofilin into the Cellanimat according to environmental signals they detect. The amount of profilin and cofilin in the Cellanimat is balanced by the absorbing behaviour of receptor agents. Specifically, the receptor agent diffuses profilin and absorbs cofilin when its state is active, and does the opposite activities when it is inactive.

1.1.17. Filament Cycle

The design of the actin filament cycle is partially based on the model constructed by Bentley and Clack [2]. All modeled proteins participate in this cycle. Particularly, actin agents and Arp2/3 agents maintain state automata to facilitate the cycle procedure. Agent states are summarized in Table 3-3 and an overall picture of filament cycle is given in Figure 3-3.

---

[22] Monomer actins are actins that are not in filament. Sequestered actin is one kind of them.

[23] Without cofilin, if the market drops suddenly, the Cellanimat may be unable to behave (contract) correspondently, as actins will block the cell contraction.

Protein	State	Description
Actin	SA	Sequestered actin
Actin	PA	Actin activated by profilin
Actin	WA	Actin activated by WASP
Actin	FA	Actin in a filament
Arp2/3	NAP	Normal Arp2/3
Arp2/3	WAP	Activated by WASP
Arp2/3	FAP	Arp2/3 in a filament

Table 3-3 Protein States

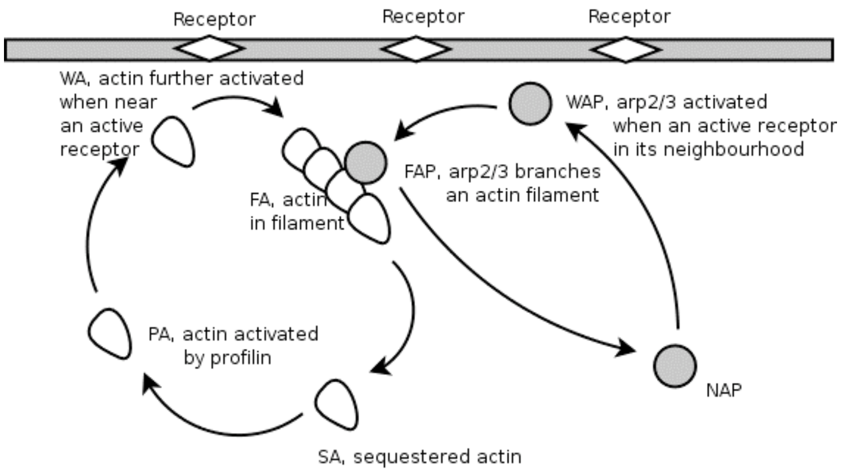

Receptor
Receptor
Receptor
WA, actin further activated when near an active receptor
WAP, arp2/3 activated when an active receptor in its neighbourhood
FAP, arp2/3 branches an actin filament
FA, actin in filament
PA, actin activated by profilin
NAP
SA, sequestered actin

Figure 3-3 Filament Cycle

1.1.17.1.The Hydrolysis of ATP to ADP

The hydrolysis of ATP to ADP is one of key forces that disassemble the actin network. In the system, this process is modeled by a timer. For example, when an actin agent joins a filament, it starts the timer. A few time steps later, the actin agent changes its state back into sequestered actin when the timer is over a preconfigured time limit. The timer does not produce the severing effect as agents (timer starts earlier) at point end always dissociate before those (timer starts later) at barbed end.

1.1.17.2.Profilin and Cofilin Streaming

Actin agents also diffuse and absorb profilin and cofilin in the filament cycle. Particularly, actin agents diffuse cofilin and absorb profilin when they change their states from SA to PA (see Figure 3-3), and absorb cofilin and diffuse profilin when they update from WA to FA (see Figure 3-3). The result of such behaviours is that profilin streams towards the barbed end of the filament while cofilin streams at the opposite direction.

1.1.18. Receptor

Being a major component of the transduction pathway, receptor agents receive various technical indicators from the environment and initiate proteins' reactions. Moreover, receptor agents' movements can be directly mapped into trading behaviours. Therefore receptor agents provide an ideal place to formulate different investing strategies.

1.1.18.1.Overview of Receptor

Each receptor agent can be regarded as a single system. It receives inputs from environment and produces outputs to affect cellular proteins' interaction (see Figure 3-4 [FIG]). The actual work of computing the inputs is delegated to a Strategy object which will be discussed in next section.

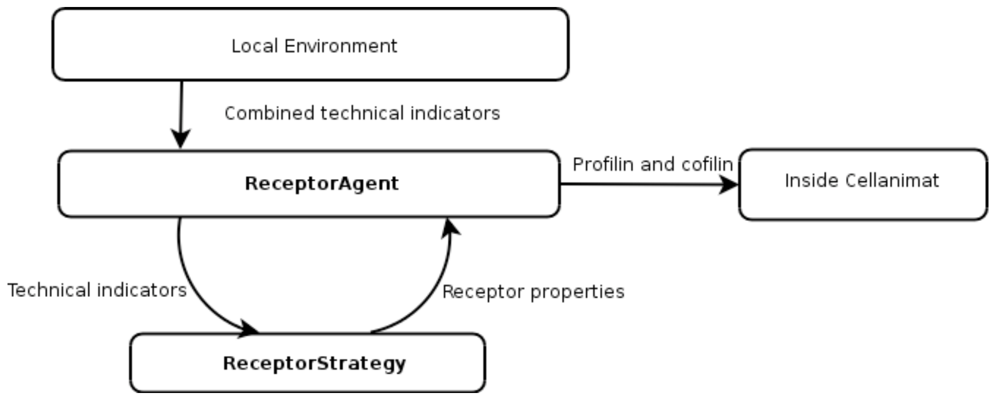


Figure 3-4 Overview of receptor agent

1.1.18.2.Movement

In the artificial cytoskeleton created by Bentley and Clack, one Push-out[24] procedure is always accompanied with a Pull-in procedure at the far end. The reason is that their Cellanimat has to keep volume constant. However in this project, the Cellanimat's volume changes all the time (due to trading). Therefore, its growth (Push-out) and shrinkage (Pull-in) is decoupled in the design. In brief, the receptor agent protrudes when it is pushed by a growing filament, and it shrinks when it is lack of actin support in the neighbourhood. Beside normal protrusive and contractile movements, receptors can also randomly flow along the membrane or forced to move on membrane by another receptor agent. Table

[24] Push-out is a procedure that growing filament causes Cellanimat growth. It looks like filament pushes Cellanimat membrane outwards. Pull-in is the opposite procedure but caused by lack of filament support.

3-3 summarizes all possible receptor movements. Within each time step, a receptor agent can only perform one of four described types of movement.

Movement	Description
Protrusive movement	Caused by growing filament inside Cellanimat. Receptor agent will move into an adjacent non-Cellanimat grid cell. Such movement will trigger a buying procedure.
Shrinking movement	Caused by lack of filament support inside Cellanimat. Receptor agent will move into an adajacent grid cell which containds CytoplasmAgent and replace it. Such movement will trigger a selling procedure.
Random movement	Receptor randomly swap position with its neighber MembraneAgent. No buying or selling procedure is triggered.
Forced movement	Receptor tries to push its neighbour agent away if they contains different types of strategies. If its neighbour agent is blocked, receptor agent itself will try to move away. No buying or selling procedure is triggered.

Table 3-3 Summary of receptor movements

1.11 Strategy Design

Ability to formulate different investing strategies is a key objective of this project. In the design, *Strategy* pattern together with *Property* pattern are employed to achieve a high extensibility of formulating the investment strategy.

1.1.19. Strategy Overview

I have designed a ReceptorStrategy class to encapsulate the real algorithm that conducts the major part of transformation from external signals to internal signal.

Each ReceptorStrategy object, in fact, maintains a multi-layer neural network. The input of such neural network is combined chemical signals obtained from neighbour substance agents. The output is a set of attribute values wrapped in ReceptorProperty object which will be used to configure the receptor agent later. Once the configuration is finished, the receptor agent can diffuse or absorb proteins and make a movement according to its attributes.

