MAP-ADAPTIVE ARTIFICIAL INTELLIGENCE FOR VIDEO GAMES

Laurens van der Blom, Sander Bakkes and Pieter Spronck
Universiteit Maastricht
MICC-IKAT
P.O. Box 616
NL-6200 MD Maastricht
The Netherlands
e-mail: l.vanderblom@student.unimaas.nl, {s.bakkes,p.spronck}@micc.unimaas.nl

ABSTRACT

This paper proposes an approach to automatically adapt game AI to the environment of the game (i.e., the so-called map). In the approach, a particular map is first analysed for specific features. Subsequently, an automatically established decision tree is applied to adapt the game AI according to the features of the map. Experiments that test our approach are performed in the RTS game SPRING. From our results we may conclude that the approach can be used to automatically establish effective strategies dependent on the map of a game.

INTRODUCTION

Throughout the years, video games have become increasingly realistic with regard to visual and auditory presentation. However, artificial intelligence (AI) in games has not yet reached a high degree of realism. Typically, game AI is based on non-adaptive techniques [9], which prevents it to adequately adapt to changing circumstances. Adaptive game AI , on the other hand, has been explored with some success in previous research [2, 5, 8].

An important component of adaptive game AI is the ability to automatically establish effective behaviour dependent on features of the game environment (i.e., the so-called map). This is called 'map-adaptive game AI'. In this paper we will investigate how to analyse and exploit features of the game environment for the purpose of establishing effective game strategies.

The outline of the paper is as follows. We will first present our approach to establish map-adaptive game AI. Subsequently, the experiments that test our approach are discussed. The experimental results are discussed next. Finally, we provide conclusions and describe future work.

APPROACH

Our approach to establish map-adaptive game AI consists of three components: (1) definition of the features of a map, (2) determination of compound actions of map-adaptive strategies, and (3) automatic construction of a decision tree of map-adaptive strategies.

We establish map-adaptive game AI in an RTS game environment, i.e., a simulated war game. Here, a player needs to gather resources for the construction of units and buildings. The goal of the game is to defeat an enemy army in a real-time battle. We use RTS games for their highly challenging nature, which stems from three factors: (1) their high complexity, (2) the large amount of inherent uncertainty, and (3) the need for rapid decision making [1]. In the present research we use SPRING, illustrated in Figure 1, which is a typical and open-source RTS game. A SPRING game is won by the player who first destroys the opponent's 'Commander' unit.

Features of a Map

To automatically establish map-adaptive game AI, we start by defining a basic set of features that will play an essential role in the strategy of a player. For our experiments, we decided to use the following five features of a map.

1. RN: Number of metal resources.

Figure 1: Screenshot of the SPRING game environment. In the screenshot, the artillery unit on the left attacks a target across the river.

2. RD: Resource density.
3. NR: Presence of relatively narrow roads (e.g. due to obstacles such as mountains and rivers).
4. CL: Number of cliffs.
5. DE: Distance between base locations.

Feature values will be used to automatically construct a decision tree of map-adaptive strategies. If we would allow all possible feature values, an explosion in the number of nodes in the decision tree would occur. We therefore divide the range of feature values in bands [3], such as 'None', 'Few' and 'Many'. Naturally, game maps can vary in size. Therefore, feature values are scaled proportionally to the size of the map.

Map-Adaptive Game AI Actions

After analysis of features of a map, game AI is established on compound actions of map-adaptive strategies. For our experiments, we decided to use the following seven compound actions.

1. Construction of metal extractors at near metal resources.
2. Construction of metal extractors at far away metal resources.
3. Placement of offensive units at relatively narrow roads.
4. Placement of offensive units at own base.
5. Placement of artillery on cliffs.
6. Protection of operational metal extractors.
7. Protection of artillery on cliffs.

The defined actions can be used to establish offensive as well as defensive stances of game AI.

Decision Tree of Map-Adaptive Strategies

A decision tree is a tree where each internal node analyses a feature, each branch corresponds to a band of feature values, and each leaf node assigns a classification. Since we are dealing with discrete feature values, the decision tree is called a 'classification tree'. The leaves of such a tree represent classifications and the branches represent conjunctions of features that lead to those classifications. Classification trees also enable disjunctive descriptions of the features of the map.
In the SPRING game, each feature of the map can be expressed by discrete values (e.g., the number of resources and the resource density). For constructing a decision tree for the SPRING game, we employ the ID3 learning algorithm [4, 6].

