Myrmidia – Case-based reasoning for Warhammer fantasy battle army building

Glenn Rune Strandbråten and Anders Kofod-Petersen

Department of Computer and Information Science,
Norwegian University of Science and Technology,
7491 Trondheim, Norway
glennrune@gmail.com,anderpe@idi.ntnu.no

Abstract. Constructing a viable army for Warhammer Fantasy Battle is a very complicated problem that entrails choosing between many races, many different units and equipment. In many ways, the construction of a viable army is a constraint satisfaction problem. There are many hard constraints based on the game mechanics and many soft constraints, such as the number of miniatures available, the capabilities of the enemy and the style of playing; and the fact that any choice made influences all the other choices possible. The work presented here presents an initial case-based reasoning implementation in jColibri for construction armies in the Warhammer domain.

Key words: case-based reasoning, constraint satisfaction, strategy games

1 Introduction

Constructing a good army for Warhammer Fantasy Battle (WHFB) is a very complicated constraint satisfaction problem. Choosing a successful army is naturally not the only requirement for actually winning a battle; mastery of that army and tactics are obviously also important. As warhammer includes dices and different strengths and weaknesses of the troops some level of probability and statistics do play a role. However, a clever general can easily shame the math. The first step is showing up with a viable army. Choosing an army has many constraint, not only what is legal. Constraints might include available miniatures, tactical consideration, properties of the opponent, or playing style, such as infantry heavy, fondness of cavalry, or being partial to artillery or magic.

The work presented here is a decision support system for constructing viable armies that the player can bring to the table by applies case-based reasoning (CBR). That is, the decision support system is only involved in constructing the roster all actual game playing is in the hands on the player. The application developed uses the relevant domain knowledge from the game together with cases describing actual battle fought to suggest an army to a player, using his preferences and constraints. Applying CBR takes the army constructing problem beyond probabilities and statistics, which are used solely in (commercial) available solutions. We argue that applying a lazy learner instead of rule induction captures the many different constraints that can occur.

The rest of the paper is organise as follow: Section 2 gives a short introduction to the Warhammer Fantasy Battle game mechanics; Section 3 gives a short overview of related work on CBR and constraint satisfaction problems; Section 4 shortly describes the domain model used; Section 5 describes how the four steps in case-based reasoning is used, in particular how similarity is calculated; Section 6 shortly describes the use of jColibri; Section 7 describes the evaluation; Finally, the paper ends with a summary and outlook on future work.

2 Warhammer Fantasy Battle

Warhammer Fantasy Battle is a turn-based miniature figure board game, where each player leads an army to battle against another player There are 15 different races to choose from and each race have many different units that can be used. The game is played using miniatures made by plastic or pewter on a 25mm scale and each units sits upon a square or rectangular base.

Strict rules govern the game play and army creation process. On the battlefield the individual units strength and abilities are matched against the opponent, and in conjunction with the lay of the land and the roll of the dice determine the winner. This essentially mean that the same two armies can do battle several times and the results will most likely be different each time.

Each of the 15 different races have a host of different units that can be bought and added to the players collection; additionally one unit may have different equipment from another unit of the same race and type. Classification determines what type of unit it is, and is divided into many categories, such as: infantry, monsters, cavalry and war machines.

Additionally some units are either a lord or a hero, these are powerful units which leads the army to war. They are not separated in their own classification, but can belong to any classification (usually infantry with the ability to purchase a mount). Lords are the most powerful unit in an army, with fearsome martial- or magical might. Heroes are lesser in might than lords, but still worth a score of ordinary warriors.

Table 1. Dwarf lord and dragon characteristics

Unit	Movement	Weapon skill	Ballistic skill	Strength	Toughness	Wounds	Initiative	Attacks	Leadership	Type
Dwarf lord	3	7	4	4	5	3	I	4	10	In
Dragon	6	7	0	7	6	7	2	6	9	Mo

Characteristics describe all the aspects of a unit through nine associated attributes. Each attribute have a numeric value in the range from 0 to 10, all these attributes are uniquely described in a table for each unit and associated classification. Attributes includes: how many attacks a unit has, how fast it

can move, how courageous they are and how much damage they can sustain. Table 1 exemplifies the difference between a dwarf lord and a dragon (higher is normally better). Each individual unit have a base cost associated with it, this cost represents how good the unit is. As the player adds better equipment, e.g.: weapons and armour to the unit, the cost increases.