Other than conduct calculation for receptor agents, ReceptorStrategy objects also implement a method to return a signature of the strategy which varies from -1 to +1. This signature is used as regulator of receptor agents' forced movement.

1.1.20. Strategy Extensibility

As mentioned in Chapter 2, there are tremendous amount of indicators being used in real market. Moreover, most indicators are preferred by some traders but distrusted by others. Such nature of indicators requires strategy class highly adaptable. In the system, a container class is employed to store technical indicators. And so addition of new indicator and removal of unnecessary indicators can be accomplished without modifying source code

Implementation

In this chapter, I will first introduce software that are used in this project in section 4.1, and then quickly walk through general implementation issues in section 4.2. In the remaining sections, I will discuss about detailed issues about implementation.

1.12 Development Environment

1.1.20.1.MinGW 4.0 and GDB 5.2.1

The main part of the system is written in C++ language, compiled by MinGW 4.0 and debugged with GDB 5.2.1. Choosing C++ as the main developing language is because: 1) the project is initiated with Bentley and Clack's implementation of artificial cytoskeleton which is written in C; 2) time complexity is critical as the system contains thousands of agents; 3) the design of system is object oriented.

1.1.20.2.Java 2 Standard Edition JDK 5.0

In addition to C++, Java is also used in some part of the project. Particularly, Java is used for database initialization and technical indicator normalization because of its powerful string library and convenient JDBC library.

1.1.20.3.Reuters 3000 Xtra

This is a client program used to access Reuters real market data. It provides price and volume historical data of most companies in the market. Moreover, it supports different kinds of technical analysis, such as moving average, open interest and momentum.

1.1.20.4.MySQL 4.0

MySQL 4.0 is used to maintain price and volume history data. It also stores technical indicators and records trading history during Cellanimat's life time.

1.1.20.5.Eclipse 3.1 and CDT 3.0

Eclipse and its plug-ins are used as IDE in this project as it is free and powerful.

1.13 General Implementation Issues

1.1.21. Class Structure

The Cellanimat context keeps a 2D array of *GridCell* objects and a reference to *CellContextAgents* object which encapsulates references to all agents within the context. *GridCell* object's dynamic behaviour is achieved by delegating its behaviour to the *CellAgent* object it contains a reference to. Figure 4-1 gives the class hierarchy of Cellanimat.

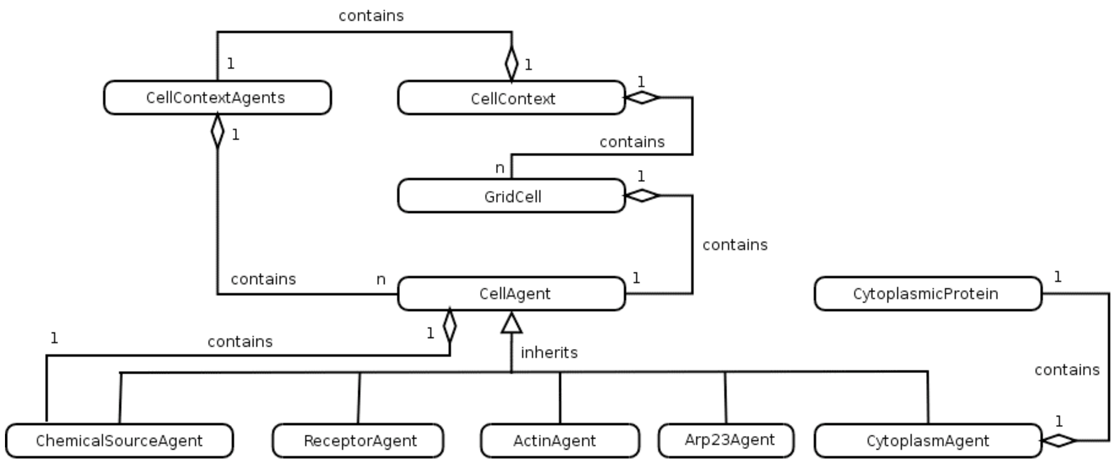


Figure 4-1 Class hierarchy of Cellanimat

Agents also keep a reference to the grid cell that they are living in. This reference is essential to agents because it provides them the ability of accessing neighbour grid cells.

1.1.22. Initialization

The Cellanimat initially lives at the centre of the context. Proteins inside Cellanimat have been initialized based on their possibilities of existence defined in context configuration. One portion of source agents are randomly spread in the environment, others are initialized to fill in the Cellanimat.

Each grid cell holds a list (called *neighbours*) of references to its neighbour grid cells. The references to those neighbour grid cells are carefully placed in the list (see Figure 4-2 [FIG]) in a way that relative location of its neighbour can be calculated by simple arithmetic equations. For example, if we know one neighbour grid cell is *neighbours*[*i*] (*i* is the index of reference in *neighbours* list), the opposite grid cell must be *neighbours*[*(i + 4) % 8*]. With such encoding, process of locating neighbourhood and direction is greatly simplified.

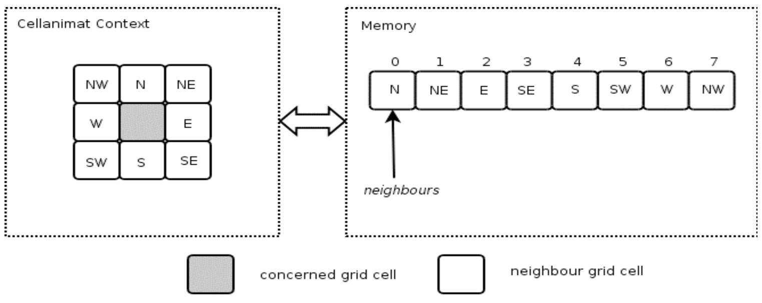


Figure 4-2 neighbour grid cell encoding[25]

This structure of CellGrid object has some spatial overheads. But by doing so, agents' vision of overall context has been blocked and the latter implementation of agents' rules is limited within a local neighbourhood and is therefore simplified.

1.1.23. Update

In each time step, the Cellanimat context updates its contents in two ways: synchronous and asynchronous. Synchronous update imitates events that happen at same time around the context, e.g. diffusion. Asynchronous update is used for updating agents' states, including agent to agent interaction that will be computationally expensive to do synchronously, e.g. agents' movement. Table 4-1 gives a summary of synchronous and asynchronous update.

Type of Update	*Involved Agents*	*Description*
Synchronous update	CytoplasmAgent	CytoplasmAgent needs synchronous update to diffuse cytoplasmic proteins into its neighbour agents
Asynchronous update	ChemicalSubstanceAgent CytoplasmAgent ReceptorAgent ActinAgent Arp23Agent	Those agents need asynchronous update perform agent to agent interaction

Table 4-1 Summary of updates

To perform asynchronous update, the CellContext object must maintain a container of agents in the context which stores agents' references in a random order. So during asynchronous update, agents in that container will be iterated and updated respectively. However, some agents need to access to the container as they have to insert new agents or remove olds, e.g. receptor movement. In order to keep encapsulation of CellContext class (to localize the vision of an

[25] Codes in Figure 4-2 [FIG] are shorts for orientations. For example, NE stands for north east.

individual agent) and provide accessibility of container to individual agents, I use a separate class, named as CellContextAgents, to act as agents container. As agents can be removed or inserted into this container, its data structure must be flexible so that insertions and removals could happen during its iteration. Because of this flexibility requirement, in the implementation, a double link list is maintained in the CellContextAgents class as the container.

1.14 Database Implementation Issues

The data source of the system used to be implemented as a normal text file. However, the system is designed to be extensible, e.g. it should have an ability of adding new technical indicators. Since it is expensive to modify the internal data structure of data source, the data source of the system has been shifted into a MySQL database.

