Resource Management in complex environments: an application to Real Time Strategy Games

Souza, Thiago A. Ramalho, Geber L. Queiroz, Sérgio R. M.

Federal University of Pernambuco, Informatics Center, Brazil

Figure 1: Starcraft Broodwar with PICFlex

Abstract

Resource management is concerned with the optimal application of currently available resources and those that are to become available in the future to achieve goals. Normally, when resources are scarce, this is not an easy task, especially when the environments are real-time, partially observable, dynamic and uncertain. Despite being a common task in real time strategy games (RTS), there is little research in resource management applied to RTS games. In this study, we use the RTS to propose an original approach to decision-making involved in managing resources in complex environments. We develop new techniques and concepts such as investment policy and reuse some existing ones. We made several simulations and the results were very promising.

Keywords: artificial intelligence; computer games; real time strategy; resource management

Authors' contact:
`{tas3,glr,srmq}@cin.ufpe.br`

1 Introduction

Resource management is the process of using resources as efficiently as possible. These resources can be tangible like equipment, financial resources, time and even human resources as employees (Resource Management, 2013). Manage resources range from managing simple things as the clothes to be worn in a common day to managing the resources of a big city. The latter is a good example of what we will call in this study a complex environment: an environment where actions occur in real time, which is partially observable, dynamic and non-determinist (Russell & Norvig). Because of these characteristics, resource management in complex environments can be very hard. Besides that, these environments often present conflicting goals and a strong and multiple (uncontrolled) interactions among variables of the model. For instance, meeting a demand from the citizens, a mayor decides to build a viaduct in order to improve traffic flow. However, due to the characteristics of the neighborhood, the viaduct may increase thefts and murder rates, and it also can cause an undesirable impact in the sightseeing.

RTS games are a good laboratory for learning how to create algorithms to manage resources, since all the aforementioned problems of in complex environments are present in RTS (Churchill & Buro, 2011) and (Weber, Mateas, & Jhala, 2010). Indeed, RTS games require the creation of items such as armies, buildings, and of technologies, each of the demanding resources to be created. To achieve victory, the player must then to perform optimal investments since the resources are limited.

Despites, its importance in RTS, resource management has not been well studied. Few works are devoted to this issue and most of them work with the concept of build-order, an order in which units and structures must be produced (Churchill & Buro, 2011). Build-order is narrow view of resource management activity, and in most cases, this build order does not take into account the dynamic evolution of the context during the game.

The main objective of our work was to propose a system capable of performing resource management, particularly in real time strategy games. Given the complexity of the problem, instead of creating a supposedly complete or detailed model and, then testing it, we follow the computer abstraction tradition (Holte & Choueiry, 2003) (Polya, 1945) and the lean startup philosophy (Ries, 2011): start with a simple approach and then to go to a continuous cycle of tests and improvements. We based our approach on the concept of "investment policy" (a simplification of real life economics one) that establishes a fixed resource management behavior. For instance, a possible investment policy is to spend 40% of the resources on the army units, 35% on buildings, and 25% on technology. Then, we have improved the model in order to take into account the context, and to give more flexibility to the decision. At this point, we have introduced the possibility to change the investment policy dynamically, as well as the idea of flexible spending margins.

In this study, we have used a computer game StarCraft Broodwar as a laboratory for experimentation, where we implemented our techniques and collected the results to be analyzed. Various models, going from the simplest to the more sophisticated ones, were created, tested and analyzed. Each of the models were compared, confronting the same computational and intelligent opponent: the BTHAI bot (created by Dr. Johan Hagelbäck[1]), in the RTS game Starcraft Broodwar. The results obtained via several simulations were very promising and taught us which concepts/models produced better performance.

In the next section we present the main problems related to RTS games and resource management in detail. Then we present the state of the art research in RTS games and resource management as well some other aspects related to this field. In Section 4 we present our original solution for resource management and its application RTS. Then, we present and discuss the obtained results. Finally, we conclude our remarks and we mention some direction for future work.

2 The Problem

RTS games are simulators in which players instruct units in real time to collect the various types of available resources, train units, construct buildings, and eventually destroy opponents to win the match. Since creating army units, buildings and technologies demand resource, which are limited, an appropriate management of resources is crucial in RTS (Buro, 2003).

In order to better understand the problem of resource management in RTS games, we will begin to describe the main elements one can find in such a game. Although there is no consensus on the elements of an RTS game, there are some elements and concepts common to most of these, which are described below.

Units - A unit is any object present in the game with which players can interact. The units are usually fighters, able to attack and destroy other units or buildings. Usually one type of unit can only be produced by a type of building. Through technological development, they may be specialized in ground attack, or long-range airstrike. Each unit has different valued characteristics such as range, hit points, energy, speed, for different purposes. It is up to the player to decide how many and which units to invest due to its strategy. Example: soldiers, citizen, etc.

Buildings - The buildings are commonly used for the production and upgrade of units, storage resources, technology development or defense. The buildings may also serve as storage facility for units, for example, the supply depot of Starcraft Broodwar.

