Case-Based Reasoning for Military Naturalistic Decision Making

by

Ying Zhang

A dissertation Submitted in fulfilment of the requirements for the award of the Degree of Doctor of Philosophy

School of Management

University of Surrey

July, 2008
Acknowledgements

I would like to express my gratitude to all those who help and encourage me to complete this dissertation.

I would like to express my gratitude to my supervisor Dr. Panos Louvieris for his guidance, encouragement, and friendship over the past years. I am also indebted to Prof. Maria Petrou for her invaluable suggestions, comments, and constant help even after she moved to Imperial College.

Thanks for Dr. Terry Windeatt for reviewing my dissertation at the final stage. Thanks for Gareth White for his help with using VR-FORCES.

It is not easy experience for me to research about Artificial Intelligence in a management school, but I shall thank all my friends in SOM for their encouragement, support, friendship, and distractions.

I greatly appreciate the financial support for this dissertation provided by Surrey DTC and MoD.

I owe a debt of gratitude to my family for their incredible support throughout these years. I would like thank my brother for his immense help. Last but not least I would like to thank my parents for their endless encouragement, tolerance and patience.
Abstract

The more experience decision makers have in a particular decision domain, the more likely they are to employ Naturalistic Decision Making. In high tempo command and control military environments it is important that the human commander is in a position to make the appropriate decisions. Experienced human commanders often make decision based on the principles of Naturalistic Decision Making. Our approach employs Case Based Reasoning as a metaphor by identifying previous cases that match the problem in hand and accordingly provides the decision maker with the appropriate option. Cases based on the sample scenario are represented by vectors. The high dimensionality of data makes the adaptation very difficult. Secondly, interactions between solution parts have significant impact on the complexity of adaptation.

In this thesis, cases are visualised, and divided into clusters by unsupervised cluster mapping. This not only gives an insight into the case base, but also simultaneously allows high dimensional vectors to be transformed into two dimension map location representations. This process is applied to both of the case problem space and the case solution space. To achieve better adaptation results, cases with low ‘Problem-Solution Regularity’ are filtered first. New cases are then mapped to one of the available clusters. Subsequently, if an exactly analogous scenario cannot be found, the most similar cases in this cluster will be retrieved and input into a neural network. The neural network has been trained using as input the location differences in the problem space of pair of cases that belong to the same cluster and as output the location differences in their solution space. The new location of the adapted solution can be generated by the trained neural network if the input is the location difference in the problem space between the new cases and the nearest case to it from the corresponding cluster. Finally, the two dimensional location of the solution is transformed back to the original high dimensional Course of Actions. The obtained result is stored in the knowledge base for future reference. A military scenario is employed as a case study.
The sheer complexity and dimensionality of the battlefield rarely allow commanders to make decisions using rational, analytical methods in a timely, efficient and effective manner. Meanwhile, current military decision models are often regarded as stereotypical, predictable and doctrine limited. They fail to provide realistic characterisation of variability, flexibility and adaptability. Our research tackles this interesting and challenging problem in this complicated area. We propose to implement the Naturalistic Decision Making model by Case-Based Reasoning, to imitate the decision process of experienced military human commanders. As proof of concept, the results demonstrate that this approach can help identify the fundamental cognitive processes of military decision making and assist human commanders in military situation. Although there is scope for future work and improvement, this research provides a great experiment to achieve a more flexible, practical, adaptable decision model.
# Table of Contents

Acknowledgement ........................................................................ i  
Abstract ..................................................................................... ii  
Table of Contents ....................................................................... iv  
List of Figures ............................................................................ viii  
List of Abbreviations and Acronyms .......................................... xi  

Chapter 1 Introduction ................................................................ 1  
1.1 Motivation ........................................................................... 1  
1.2 Problem formulation ............................................................. 2  
1.3 Contributions ....................................................................... 4  
1.4 Organisation of the thesis ...................................................... 6  