1.1.23.1.Database organization

All stocks information, technical indicators and trade record are modeled as relations in the database. And so does strategy settings. A complete E-R diagram is given in Figure 4-3

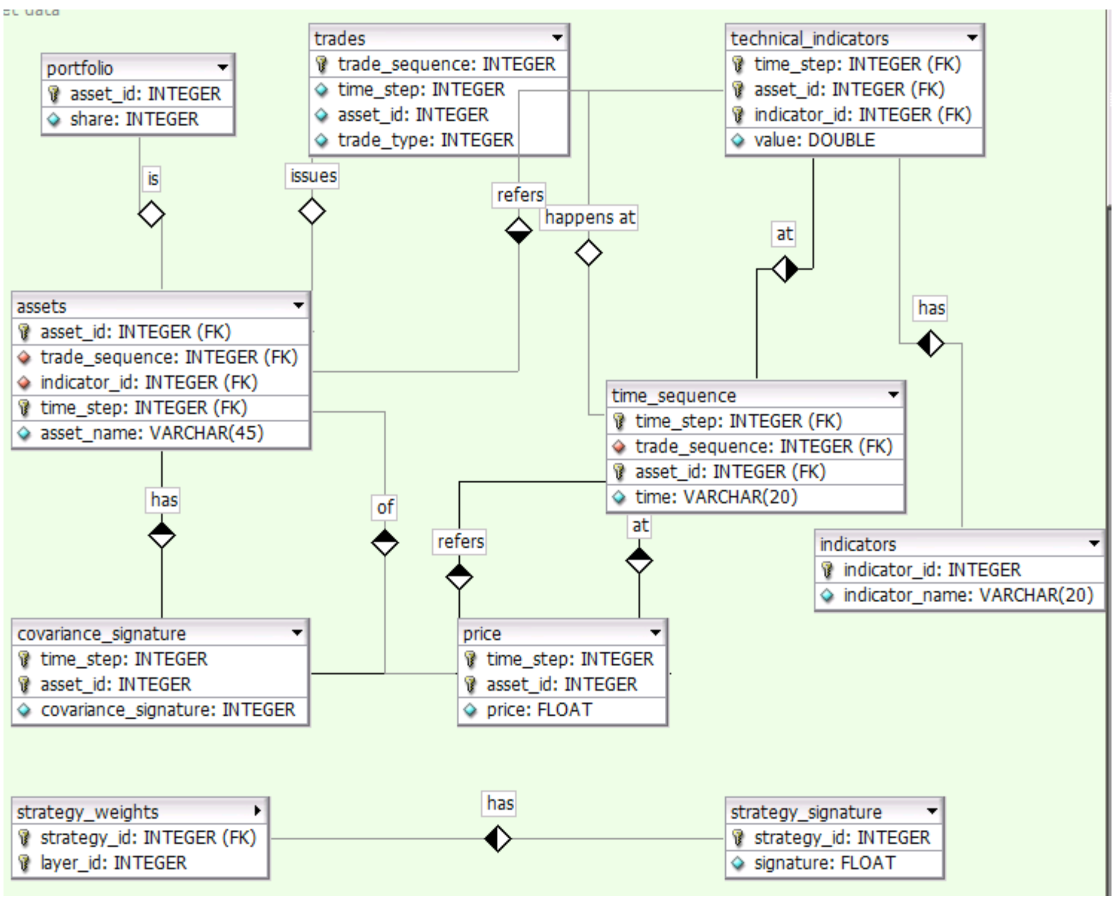


Figure 4-3 ER Diagram of the database

1.1.23.2.Indicator Preparation

In the system, receptor agents need to combine different environment signals in its neighbourhood. Therefore a normalization of some technical indicators is required to remove the inherited price difference between stocks before the system starts.

1.15 Environment Implementation Issues

Substance agents move around the environment. When they move into the membrane of the Cellanimat or they are eaten by the Cellanimat, they will be removed from *CellContextAgents* container and therefore stop updating in each time step until they move back into the environment again. To implement the process of substance agents moving into other agents, each agent maintains a reference to the substance agent it contains.

1.1.24. Intracellular Movement

Substance streaming is implemented inside each cellular agents. They behave in the same way as following: for each neighbour agent, the concerned agent will conduct a comparison of covariance between substances itself and its neighbour agent contain. If the result is negative covariance, the concerned agent will force its neighbour agent to swap its substance with its neighbour agent.

1.1.25. Covariance Algorithm

As covariance is used to regulate substance agent's movement both inside and outside of Cellanimat, its performance and quality are keys to high performance of Cellanimat. Theoretically, calculating precise covariance requires too much time and memory space. So I introduce a less precise yet rather faster algorithm to estimate covariance of two stocks. The algorithm requires maintaining a 32-day price moving average and price history of last 32 days. As the first step, the algorithm will ask FinanceDataManager class for an unsigned integer variable as covariance signature of concerned asset. The signature is coded like this: every bit in the signature corresponds to the relation of every day's price to the moving average. If a day price is higher than the average, the corresponding bit of that day is set to 1; otherwise it is set to 0. Figure 4-4 shows two example signatures.

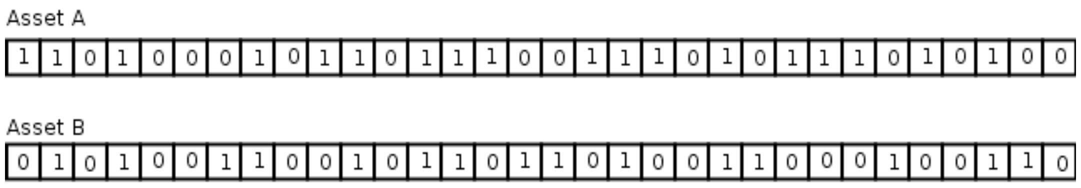


Figure 4-4 Covariance signature examples

As shown in Figure 4-7, asset A moves to different direction from B in the first day. But after that, A and B move together for five days until they diverge again. When the algorithm receives two signatures of compared stocks, it starts calculating the covariance: 1) initialize a float as covariance with zero; 2) compare each pair of bits in signatures, if they are same, add +0.03125 (1/32) to covariance, else add -0.03125(-1/32); 3) return the final result of covariance. For example, covariance of A and B in Figure 4-4 is 0.0625.

1.16 Cellanimat Implementation Issues

1.1.26. Cytoplasmic Proteins

As discussed in Chapter 3, cytoplasmic proteins are small and therefore implemented by wrapping in CytoplasmicProtein class as attributes. In such case, profilin and cofilin are coded as two float type attributes. To facilitate both synchronous (e.g. protein diffusion), CytoplasmicProtein class maintains buffers for each type of small protein.

1.1.26.1.Synchronous update

Although both profilin and cofilin are diffused inside Cellanimat, diffusions of them are differed in details. Profilin is diffused by using the Glazier & Graner method, same as Bentley and Clack used in [4] which is shown in Figure 4-5 below.

```
If cytoplasm.profilin > profilin.threshold
    neighbour-cytoplasm.profilin += p / n
    cytoplasm.profilin = profilin.threshold

p: cytoplasm.profilin(old) - profilin.threshold
n: number of CytoplasmAgent in its neighbourhood
```

Figure 4-5 Profilin diffusion

Cofilin is using another method to model diffusion as shown in Figure 4-6 below.