When preparing for a battle the two players determine the total army points for each player to be used during the game. Common values are between 1000 and 4000, where 2500 point is the de facto standard. To create the army the player selects the units she wants to use for the battle while using as many points possible, but still under the total allowed points. As it is difficult to use exactly all the points, both armies are usually of a different value, but as close to the limit as possible.

There are however a few rules that must be followed when creating the army, you cannot choose ten of the meanest unit there is and be done with it. The player should first select a General, which represents the player and leads the army on the battlefield. The rest of the army is selected based on rules governing the following unit categories: Lords, Heroes, Core Units, Special Units and Rare Units. All equipments and/or mounts purchased to individual units or groups, counts towards the total point usage within each category (see Table 2).

Table 2. Rules for points usage when constructing an army

The army must consist of at least three units, and one lord or hero to be the general		
	Points limit	**Duplication choices**
Lords	Up to 25 %	No limits
Heroes	Up to 25 %	No limits
Core	25 % or more	No limits
Special	Up to 50 %	Up to 3 (6 if you use 3000 or more army points)
Rare	Up to 25 %	Up to 2 (4 if you use 3000 or more army points)

Lords and heroes are powerful leaders of armies, thus only 25 % of the agreed upon points can be spend on each. Core units are the basic units of your army and the ones you will have the most off, e.g.: basic footmen and cavalry. You must use a minimum of 25 % of the army points on core units. Special units are elite troops which perform great on their own or, as motivators for lesser soldiers. You can spend as much as 50 % of the points on these units. Rare units are very powerful units, e.g. monsters, weird war machines and elite solders of unsurpassed skill. You can use up to 25 % of the points on these units.

3 Related Work

At the time of writing the authors are not aware of any research applying case-based reasoning to army build in Warhammer, or in any other strategy game.

However, the problem of constructing a viable army is not disjoint from any other research area. In many ways, this problem resembles a *constraint satisfaction problem* (CSP). There are different constraints on the number of different type of units that can be used (see Table 2) and the different ways of building each unit (e.g. equipment and special troop types). All of these constraints must be satisfied, whilst still building an army that can beat the opponent.

I can be argued that the adaptation phase in case-based reasoning resembles a constraint satisfaction problem in general, and dynamic constraints satisfaction in particular [1]. They both share the same idea of not solving problems from scratch as this is inefficient. Most research on case-based reasoning in the context of constraint satisfaction has been been directed at combining CBR and CSP in the adaptation phase. This includes configuration design and assembly sequence generation for assembly of motors [2], holiday scheduling [3], timetable scheduling [4] and process design [5]. In the latter, the idea is to retrieve relevant existing designs (of compression stations) and adapt them in cooperation with a domain expert to improve the design. The case-based reasoner would retrieve existing case that would satisfy a set of mandatory constrains. From this case a set of parameters covering the target constraints would be selected and solved by a CSP solver.

Others have investigated the use of case-based reasoning as the main tool for solving constraints. This includes manufacturing, where for loading autoclaves [6] is a well known case and metal casting [7]. Finally, the problem of building a viable army resembles in particular a rostering problem,where different people (troops) with different capabilities are required at a specific time and place. This has been approach by e.g. Beddoe and Petrovic [8] by using case-based reasoning in conjunction with tabu search.

The approach presented here draws inspiration from applying case-based reasoning to constraint satisfaction problems. However, contrary to most related work we do not include specific CSP solvers.

4 Domain model

The domain model developed consists of two parts: the official rules for warhammer, both the core rules and the specific rules for the races; and date from real battles fought and recorded by Trondheim Warhammer Club (WarTrond). The data from the official rules constitutes the general domain knowledge, whereas the data from WarTrond constitutes the cases.

The domain model was developed through three iterations of modelling, implementation and testing. After the three iterations the domain model was finalised. The final entity-relationship model is depicted in Figure 1.