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 features and their values. That is, the hypothesis space consists of all possible disjunctions of conjunctions of the features. The algorithm maintains only a single 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 features A. V_a is the set of values corresponding to feature 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 feature $a \in A$ of a specific example x. Moreover, $E_{v,a}$ is a subset of E, such that

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

The function I is 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. This implies that the learning algorithm has a preference for short trees, with the features with a high information gain located near the root of the tree.

EXPERIMENTS

This section discusses the experiments that test our approach. We first describe the process of constructing the decision tree and then experimental setup.

Constructing the Decision Tree

We use the ID3 learning algorithm to construct the decision tree from experimentally determined training data,

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 training data is given in Table 2 (Appendix A). The learned decision tree is displayed in Figure 2 (Appendix B). We give two observations on the constructed decision tree.

First, the feature 'number of metal resources' is placed at the root of the tree. This implies that the number of metal resources is the most important feature when analysing a map. This result fits our expectation, for metal resources are vital to expanding the base and building the first units.

Second, if the number of metal resources is low, the constructed decision tree considers the 'resource density' as the next-most important feature of the map. If the number of metal resources is high, however, the resource density is the least important feature.

Experimental Setup

To test our approach, the map-adaptive game AI will be pitted in a game against the same game AI without map-adaptive capability. We found one game AI which was open source, which we labeled 'AAI' [7]. We enhanced this AI with the capability to adapt its behaviour dependent on the features of the map.

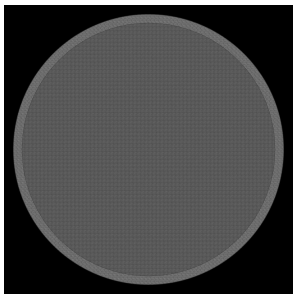
The map-adaptive game AI is tested in real-time games, which took place on five maps that are described in the next section, namely (1) Speed Ball, (2) Speed Ball Ring 8-way, (3) Mountain Range, (4) Small Supreme Battlefield v2, and (5) No Metal Speed Metal. Each experimental trial is repeated five times.

In addition to pitting the map-adaptive game AI against a computer opponent, our approach was also tested against a human opponent. Our expectation was that when pitting it against a superior human opponent, different behaviour will be evoked from the map-adaptive game AI.

RESULTS

In this section we provide a detailed discussion of the obtained results. For each map, we present the following three items: (1) the characteristics of the map, (2) the obtained results, and (3) a discussion of the obtained results.

Map 1: Speed Ball



The Speed Ball map has many resources and the players are always relatively close to each other. This allows us to determine whether the map-adaptive game AI attempts to protect its own units. In the black area nothing can be constructed. 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 different AI behaviour will be observed when a relatively idle human player is involved as the second player.

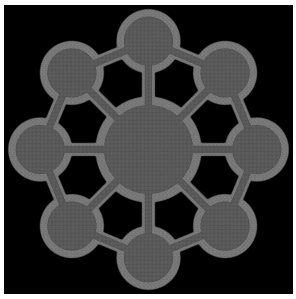
Gameplay on this map is focussed on effectively responding to the following features of the map: 'Number of Resources', 'Resource Density' and 'Distance between base locations'. Action '1+2,4,6' is expected to be executed, which corresponds to example X37 from Table 2 (Appendix A).

Obtained Results

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

When a human player was involved, the map-adaptive game AI behaved differently. One time 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 more on gathering resources before actually attacking its opponent.

Map 2: Speed Ball Ring 8-way



The Speed Ball Ring 8-way map has many similarities with the Speed Ball 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, which implies that they are considered as 'Narrow Roads'. There is a relatively large number of resources available on this map. Players are positioned randomly at the beginning of the game, thus the distance between the players can vary.

In this map we focus on all of the features other than 'Number of cliffs'. It is expected that actions '1+2,3' or '1+2,3,4,6' will be executed, corresponding to the examples X16 and X40 respectively, from Table 2.