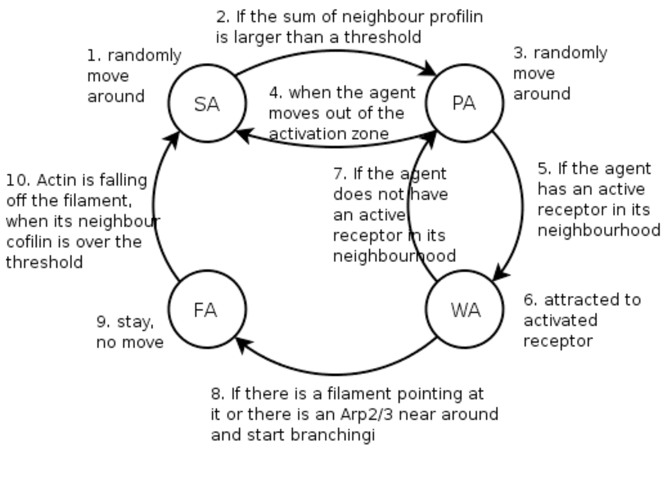
```
neighbour-cytoplasm.cofilin += p / n
cytoplasm.cofilin = p / n

p: cytoplasm.cofilin(old)
n: number of CytoplasmAgent in its neighbourhood
```

Figure 4-6 Cofilin diffusion

The consequences of those two kinds of diffusion are slightly different. If the profilin and cofilin streaming is not considered, profilin is always higher in an area that is near to membrane to keep an activation zone for actin and Arp2/3, while cofilin tends to be equally diffused throughout the Cellanimat.

1.1.27. Actin

Actin protein is modeled as ActinAgent class which has attributes of status, barbed end direction and a timer. The direction is used to make sure that filament always grow linearly[26]. The pointed end direction can be easily computed by barbed end direction. Based on Bentley and Clack's implementation in [4], in each time step, actin agent will apply a set of rules in the sequence of 1) rules of substance streaming; 2) rules of status update; 3) rules for affecting its neighbour agents and 4) rules of movement. Except first group of rules, rules are chosen to apply based on status of actin. Figure 4-7 shows a state transition diagram with detailed explanation and Table 4-3 and following explanation give a summary description of rules that applied by ActinAgent.



2. If the sum of neighbour profilin is larger than a threshold
1. randomly move around
3. randomly move around
SA
4. when the agent moves out of the activation zone
PA
10. Actin is falling off the filament, when its neighbour cofilin is over the threshold
7. If the agent does not have an active receptor in its neighbourhood
5. If the agent has an active receptor in its neighbourhood
FA
WA
9. stay, no move
6. attracted to activated receptor
8. If there is a filament pointing at it or there is an Arp2/3 near around and start branchingi

Figure 4-7 State transition of Actin agent

[26] In the real implementation, only one direction is maintained, as pointed and barbed directions are always opposite to each other.

In Figure 4-7, four possible status of ActinAgent are shown: SA, sequestered actin, it is the status when actin is created; PA, profilin activated actin; WA, receptor activated actin; FA, actin as component of a filament. The number in Figure 4-7 clearly shows the life cycle of an ActinAgent object.

Rule	Type	Applied Status
Substance movement	Substance streaming	All status
Status update	Status update	All status
Actin recruit	Affect neighbour	FA
Fall off filament	Affect neighbour	FA
Random movement	Movement	SA, PA
Attracted movement	Movement	WA

Table 4-3 Summary of ActinAgent rules

In Table 4-3, *substance movement* rule is always the first rule applied to the agent. It may arouse its neighbour agents to swap substance with their neighbour agents; *status update* rule will update the agent's internal status based on description in Figure 4-7. It is always the second rule to apply; *actin recruit* rule applies when ActinAgent is in FA status and at the end of its barbed end direction, there is an ActinAgent in its WA status. The recruit procedure will modify target ActinAgent's pointed end direction and status into FA; *random movement* rule will cause agent to randomly move into a neighbour grid cell that contains a CytoplasmAgent; *attracted movement* rule will cause a movement of current agent to a grid cell that is neighbour to an active receptor. The agent can only move into a grid cell that contains CytoplasmAgent.

1.1.28. Arp2/3

Arp2/3 protein is modeled as Arp23Agent class which has attributes of state, branching direction and a timer. Pollard's [11] model is used to implement the Apr2/3 branching. However, in his model, Arp2/3 will start a new filament branch at angle of approximately 70 degree, which is hard to implement in the raster world. So 90 degree is implemented in the system.

Same as ActinAgent, Arp2/3 agent applies several rules in each time step. Again, the implementation of Arp2/3 life cycle is based on Bentley and Clack's model [2]. State transition diagram and detailed explanation are given in Figure 4-8. Table 4-4 provides a summary of Arp2/3 rules.

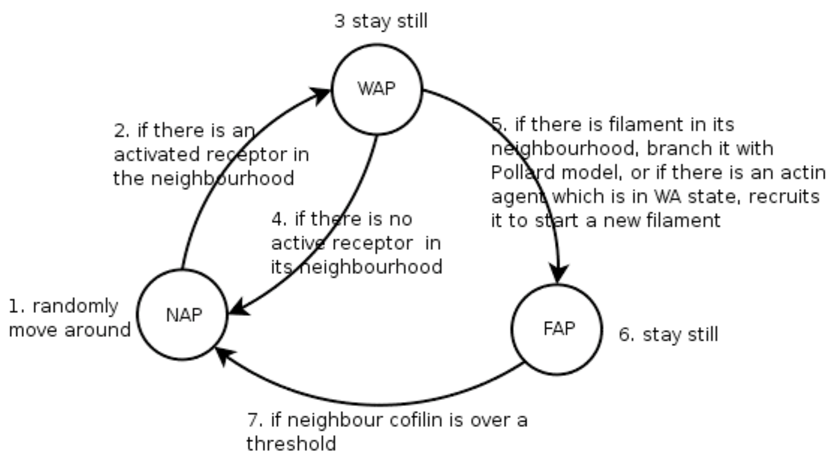

3 stay still
WAP
2. if there is an activated receptor in the neighbourhood
4. if there is no active receptor in its neighbourhood
5. if there is filament in its neighbourhood, branch it with Pollard model, or if there is an actin agent which is in WA state, recruits it to start a new filament
1. randomly move around
NAP
FAP
6. stay still
7. if neighbour cofilin is over a threshold

Figure 4-8 State transition of Arp23Agent agent

Rule	***Type***	***Applied State***
Substance movement	Substance streaming	All states
State update	State update	All states
Initiate filament	Affect neighbour	FAP
Branch filament	Affect neighbour	FAP
Random movement	Movement	NAP
Attracted movement	Movement	WAP

Table 4-4 Summary of Arp23Agent's rules

In Table 4-4, I use the same taxonomy as Table 4-3. *State update* will update its internal state based on transition described in Figure 4-8; *initiate filament* rule will modify its neighbour ActinAgent's state and directions; *branch filament* rule also bind its neighbour ActinAgent, but only at specified direction; *attracted movement* happens when Arp2/3 agent in WAP state, and it will look for filament in a 5x5 neighbourhood. If find one, the agent will try to move towards the filament. *Substance movement* and *random movement* rules are the same as ActinAgent's.

1.1.29. Receptor

Receptors are modeled as ReceptorAgent class which holds a reference to a ReceptorStrategy object and a ReceptorProperty object which encapsulates all receptor's attributes. The receptor always updates its ReceptorProperty object first in each time step, and then protrudes or contracts according to that ReceptorProperty object. ReceptorProperty object's update is delegated to the ReceptorStrategy object. Table 4-5 summarizes all attributes a receptor may have.

Attribute	Type	Description
State	bool	Can be either ACITVE or INACTIVE. When it is ACTIVE, the receptor will trigger a serial of protein reaction which evetually will result in Cellanimat growth
CytoplasmicProtein	CytoplasmicProtein	Amount of profilin and cofilin that will diffuse to / suck in from cytoplasm of the Cellanimat. Profilin and cofilin are coded as float number. Positive number means diffuse and negative number means suck in
Moveing possibility	float	When the movement direction (protrude or contract) of receptor is decided, the receptor will roll a random number between 0 and 1 against this attribute, if it is larger than this attribute, the receptor moves, otherwise it stay still

Table 4-5 Summary of receptor attributes

1.17 Strategy Implementation Issue

1.1.30. Technical Indicators

Technical indicators can be obtained by using Reuters 3000 Xtra. However, data need cleansing before fed into the system because indicators of different stocks will be combined before they are passed to the receptor agent. Stock prices are normalized by dividing their 200-day price moving average. The covariance signature also needs to be calculated and stored in the database. One more thing worth notice is that there are some data missing in some period of time. It has been ignored instead of interpolation as the Cellanimat is not a real time system.