1http://bth.se/tek/jhg.nsf/pages/273c7bbb699ba91cc1256bf80044ef09

Resources - Resources are prerequisites for the production of units, buildings or developing technologies. These resources are scattered through the map and need to be collected during the game. Resources are required for training a unit, construction of a building or development of a technology. They function as an economic currency as following example: to build a Terran Marine it is necessary to have 50 minerals in Starcraft Broodwar. Each unit, building, or technology tech tree in the game has its resource requirements, i.e., if the player does not have the resources to a specific unit it cannot be create in that moment. Each RTS game has its own set of resources, in Starcraft Broodwar, for example, the player must collect ore and gas, whereas in Command & Conquer, only ore.

Technologies and Updates - The technologies and updates are quite common in this type of game, determining that the player has a better chance to fight with his enemy, because the development of such technologies will improve the player's units. The developed technologies can improve important characteristics of an army, such as speed of movement, attack power, defense capability, etc.

Fog of War - Is the lack of complete environment visualization used in a RTS game. In strategy games this aspect can be perceived by the darkened area of the map that despite the fact that the terrain has been already explored, the player does not know what occur in that region at that moment.

2.1 Resource Management in RTS

RTS games are usually won by players who destroy their opponents first. This goal can be achieved in several ways. For example, a player might try to rush and surprise his opponent. In order to accomplish this, he invests resources to build an attacking army early in the game, at the expense of investing in buildings that are more important in later stages of the game. If the opponent invest in technological development and does not have a good defense at this early stage, the player who is attacking succeeds easily. But if the defender is able to handle the onslaught and remains alive, this could be a fatal blow to the striker who has "wasted" his initial resources and will, from now on, lag behind the defender. Therefore, the choice of strategies of the players can determine the outcome of the game. Another way to win the match would be to invest resources in buildings, assemble large armies and develop technology collecting as many resources as possible, so that more investments are possible, and at a late stage in the game engage the whole army against the enemy at once. Between the former (rush) and the latter (patience), there are numerous possibilities and adaptations that may occur during the game depending on the opponent's strategy.

As will be seen in the literature review, we found no studies that enumerate and systematize the various problems encountered in the management of resources in RTS. We made an effort to produce a preliminary identification of these problems. In order to achieve this we started by using our own experience as players of RTS games and then we interviewed ten experimented players. We were able to identify frequent problems cited by them, which made us believe that these are the most common problems involving resource management in RTS. In what followings, we describe some problems of resource management in RTS. The following questions were asked to each of the players:

- What are the major challenges related to the management of resources do you see?
- How do you see the influence of real time behavior of this type of game on creating extra difficulties for you?
- How do you analyze the overall context of the game and make decisions about investments and resource management?

2.1.1 Limited resources

In a RTS game, whereas the resources are limited, in any situation where we need to invest resources to get something there is potential for having a problem.

2.1.2 Too many options

A player during a RTS game is faced with a myriad of investment choices available. For instance, types of rider, soldiers, several technologies that can be developed, buildings that can be constructed, etc.

2.1.3 Quantities to consider

More than just evaluate and choose the investment item, the player must decide how many items it will create at once. An example is the following: the player at a specific time of the game decides to create soldiers, but also needs to decide how many soldiers it will build.

2.1.4 Options difficult to assess (multi criteria)

One aspect is quite complex: how to compare investment options at a specific time? A priori comparison of items of the same class seems to be easy and straightforward. But even when these options are investment items of the same class, for example, a soldier or a knight - both are items of combat that have similar attack and defense values, but with different characteristics - comparing them is a difficult task because the current context of the game is crucial for this comparison. A rider may be better than the soldier when you are playing against one kind of army, but against another one a rider is totally inefficient and a soldier would be a more interesting choice. When it comes to compare different classes of items the difficulty is even greater. How to compare in a specific situation if it is best to create an airport or to develop a technology, or even a new squad of soldiers?

2.1.5 Dependency between options

Normally the RTS game has dependencies between the items, i.e., in order to create item A, item B should already exist. A clear example of dependency among items that exist in most RTS games are the deposits of units. The units cannot exist without constructing a building first to serve as their deposit.

2.1.6 Time

Time itself is a scarce resource in a RTS game. It is an orthogonal factor to all decisions in RTS games. The RTS game takes place in real time and all players should accelerate their actions so that they can build the largest empire in the shortest time. For example, if a player is slow to make decisions at the beginning of the game, he does not create an effective army that can defend him and can be easily defeated by an opponent that adopted a rush strategy.

Evaluate and decide whether it is better to invest a specific amount of resources in the current time or postpone the investment for another time when it may be more appropriate is a rather complicated task. An example to illustrate this difficulty follows: how to decide whether it is better to invest X now to create N infantry units or wait T turns to develop a weapons technology that will cost X + Y?

2.1.7 Context

It is easy to see that when his empire live a peaceful moment the best decisions are often very different from the ones in a situation where the player's empire is being attacked by an enemy army with a devastating force of attack. For example, when the player is being attacked and there are no soldiers to defend him, it is ill-advised to create workers instead of soldiers. The context of the game is something very subjective and it is essential in the evaluation of options and strategy changes.

2.1.8 Combination between assessments

Throughout the game the previous problems clearly mingle and the context changes at every moment. In a real-time environment the time factor is orthogonal to all the problems mentioned. When real-time is added to the problem of multiple choice, the decision problems become even more complex, because an item A may be better than an item B at time t_0, but may not be interesting in time t_{0+i} which is the time in which the investment is made.