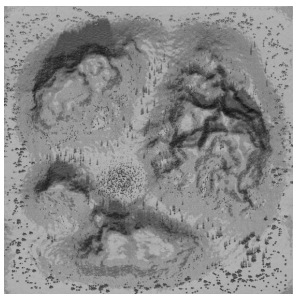
Obtained Results

When the map-adaptive game AI played against another computer-controlled player, the game AI often performed action '1+2,3', as expected. There was only one occurrence of action '1+2,3,4,6' being performed, 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 map-adaptive game AI fared well against the human player with the same actions as expected, but again the human player had to remain inactive for the most part in order to provoke the game AI to exhibit behaviour that was expected of 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'.

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 caused in particular by the distance between both players, explaining the large difference in classifications.

Map 3: Mountain Range



The Mountain Range map does not have many metal resources. Additionally, the mountains are obstacles that obstruct the roads, making navigation relatively difficult. Each location for constructing a base on this map has its own advantages and disadvantages. The use of metal storages is recommended on this map, because of diminishing metal resources.

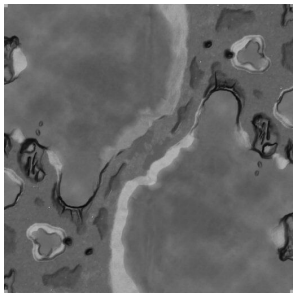
The map is relatively large. Therefore, the distance between the base of two players will typically remain large in a one-versus-one battle, as is the case here. The focus of the decision tree lies on the features 'Number of Metal Resources', 'Resource Density', and 'Narrow Roads'. We expected that the game AI would execute action '1,3', corresponding to example X4 from Table 2.

Obtained Results

We observed that on this map the map-adaptive game AI was strongly focussed on gathering nearby metal resources early in the game. Additionally, the game AI was blocking the passageways between the mountains and, if applicable, between the edges of the map and the mountains. The game AI protected the expanded position, and launched an attack on the opponent when it constructed a relatively large army. This strategy was consistently observed against both the computer-controlled as well as the human opponent.

We observed that early in the game, relatively few attacks would take place. This is caused by the relatively large distance between the base of each player. As lower-level subroutines will not be called for handling an attack, the map-adaptive game AI can focus on its first priority: establishing and expanding a strong base. Thus, we observed that in all cases action '1,3' was executed, as expected.

Map 4: Small Supreme Battlefield v2



The Small Supreme Battlefield v2 map contains a long bottleneck in the centre of the map. On each side of the map there is a large area of water. On the other two sides of the map there are small areas of land. The map has relatively few metal resources, some of which are available in the water areas. We expected that the game AI would execute action '1−2,3', corresponding to example X10, or '1−2,3,4,6', corresponding to example X34, from Table 2.

Obtained Results

When in competition against the computer-controlled opponent, the map-adaptive game AI had a preference for executing action '1−2,3'. In most of the cases the game AI blocked the bottleneck with offensive units. In other cases the focus of the game was not so much on the bottleneck, but more on the water area, on which the battle continued by sea units and submarine units. The computer-controlled player always built its base on land located in the corner of the map, which implied that the distance between the bases remained fairly large.

Identical behaviour was observed when in competition against a human player. On one occasion the human player constructed the base nearby the base of the map-adaptive game AI. This led the AI to increasingly protect its base and the metal extractors by use of offensive units.

Map 5: No Metal Speed Metal

The No Metal Speed Metal map is based on the Speed Metal map. As the name implies, however, this particular map does not contain any metal resources. The lack of metal resources makes it difficult to quickly produce units and expand the base. A challenge for the established map-adaptive game AI was that it was not trained under circumstances where no metal resources were present on the map.

It was expected that the game AI would choose action '1−2,3', corresponding to example X10, or '1−2,3,5', corresponding to example X11 and X12, from Table 2. Actions '1−2,3,4,6', corresponding to example X34, and '1−2,3,5,6,7', corresponding to examples X35 and X36, were considered alternative possibilities.

