Map-Adaptive Artificial Intelligence for Video Games

Laurens van der Blom

June 12, 2007

Abstract

Developing artificial intelligence (AI) in today's games – classic as well as commercial – is difficult. In real-time strategy (RTS) games, game AI is typically modeled after the behaviour of its opponent(s) (i.e., opponent modeling). In this paper, we propose a machine learning approach to establish effective game strategies based on the structure of the environments of the game (i.e., the so-called maps). In the approach, a decision tree is constructed that is applied to the structure of the maps. This way, strategies are devised that are used to defeat the opponent(s). Experimental results obtained in the video game Total Annihilation Spring indicate that it is possible to use a decision tree to construct effective strategies. Classifications of the attributes of the maps are successful, but actions suggested by the decision tree are not always fully executed, because of external influences.

Keywords: Video games, RTS, machine learning, map-adaptive behaviour.

1 Introduction

This introductory section provides some background information about game AI in RTS games and an application of it. Additionally, our problem will be described and an overview of this paper will be given.

1.1 Background

Classic and commercial games make use of AI in order to provide a good challenge for the player or in order to support the player in an effective fashion. Simple AI is commonly used in the games, but, depending on the circumstances, more complex forms of AI can adapt themselves to the behaviour of the player(s); they are modeled after their behaviour, so that they can support or play competitively against (human) players accordingly. However, in this paper, we will consider an aspect of game AI in RTS games that does not seem to be as widely used, most probably because it cannot always be applied successfully in the games due to its complexity. This aspect is the structure of the map in which the game AI is situated. Arguably, using the structure of the map is effective in games genres where the focus lies on the long-term effect of game actions (e.g., conquering a part of the map or even winning at the end of the game). One of such game genres is RTS. In RTS games, players construct bases, gather resources, build units and dispatch them to defeat their opponent(s). To win an RTS game, the player has to perform highly strategic actions in order to fulfill one or more conditions – a commonly used condition is defeating the opponent(s). They have to take into account, among others, the behaviour of their opponents, the strengths and weaknesses of their units and the structure of the map.

In this paper, we focus on one technique from machine learning, namely decision trees. Decision trees can be particularly useful in applications where discrete values are common and only simple decisions need to be made. Our choice for decision trees stems from their ability to describe any Boolean function and approximate a large set of functions correctly, when done right.

An example of how the structure of a map can be used to devise strategies in order to defeat the opponent(s) is the following: If the map in a game is split in half because of mountains or rivers and the game AI's base is on one side and the opponent on the other, then one of the most logical steps to do at first is to conquer the only roads available towards the game AI's area, making it difficult for the other player to intrude. In other words, the game AI is playing defensively for the most part. This is achieved by using a decision tree that can classify such attributes of the map.

1.2 Problem Description

In this paper we investigate to what extent game AI can establish a successful strategy for defeating the opponent(s) in a RTS game, depending on the structure of the map. A typical and open-source RTS game called Total Annihilation Spring[1] will be used as our environment to implement the decision tree into and to conduct our experiments in. Moreover, we assume that the structure of the entire map is known from the beginning of a game (i.e., perfect information is available).

[1] See also http://spring.clan-sy.com.

1.3 Overview

First, a description of our approach to determine strategies based on the structure of the map is given in Section 2. Second, the experiments that test our approach are described in Section 3, where the results are shown and discussed as well. Finally, in Section 4 the experimental results are reviewed, the conclusion to our research is provided and suggestions for future research are given.

2 Approach

This section describes what a decision tree is and how we use it in order to determine strategies based on the structure of the map. The decision tree is trained with a number of examples, which consist of input data with values for attributes of the map and the corresponding target output data in the form of actions of the game AI.

The decision tree is discussed in Section 2.1. Section 2.2 describes how it is implemented in the game Total Annihilation Spring.

2.1 Decision Tree

A decision tree is a tree where each internal node tests an attribute, each branch corresponds to an attribute value and each leaf node assigns a classification. The decision tree is a predictive model, which maps observations about an item to conclusions about its target value. Since we are only dealing with discrete values, such decision trees are also called classification trees. The leaves represent classifications and the branches represent conjunctions of attributes that lead to those classifications.

In the game Total Annihilation Spring, each attribute of the structure of the map can be expressed in discrete values, such as the number of resources and their densities, or the number of obstacles like mountains, rivers and cliffs. The decision tree also allows disjunctive descriptions of the attributes of the map in the game.