2.1.9 Recommendations for the problem's solution

We believe that a good RTS game playing requires the combination of activities deliberately planned and real-time reactions to game changing conditions. The best solution for management and investment of resources will be the one that results in the highest game scores. It is expected that the solution proposed by this study meets all these requirements:

- Fast – a solution to the problem of resource management must be fast enough so that the decision making of investments do not delay the game progress.
- Adaptable – the solution must be tunable to the constant changes inherent in an RTS game, by adjusting it to the current state. The suitability of the investments must meet both the instant needs as well as the long-term needs.

- Performance – the system must meet the performance standards for real-time games.

3 Related Work

An approach to the problem of resource management in a RTS game is based on the concept of marginal utility, which means in economics, the satisfaction of the consumer (Harmon, 2002). The game units are consumable objects and the player is the consumer. Each consumable object has a priority of the investment according to ratio between its marginal utility and its cost. The latter is measured by the monetary cost of the item and its production time. The former is calculated as follows: an utility is assigned to an objective (for instance, to build a tower), a plan is created to achieve this objective, utilities are assigned to each sub-objective of the plan (for instance, to collect wood, to collect gold, to build a worker, etc.). This approach has two main problems: (a) it is complicated to build plans for all possible situation in a RTS game, (b) the attribution of utilities are arbitrary. Besides that, no experimental evaluation is presented to validate the approach.

A large part of the research concerning resource management is related to the concept of build-order. Researchers Churchill and Buro define build-order as the order in which units and structures are produced in a RTS game (Churchill & Buro, 2011). Build-order is then related o resource management, as it depends, among other things, on existing resources and those that will be collected. However, build-order is actually more restrict than resource management, as it considers, a priori, the application of resources as static, when a RTS game is dynamic, in the contrary. We understand that the order in which things are created in an RTS game should be contextualized. For instance, whether creating a defense tower must come before creating a worker or an attack unit depends on the game context. Working with an ordered and fixed list of investments, no matter what is happening in the game, may be not good enough to deal with the complexity RTS environment.

The study of Kovarsky and Buro (Kovarsky & Buro, 2006) is focused on optimizing the build order in RTS games. The research aims to optimize the collection of resources and the creation of units and buildings in the early stages of RTS games, which means that once optimized, the build order will not change anymore. It considers two types of optimization problems: minimizing the time to reach a certain goal, for example, build two tanks and five soldiers, or maximize the amount of resources at a specific time, i.e., maximize gold collection in ten minutes. The approach seems interesting but does not take into account the dynamic evolution of the context during the game.

The study of Churchill and Buro (Buro & Churchill, 2012) focuses on optimizing the planning of investments, assuming that the build order is already given (Churchill & Buro, 2011). They try to find actions that achieve the implementation of the build order as fast as possible. In other words, the point is not to decide on what to invest, but how to minimize the time interval of the investment plan. Acting fast is very important in a RTS game due to the fact that players act asynchronously, however this is marginal issue in resource management.

Another study (MCCoy & Mateas, 2008) addresses the problem of build order on an ad-hoc basis, so that the construction of army units, technological development and construction of buildings are made according to different "strategies". In every moment of the game only one strategy is enabled by starting with an opening strategy that they call *InitialStrategy* responsible for the initial growth of the economy and military production. The strategy that follows after this is the *TierStrategy* which is responsible only for military growth, building a soldier for every worker created. With an ad-hoc strategy that is responsible for building two barracks and a blacksmith; beyond that, two soldiers that are built for each worker.

We have reused some ideas and concepts from the state of the art. We have applied Kovarsky ans Buro's idea of having a module that decides which object to create when there is demand for other objects or resources. Harmon's work inspired us to borrowed concepts from real world economics to deal with RTS resource management. From the work of MCCoy and Mateas, we have adopted the idea of having a limited set of strategies that, when adopted, may change the investment priorities.

4 PICFlex: an approach to resource management based on Contextual and Flexible Investment Policy

The problem was so complex that we had to take a set abstractions and reformulations so that we could make simplifications. The number of possible states in a finite environment, dynamic and partially observable is potentially very large. The combination of all the variables in all states that these results can take on a huge set of possible states make this problem intractable by solutions based purely on exhaustive search (Cunha & Chaimowicz, 2010). To create rules to handle each one of the possibilities and nuances of context seem unreasonable, because besides the dynamics of the environment there is also the fact that it is partially observable, which brings the uncertainty factor. Additionally, the environment can have constraints of time - which could even be real-time constraints - which would likely be disregarded because the amount of possibilities that should be analyzed. So to overcome all the problems mentioned above, and to simplify the decision making investing module, a higher level of abstraction has proved essential, since it is believed to enter the complex combination of numerous variables involved makes any solution apart from being found.

4.1 Investment policy

The central concept of our model is the investment policy, around which other concepts are articulated, such as investment classes, investment context, policy flexibility, investment demand, etc.