Obtained Results

In competition against the computer-controlled game AI, both players focused on constructing offensive units to oppose their opponent. In addition, the map-adaptive game AI utilised units for protecting the entrance of its own area. Similar behaviour was observed when competing against the human player. In one case, the human player constructed the base in the centre of the map, near the map-adaptive game AI. This led to a different classifaction, and thus different behaviour from the map-adaptive game AI.

Though the map-adaptive game AI was not trained for circumstances where no metal resources are present, it was able to utilise the learned decision tree by traversing the node for 'few metal resources'. However, it did not exhibit behaviour suitable for defeating the computer-controlled player.

A summary of the experimental results is provided in Table 1.

DISCUSSION

Our approach to map-adaptive game AI should not be confused with machine-learning techniques that allow the game AI to adapt to novel situations. Rather, in our approach, we implemented the map-adaptive game AI as a high-level planner of strategic actions. We allowed low-level actions, such as handling an imminent threat of the opponent, to interfere with the established high-level plan.

In a typical RTS game, early phases of the game are focussed on planning the construction and expansion of the base. Later phases of the game are typically focussed

	Computer	**Human**
Map 1		
Speed Ball		
Classification	X37 (100%)	X37 (100%)
Win:Loss	3:2	0:5
Map 2		
Speed Ball Ring 8-way		
Classification	X16 (80%) X40 (20%)	X16 (60%) X40 (40%)
Win:Loss	4:1	0:5
Map 3		
Mountain Range		
Classification	X4 (100%)	X4 (100%)
Win:Loss	4:1	0:5
Map 4		
Small Supreme Battlefield v2		
Classification	X10 (100%)	X10 (60%) X34 (40%)
Win:Loss	3:2	0:5
Map 5		
No Metal Speed Metal		
Classification	X10 (100%)	X10 (80%) X34 (20%)
Win:Loss	0:5	0:5

Table 1: Classifications of the map-adaptive game AI in competition against the computer-controlled player and against the human player.

on engaging in offensive or defensive actions. However, if an opponent would decide to attack relatively early, a player would be forced to abandon the established plan and focus on combat. Therefore, our implementation of map-adaptive game AI as a high-level planner of strategic actions, is particularly suitable for RTS games.
For games from other genres, our implementation of map-adaptive game AI may not necessarily be the most suitable one. Game developers should consider that to apply our approach in practice, a balance should be found between pursuing a high-level map-adaptive plan, and allowing the game AI to respond to low-level actions.

CONCLUSIONS AND FUTURE WORK

In this paper we proposed an approach for establishing map-adaptive game AI. In our approach the game environment, in the form a so-called map, is first analysed for specific features. Subsequently, a decision tree of map-adaptive strategies is constructed automatically. Experiments to test our approach were performed in the RTS game Spring. Our experimental results show that against a computer-controlled opponent, the map-adaptive game AI consistently constructed effective game strategies. The map-adaptive game AI was outperformed by a human opponent. However, observations showed that the map-adaptive game AI responded to strong play of the human player by adapting its own strategy. From these results, we may conclude that the established map-adaptive game AI can be successfully used to construct effective strategies in RTS games.
In future work, we will incorporate a self-adaptive mechanism to enable the game AI to automatically refine the constructed decision tree. This mechanism will be particularly suitable for games where effective map-adaptive game AI should be established online, on the basis of relatively little training data.

ACKNOWLEDGEMENTS

We extend our gratitude to the anonymous reviewers for their insightful feedback.
This research was funded by a grant from the Netherlands Organization for Scientific Research (NWO grant No 612.066.406).

REFERENCES

[1] Michael Buro and Timothy Furtak. RTS games and real-time AI research. In *Proceedings of the Behavior Representation in Modeling and Simulation Conference (BRIMS)*, 2004.

[2] Pedro Demasi and Adriano Cruz. Online coevolution for action games. *International Journal of Intelligent Games and Simulation*, 2(3):80–88, 2002.

[3] Roger Evans. Varieties of Learning. In S. Rabin, editor, *AI Game Programming Wisdom*, pages 571–575. Charles River Media, 20 Downer Avenue, Suite 3, Hingham, Massachusetts 02043, United States, 2002.