To learn the decision tree, the ID3 learning algorithm is used [2, 3]. In short and informally, the algorithm is as follows:

1. Determine the attribute with the highest information gain (which is explained below) on the training set and use this attribute as the root of the decision tree.
2. Create a branch for each of the values that the attribute can have.
3. For each branch, repeat the process with the subset of the training set that is classified by that branch, as long as there are unprocessed attributes remaining. Otherwise, the algorithm stops here.

The ID3 algorithm performs a simple-to-complex, hill-climbing search through the hypothesis space, which consists of all possible decision trees for the given attributes and their values. That is, the hypothesis space consists of all possible disjunctions of conjunctions of the attributes.

The algorithm maintains only a single current decision tree, performs no backtracking in its search and uses all training instances at each step of the search during the training process. Its evaluation function is the information gain, which is defined as

$$G(E,a) = I(E) - \sum_{v \in V_a} \frac{\mid E_{v,a} \mid}{\mid E \mid} I(E_{v,a}) \tag{1}$$

where E is the set of all training examples and a is an attribute from the set of all attributes (i.e., $a \in A$). V_a is the set of values corresponding to attribute a; that is, it is a set of values, such that

$$V_a = \{v \mid value(a,x) = v\} \tag{2}$$

for all $x \in E$, where $value(a,x)$ defines the value for attribute $a \in A$ of a specific example x.

Moreover, $E_{v,a}$ is a subset of E, such that

$$E_{v,a} = \{x \in E \mid value(a,x) = v\} \tag{3}$$

Finally, the function I is generally defined as

$$I(E) = \sum_{c \in C} -\frac{\mid E_c \mid}{\mid E \mid} \log_2 \frac{\mid E_c \mid}{\mid E \mid} \tag{4}$$

where C is the set of all possible classifications (i.e., the actions that the game AI should perform) and E_c is a subset of E with classification c; that is, it is a set of training examples, such that

$$E_c = \{x \in E \mid class(x) = c\} \tag{5}$$

for all $x \in E$, where $class(x)$ defines the classification of a specific example x.

The function I is also called the entropy, which measures the impurity of the set of all examples. In other words, information gain is the expected reduction in entropy caused by partitioning the instances according to a given attribute. Note that the algorithm is greatly simplified when working with Boolean classifications [4].

Finally, the algorithm has a preference for short trees, with the attributes with high information gain located near the root.

2.2 Total Annihilation Spring

Total Annihilation Spring is an open-source RTS game. Therefore, it is possible to customise every aspect of the game based on one's needs. The game engine has functions that determine whether parts of a map consist of

rivers, mountains, resources, et cetera. By making additions to the source code, it is possible to obtain data required to work with the decision tree in order to devise new strategies.

These strategies are used for as long as the game lasts. The game AI can play either defensively or offensively, depending on what it has observed from the map it is situated in. A list of general rules that we will use in the game is shown below. These rules deal with the structure of the map. The rules are based on the human logic; in other words, the game AI is told what human players would do if they were in a particular situation during the training stage. That is, the game AI should play like a human player. The decision tree will be trained based on these rules.

1. If there are metal resources in the vicinity and if they are close to each other (i.e., high density), then construct metal extractors near them as soon as possible. Otherwise place defensive units near the furthest metal resources and construct metal extractors near the closest ones first and then near the furthest ones.
2. If there are relatively narrow roads because of obstacles, such as mountains and rivers, that make entry into another area difficult, then place offensive units at these roads as soon as possible.
3. If there is a sufficient number of cliffs, then place artillery on top of them.
4. If the opponent's base is close by, then protect the units. If the number of metal resources is low as well, place offensive units at own base as soon as possible.

The rules imply playing either defensively or offensively and, depending on the situation, taking appropriate actions. To make the rules more specific for more different kinds of situations, we will use training data that consist of many different combinations that result in defensive or offensive stances of the game AI and the corresponding actions. Such specific rules are given in Table 6 (Appendix A), in which actions from the following list will be referred to:

1. Construct metal extractors at near metal resources.
2. Construct metal extractors at far away metal resources.
3. Place offensive units at relatively narrow roads.
4. Place offensive units at own base.
5. Place artillery on cliffs.
6. Protect operational metal extractors.
7. Protect artillery on cliffs.

A decision tree is constructed, which is based on the data from Table 6. The resulting decision tree will be used in the game, accompanied with the necessary alterations to the source code so that recognition of the attributes of the map is possible, on which the decision tree can be used to determine a strategy for defeating the opponent in the rest of the game.