Since technical indicators are stored in the database in a way that the system does not contain any encoded information of which indicators are used, new indicators can be introduced without modifying the source code. The only requirement is to carefully configure the weight matrix of strategy objects so that their inputs match the number of technical indicators..

1.1.31. Strategy Class

A strategy class has two public methods, *applyStrategy* and *getStrategySignature*. The former receives a reference of a ChemicalSignal object and another reference of a ReceptorProperty object. This method is supposed to mimic a calculation over an encoded neural network and store the output of the neural network into ReceptorProperty object which is later used to direct ReceptorAgent's behaviour.

Before the Cellanimat context starts running, a set of strategy objects are created and pooled. When Cellanimat is running, every time a new receptor is created, a randomly chosen strategy object is passed to the constructor of ReceptorAgent. By pooling strategies, the time for creating and deleting strategy objects is saved and a means of controlling strategy is also provided.

1.1.32. Strategy Object

Within each strategy object, a set of weight configurations is maintained. Obviously, different configuration is responsible for different outputs, i.e. receptor's attributes, and therefore accountable for different Cellanimat behaviour given same environment signals. The data structure of weight configurations is carefully designed so that the structure of encoded neural network can be different in each strategy object. Details of data structure can be read in ER diagram in the previous section. Figure 4-9 shows a simple example of strategy configuration.

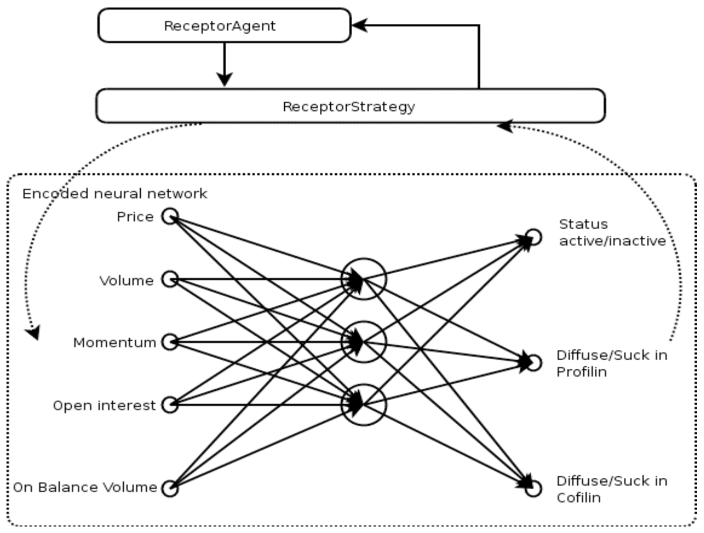


Figure 4-9 A simple configuration of ReceptorStrategy object

Figure 4-9 clearly shows a strategy that only concerns about five technical indicators. Due to space, weights and biases are not given out explicitly in Figure 4-9. Other types of indicators are not in Figure 4-9, but it does not mean they are not passed in ReceptorStrategy object from receptor agent. They are connected in above neural network with zero weight and biases. Again, due to space, they are not shown in Figure 4-9.

1.1.33. Strategy Signature

Other than conduct calculation for receptor, ReceptorStrategy object also implements a method to return a signature of float type which varies from -1 to +1. It is an essential director for receptor agent's forced movement. Currently, the signature is set by the user who configures the weights matrix of the strategy object.

Evaluation

In this chapter, I will start with a discussion about tests of the system. The focus will be the test plan, data and environment settings, and result analysis. Then I am going to give a critical evaluation of the system.

1.18 Testing

1.1.34. Test Plan

This test is used to ensure that the system function well and has ability to achieve project's objectives mentioned in Chapter 1. This test is not a demonstration of formulating a "winning" strategy. And for the simplicity, configurations of the Cellanimat are only within receptor agents' strategies. In order to perform an effective test, I set up following questions to be answered by testing the system:

1. Can the system behave dynamically according to its environment?
2. How does the Cellanimat behave when it contains only one kind of strategy object, specifically, will different strategy configuration cause different behaviours?
3. Is the Cellanimat able to perform normally when two contradict strategy are used by receptors?

The test will be performed on real market data from Rueters. Details of testing data are given in next section. There are two strategy objects are configured. One is for momentum strategy, and the other is for contrarian strategy. The system will run three times. In the first run, receptors can only use momentum strategy object. In the second run, only contrarian strategy is available to receptors. In the third run, both strategies are used. To answer question 1, screenshots of the system are taken. To answer question 2, the Cellanimat's volume is monitored in each time step and compares between different Cellanimats. To answer question 3, observe the third run to see if there is any abnormality[27] emerges.

1.1.35. Test Data

The data used to test the system is chosen from real market, particularly, they are stock prices and indicators from Rueters. Price figures of testing data are

[27] Abnormality means the Cellanimat shrinks too much or grows too big at some time.

available in Appendix D.

Parameters of Cellanimat context is configured by me and given in Appendix C as column of “value”. Table 5-2 gives settings of strategy objects used in tests. It is a single layer neural network as shown in Table 5-2.

	State	***Profilin***	***Cofilin***	***MovingPossibility***
10-day MA	1.6	0.06	0.06	-3
20-day MA	-0.7	-0.02	-0.02	1
50-day MA	-0.55	-0.02	-0.02	1
10-day MA	-0.25	-0.02	-0.02	1
RSI	0	0	0	0
ROC	0	0	0	0.6
Momentum	0	0	0	0.5

Table 5-1 The first strategy configuration

	State	***Profilin***	***Cofilin***	***MovingPossibility***
10-day MA	-1.6	-0.06	-0.06	3
20-day MA	0.7	0.02	0.02	-1
50-day MA	0.55	0.02	0.02	-1
10-day MA	0.25	0.02	0.02	-1
RSI	0	0	0	0
ROC	0	0	0	-0.6
Momentum	0	0	0	-0.5

Table 5-2 The second strategy configuration[28]

1.1.36. Testing Result

The system runs three times and 1000 time steps for each time. The result is shown as below:

[28] These two configurations are deliberately set to opposite.