The main information about the cases are stored in *Cases* and *Armies*, which together represent the player race, opponent race, outcome, player unit and points. Each of the armies in the case contains a number of units described in *Unit*. This class is modelled based on the general characteristics of a unit, such as attacks, movement and weapon skill, as well as the information about the unit

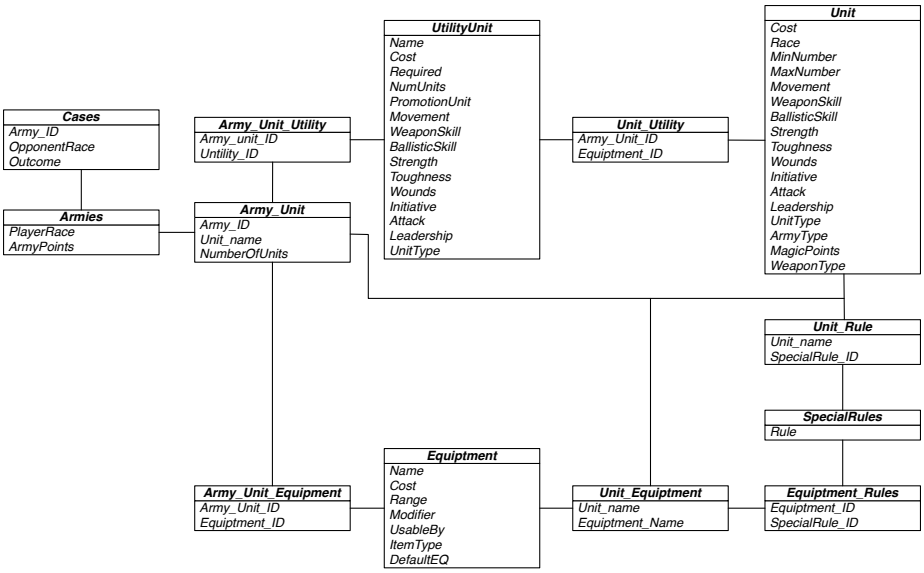

Fig. 1. Entity-relationship model of the domain

type, such as minimum and maximum number of troops and weapon type. The *Unit_Rules* and *Special_Rules*, which contains any special rules for the unit, the information about the unit is complete. The final information about unit is their equipment, which is contained in *Equipment*. Together these classes contain all relevant information about units. The *UtilityUnit* class is used to inform the program that this is attached to another unit, such as war machine crew.

5 Case-based reasoning component

Following the domain model described in Section 4, queries to the case base must at least contain features describing which race the player wishes to use, the opponents race and the number of points. Whilst this is sufficient to retrieve suitable cases more specific information will restrict the number of cases and give the similarity function more to work with; thereby (hopefully) enhance the result. There are no restrictions on the number of constraints used.

The case-based reasoning component follows the four steps in the CBR cycle as described in [9]. Each of the four steps are described in the following sections.

5.1 Similarity (Retrieve)

The case base contains all valid cases following the domain model in Section 4. The similarity is calculated using k-nearest neighbour. This approach was chosen as it quickly can distinguish between cases.

The similarity is calculated based on a weighted sum of the three parameters: army, opponent and outcome. Equation 1 calculates the weighted average of the three parameters, where A = 'Army', AW = 'Army Weighted', O = 'Opponent', OW = 'Opponent weighted', Ou = 'outcome', and OuW = 'Outcome Weighted'.

$$Similarity = \frac{\sum A \times AW + O \times OW + Ou \times OuW}{\sum AW + OW + OuW} \quad (1)$$

Each of the components in the above equation has their own simple similarity function tailor made to the specific part of the case.

5.2 Adaptation (Reuse)

The reuse phase in this implementation consists of two steps: first a retrieved case is passed through a naïve and secondly it may be passed through a more exhaustive adaptation. The naïve adaptation is used to substitute data in the query with data from the retrieved case.

Once the naïve adaptation phase is done the exhaustive adaptation process is started. This process may perform adaptation of the case. This adaptation investigates the query case to see if any of the army construction rules (see Section 2) has been violated, such as "no general", "too few core unit points", or "too many units in group".

The adaptation phase uses its own similarity function when calculating unit similarity. It simply calculates the difference between the parameters describing each unit (characteristics, unit type, army type, cost and weapon type). The characteristics are calculated as the average of the similarity of the nine individual characteristics. The cost is calculated as an interval similarity. The unit type, army type and weapon type is calculated from look-up tables where the similarity between different the different types has been estimated. For unit type, e.g. two infantry units has the similarity of 1.0, monstrous beasts and monstrous infantry as the similarity of 0.8, and cavalry and war machines are not similar.

At the end of these two adaptation phases the query case now contains the most similar retrieved case along with any adaptation necessary to fit the original requirements from the user.

5.3 Revise

The revise phase in Mymidia is a completely manual task for the user. The user is able to change almost any aspect of the case. There are two types of limitations on the possibilities of revision: *limitation* and *invalidators*. The former is changes that the user interface cannot change, yet if changed would not invalidate the results; whereas the latter is the player race and army points, which would invalidate the result.