There are two things to be noted about this decision tree with respect to the game. First, there cannot be too many values for each attribute, otherwise there would be an explosion in the number of nodes in the decision tree. That is why we use terms like "Few" and "Many" in Table 6, which divide the entire range of values of an attribute into bands [1]. These bands are determined by observing the map and computing the proportions of the attributes, relative to the size of the map. Second, classification occurs only when one requirement has been met: the bases of both players must be established first by constructing at least one particular building, namely the so-called K-Bot Laboratory. As the game AI is programmed to construct at least one K-Bot Laboratory, we therefore assume in our experiments that every base always has at least one such building. Classification by the decision tree is also relative to the current position of the game AI's base on the map. Therefore, results may vary if different starting locations are used at the beginning of a game.

3 Experiments

This section discusses the experiments that test our approach. The learned decision tree is described in Section 3.1. Section 3.2 shows how the experiments have been set up that test the performance of this decision tree. The results of the experiments can be found in Section 3.3, where these results are discussed as well.

3.1 Constructing the Decision Tree

The learned decision tree is (partially) shown in Figure 1 (Appendix B). It has been constructed from the data in Table 6 by applying the ID3 algorithm to the data. There are some things to be noted about the decision tree. First, it shows that the number of metal resources is the most important attribute of the map, which makes sense, because resources are needed in order to expand the base and build units before any actions can be taken. Second, if the number of resources is low, it considers the density of the resources as the next-to-most important attribute of the map. Otherwise, it is the least important attribute. Other attributes are more balanced in the decision tree. It is interesting to observe such extremities, but it can only be an intuitive thing, because the decision tree has been constructed with data that are based on what a human player would do under certain

circumstances. This way, the game AI will also behave like a human player with respect to the structure of the map, as intended from the beginning. That is why it is good to see that the decision tree turned out to be one that we have expected from the available data.

This decision tree is implemented into the game AI by constructing a connection between the classification of the decision tree, which is available as soon as there is a K-Bot Laboratory in all bases, and the game AI. However, due to the high complexity of the game's AI, modifying the game AI's actions in real-time based on the classification of the decision tree may not always be successful. It is not entirely known how some components of the game AI interact with each other.

3.2 Setup of the Experiments

In all of the experiments, there are only two players involved, namely the game AI playing against another computer-controlled player and the game AI playing against a human player. To test the performance of the decision tree, five maps will be used with attributes that the decision tree is supposed to classify. To compensate for variations in the results due to (un)fortunate events during one or more of the games, we will play both kinds of games mentioned earlier five times on every map. The performance is measured with "success" and "failure" and it is done by observing the actions of the game AI. We need to see whether actions suggested by the decision tree are actually executed. Since perfect environmental information is available, we can see everything that is happening on the map, therefore judgements are made more easily.

Moreover, during each game, statistical data about the decision tree will be collected automatically, so that we can see whether the attributes of the map are correctly classified. Then we consider every action that is suggested by the decision tree as a success if and only if these actions are also actually executed and completed. That is, it is not considered as a success when a suggested action is executed, but gets interrupted by something else. Thus, we will focus on the classification performance and the game AI's performance separately.

First, we begin by letting the game AI play against another computer-controlled player five times in all five of the maps. During the game we rate our game AI's actions by observing what it is doing and deciding whether its actions meet (or even exceed) our expectations. Statistics on the decision tree are also automatically collected at the same time.

Second, a human player will compete five times against the game AI in all five of the same maps. During these games, actions of the game AI are again observed. Because a human player can add variations in the gameplay, the game AI is expected not to repeat itself in its actions. In "normal" games, results may also vary, because each game is different by nature.

The following five maps have been used in the experiments: Speed Ball, Speed Ball Ring, Mountain Range, Small Supreme Battlefield and No Metal Speed Metal.[2] Pictures of these maps are available in Appendix C.

3.3 Results of the Experiments

Results of the experiments are shown and discussed here. In order to maintain a good overview, this sub-section has been divided into five parts that correspond to each map that has been used in the experiments. In each of the parts, we first describe the map and what attributes we want to test specifically, so that the results are better understood. Second, we describe what our expectations are from the map. Third, the results obtained from the games on the map are shown – first the subjective ones in the form of observations and descriptions, then the objective ones in the form of statistical tables – and, finally, these results will be discussed.

Map 1: Speed Ball

See Figure 2(a) for the map and Table 1 for the statistics.

Map Description

This map has many resources and the players are always close to each other, so that we can see whether the game AI does attempt to protect its own units. In the black area nothing can be constructed.

However, there is a possible disadvantage to this map: When two AI players compete against each other, they may become more offensive and attack each other early in the game. It is therefore expected that better results are achieved when a relatively idle human player is involved as the second player.

This map tests the following attributes in particular: "Number of Resources", "Resource Density" and "Enemy Distance". Action "1+2,4,6" is expected to be executed, which corresponds to example X37 from Table 6 (Appendix A).