We name as **investment policy** the spending pattern of resources according to the type of demands. Formally, an investment policy P is given by:

$$P = \{[c_1, t_1], [c_2, t_2], \dots, [c_n, t_n]\}, \qquad \text{where} \sum_{i=1}^{n} t_i = 100\%$$

where c_n is the n-th investment class and t_n is the reference target for investment in the n-th investment class. In other words, percentage values that represent spending targets for each investment class.

Investment Policy choice - The experienced human player perceives the nuances of the game over the course of the match. The general context changes: defending his empire of an opponent's attack, attacking the enemy, or even in a peace time. Depending on the context player adopts different strategies that unfold in different investment policies and behaviors. For example, targets investment in times of peace are usually different from those when the player is under attack or conducting attacks against enemies. Therefore it is not recommended to have a single policy. As the environment is dynamic, context switches and a proper investment policy at a time may not be as proper as before. You need to choose the best policy according to the context. in other words, have goals for each class of investment that are consistent with current reality. This led us to formulate the concept of **investment policy choice function**, *F*, which is formally defined as

$$F(P, C) \rightarrow P'$$

where *P* is the current investment policy, *C* is the current game context and P' is the new policy chosen by *F*, and *P'* supposedly better adapted to the new game context.
We have created two implementations of F for us to use and test in our implementations of PICFlex. The first is called **simple** (*Fs*) that regardless of the context always returns the same policy, i.e., the investment policy is never changed. It can be defined as follows:

$$Fs(P, C) \rightarrow P$$

The second implementation of F is more sophisticated and analyzes the game context to choose a more appropriate investment policy. This function analyzes a set of pre-existing investment policies that have been identified as appropriate by experts to specifics contexts. We call this function **adequate** (*Fa*), it creates a new investment policy which adapts PICFlex to the new reality of the game. *Fa* is defined below:

$$Fa(P, C, G) \rightarrow P'$$

where P is the current investment policy, C is the context in question, and G is a set of pre-existing policies. In practice, F allows switching policies to adapt better to the game context.
It should be noted that it is possible to create numerous versions of F more sophisticated than Fa function, not needing to rely on pre-existing policies. However, for the scope of this work and in order not to complicate without apparent necessity, we adopt the Fa function as an alternative version of the even simpler version, *Fs*.

Context - The context of a game consists of variables that describe its current state. We know that the context is something very complex, but to simplify our overall implementation we have defined only three contexts; they are: "peacetime", "under attack" or "performing attack".

Execution's strategy of Investment Policy - The investment policy will set spending goals for each investment class, but it does not say how these goals should be pursued. It is understood that, in the long term, the adoption of the policy will provide "balance" in spending per class according to the proposed goals, but the process of pursuit of these goals may not be linear. As the goal can work as a spending ceiling, there may be complications in policy execution. For example, suppose that the goal of spending on buildings units is 25% of the resources, and expenditures made so far to this specific class are 23%. If it becomes clear in the context of the game, it would be necessary to create a new building unit that would raise the percentage spent for this class to 27%, the player should do or not such investment? If he follows strictly the policy should not do it, but the player could build that unit, because he understands the urgency and usefulness, and pay the "overflow" of the goal in future investments, bringing the percentage of spending on buildings to the original level of 25% as specified in the policy.

To explain the way or whether one can pursue the goals we have introduced the concept of implementation of the investment policy strategy *S*, which is formally defined

$$S(P, D, H) \rightarrow boolean$$

where P is the current investment policy, D is the current demand and H is the set of expenditures made by PICFlex throughout the game. The S function receives a demand that may be creating an item of army, or the development of any technology and must authorize it or not based on *H* and *P*.
We have created several variations of S to use in our implementations of PICFlex. The **generous function** *St* always allows demand to be made returning *true* whatever the investment policy, demand and set of expenditures. This strategy turns out to nullify the effects of the investment policy chosen once given a demand, it shall be permitted regardless of the set of expenditures or investment policy. Formally, *St* and is defined by

$$St(P, D, H) \rightarrow true$$

The second implementation of S is called **accept once** *So* allows a demand to break the ceiling of the investment target only one time, the second demand (which break the ceiling of the target) will be denied until the balance of spending is reestablished.

$$So(P, D_1, H_1) \rightarrow true$$

$$So(P, D_2, H_2) \rightarrow false$$

$$\ldots$$

$$So(P, D_n, H_n) \rightarrow false$$

where $D_{1\ldots n}$ go beyond the target ceiling $D_{2\ldots n}$ consecutive to D_1.

The third implementation we model S is called **rigid** Sr and does not allow a claim, which break the ceiling of the target of a class of investment is met and is defined as follows

$$Sr(P, D_1, H_1) \rightarrow false$$

if the cost of creating D_1 extrapolates the target of his class. In practice, this strategy does not allow spending beyond the goals.

The fourth implementation of S we have created is called **observed growth** Sg and is a little different than the previous one so that it allows the demands that exceed the target ceiling of a class X are met until a demand of a class Y is rejected by extrapolating its target. Defined as follows

$$Sg(P, D, H) \rightarrow boolean$$

For exemple:

1. $Sg(P, D_1, H_1) \rightarrow true$
2. $Sg(P, D_2, H_2) \rightarrow true$
3. $Sg(P, D_3, H_3) \rightarrow false$
4. $Sg(P, D_4, H_4) \rightarrow false$

where D_1, D_2, D_3 and D_4 are consecutive and extrapolate the target and D_1, D_2, and D_4 and are class X but D_3 is of class Y. If the step 3 had not happened the step 4 could return *true*.