Chapter 2 Decision Making ....................................................... 8  
2.1 Classical decision theory ...................................................... 8  
2.2 Naturalistic Decision Making ............................................... 10  
2.3 Recognition Primed Decision making ................................. 11  
  2.3.1 Situation Awareness ..................................................... 12  
  2.3.2 Mental Simulation ..................................................... 14  
  2.3.3 Different variations of RPD ......................................... 15  
2.4 Related Military Background .............................................. 16  
  2.4.1 Military Course of Action (COA) .................................. 16  
  2.4.2 Military decision-making process ................................. 18  
  2.4.3 OODA Loop ........................................................... 18  
2.5 Artificial Intelligence (AI) in Military decision aid systems ...... 20  
2.6 Related work about NDM .................................................... 22  
  2.6.1 RPDAgent ................................................................. 22  
  2.4.2 Adaptive BDI .......................................................... 23  
  2.4.3 Composite Agent (CA) ............................................... 24  
  2.4.4 Long Time Memory(LTM) .......................................... 25  
2.7 Implement NDM Model with CBR ...................................... 27  
2.8 Conclusions ....................................................................... 32  

Chapter 3 Case-Based Reasoning ............................................. 34
3.1 Introduction ................................................................. 35
3.2 CBR cycle ................................................................. 38
3.3 A Classification of CBR Applications ......................... 45
3.4 CBR in Military applications ........................................ 45
3.5 Summary ................................................................. 47

Chapter 4 Case Representation .............................................. 49
4.1 Case representation .................................................... 49
4.1.1 Symbolic Representation ......................................... 54
4.1.2 Binary (Nominal) Representation ............................. 54
4.1.3 Ordinal Representation ............................................ 55
4.1.4 Value Representation ............................................. 57
4.2 Scenario representation ............................................... 57
4.3 COA formulation ...................................................... 59
4.4 Summary ............................................................... 61

Chapter 5 Case Clustering ................................................... 63
5.1 Clustering ............................................................... 63
5.1.1 Distance Measures ................................................ 64
5.1.2 Clustering Methods ............................................... 67
5.1.3 Non-hierarchic clustering methods (NHCM) ............. 68
5.1.4 Hierarchic clustering methods (HCM) ....................... 69
5.1.5 Single-link clustering method ................................. 70
5.1.6 K-Means clustering ............................................. 71
5.2 Self Organised Map (SOM) ........................................... 73
5.3 Clustering by SOM .................................................. 77
5.4 Shape of the cluster .................................................. 78
5.5 Some further topics about SOM .................................... 78
5.6 Conclusions .......................................................... 82

Chapter 6 Case Base Visualisation ......................................... 85
6.1 Visualisation .......................................................... 86
6.1.1 PCA .............................................................. 86
6.1.2 Multidimensional scaling (MDS) .............................. 86
6.1.3 Sammon Mapping ................................................ 87
6.1.4 SOM ............................................................. 88
6.1.5 Visualization induced SOM (ViSOM) ....................... 91
6.2 Visualisation in related CBR systems ............................ 94
6.3 Visualisation for CBR ............................................... 95
6.4 Similarity assumption .............................................. 98
6.5 A simple example ................................................... 99
Chapter 7 Case Quality Assessment ..................................................... 106
7.1 Case Quality Analysis in CBM ......................................................... 107
7.2 Problem-Solution Regularity .............................................................. 109
7.3 Correlation for case quality measuring ............................................. 111
7.4 Experiment result ............................................................................. 112
7.5 Conclusions ...................................................................................... 117

Chapter 8 Case Adaptation ............................................................... 118
8.1 Introduction ................................................................................ 118
8.2 Related Work ............................................................................. 119
8.3 A direct approach and its problems ............................................... 121
8.3.1. BP network ............................................................................ 122
8.3.2. Hidden Units ......................................................................... 125
8.4 Improved approach ......................................................................... 126
8.5 How to find the target case solution .................................................. 127
8.6 An example .................................................................................. 128
8.7 Conclusions .................................................................................. 131