Results

We observed that the players did not focus on gathering resources and building an army when the game AI played against a computer-controlled player. They became offensive early in the game and kept battling while constructing buildings and units, usually of the cavalry and artillery types, which assisted them in defeating their opponent. This occurred in each of the five times that they played in this map.

On the other hand, when a human player was involved in this game, depending on the actions of the human player, the game AI behaved differently. One time

[2] All of the maps have been downloaded from the internet at http://spring.unknown-files.net/category/13/Maps/.

the human player played offensively and successfully defeated the game AI, but in the meantime the game AI countered by also playing offensively. Another time the human player stayed in the background and remained relatively idle for a long period of time, only taking actions in order to defend himself. As a result, the game AI focused itself more on gathering resources before actually attacking its opponent. So, in some cases, the game AI's actions were recognized as a result from the decision tree. In other cases, it did not perform the actions as expected.

	Computer	**Human**
Classification	X37 (100%)	
Wins	3	0
Action 1 Success Rate	20%	60%
Action 2 Success Rate	20%	60%
Action 4 Success Rate	60%	80%
Action 6 Success Rate	0%	40%
Decision Tree Effectiveness	40.00%	68.00%

Table 1: Statistics from all games against a computer-controlled player and a human-controlled player on the map Speed Ball.

Discussion

An interesting observation is that the performance of the decision tree depends on what the other player is doing. When the human player stays in the background most of the time, the game AI tends to focus more on preparing itself for attack, rather than attacking immediately. However, when the game AI is playing against a computer-controlled player, both of them are not as merciful towards each other. This is likely to be caused by other subroutines of the game AI taking over, because not all subroutines of the game AI depend on the decision tree, due to the complexity of the game AI.

Statistics indicate that classification of the features of the map by the decision tree has been successful in all cases. All actions have been executed only a few times in the games. The only exception is action 6: it has not been executed at all in the game against the computer-controlled player, in spite of the decision tree indicating that it should do so. Again, this is likely to be caused by other subroutines of the game AI.

Map 2: Speed Ball Ring

See Figure 2(b) for the map and Table 2 for the statistics.

Map Description

This map has many similarities with the previous map. Instead of one big circle, however, this map is divided into eight equally sized circles and one larger circle in the middle. All of the circles are interconnected. The pathways that connect the circles are not very wide, so they count as "Narrow Roads". There are also plenty of resources available on this map. Since players are positioned randomly at the beginning of the game, the distance between the players can vary, and therefore also the results.

In this map we focus on all of the attributes other than "Cliffs". It is expected that actions "1+2,3" or "1+2,3,4,6" are executed, corresponding to the examples X16 and X40 respectively, from Table 6.

Results

When the game AI played against another computer-controlled player, the game AI performed action "1+2,3" in some of the times that it has played in this map, as expected. There has been only one occurrence of action "1+2,3,4,6", when the game AI was in the middle circle and decided to place offensive units at the entrance of the pathways connecting to the other circles, while the other player was on one of the outer circles.

The game AI also fared well against the human player with the same actions as expected, but the same issue from the previous map occurred here again: The player had to remain inactive for the most part in order to provoke the game AI to do things that were expected from it. Otherwise, more offensive subroutines of the game AI would take over and battles occurred early in the game. There were two occasions where the game AI was located in the middle of the map, resulting in a different action, namely "1+2,3,4,6".

	Computer	**Human**
Classification	X16 (80%) X40 (20%)	X16 (60%) X40 (40%)
Wins	4	0
Action 1 Success Rate	40%	60%
Action 2 Success Rate	40%	60%
Action 3 Success Rate	80%	100%
Action 4 Success Rate	20%	60%
Action 6 Success Rate	20%	40%
Decision Tree Effectiveness	50.00%	70.00%

Table 2: Statistics from all games against a computer-controlled player and a human-controlled player on the map Speed Ball Ring.

Discussion

The same situation from the previous map also occurred here. The game AI depended on what the other player was doing. The human player was sometimes easygoing and waited for the game AI to actually perform the actions that the decision tree had suggested as part of its classification. The computer-controlled player was far less forgiving, resulting in offensive stances from both parties.

Because starting positions of both players were random, classifications by the decision tree were also different each time we started a new game. This is in particular caused by the distance between both players, explaining the large difference in classifications.

Again, all actions were only executed a few times throughout all games. Only action 3 – placing offensive units at the narrow roads – was executed most often in both types of games. Considering the structure of the map, that was not too surprising. In most cases, the distance between the players remained fairly large, so that actions 4 and 6 were not executed as often.

Map 3: Mountain Range

See Figure 2(c) for the map and Table 3 for the statistics.