Once the user is satisfied with the the case in question and wants to test it in a battle the case is stored in the case-base in a case vault. This initial stored case in the vault cannot be accessed by the retrieve phase.

5.4 Retain

The retain phase is a pseudo manual phase, where the user must add the result of a battle for case. Based on this result the following actions can be carried out:

- If the case results in a victory or a draw that value is updated in the case base. This moves the case from the vault to the normal case base, thus making it accessible to the retrieve phase.
- If the case results in a defeat that case is deleted.
- Finally, if the result remains unknown the case remains in the vault.

6 Implementation

The system was implemented in Java using the jColibri framework together with the Apache Derby database and Hibernation mapping tool. This section will shortly describe how jColibri was used in the implementation. The details on Apache Derby and Hibernation is outside the scope of this paper.

jColibri is a CBR framework that is fully implemented in Java. It includes several out-of-the-box features and interfaces that allows developers to create new functionality or tweak existing functionality. jColibri is easy to set up and integrate into a project. JColibri also supports a wide range of databases through Hibernate. Ontologies can be included through OntoBridge, which simplifies the use of ontologies to create knowledge intensive CBR application [10].

Through the use of interfaces in all core components of jColibri any application specific class can easily and seamlessly integrate into and used in conjunction with out-of-the-box features. The similarity calculation is divided into local and global similarity functions [11].

The changes to jColibri in Myrmidia is primarily related to the local and global similarity functionality (see Section 5) and the distinction between the problem description, solution and justification of the case. In our implementation this distinction has been left out as there are no distinction between the problem description and the solution. This has been done for the following reasons:

1. It was difficult to categorise the features as either being a description or a solution;
2. As it was difficult to categorise the features the most cost efficient solution with respect to minimise number of code lines was to have the problem description and solution in the same case hierarchy;
3. It felt restrictive and impairing on the functionality and the users' possibilities to define the *perfect* query.

7 Evaluation

Since only a limited number of races has been modelled and the number of cases is limited, a thorough performance test is difficult to conduct. However, each of the components and functions have been tested one their own.

7.1 Unit weight similarity

In order to maximise to unit similarity calculations (see Section 5.2) and adaptation matching results the associated weights for unit similarity can be set accordingly. This test was conducted by calculating two relatively dissimilar units, within the same race, with every possible weight configuration. The result of these calculations were not very surprising since the calculations is based on different features of the unit. Once the "best" weights were found they were used to calculate the similarity between one unit and all the others within a race. The results were then evaluated as either satisfactory or not and the ration between these two were calculated. The results showed that one universal unit similarity weight configuration is difficult, if not impossible to find. This is based on the observation that what is "good" from some units are far from accurate for others.

7.2 Case adaptation

The case adaptation stage is quite a complex function with several random elements. This makes it unpredictable and difficult to control. To verify that neither random elements or rule set verification was involved a specialised test was conducted. This specialised test posted a static query to the system that returned five cases and sent each case through the adaptation 10 000 times, totalling 50 000 trials.

The results of theses tests indicated that in cases that required much adaptation (e.g. cases with few army points begin used as the basis for cases with many army points) are prone to add the same unit several times. This is clearly a result of searching for the most similar unit several times. This type of unit affinity impacts the diversity of the army and may impair the overall effectiveness of an army. This is an open question and requires real gaming tests to clarify.

Another issue that arose is the fact that lords and heroes uses more that the allowed 25 % of points. Theses offending lords and heroes have a tendency to loose all their extra equipment. Thus, it is clear that this part of the adaptation requires some redesign.

7.3 Race exchange

An interesting side effect of this application is the ability to create a new army from scratch by basing it on an existing army from another race. This functionality was developed to act in the event that one or more of the kNN results are of another race than the target race. This occur if: there are no cases with the target race available; the existing case is less similar that other cases with a different race; or if k is larger than the number of cases with the target race.