Chapter 9 Experimental Design ........................................................... 133
9.1 Methodology .............................................................................. 133
9.2 Data Collection ............................................................................ 134
9.3 Case Representation ..................................................................... 136
9.3.1 Scenario Representation ..................................................... 137
9.3.2 Case Solution Part ............................................................ 140
9.4 Case Clustering ............................................................................ 145
9.5 Case Adaptation ......................................................................... 148
9.6 Evaluation ................................................................................ 148
9.7 Conclusions ................................................................................ 155

Chapter 10 Conclusions and Future Work ............................................ 156
10.1 Contributions ............................................................................. 156
10.2 Future work ............................................................................. 161
10.2.1 RPD model .......................................................................... 161
10.2.2 Other possible approaches for adaptation ................................ 162
10.2.2.1 Kriging ........................................................................... 162
10.2.2.2 Hebbian Network ......................................................... 166
# List of Figures

<table>
<thead>
<tr>
<th>Fig.</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Simple Match (Level 1 Recognition)</td>
<td>14</td>
</tr>
<tr>
<td>2.2</td>
<td>Diagnose Situation (Level 2 Recognition)</td>
<td>14</td>
</tr>
<tr>
<td>2.3</td>
<td>Military command styles</td>
<td>17</td>
</tr>
<tr>
<td>2.4</td>
<td>OODA loop</td>
<td>19</td>
</tr>
<tr>
<td>2.5</td>
<td>Implement NDM with CBR</td>
<td>31</td>
</tr>
<tr>
<td>2.6</td>
<td>Three different conditions</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>CBR Cycle</td>
<td>44</td>
</tr>
<tr>
<td>3.2</td>
<td>A Classification Hierarchy of CBR Applications</td>
<td>45</td>
</tr>
<tr>
<td>4.1</td>
<td>Fuzzy logic representation of time</td>
<td>56</td>
</tr>
<tr>
<td>5.1</td>
<td>Neighbourhood examples of SOM</td>
<td>74</td>
</tr>
<tr>
<td>5.2</td>
<td>Examples of topological neighbourhood</td>
<td>76</td>
</tr>
<tr>
<td>5.3</td>
<td>Learning Rate decrease monotonically while the number of cycle increase</td>
<td>79</td>
</tr>
<tr>
<td>5.4</td>
<td>Radius decrease monotonically while the number of cycle increases</td>
<td>80</td>
</tr>
<tr>
<td>5.5</td>
<td>The neighbourhood function as the distance and cycle number change</td>
<td>80</td>
</tr>
<tr>
<td>5.6</td>
<td>Eigenvectors of the dataset</td>
<td>82</td>
</tr>
<tr>
<td>6.1</td>
<td>Honkela's suggestion for visualization</td>
<td>88</td>
</tr>
<tr>
<td>6.2</td>
<td>Squares and Hexagons</td>
<td>89</td>
</tr>
<tr>
<td>6.3</td>
<td>Decomposition of the SOM updating force</td>
<td>92</td>
</tr>
<tr>
<td>6.4</td>
<td>PCA for Iris data</td>
<td>93</td>
</tr>
<tr>
<td>6.5</td>
<td>Sammon mapping for Iris data</td>
<td>93</td>
</tr>
<tr>
<td>6.6</td>
<td>SOM of Iris data</td>
<td>93</td>
</tr>
<tr>
<td>6.7</td>
<td>ViSOM for Iris data</td>
<td>93</td>
</tr>
<tr>
<td>6.8</td>
<td>Feature values of Iris data</td>
<td>96</td>
</tr>
<tr>
<td>6.9</td>
<td>U-Matrix shows the solution values</td>
<td>96</td>
</tr>