Map Description

This map has some notable attributes. First of all, there are not many resources – only enough in order to get a head start in the game. Second, the mountains are obstacles that obstruct the roads, making navigation relatively more difficult. Each location for the base on this map has its own advantages and disadvantages. Metal storages are recommended in this map, because of diminishing resources.

This is a relatively large map, actually intended for three-versus-three battle games, so the distance between the two players in a one-versus-one battle game will remain large for the most part. The focus of the decision tree lies in the "Number of Resources" – but more importantly "Resource Density" –, and "Narrow Roads" attributes. We expect that the game AI executes action "1,3", corresponding to example X4, from Table 6.

Results

The first thing that stood out is that the game AI actually payed attention to gathering nearby resources early in the game and blocking the passageways between the mountains and, if applicable, between the edges of the map and the mountains. Then it kept this up until it was powerful enough to attack. This happened in every game against a computer-controlled player as well as a human player, no matter how tough the opposition was.

Moreover, there have been occasions where the game AI also attempted to gather resources that were much further away – these resources are starting resources and they are generally placed far away from each other, so that every player has their own set of resources at the beginning of the game. Apparently, this corresponds to action 2 and this only happened when the game lasted long enough and the local resources were depleted, but this was not a result from the decision tree and therefore was not recorded in the statistics.

Discussion

The results exceed our expectations. Based on human judgement, the player would indeed attempt to gather metal resources and obstruct the narrow passageways by placing offensive units there. When the game AI had to decide, it performed these actions as expected, but it was surprising to see that it happened in all of the games. A possible explanation follows.

	Computer	**Human**
Classification	X4 (100%)	
Wins	4	0
Action 1 Success Rate	100%	100%
Action 3 Success Rate	100%	100%
Decision Tree Effectiveness	100.00%	100.00%

Table 3: Statistics from all games against a computer-controlled player and a human-controlled player on the map Mountain Range.

Due to the high density of metal resources at the starting positions, the game AI focused itself on gathering the resources in all games. The fact that the map is rather large is also responsible for this. This means that both of the players were far away from each other, so there was little influence from both sides. Because of this, other subroutines of the game AI did not have an influence on the outcome of the decision tree. Therefore, preparing for battle was the first priority, just as the decision tree suggested.

Map 4: Small Supreme Battlefield

See Figure 2(d) for the map and Table 4 for the statistics.

Map Description

What is noticeable about this map is that it has a large bottleneck in the middle of it. On both sides of that bottleneck, there is only water. On the other two sides of the map there are small areas of land. There are not many resources, and they are also available in water. This map is used in order to especially test the "Number of Resources", "Resource Density", "Narrow Roads" and "Enemy Distance" attributes of the decision tree.

We expect that the game AI executes action "1−2,3", corresponding to example X10, or "1−2,3,4,6", corresponding to example X34, from Table 6.

Results

In this map, the game AI seemed to perform quite well. Not all actions were properly executed, but all of them were recognizable. When it played against another computer-player, it had the preference to execute action "1−2,3", although it did not block the small road with offensive units in some cases, because the players can also travel through water and fight on – or even under – water. The computer-controlled player always built its base on land at the corner of the map, meaning that the distance between the two bases remained fairly large.

However, as the human player can perform unexpected actions, when he once built his base closer by the game AI's base, the game AI also attempted to protect its base and the metal extractors by using offensive units. Unfortunately for the game AI, this did not stop

the human player from winning all of the games on this map.

	Computer	Human
Classification	X10 (100%)	X10 (60%) X34 (40%)
Wins	3	0
Action 1 Success Rate	20%	40%
Action 2 Success Rate	40%	40%
Action 3 Success Rate	40%	40%
Action 4 Success Rate	N/A	60%
Action 6 Success Rate	N/A	40%
Decision Tree Effectiveness	50.00%	53.33%

Table 4: Statistics from all games against a computer-controlled player and a human-controlled player on the map Small Supreme Battlefield.

Discussion

The results almost meet our expectations. Although the classifications were correct according to our expectations, the statistics indicate that all actions have been performed only in a few games of both kinds. Our explanation for this phenomena is that it is – again – caused by other subroutines of the game AI. This is in particular because of the size of the map. It is not very big, and hence both players have an influence on each other. Despite of that, though, the classification was successful in all games.

Map 5: No Metal Speed Metal

See Figure 2(e) for the map and Table 5 for the statistics.

Map Description

This small map has particular properties that challenge the decision tree's ability to classify the attributes correctly. It is based on the original Speed Metal map, which is relatively popular amongst the players of the game.