A **demand** for PICFlex is a need for some investment item and is defined as

$$D = \{Object, BuildTime, Cost, Class\}$$

where *Object* is something that needs to be created, build time is the time that the object takes to be created, cost is the price paid to create a new *Object* and *Class* refers to the investment type to which the *object* belongs.

When demand comes an investor should consider it and decide whether to answer or not, and for simplicity, we divide into two types: **permanent** and **circumstantial**. A perennial demand is the demand that happens steadily, without any extraordinary event happens. This demand is scheduled and when the time comes PICFlex will analyze and decide whether it will meet it or not. To illustrate an example of a perennial demand imagine if this case our model was applied to a mayor of a municipality, then a perennial demand would be garbage collection, police patrol neighborhoods or even checking the lighting. The other type of application, the circumstantial situations happens when the environment does not create any scheduled necessity. Using again the example of city hall, an example of a situational demand would be the building of a new public school when the student population grew.

We can also divide the circumstantial demands on internal or external demands. Internal demands are those that arise from the internal resource manager as the example of the school described in the previous paragraph circumstances - in this case the Investor is the mayor. Have external demands are caused by some external entity. To illustrate using the town hall, an external demand would be hiring more teachers who had been given for the creation of a federal law - in this case the external agent is the federal government.

The way that comes PICFlex demand takes into account the type of demand and its urgency. The PICFlex for some circumstantial and urgent situation may find that an external demand has priority over an internal, but the most common situation is to treat the PICFlex internal demands with higher priority than the external and treat circumstantial demands with higher priority over perennial.

TABLE 1 PROPORTIONS

Defensive policy	
Army	30%
Building	35%
Upgrade	20%
Tech	15%

4.2 PICFlex architecture

In this section we describe the module that will do the necessary tests and finally take the decisions. To simulate the behavior of PICFlex, we use the simulation platform game Starcraft Broodwar provided by the company Blizzard Entertainment which makes possible the testing of the intelligent module.

The process of decision making in a complex environment involves many aspects that make the task quite complicated. Therefore, we conducted several observations matches experienced RTS players to identify the techniques used. The aim was to identify aspects of the experienced player that led to victory. We understood that if PICFlex had intelligent behavior similar to the human player their success rate would be good. Then, we find the frequent presence of behavioral aspects described in detail below.

Class proportions - we found that players obey a proportion of investment of resources among investment classes. Basically, it is a share of the investments can be made in a class. A potential distribution is shown at Table 1.

Investment item - all things of RTS games that can be created we named investment items. They are the possible investments that PICFlex can make. When investments are made and the items are purchased, going to be called "objects", we can say that the investment item is a class and object is an instance of this class. Each object has own life and behavior as well as characteristics or properties which each of these have an assigned value. These are items that, through their properties, change the current state of the environment. The properties common to investment items are:

- Cost of creation – the value in monetary resources of the item.
- Creation time – the time it takes to create.
- Required objects – the item has a list of objects that should exist prior to its creation be possible.
- Builder / Creator – item type that uses resources to create the item in question.
- Role – the role that develops in the setting item.

Basic investment item - we created the concept of basic investment item for this study that is the most representative item. In Starcraft, for example, for the Terran race, the *Marine* was chosen as the most representative, as well as being the first army item that can be created, it is inexpensive, and has good attack characteristics. This item will be used as a basic reference for a balanced growth pattern as discussed below.

Balanced growth pattern - we found that the armies of the players grew up in a balanced way conducted by an implicit pattern of all players. This pattern requires that for every X items of type A are needed Y of type B. So whenever this amount of items is out of the pattern the PICFlex should schedule creation of units required for pattern to be reestablished. We chose the Terran Marine as the basic item, thus balancing the army's growth will be based on the number of reference Terran Marine existing.

Trigger rules - rules that trigger investments scheduling events according to specific situations and ultimately creates circumstantial demands. For example, when there are few items to store units, or insufficient, the players commonly order the creation of more of these.

Prerequisites - set of prerequisites must be provided so units can be created. These prerequisites can be resources or other units that must exist before starting their creation. The graph obtained by listing all units prerequisites and removing transitive edges is called the *tech tree* (Churchill & Buro, 2011).

Posture - we created the concept of posture that is a set of values assigned to the properties of the investment policy. In technical terms we can say that an posture is an instance of an investment policy. For simplicity, we created three basic postures and essential PICFlex can assume that during the game: initial, defensive and aggressive. In practice, the posture determines the way that the PICFlex behaves in a given time of game - may assume another policy at any time. It is the posture that will determine the PICFlex take a more aggressive behavior and invest in a great army in order to storm the opponent or get more defensive strengthening their buildings and defense units. With regard to implementation, the posture toggles a set of attribute values of the investment policy that define the behavior of PICFlex. In the initial posture, the PICFlex obeys ad-hoc rules. Defensively, the PICFlex main goal is to invest in the growth and strengthening of army building buildings for the production of army units and performing upgrades / techs that will strengthen such units and buildings. In aggressive, the goal is to increase the power of the army increased in numbers and firepower.