1.1.36.1.Screenshots

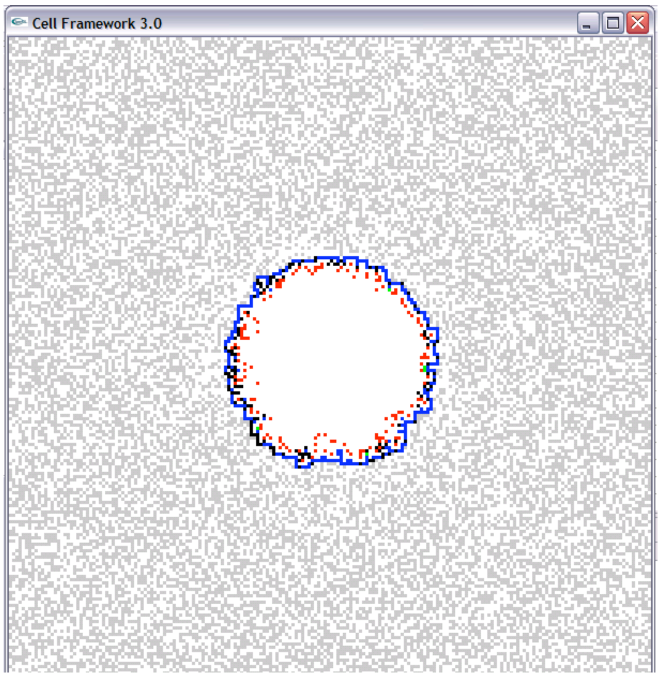


Figure 5-1 the Cellanimat at time step 50

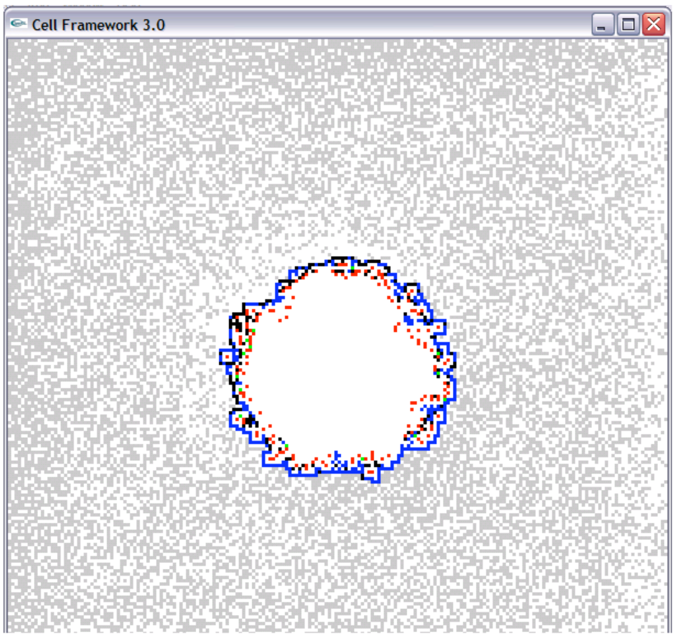


Figure 5-2 the Cellanimat at time step 100

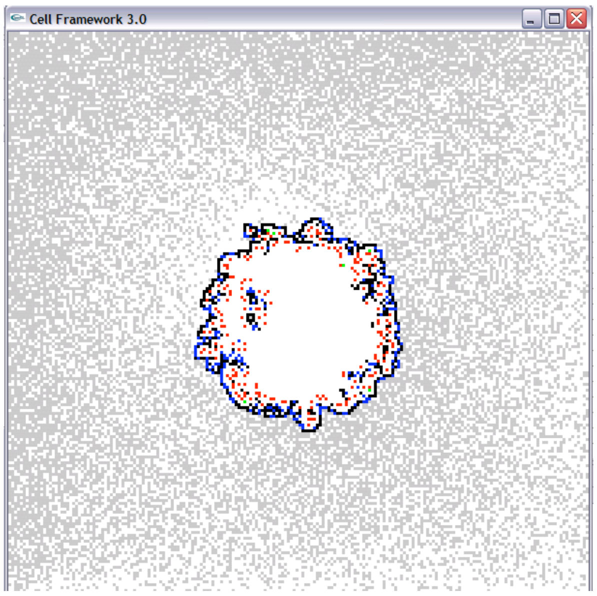


Figure 5-3 the Cellanimat at time step 200

Figure 5-1, Figure 5-2 and Figure 5-3 show that the Cellanimat dynamically change its shape. More importantly, in those figures, there are many tiny dots inside the Cellanimat and near its membrane. They are proteins that form an actin network. And this shows the actin dynamic within the Cellanimat.

1.1.36.2. Volume of the Cellanimat

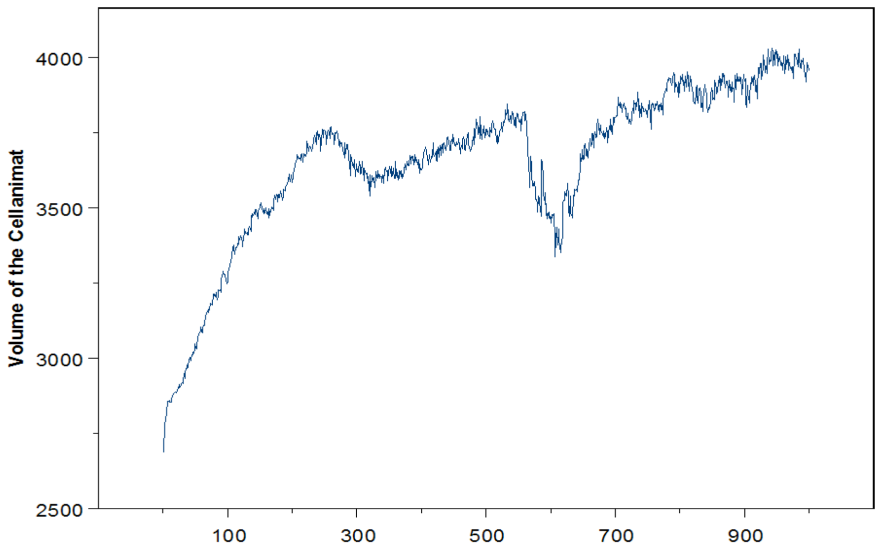


Figure 5-4 Volume history of the first run in the test

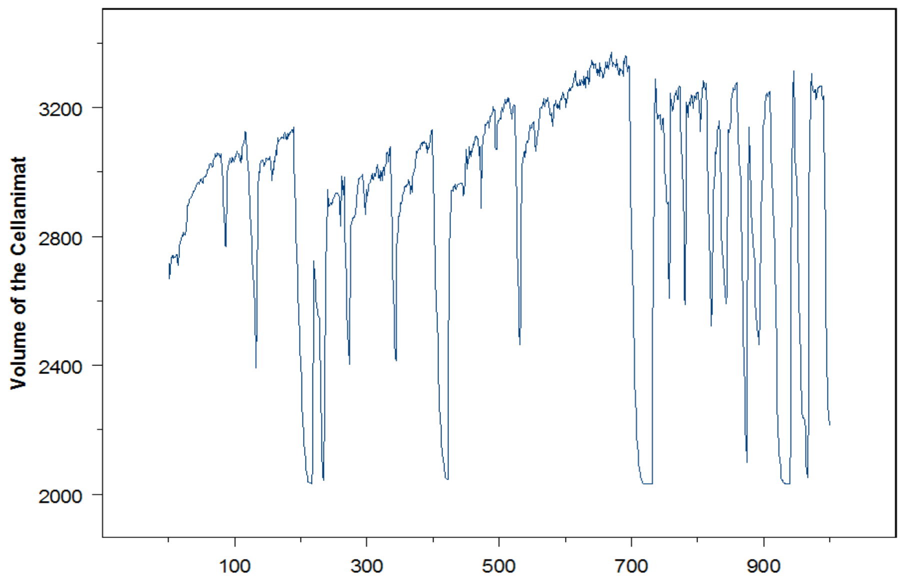


Figure 5-5 Volume history of the second run[29]

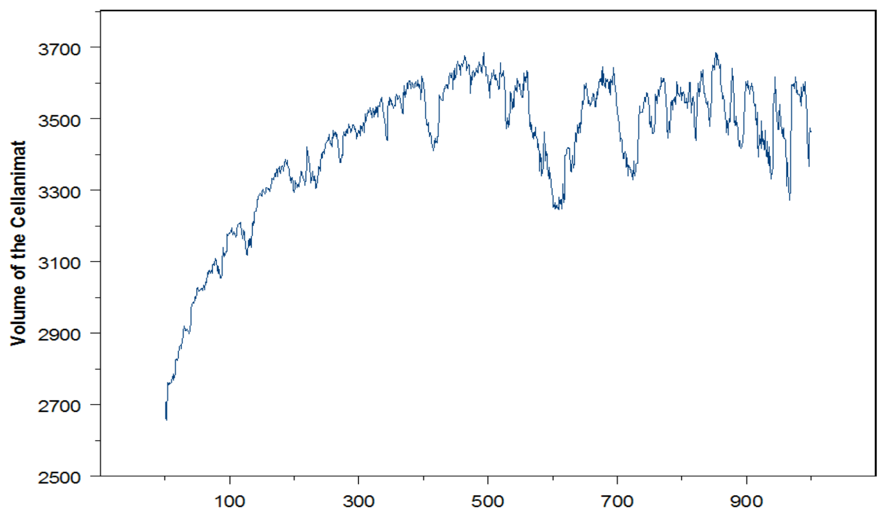


Figure 5-6 Volume history of the third run

The first two figures show that different configurations in receptor agents affect the overall behaviour of the Cellanimat. This gives a positive answer to the second question raised in section 5.1.1. The third figures above shows that different strategy configurations can coexist in the Cellanimat.