The lack of metal resources makes it difficult to quickly produce units and expand the base and the decision tree does not have explicit information about lack of resources, although one could classify "no resources" as "few resources". The upper half and bottom half of the map are abysses where only some units can walk on, so we can say that there are cliffs that most units cannot trespass. The narrow roads also form a bottleneck for most units. Only in the middle part of the map, the area is large enough to fight properly, and, if the players' areas are weakly protected, battles in either one of the bases are possible in a short period of time. It can be said that this is a rather extraordinary map.

The following attributes are in particular tested: "Number of Resources", and indirectly "Resource Density", "Narrow Roads" and "Cliffs". It is expected that the game AI will choose action "1−2,3", corresponding to example X10, or "1−2,3,5", corresponding to example X11 and X12, from Table 6. Actions "1−2,3,4,6", corresponding to example X34, and "1−2,3,5,6,7", corresponding to examples X35 and X36, are also other possibilities. This is quite a number of expectations, but because it is not known beforehand whether we can consider the edges of the ground as cliffs or not, there are more possibilities than usual.

Results

The game AI hardly performed any actions based on the classification from the decision tree in all games against the computer-controlled player and the human player. Both players focused on building a force to oppose their opponent, although the game AI did attempt to place offensive units at the entrance to its own area.

Moreover, there was one instance where classification by the decision tree was different. The human player initially constructed the first bit of his base in the middle of the map and that resulted in a close distance between both players, hence also resulting in a different classification.

	Computer	Human
Classification	X10 (100%)	X10 (80%) X34 (20%)
Wins	0	0
Action 1 Success Rate	0%	0%
Action 2 Success Rate	0%	0%
Action 3 Success Rate	100%	100%
Action 4 Success Rate	N/A	0%
Action 6 Success Rate	N/A	0%
Decision Tree Effectiveness	50.00%	33.33%

Table 5: Statistics from all games against a computer-controlled player and a human-controlled player on the map No Metal Speed Metal.

Discussion

This is a rather surprising result. Despite the fact that classification by the decision tree was successful in all games, none of the actions were executed at all, with the exception of action 3. This is caused by the lack of metal resources. The decision tree does not take into account resources other than metal. Apparently, the edges were also not considered as cliffs.

This is also the first time where the decision tree's effectiveness is higher in the game against the computer-controlled player than in the game against the human player, most probably caused by the fact that the human player constructed the first bit of his base in the middle of the map. As a result, actions 4 and 6 were added to the list, but they did not get executed, most likely because of other subroutines of the game AI taking over since the distance between both players was small. As a result, the effectiveness score of the decision tree is also reduced.

4 Conclusion

This paper discussed our approach to establish a successful strategy for defeating the opponent(s) in an RTS game, depending on the structure of the map. The discussion of the results from the experiments will be summarized here. Following after, we will conclude this paper with a general conclusion and suggestions for improvements in a future research.

4.1 Summary of Discussion

As can be observed from the results, the performance of the decision tree applied in the game Total Annihilation Spring is fairly high. In all maps, the decision tree can correctly classify the attributes of the map that it has encountered in the game. Yet, some of the results were not as expected.

The variations in the results may be primarily caused by other subroutines of the game AI (e.g., the ones dealing with attacking the opponent) taking over from the subroutines dealing with the map structure. Attention to the structure of the map degrades when the game AI progresses in the game, because of the opponent's actions gaining more influence on the actions of the game AI. Thus, as the game progresses, the game AI continues to devise strategies in order to defeat the enemy, mostly basing its actions on what the enemy is doing. Therefore, it seems that this application of the decision tree is effective in early phases of the game that most concern planning, building bases and expanding them.

Interestingly enough, the "Cliffs" attribute was never used. This raises the question whether the cliffs are detected properly, which requires more investigation or perhaps more maps that are suitable for testing.

4.2 General Conclusion

We have seen a successful application of the decision tree in the game, in the sense that it can classify the attributes of the map structures correctly. However, depending on the attributes of the map and the actions of the opponent, results from this classification still vary. When a human player is involved in the game, the game AI is not able to gain a major advantage from the structure of the map, because the human player typically plays better in this RTS game than the game AI does. That explains why the game AI always loses against the human player in our experiments. However, against a computer-controlled player, the game AI is still able to win. Thus, the use of the structure of the map with the help of a decision tree contributes well to establishing strategies for defeating the enemy, but these strategies are unfortunately not always actually used due to external influences.

4.3 Suggestions for Improvements

To improve the performance of the decision tree, the decision tree can be more refined by training it with more specific training examples (for example, by taking into account that maps may not contain any resources at all or by using more different attributes or different actions).