5 Evaluation

We have implemented various investment policy variations, going from the simplest to the more sophisticated ones. Then we have compared them using the StarCraft Broodwar simulation platform. This section presents these implementations, the platform and the obtained results.

5.1 The six implementations PICFlex

To evaluate the techniques studied and developed by ourselves, we create six incremental implementations PICFlex. The implementations were called "Player 1" to "Player 6" and will be described below in detail.

Player 1 (random player) - The first implementations PICFlex is completely random and their behavior with respect to resource management is impossible to predict. There is not investment policy in its behavior nor concept of balanced growth pattern neither concept of class proportions. Also there is not the routine that schedules army's growth investing in basic item. Like all implementations will be described, the player begins his first game with an open strategy that is enough to not be defeated in the match early in the game. Hence forth, this player starts to make decisions on resource management at random and without any control.

Player 2 (investment policy random player) - The second player was created in order to evaluate our concept of investment policy. The Player 2 is a modification of Player 1, described in the previous section, but has army's growing routine based on the basic investment item - a routine that every X turns creates a basic item; investment policy – has investment policy, but it is randomly set at the beginning of the game and immutable; balanced growth pattern – there is, however random.

Player 3 (investment policy fixed player) - The Player 3 is a change from Player 2 made to evaluate the concept of investment policy, but this time with the values of the attributes of the policy chosen by experts.

Player 4 (balanced growth player) - The fourth player we created is identical to Player 3, but with the change in the army's pattern of growth was defined by experts.

Player 5 (toggled policy by postures player) - The only difference between the Player 5 and above is the fact that for Player 4 investment policy is fixed, so their behavior is the same for any game situation. However, for Player 5 policy changes during the match (by changing **posture**) and thus this player assumes behaviors appropriate to the different contexts of the game.

Player 6 (flexible policy player) - The player of this section is again equal to the previous player, but with the difference that the proportions of investment classes are flexible.

Player 6 in details - This section aims to describe how PICFlex, in its most complete form, behaves - the Player 6 is the implementation of PICFlex that aggregates all the concepts mentioned in the previous sections.

In this paper we treat the RTS game in which empery begins with a few units and resources available in its economy – often just enough to build a few workers. Then, once the game begins the initial posture is assigned ant toggles the policy's attributes. The initial posture creates trigger rules for construction of essential units at the beginning of the game, for example, the construction of a Barracks a Academy, twenty Marine and six Medics. These investments are scheduled because the opponent used to evaluate the solution attacks the empire coordinated by our player early in the game. If these investments are not made in an ad-hoc way, there will not be time enough for PICFlex identify the need and then it lose the game most of the times.

When the empire player reaches a level of maturity specific set of ad-hoc manner, the PICFlex assume defensive **posture**. The defensive aims of strengthening the empire as a whole inducing PICFlex choose to create multiple buildings, train army units and develop a tech tree. Investments in buildings and army classes are covered with proportions (at this time) slightly larger, as is the vast majority of possible investments in RTS games and therefore require more resources especially at this moment when the army needs to grow quickly.

One aspect that was incorporated Player 6 for a balanced growth pattern that governs the growth of the army with the types of units that will be created and the amount of each type. But in order to the army to grow there must be a rule that fires a basic investment item creation trigger for every X turns. Several simulations were performed to find the best configuration of proportions between the items.

The game follows and investment items are allocated and investments are made. The empire grows and as it grows some criteria are analyzed in the current environment to the change of posture. The rule that triggers the change of current posture – that is defensive to **aggressive** posture – is based on criteria *ad-hoc*. So PICFlex that could invest in technology, buildings, and now an army can then invest heavily in increasing the attacking army is already created so strengthened - by technology developed previously - and grow more rapidly by buildings constructed.

To determine the time to attack the opponent we defined a simple rule - because this analysis is very complex and it is not the focus of this study. Basically checks if the ratio of the number of units of the attack squad with the number of all units of the army is greater than 50%. Thus, the PICFlex realizes the right moment to attack the enemy and, in addition, aggressive posture is assigned to it (if not already in this position), which is the most appropriate approach for moments of attacking the enemy.

When PICFlex takes aggressive posture that will have more resources to invest in the army – in proportional terms, because as has been said, the posture toggles a set of variables and among which are the class proportions. For example, when the posture taken by PICFlex is **aggressive**, the investment class army will have more investment quota than all other classes (building, tech/upgrade), which will allow larger amounts of resources to be spent in this class.

5.2 The simulation platform

In order to deploy the solution and run the tests on digital computers we needed a platform that would make it possible to create AI modules and tests for the analysis of the results. The BWAPI 3.7.2[2] (BroodWar API) is a framework for C++ open source that made possible the creation of artificial intelligence modules for Starcraft Broodwar. Using BWAPI, developers can get information about the players and the individual units of Starcraft as well as issue a variety of commands to units, opening the door for AIs with custom algorithms. This API has been used to study aspects of AI RTS games (Buro & Churchill, 2012).