<tr>
<td>6.10</td>
<td>How well the prototype vector reflects the new case(Vesanto 1999)</td>
<td>97</td>
</tr>
<tr>
<td>6.11</td>
<td>Quantization error plot (Vesanto 1999)</td>
<td>97</td>
</tr>
<tr>
<td>6.12</td>
<td>Case problem space for dataset1</td>
<td>101</td>
</tr>
<tr>
<td>6.13</td>
<td>Case solution space for dataset1</td>
<td>102</td>
</tr>
<tr>
<td>6.14</td>
<td>Case problem space for dataset2</td>
<td>103</td>
</tr>
<tr>
<td>6.15</td>
<td>Case solution space for dataset2</td>
<td>104</td>
</tr>
<tr>
<td>7.1</td>
<td>Case coverage and reachability</td>
<td>108</td>
</tr>
<tr>
<td>7.2</td>
<td>Example of case problem space</td>
<td>110</td>
</tr>
<tr>
<td>7.3</td>
<td>Example of case solution space</td>
<td>110</td>
</tr>
<tr>
<td>7.4</td>
<td>KNN result for Boston housing database</td>
<td>113</td>
</tr>
<tr>
<td>7.5</td>
<td>KNN result for Water Treatment Plant (K=1)</td>
<td>114</td>
</tr>
<tr>
<td>7.6</td>
<td>KNN result for Water Treatment Plant (K=3)</td>
<td>114</td>
</tr>
<tr>
<td>7.7</td>
<td>KNN result for Water Treatment Plant (K=5)</td>
<td>116</td>
</tr>
<tr>
<td>7.8</td>
<td>KNN result for Water Treatment Plant (K=7)</td>
<td>116</td>
</tr>
<tr>
<td>7.9</td>
<td>KNN result for dataset2 (K=3)</td>
<td>117</td>
</tr>
<tr>
<td>8.1</td>
<td>Gradient descent algorithm (Mitchell 1997)</td>
<td>123</td>
</tr>
<tr>
<td>8.2</td>
<td>BP network</td>
<td>124</td>
</tr>
<tr>
<td>8.3</td>
<td>The structure of the BP network with SOM data input and output</td>
<td>126</td>
</tr>
<tr>
<td>8.4</td>
<td>Another approach to train the BP network</td>
<td>127</td>
</tr>
<tr>
<td>8.5</td>
<td>Problem space ViSOM(30x15)</td>
<td>129</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>The Contents of the Major Parts of a Case</td>
<td>50</td>
</tr>
<tr>
<td>4.2</td>
<td>Our Case representation</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Symbolic representation</td>
<td>54</td>
</tr>
<tr>
<td>4.4</td>
<td>Binary representation</td>
<td>55</td>
</tr>
<tr>
<td>4.5</td>
<td>Ordinal representation</td>
<td>55</td>
</tr>
<tr>
<td>4.6</td>
<td>Fuzzy ordinal representation</td>
<td>56</td>
</tr>
<tr>
<td>4.7</td>
<td>COA representation matrix</td>
<td>61</td>
</tr>
<tr>
<td>4.8</td>
<td>An example of waypoint movement</td>
<td>61</td>
</tr>
<tr>
<td>4.9</td>
<td>COA representation matrix for the example in Table 4.8</td>
<td>62</td>
</tr>
<tr>
<td>6.1</td>
<td>Dataset1</td>
<td>100</td>
</tr>
<tr>
<td>9.1</td>
<td>Grid representation</td>
<td>139</td>
</tr>
<tr>
<td>9.2</td>
<td>Scenario representation</td>
<td>140</td>
</tr>
<tr>
<td>9.3</td>
<td>Troops section route representation</td>
<td>141</td>
</tr>
<tr>
<td>9.4</td>
<td>A case representation example of Fig.9.4</td>
<td>144</td>
</tr>
<tr>
<td>9.5</td>
<td>An example of adapted result</td>
<td>148</td>
</tr>
<tr>
<td>9.6</td>
<td>Evaluation of suggested COAs</td>
<td>152</td>
</tr>
<tr>
<td>9.7</td>
<td>Evaluation Result of different approaches</td>
<td>153</td>
</tr>
<tr>
<td>9.8</td>
<td>ANOVA result of different approaches</td>
<td>154</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
<td></td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>ACTA</td>
<td>Applied cognitive Task Analysis</td>
<td></td>
</tr>
<tr>
<td>ADS</td>
<td>Advanced Distributed Simulation</td>
<td></td>
</tr>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
<td></td>
</tr>
<tr>
<td>APSS</td>
<td>Anticipatory Planning Support System</td>
<td></td>
</tr>
<tr>
<td>BDI</td>
<td>Belief, Desire, Intention</td>
<td></td>
</tr>
<tr>
<td>BMU</td>
<td>Best Matching Unit</td>
<td></td>