Moreover, the decision tree appears to be only used at the beginning of the game and attention to the structure of the map degrades as the game progresses. Ways to prevent that from happening could be implemented, such as making components of the game AI – such as ones that react directly on the actions of the opponents through opponent modeling – more dependent on the decision tree than they are in the current situation.

Also, we had to assume that perfect information about the maps is available. Normally, an RTS game does not provide such information by making use of the so-called "Fog of War", which hides unexplored parts of the maps. Therefore, in a future research, our approach can be extended in order to cope with imperfect information.

Lastly, the classifications by the decision tree are dependent on what the other player is doing, because the "Enemy Distance" attribute (i.e., the distance to the opponent's base) is dependent on that. That way, we can make better use of the map. For instance, if the enemy is close by, the game AI has different priorities with respect to the structure of the map than when the enemy is far away. By adding more of such attributes directly relating to the behaviour of the player, we can possibly make the decision tree more accurate and effective.

References

[1] Evans, R. (2002). Varieties of Learning. *AI Game Programming Wisdom* (ed. S. Rabin), pp. 571–575. Charles River Media, 20 Downer Avenue, Suite 3, Hingham, Massachusetts 02043, United States.

[2] Fu, D. and Houlette, R. (2004). Constructing a Decision Tree Based on Past Experience. *AI Game Programming Wisdom 2* (ed. S. Rabin), pp. 567–577. Charles River Media, 20 Downer Avenue, Suite 3, Hingham, Massachusetts 02043, United States.

[3] Mitchell, T. (1997). *Machine Learning*, Chapter 3: Decision Tree Learning, pp. 52–80. McGraw-Hill, 2 Penn Plaza, New York 10121-0101, United States.

[4] Russel, S. and Norvig, P. (2003). *Artificial Intelligence*, pp. 653–664. Prentice Hall, Upper Saddle River, New Jersey 07458, United States, second edition.

A Training Examples

Example	Attributes					Action(s)
	RN	RD	NR	CL	DE	
X1	Few	High	No	None	Far	1
X2	Few	High	No	Few	Far	1
X3	Few	High	No	Many	Far	1,5
X4	Few	High	Yes	None	Far	1,3
X5	Few	High	Yes	Few	Far	1,3
X6	Few	High	Yes	Many	Far	1,3,5
X7	Few	Low	No	None	Far	1−2
X8	Few	Low	No	Few	Far	1−2
X9	Few	Low	No	Many	Far	1−2,5
X10	Few	Low	Yes	None	Far	1−2,3
X11	Few	Low	Yes	Few	Far	1−2,3,5
X12	Few	Low	Yes	Many	Far	1−2,3,5
X13	Many	High	No	None	Far	1+2
X14	Many	High	No	Few	Far	1+2
X15	Many	High	No	Many	Far	1+2,5
X16	Many	High	Yes	None	Far	1+2,3
X17	Many	High	Yes	Few	Far	1+2,3
X18	Many	High	Yes	Many	Far	1+2,3,5
X19	Many	Low	No	None	Far	1+2
X20	Many	Low	No	Few	Far	1+2
X21	Many	Low	No	Many	Far	1+2,5
X22	Many	Low	Yes	None	Far	1+2,3
X23	Many	Low	Yes	Few	Far	1+2,3,5
X24	Many	Low	Yes	Many	Far	1+2,3,5
X25	Few	High	No	None	Close	1,4,6
X26	Few	High	No	Few	Close	1,4,6
X27	Few	High	No	Many	Close	1,5,6,7
X28	Few	High	Yes	None	Close	1,3,4,6
X29	Few	High	Yes	Few	Close	1,3,4,6
X30	Few	High	Yes	Many	Close	1,3,5,6,7
X31	Few	Low	No	None	Close	1−2,4,6
X32	Few	Low	No	Few	Close	1−2,4,6
X33	Few	Low	No	Many	Close	1−2,5,6,7
X34	Few	Low	Yes	None	Close	1−2,3,4,6
X35	Few	Low	Yes	Few	Close	1−2,3,5,6,7
X36	Few	Low	Yes	Many	Close	1−2,3,5,6,7
X37	Many	High	No	None	Close	1+2,4,6
X38	Many	High	No	Few	Close	1+2,4,6
X39	Many	High	No	Many	Close	1+2,5,6,7
X40	Many	High	Yes	None	Close	1+2,3,4,6
X41	Many	High	Yes	Few	Close	1+2,3,4,6
X42	Many	High	Yes	Many	Close	1+2,3,5,6,7
X43	Many	Low	No	None	Close	1+2,4,6
X44	Many	Low	No	Few	Close	1+2,4,6
X45	Many	Low	No	Many	Close	1+2,5,6,7
X46	Many	Low	Yes	None	Close	1+2,3,4,6
X47	Many	Low	Yes	Few	Close	1+2,3,5,6,7
X48	Many	Low	Yes	May	Close	1+2,3,5,6,7