[29] The reason why the volume of the second Cellanimat oscillates during the test is unknown.

1.19 Critical Evaluation

Although the system works well as shown in tests, there are some places need improvements if I have a chance to redo the project.

1.1.37. Covariance calculation

Covariance calculation regulates two kinds of movements in the system; therefore it is worth of extra working to improve its performance. The algorithm given in Chapter 4 can only get an approximate covariance of prices. However, as Reilly and Brown suggested in [9], in practice, it is the covariance of rate of return receives more concerns. Moreover as shown in Chapter 4, I concern too much about time consuming problem of covariance calculation. With a second thought, if the system is only used for research, covariance does not have to be calculated at run time. On the contrary, it can be prepared before the start of the system, just as I did with technical indicators.

1.1.38. Strategy Signature

Strategy signature is a variable I created to compare between strategies. Compare to covariance, it does not to be very precise, as there are a lot of receptors on membrane, and there is not very much space for them to perform forced movements. However, the problem of strategy signature is that it is hard to assign a number (in the case of first way) or calculation is coupling with technical indicators which should be highly extensible. So if I can restart the project, I will spend more time on re-design the strategy signature so that it can be calculated by experience.

Conclusion

In this chapter, I will give a brief summary of the project and then review project's aims and objectives.

1.20 Summary

In this project, I have built up a system in which a Cellanimat survives and uses its reactions of financial information to manage an equity portfolio. Most of cytoskeleton part of the project is based on Bentley and Clack's previous work on artificial cytoskeleton [1]. The system contains multiple agents and employs cellular automata techniques to model the Cellanimat context.

The system is mainly implemented in C++. Java had also been used in some places where C++ is inconvenient. Database techniques are employed to support system at run time.

System parameters are configured by users before the system starts to run. On one hand, the system will transform real market data into Cellanimat's internal protein reactions at run time. On the other hand, the Cellanimat's behaviour is mapped back into trading behaviour in financial domain.

1.21 Review of aims and objectives

Tests in chapter 5 show that the main aim of this object has been achieved with the constructed Cellanimat. The Cellanimat can dynamically react to financial signals and therefore perform trading activities. The investment strategies have been implemented as a configuration of the Cellanimat context and its receptor agents. The third test described in chapter 5 shows that contradictory investment strategies can coexist within the Cellanimat. To sum up, the Cellanimat provides a platform for a user to formulate investment strategies.

Further Work

The system provides a very interesting platform for modeling investing decision. Yet many places of the system are worth further consideration. In this chapter I will outline some interesting questions that we can work at in the future.

1. Experiment on learning ability of the system. It is obvious that the system can be evolved by genetic algorithm. Some interesting questions related to GA design are: 1) How to design the chromosome representation when there are so many parameters waiting for configuration? 2) How to combine individual learning of receptors and overall learning of Cellanimat? 3) Is it possible the system could learn an optimized configuration in different types of market?

2. Extend the system with different investment instruments, such as bonds, derivatives and foreign exchange. All those instruments have their unique features which may affect the parameters setting of the system. Combination of instruments may result in a much more complex configuration of the system.

3. Extend the system to model loop 2 or even loop 1 (see section 2.1.1). Without question, risk control functionality has to be implemented in this extension. Some related questions would be: 1) How to map large number of potential instruments into the environment? 2) How to model re-allocation with cellular behaviour?

Appendix A – System Manual

1.1.38.1.Software Requirement

In order to run the system, following software must be installed correctly:

- MingW 4.0 compiler
- Any kinds of database with ODBC connection
- Glut32 library

1.1.38.2.System requirement

- Intel Pentium M processor 1300MHz
- 768MB of memory
- Windows XP system

1.1.38.3.File Structure on CD

- Report.doc, the report of this project
- CellContext.rar, the source code of the system
- Create.sql, sql statements that are used to create system required tables

1.1.38.4.Database Construction

1. Run the create.sql script in the CD.
2. Fill in time sequence data into table "time_sequence".
3. Fill in assets' information and indicators' information into "assets" and "indicators" tables.
4. Flush in price data and indicators according to the date stored in the "time_sequence" table.
5. Configure the strategy signature and strategy weights.

Appendix B – User Manual

1.1.38.5.Preparation

Before the system starts, you need to make sure:

1. Glut32 library has been correctly installed
2. ODBC has been configured and the connection has been successfully tested.
3. Price and technical indicators data has been loaded into the database.

1.1.38.6.Running

1. Unzip the CellContext.rar
2. Go to Debug directory and double click the CellContext.exe
3. The system starts with a pause state. You can choose to press one of following keys:
    a) *U, it can un-pause the system if it is in a pause state.*
    b) *P, it can pause the system if it is running.*
    c) *N, it can run the system for one time step. Using this key, you can run the system step by step.*
    d) *Q, quit the system.*

1.1.38.7.Output

There are three kinds of outputs:

1. Animation of the Cellanimat evolvement in the main window.
2. A log file named as cellanimat_log.xml. It records much useful information about the Cellanimat in each time step.
3. A log file named as portfolio_log.xml. It records summaries of the portfolio such as cash, value of the portfolio and the number of shares that are hold.

1.1.38.8.Useful tools

Appendix C – Context Configuration

Parameter	Type	Value
InitialRadius	int	30
ActinProbability	float	0.5
Arp23Probability	float	0.2
CytoplasmProbability	float	0.3
ReceptorProbability	float	1
SubstanceProbability	float	0.5
ActinProfilinThreshold	float	0.01
ActinCofilinThreshold	float	0
Arp23CofilinThreshold	float	0
ActinPATimeLimit	int	10
ActinWATimeLimit	int	20
ActinFATimeLimit	int	30
ArpWAPTimeLimit	int	10
ArpFAPTimeHighLimit	int	30
ArpFAPTimeLowLimit	int	20
CovarianceThreshold	float	-0.25
CytoplasmProfilinLimit	float	0.06
CytoplasmCofilinDiffuseRate	float	0.3
ActinCofilinDifferenceLimit	float	0.5
ActinSAToPASuckInProfilin	float	0.01
ActinSAToPADiffuseCofilin	float	0.01
ActinWAToFADiffuseProfilin	float	0.01
ActinWAToFASuckInCofilin	float	0.01
ReeptorStrategyDifferenceThreshold	float	1

Context configuration of the Cellanimat and their default value

Appendix D – Testing Data

Following figures shows stock prices that are used for testing the system.