</tr>
<tr>
<td>BP</td>
<td>Back Propagation</td>
<td></td>
</tr>
<tr>
<td>CA</td>
<td>Composite Agent</td>
<td></td>
</tr>
<tr>
<td>CGF</td>
<td>Computer Generated Forces</td>
<td></td>
</tr>
<tr>
<td>CBR</td>
<td>Case-Based Reasoning</td>
<td></td>
</tr>
<tr>
<td>CBM</td>
<td>Case-Based Maintenance</td>
<td></td>
</tr>
<tr>
<td>CLIQUE</td>
<td>Clustering High-Dimensional SPace</td>
<td></td>
</tr>
<tr>
<td>COA</td>
<td>Course of Action</td>
<td></td>
</tr>
<tr>
<td>CSF</td>
<td>Critical Success Factor</td>
<td></td>
</tr>
<tr>
<td>DARPA</td>
<td>Defence Advanced Research Projects Agency</td>
<td></td>
</tr>
<tr>
<td>DART</td>
<td>Dynamic Analysis and Re-planning Tool</td>
<td></td>
</tr>
<tr>
<td>DBSCAN</td>
<td>Density-Based Spatial Clustering of applications with Noise</td>
<td></td>
</tr>
<tr>
<td>DENCLUE</td>
<td>DENsity-based CLUstEring</td>
<td></td>
</tr>
<tr>
<td>FIPA</td>
<td>Foundation for Intelligent Physical Agents</td>
<td></td>
</tr>
<tr>
<td>GE</td>
<td>Generalised Episode</td>
<td></td>
</tr>
<tr>
<td>HCM</td>
<td>Hierarchic Clustering Methods</td>
<td></td>
</tr>
<tr>
<td>KNN</td>
<td>K- Nearest Neighbor</td>
<td></td>
</tr>
<tr>
<td>KQML</td>
<td>Knowledge Query and Manipulation Language</td>
<td></td>
</tr>
<tr>
<td>LTM</td>
<td>Long Time Memory</td>
<td></td>
</tr>
<tr>
<td>LWDB</td>
<td>Land Warfare Data Base</td>
<td></td>
</tr>
<tr>
<td>MDS</td>
<td>Multidimensional scaling</td>
<td></td>
</tr>
<tr>
<td>METT-T</td>
<td>Mission, Enemy, Terrain, Troops, and Time</td>
<td></td>
</tr>
<tr>
<td>ModSAF</td>
<td>Modular Semi-Automated Forces</td>
<td></td>
</tr>
<tr>
<td>MOP</td>
<td>Memory Organisation Packet</td>
<td></td>
</tr>
<tr>
<td>NDM</td>
<td>Naturalistic Decision Making</td>
<td></td>
</tr>
<tr>
<td>NHCM</td>
<td>Non-Hierarchic Clustering Methods</td>
<td></td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>NN</td>
<td>Nearest Neighbour</td>
<td></td>
</tr>
<tr>
<td>OODA</td>
<td>Observe, Orient, Decide, Act</td>
<td></td>
</tr>
<tr>
<td>OPTICS</td>
<td>Ordering Points to Identify Clustering Structure</td>
<td></td>
</tr>
<tr>
<td>PAM</td>
<td>Partitioning Around Medoid</td>
<td></td>
</tr>
<tr>
<td>PCA</td>
<td>Principal Component Analysis</td>
<td></td>
</tr>
<tr>
<td>RA</td>
<td>Reactive Agent</td>
<td></td>
</tr>
<tr>
<td>RPD</td>
<td>Recognition Primed Decision</td>
<td></td>
</tr>
<tr>
<td>SA</td>
<td>Situation Awareness</td>
<td></td>
</tr>
<tr>
<td>SCA</td>
<td>Symbolic Constructor Agent</td>
<td></td>
</tr>
<tr>
<td>SME</td>
<td>Subject Matter Experts</td>
<td></td>
</tr>
<tr>
<td>SOAR</td>
<td>States, Operators and Rules</td>
<td></td>
</tr>
<tr>
<td>SOM</td>
<td>Self Organizing Map</td>
<td></td>
</tr>
<tr>
<td>SSR</td>
<td>Scaled Strength Ratio</td>
<td></td>
</tr>
<tr>
<td>STING</td>
<td>Statistical Information Grid</td>
<td></td>
</tr>
<tr>
<td>U-Matrix</td>
<td>Unified Distance Matrix</td>
<td></td>
</tr>
<tr>
<td>ViSOM</td>
<td>Visualization induced SOM</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Military commanders use dynamic, unstructured, uncertain and temporal information of battle space to plan, analyse and decide on the best plan in order to defeat the enemy. Information gathered from various sources such as terrain, equipment, weather, resources, personnel and mission definition are integrated to help the commander develop the most appropriate course of action (COA) and the executable plan in order to achieve his/her military intent. How to make the quick and correct decision in these kinds of conditions becomes increasingly important for current military.