So in order to test our intelligent module, we needed to build a bot so it could play a lot of matches of Starcraft without human interference and collect the results at the end of the matches. We decided to use the bot of a researcher in the field of Artificial Intelligence in RTS games, the BTHAI[3], Ph.D. Johan Hagelbäck.

5.3 Results

We made experiments on our implementations PICFlex to whether the techniques used or not improve the performance of the task of managing resources. We put our players one by one to fight native AI game of Starcraft Broodwar and collect data from 150 matches for each player. The data collected were scores of our implementations of PICFlex and native IA opponent's score.

TABLE 2 RESULTS

PICFlex	Success rate	Dif. Score Mean	Dif. Score Std. Dev.
Player 1	15.33%	-1851.92	14536.95
Player 2	2.67%	-1863.62	6469.54
Player 3	20.67%	1839.42	12918.82
Player 4	25.33%	2389.47	21559.54
Player 5	26.67%	7062.02	14640.15
Player 6	42.00%	10801.01	17343.73

Players comparison. Dif. Score Mean is the mean of the difference between the player's score and the enemy's score; Dif. Score Std. Dev is the standard deviation of score's difference.

2 API C++ *StarCraft BroodWar* Interface (BWAPI) can be found at http://code.google.com/p/bwapi

3 Information about BTHAI can be found at http://code.google.com/p/bthai

The Table 2 shows the comparison between players. There are information of samples that were collected for each player as their number of wins and losses. Success rate which is the percentage of victories compared to the number of samples. Average score difference where the difference in score is the score of the player subtracted from the opponent's score. And finally, the standard deviation calculated for the difference in score. You can see that the Player 6 has the best success rate which is 42% of matches won.

We run tests of normality and we concluded that our data had not normal distribution, so it wasn’t possible to run the Student's t-Test for hypothesis testing, so we decided for using the Wilcoxon signed-rank test.

5.4 Analysis of the results

The Player 1 is completely random and it has no investment policy. This player plays unpredictably, but with a open strategy. What we realized is that this player is very innocent and it is easily defeated by the opponent; it does not make any reasoning about the context or plan of attack or defense.

The first of the players has a success rate of 15.33%, winning 23 matches. We realize that the open strategy was what determined that this player could beat that rate, since it does not even have a routine growth of military or any other planning that gauge context data to act in response. After opening strategy the player realized investments randomly which in 15.33% of the time it took the victory we understand as being fortuity.

TABLE 3 HYPOTESIS TESTS

Name	**Alt. H.**	**Name**	**Null H.**	**p-value** ($\alpha = 0.05$)	**rejected**
HA1	P2 > P1	**HN1**	P2 = P1	0.0004065	**HN1**
HA2	P3 > P2	**HN2**	P3 = P2	0.9698	-
HA3	P3 > P1	**HN3**	P3 = P1	0,001898	**HN3**
HA4	P4 > P3	**HN4**	P4 = P3	0.4412	-
HA5	P4 > P2	**HN5**	P4 = P2	0,6311	-
HA6	P4 > P1	**HN6**	P4 = P1	0,01789	**HN6**
HA7	P5 > P4	**HN7**	P5 = P4	0.003252	**HN7**
HA8	P5 > P3	**HN8**	P5 = P3	$5,53 * 10^{-5}$	**HN8**
HA9	P5 > P2	**HN9**	P5 = P2	$3,81 * 10^{-8}$	**HN9**
HA10	P5 > P1	**HN10**	P5 = P1	$6,331 * 10^{-10}$	**HN10**
HA11	P6 > P5	**HN11**	P6 = P5	0.01637	**HN11**
HA12	P6 > P4	**HN12**	P6 = P4	$7,888 * 10^{-5}$	**HN12**
HA13	P6 > P3	**HN13**	P6 = P3	$3,737 * 10^{-7}$	**HN13**
HA14	P6 > P2	**HN14**	P6 = P2	$1,415 * 10^{-12}$	**HN14**
HA15	P6 > P1	**HN15**	P6 = P1	$1,152 * 10^{-11}$	**HN15**

To understand the Table 3: Being P1 the score of Player 1 and P2 Player 2's score, we have established the following hypothesis: Alternative Hypothesis HA1: P2> P1 and HN1 and Null Hypothesis: P2 = P1. We ran the wilcox-Test with $\alpha = 0.05$ to the results were p-value = 0.0004065 which means that the null hypothesis HN1 can be rejected.

Player 2 added in the investment policy, but the policy is random. A random routine defines the attribute values of the investment policy, and from there these values are no longer adapted regardless of what happens during the game. The player also has the balanced growth by a pattern, but the pattern was such assignment taken at random, that is, the pattern was set at the start of the game randomly, once defined the pattern was not changed over, which had allowed the growth of their army so predictable. The Player 2 obtained a success rate of only 2.67% winning 4 matches which resulted in a rate worse than Player 1. We observed that the investment policy was the cause of this win rate worse than Player 1, because the investment policy exists and that player is chosen at random, what ends up waging investments that were demanded by the context. For example, the army required a Barracks, however, if the policy of investment in buildings prevent more spending, the army could not grow and the players ended up being defeated. The chosen policy can prevent an investment if the share of investment in a given class has already been attained. However, the hypothesis tests indicates that Player 2 is superior to Player 1 as the null **HN1** (**P1** = **P2**) can be rejected.