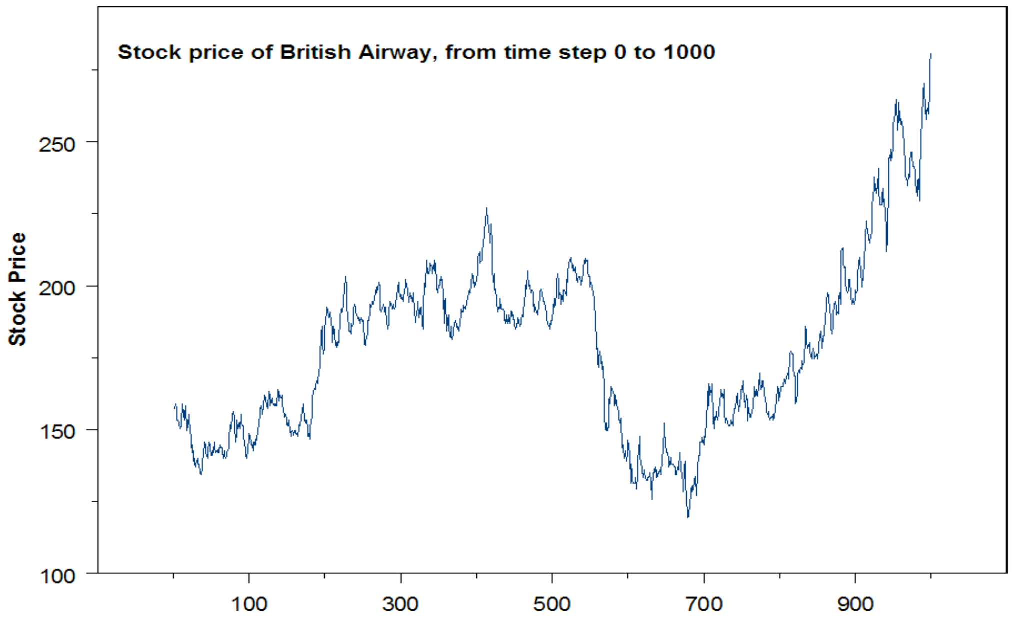
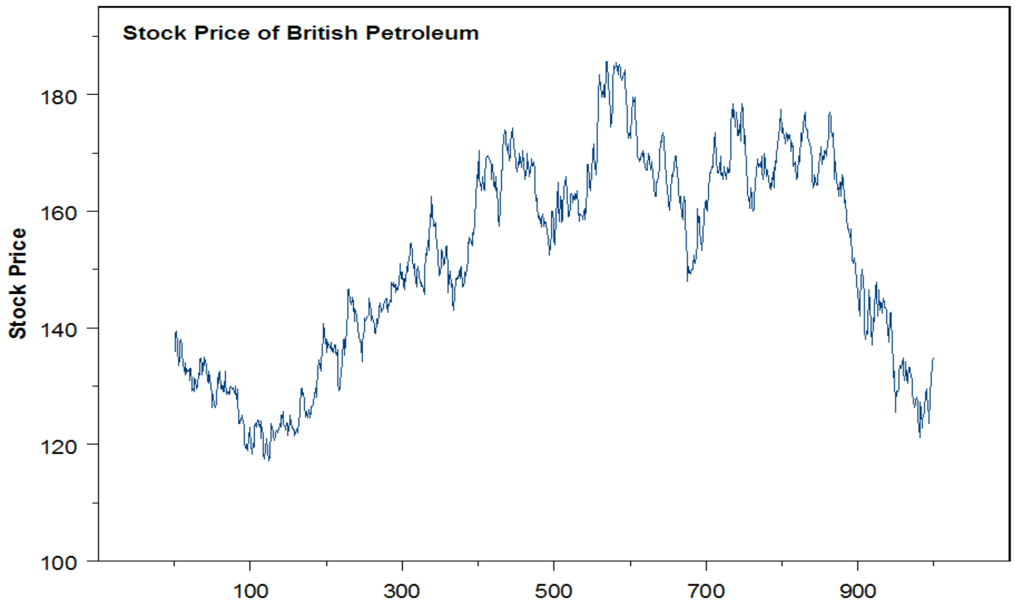


Figure D-1 Stock price of British Airway[30]



[30] This figure and following two are showing stock prices within 1000 time steps.

Figure D-2 Stock price of British Petroleum

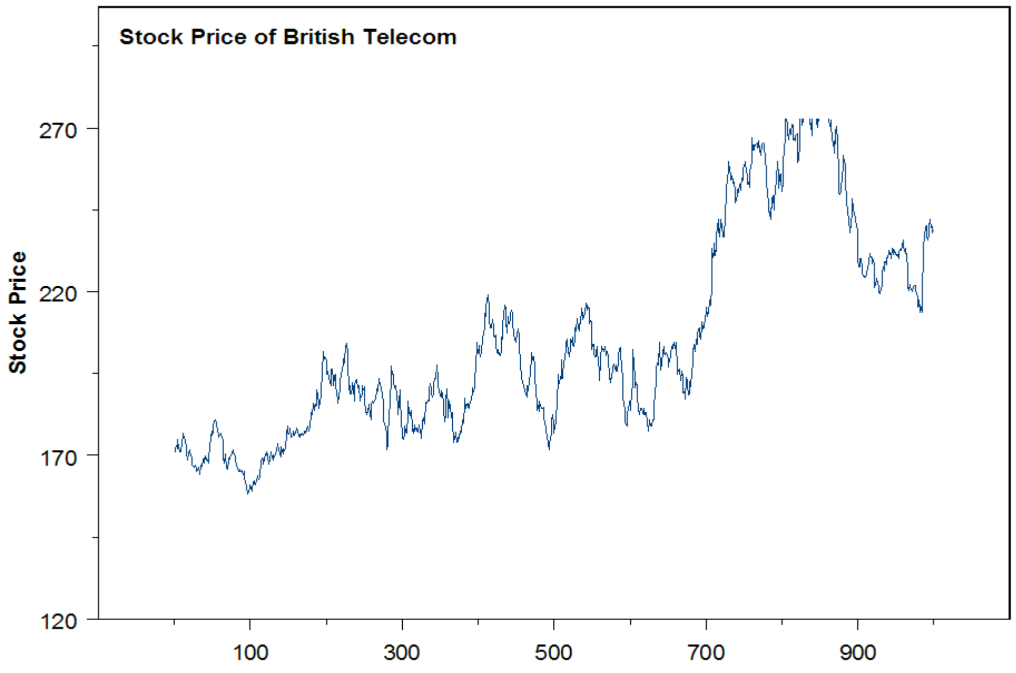


Figure D-3 Stock price of British Telecom

Appendix E – Code Listings

Only header files are printed out here. All codes can be found in the CD.

Appendix F – Bibliography

[1] Alberts, B et al. (1994) *Molecular Biology of the Cell*, Garland Publishing, Inc, New York & London, pp.821 – 847

[2] Bentley, K and Clack, C. (2004). The Artificial Cytoskeleton for Lifetime Adaptation in Morphology. *In Workshop Proceedings of the 9th International Conference on the Simulation and Synthesis of Living Systems (Alife IX). (Eds.) Bedau, M., Husbands, P., Hutton, T., Kumar, S., Suzuki, H.* Pp 13-16.

[3] Bentley, K and Clack, C. (2005). Morphological Plasticity: Environmentally Driven Morphogenesis. *Proceedings of the Eighth European Conference on Artificial Life (ECAL '05).*

[4] Bentley, K and Clack, C, 2005, The Cellanimat for Morphological Plasticity, draft

[5] Kendall, G and Su, Y (2003) "A Multi-agent Based Simulated Stock Market - Testing on Different Types of Stocks." *Proceedings of the Congress on Evolutionary Computation (CEC2003). Special Session in "Evolutionary Computation in Economics". Canberra, Australia, 8-12 December 2003,* pp. 2298-2305.

[5] Investopedia (2005), http://www.investopedia.com

[7] Kreis, T & Vale, R (1999) *Guidebook to the Cytoskeleton and Motor Proteins*, Oxford University Press, Oxford, pp.11-120

[8] Preston, T et al. (1990) *The Cytoskeleton and Cell Mobility*, Blackie, London, pp.70-86

[9] Reilly, F & Brown, K (2003) *Investment Analysis and Portfolio Management*, Thomson, New York, pp.625-689

[10] Sycara, K et al., (1996) Distributed Intelligent Agents, *IEEE Expert: Intelligent Systems and Their Applications* Volume 11 , Issue 6 pp.36 - 46

[11] Wear, M.A., Schafer, D.A. and J. A. Cooper (2000) Arp2/3 complex: control of assembly and disassembly of actin filament networks. Curr. Biol. 10: R891-R895