[4] Daniel Fu and Ryan Houlette. Constructing a Decision Tree Based on Past Experience. In S. Rabin, editor, *AI Game Programming Wisdom 2*, pages 567–577. Charles River Media, 20 Downer Avenue, Suite 3, Hingham, Massachusetts 02043, United States, 2004.

[5] Thore Graepel, Ralf Herbrich, and Julian Gold. Learning to fight. In Quasim Mehdi, Norman Gough, and David Al-Dabass, editors, *Proceedings of Computer Games: Artificial Intelligence, Design and Education (CGAIDE 2004)*.

[6] Tom Mitchell. *Machine Learning*, chapter 3: Decision Tree Learning, pages 52–80. McGraw-Hill, 2 Penn Plaza, New York 10121-0101, United States, 1997.

[7] Alexander Seizinger. AI:AAI. Creator of the game AI 'AAI', http://spring.clan-sy.com/wiki/AI:AAI, 2006.

[8] Pieter Spronck, Ida Sprinkhuizen-Kuyper, and Eric Postma. Online adaptation of game opponent AI with dynamic scripting. *International Journal of Intelligent Games and Simulation*, 3(1):45–53, 2004.

[9] Paul Tozour. *AI Game Programming Wisdom (ed. Rabin, S.)*, chapter The Perils of AI Scripting, pages 541–547. Charles River Media, 2002. ISBN: 1-58450-077-8.

A - TRAINING DATA

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 2: Training data of the SPRING real-time strategy game with respect to features 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 (e.g. due to obstacles such as mountains and rivers). "CL" means number of cliffs. "DE" means distance between base locations.

B - DECISION TREE FOR MAP-ADAPTIVE GAME AI

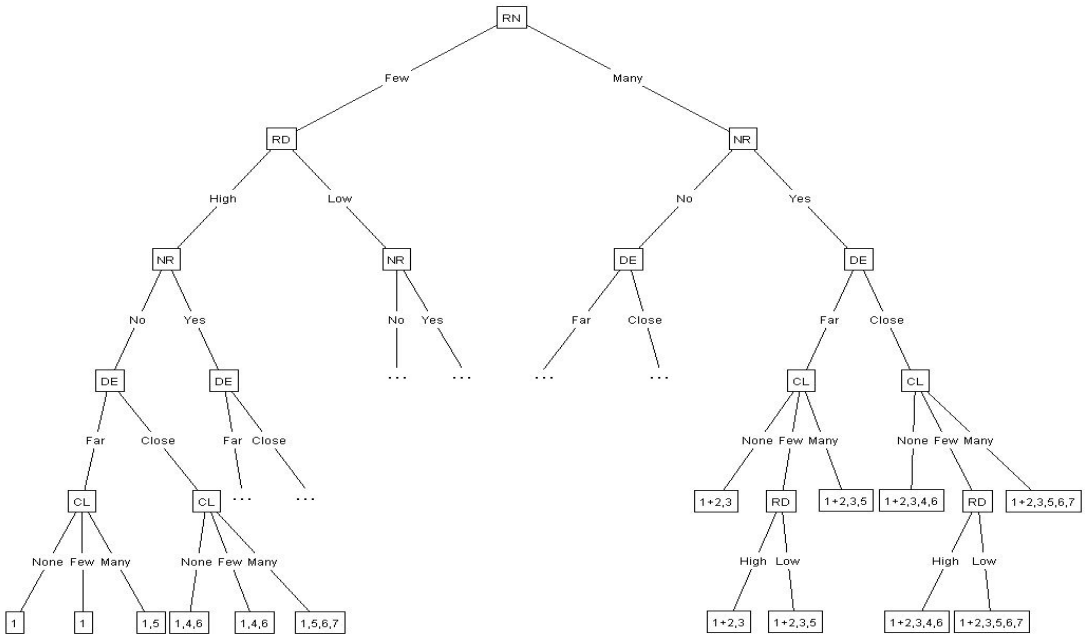


Figure 2: The automatically constructed decision tree. Only a portion of the tree is shown. Legend: "RN" means number of metal resources. "RD" means resource density. "NR" means presence of relatively narrow roads (e.g. due to obstacles such as mountains and rivers). "CL" means number of cliffs. "DE" means distance between base locations.