To the Player 3 expert human players were who have assigned values to the investment policy which caused the player had a success rate much higher than that of Player 2. This player won 31 matches and got a rate of 20.67%. The Player 3 also had more success than Player 1 which indicates that the investment policy or for a balanced growth pattern (although this random player) were the cause in the improvement of the results. Despite the number of victories of Player 3 is superior to Player 2, the hypothesis tests doesn’t show this superiority once the null hypothesis **HN2** cannot be rejected.

To Player 4 experts assigned values to the patterns that defines the balanced growth and this player was even better than his previous. It achieved a win rate of 25.33% winning 38 matches. We understand from the data that defines the growth pattern is what made this player have better results than the Player 3. However, even the number of wins Player 4 being higher than the Player 3’s wins, nothing can be said about this relationship because **HN4** hypothesis cannot be rejected.

The Player 5 obtained a success rate very close to its predecessor, got 26.67% winning 40 matches. We believe that only the switching of **postures** according to the context was not sufficient to reasonably increase the results in relation to Player 4. However the null hypothesis **Hn7** can be rejected which confirms the superiority of the Player 5 over the Player 4.

The sixth player is the most complete of all the players and got the best results when compared to others winning 63 matches with a 42.00% rate of success. The investment policy of that player besides switched according to the context, is flexible. We believe that the automatic adjustments that are made in the investment policy that players were the cause of its success, once, for example, when there was demand for some investment that the policy did not accepted, rather than deny the policy investment, the PICFlex fit the needs. Definitely Player 6 is the most superior of agents and the null hypothesis **Hn11** is rejected.

As imagined, the concepts that we insert our players, as balanced growth pattern and investment policy improved the results obtained with the exception of Player 2 that got the worst success rate. As we have previously clarified, we believe that the failure of that player is due to the fact that the policy was chosen completely at random and was not adjustable, and it ended up preventing major investments to be made because the policy was poorly chosen (randomly). The Player 3 has achieved better results than the previous ones because the policy is assigned by experts human players. The Player 4 Player 5 and had success rates very close which leads us to believe that only the switching of postures was not sufficient to significantly improve the results. The best success rate was obtained by Player 6 which has in itself all the concepts mentioned in this paper. We believe the most significant success of the latter player is due to the fact that its policy besides being switched (according to the context of the game) is flexible which gives more freedom to investments.

6 Conclusions and future work

In this paper we have presented an original artificial intelligence approach focusing on problem performing resource management in complex environments. For the sake of simulation and adherence to our scope we chose RTS games to test our approach. For this purpose, we explored the concept of investment policy, which was progressively improved to be context-sensitive and flexible.

We have showed that our investment policy approach brought positive results especially when we added to the policy contextualization and flexibility. Our simulations have demonstrated that PICFlex, our most complete model, has the potential to be implemented for resource management in commercial games.

The BOT we use as test platform presented tactical issues that could not be solved because the scope of our research was on resource management only. Beyond the limitations of BOT we use, the actual API available for testing BWAPI has several limitations and bugs still to be fixed, which made it more difficult our simulations and data collection and created a limitation in our experiment. For these reasons, we intent to make some improvements on the BOT since we believe that, once solved the tactical issues, several opportunities to improve the strategy will be possible. Another study we believe could be really interesting is the adaptation of PICFlex to other complex environments rather than RTS games, such as the decision support systems.

References

Buro, M. (2003). Real-time strategy gaines: a new AI research challenge. *Proceedings of the 18th international joint conference on Artificial intelligence*, (pp. 1534-1535).

Buro, M., & Churchill, D. (2012). Real-Time Strategy Game Competitions. *AI Magazine* .

Churchill, D., & Buro, M. (2011). Build order optimization in starcraft. *Proceedings of AIIDE* , 14-19.

Cunha, R., & Chaimowicz, L. (2010). *Um sistema de apoio ao jogador para jogos de Estratégia em Tempo Real.*

Harmon, V. (2002). An economic goal approach to Goal-directed reasoning in an RTS. *Ai Game Programming Wisdom* , 402.

Holte, R. C., & Choueiry, B. Y. (2003). Abstraction and reformulation in artificial intelligence. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences* , 1197-1204.

Kovarsky, A., & Buro, M. (2006). A first look at build-order optimization in real-time strategy games. *Proceedings of the GameOn Conference*, (pp. 18-22).

MCCoy, J., & Mateas, M. (2008). An Integrated Agent for Playing Real-Time Strategy Games. *Association for the Advancement of Artificial* .

Polya, G. (1945). *How to solve it: A new aspect of mathematical method.*

Resource Management. (11 de 07 de 2013). Fonte: Business Dictionary: http://www.businessdictionary.com/definition/resource-management.html

Ries, E. (2011). The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses.

Russell, S. J., & Norvig, P. *Artificial intelligence: a modern approach* (Vol. 74). Prentice hall Englewood Cliffs.

Weber, B. G., Mateas, M., & Jhala, A. (2010). Applying goal-driven autonomy to StarCraft. *Proceedings of the Sixth Conference on Artificial Intelligence and Interactive Digital Entertainment.*