1.1 Motivation

As the technology has developed, the amount of information the human commander needs to deal with has increased tremendously, but the time available for decision making has decreased dramatically. Since, however the human commander is frequently a human agent there is a limit to his/her capabilities, especially under fatigue in high tempo stressful military environments. Therefore, an automated system which provides information concerning the situation in hand can support the agent to succeed his/her objective. In addition, the military relies heavily on decision support systems to train their new human commanders. Furthermore, decision support systems can also be applied for COA analysis and experimenting with new war fighting doctrine.

However, current military decision models are too stereotypical, predictable and doctrine limited. They fail to provide realistic characterisation of variability, flexibility and adaptability. Also these models are too uniform, homogeneous and inflexible. They fail to incorporate the role of such factors as stress, experience, fatigue, risk attitude, etc (Pew & Mavor 1998). Additionally, most decision models
are focused on tactical decisions and ignore the decision process of military commanders at the operational level of warfare. The main reason of these is the lack of the complete understanding of how people make decisions (Sokolowski 2003a). Instead of creating the optimum decision for the given situation, a decision model which can imitate the decision process of the experienced military human commander, and identify the fundamental cognitive process is more suitable for military decision support systems.

1.2 Overview of the research

The first task is to analyse what the human commanders do and how they make decisions. Decision making research can be divided into analytical and intuitive. Analytical decision making is based on logical analysis of the decision situation, while intuitive based decisions rely upon pattern recognition and experience. Early research in decision making focused on analytical decision making. It is stated that humans always made decisions in a logical manner that maximize the decision outcome value. However, it is rarely the case that one has the time and all the information needed to apply analytical decision making in many real life situations. Since the mid 1980’s, research attention has turned to a more intuitive approach, namely Naturalistic Decision Making (NDM). NDM models the mental decision process of a decision marker in his/her natural environment and it has been formally defined as the approach people use their experience to make decisions in different domains (Zsambok 1997a).