Table 6: Examples for the real-time strategy game domain with respect to the structure of the map. Legend: "−" means first executing the action before the dash sign, then the action after the dash sign. "+" means executing the actions on both sides of the plus sign simultaneously. "RN" means number of metal resources. "RD" means resource density. "NR" means presence of relatively narrow roads due to obstacles such as mountains, rivers, et cetera. "CL" means number of cliffs. "DE" means distance to enemy's base. The distance between two bases is measured as the distance between the K-Bot Laboratories of the bases and it is only calculated when they are actually constructed, which the game AI is programmed to do. Finally, because maps vary in size, there are no fixed numbers for the numerical attributes; they are found by computing the proportions of the attributes, relative to the size of the map.

B Decision Tree

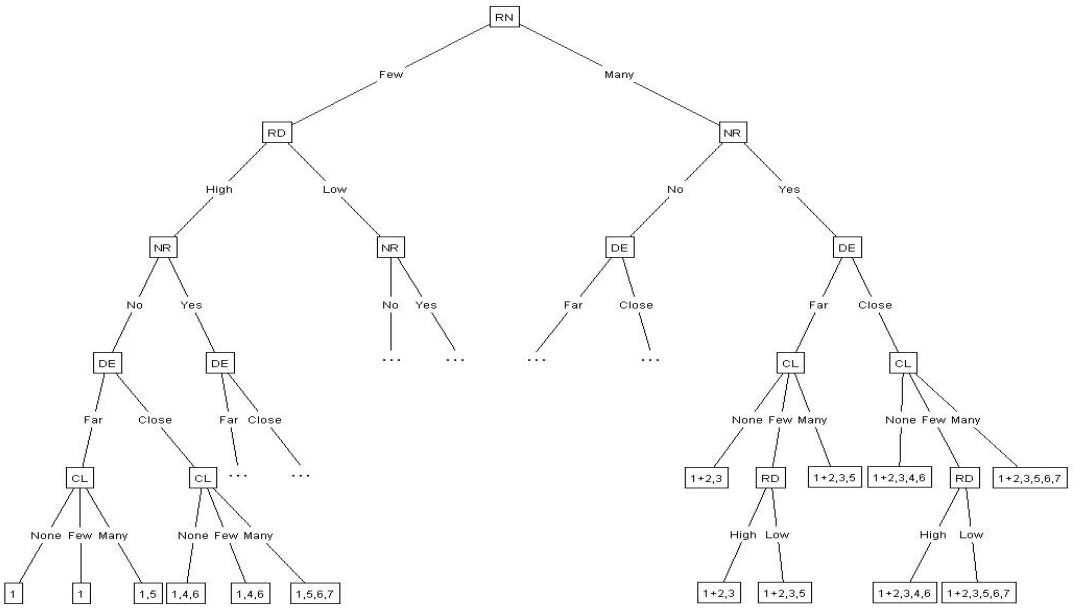
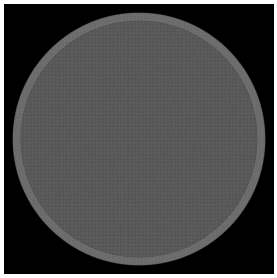
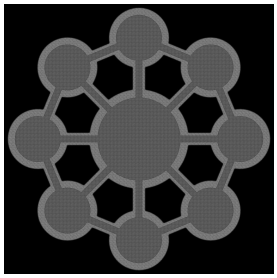


Figure 1: The result of the training process in the form of a decision tree. Only a portion of the tree is shown. Legend: "RN" means number of metal resources. "RD" means resource density. "NR" means presence of relatively narrow roads due to obstacles such as mountains, rivers, et cetera. "CL" means number of cliffs. "DE" means distance to enemy's base.

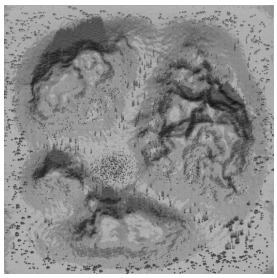
C Maps



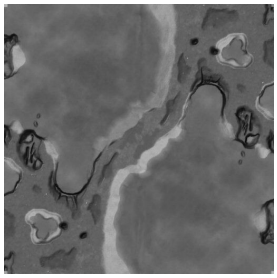
(a) Speed Ball



(b) Speed Ball Ring



(c) Mountain Range



(d) Small Supreme Battlefield

(e) No Metal Speed Metal

Figure 2: The five maps that have been used in the experiments.