Table 3 shows an example of a user asking for a Dwarf army to play High Elves. However, there are no Dwarf cases in the case base, so an existing High Elf case is adapted to suit the queried race. If the resulting army is investigated by a domain expert the Dwarf units substituting the High Elves units would appear

Table 3. Example of race unit exchange table

Player query race		Dwarf
Player case race		High Elves
Opponent query race		High Elves
Opponent case race		High Elves
Query/Case similarity		88.88 %
Outcome		Victory
Case unit	Exchanged unit	Similarity
Spearmen	Dwarf warriors	86.48 %
Sword Masters of Hoeth	Ironbreakers	89.81 %
Noble	Thane	81.48 %
Teclis	Runelord	62.96 %
Caradryan	Thane	81.48 %
Great Eagle	Gyrocopter	66.66 %
Spearmen	Dwarf warriors	86.48 %
Repeater Bolt Thrower	Organ Gun	66.66 %
Korhil	Thane	81.48 %
Repeater Bolt Thrower	Organ Gun	66.66 %
Great Eagle	Gyrocopter	66.66 %
Sword Masters of Hoeth	Ironbreakers	89.81 %

reasonably, within reason and keeping in mind that substitutions are based on actual armies. The only notable difference is that "Teclis", "Caradryan" and "Korhil", who are all named characters[1] are substituted with run-of-the-mill Dwarf lords and heroes.

8 Summary and future work

The work presented here is very much work in progress. Currently only three of the 15 different races are actually modelled in the domain model (Dwarfs, Empire and High Elves); further, not all race specific magical items has been modelled; finally, none of the recent published 8^{th} edition race books has been modelled, only the 8^{th} edition core rules. In addition, the current case-based only contains 10 cases (2 Empire, 5 High Elves and 3 Dwarf). Obviously this is a very incomplete set of cases (105 cases would be required to cover all combinations). However, the cases reflects the current and actual available domain knowledge.

There are still quite a few points where future work is required: *i*) improve the knowledge in the case-base; *ii*) improve the adaptation process in order to reduce the number of needed operations, perhaps by looking at the possibility of using a CSP solver as described in Section 3; *iii*) Implement a partly automated

[1] Named characters are special lords and heroes who often have extra special abilities compared to standard lords and heroes.

revise process; *iv*) make the system available as a web-based solutions, thereby ease the acquisition of cases and domain knowledge.

Acknowledgements

The authors would like to that WarTrond —Trondheim Warhammer Club— for their time and effort providing us the initial data for the case base.

References

1. Purvis, L.: Dynamic constraint satisfaction using case-based reasoning techniques. In: Proceedings of the CP'97 Workshop on Dynamic Constraint Satisfaction. (1997)
2. Purvis, L., Pu, P.: Adaptation using constraint satisfaction techniques. In Veloso, M.M., Aamodt, A., eds.: Case-Based Reasoning Research and Development, First International Conference, ICCBR-95. Volume 1010 of Lecture Notes in Computer Science. (1995) 289–300
3. López, B.: Holiday scheduling for city visitors. In Frew, A.J., Hitz, M., O'connor, P., eds.: Information and Communication Technologies in Tourism (ENTER'03), Springer-Verlag (2003) 252–260
4. Burke, E.K., MacCarthy, B., Petrovic, S., Qu, R.: Case-based reasoning in course timetabling: An attribute graph approach. In: Case-Based Reasoning Research and Development, Proceedings of the 4th International Conference on Case-Based Reasoning (ICCBR-2001). Volume 2080 of Lecture Notes in Computer Science. (2001) 90–104
5. Roldán, E., Negny, S., Lann, J.M.L., Cortés, G.: Constraint satisfaction problem for case-based reasoning adaptation: Application in process design. In Pierucci, S., Ferraris, G.B., eds.: Proceedings og the 20^{th} European Sympositum on Computer Aided Process Engineering (ESCAPE 20). (2010)
6. Hinkle, D., Toomey, C.: Applying case-based reasoning to manufacturing. AI Magazine **16**(1) (1995) 65–73
7. Petridis, M., Saeed, S., Knight, B.: An automatic case based reasoning system using similarity measures between 3d shapes to assist in the design of metal castings. Export Update **10**(2) (2010) 43–51
8. Beddoe, G., Petrovic, S.: Enhancing case-based reasoning for personnel rostering with selected tabu search concepts. Journal of the Operational Research Society (JORS) **58**(12) (2007) 1586–1598
9. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications **7**(1) (March 1994) 39–59
10. Garcia, J.A.R., Díaz-Agudo, B., González-Calero, P.A.: jCOLIBRI 2 tutorial. Technical report, Group for Artificial Intelligence Applications Universidad Complutense De Madrid (2008)
11. Díaz-Agudo, B., González-Calero, P.A.: A declarative similarity framework for knowledge intensive cbr. In: Proceedings on the international conference on case-based reasoning ICCBR 2001, Springer Verlag (2001) 158–172