In our research, we approximate the way experienced military commanders make decisions based on the principles of NDM, propose a new model similar to Recognition Primed Decision (RPD) (Klein 1997; Klein 1993). The most critical component of this model is maintaining situation awareness of the decision maker throughout the process. It is directly related with experience, because the more effort the decision maker puts into situational awareness, the more efficient he/she can match the decision situation to previous experience. RPD is the most popular model for NDM, although it does have some disadvantages. It does not address all the aspects of NDM. For examples, it is lack of empirical base, it neglects to specify cognitive process, it is lack of prescriptive guidance as a descriptive account and
finally it cannot distinguish between good and poor decisions or to identify errors (Klein 1997). Furthermore, RPD does not take into account the generation of new COA, it avoids from specifying what happens when people do compare COA.

Our approach employs Case-Based Reasoning (CBR) as a metaphor for identifying previous cases that match the problem in hand and accordingly provides the human commander with the most appropriate option. The origins of CBR are in research about the way people solve problems by remembering how they solved similar problems in the past. In the same way, instead of relying on general knowledge of a problem domain, CBR solves new problems based on previous experiences. When a new problem is encountered, the system retrieves similar cases from the case base and adapts the solution to fit the new problem. Furthermore, CBR has the ability of incremental sustained learning. When a problem is solved, the solution and problem description are added to the case base.

There are similarities between CBR and RPD, however RPD has some special characteristics, and pure CBR cannot capture all the aspects of the RPD process. In this thesis, we propose to apply CBR to implement a NDM model. In our research, a case may be represented as a scenario and the corresponding COA as the solution. Both are composed of complicated and high dimensional data. Thus the case base we are concerned with is far more complex than that ones encountered in the most common CBR systems.

Cases are clustered first, and then different approaches are processed according to the result, whether the cluster is typical or not. Then case filtration is applied to discard low quality cases, an approach using correlation is proposed as a measure of case quality and helps us chose the better quality cases in the case base, thus making the case adaptation easier and more accurate. Meanwhile, case visualisation is used to provide a mental simulation approach for the decision maker, as well as an intuitive understanding of the case base structure, help us discover patterns and trends in the case base. Case filtration and visualisation can also help us decide whether the system is suitable to apply CBR or not by demonstrating how the data follows "similar problems have similar solutions". If it is not, then the adaptation is so demanding that it would be better to solve the problem from scratch without employing CBR.
With the high dimensional solution space of the case base, it is extremely difficult to achieve case adaptation, as most CBR systems only deal with a single dimensional solution; especially when there are interactions among features of the solution space. In this case, the only apparent solution is to choose the nearest neighbour and skip the adaptation entirely. However, we propose to map problem space and solution space in two different neural networks first, then analyse the mapping relation between these two maps with the help of another neural network. Finally, a simple military scenario is used as a case study.

1.3 Contribution

This thesis describes the implementation of a naturalistic decision model for use in military. During its development and operation several issues were encountered. The most interesting ones, which constitute the main contribution of this thesis, are:

- A NDM model capable of mimicking decisions making process of experts in their field by applying CBR. NDM describes how experienced decision makers work problems in high stress and time poor decision situations. It has two important characteristics: the decision makers use their experience to recognize the situation and they evaluate the COA by mental simulation. It explains how people use their experience to achieve decisions without comparing the strengths and weakness of the courses of actions. In this thesis, we use case clustering for pattern recognition, different situations are discussed. Visualisation is applied for mental simulation. Meanwhile, cases are filtered and adapted to generate new COA for a sufficiently similar scenario. It solves the following problems: the RPD model cannot generate alternative COAs simultaneously; RPD's lack of empirical basis; and, RPD refrains when a comparison COA is needed. Our model combines the CBR cycle to follow the process of NDM.
A representation method for military scenarios and COA. COA are normally composed of human natural language and a graphical description, thus it is very complex to formulate. We formulate scenario by their military features first, then represent a COA by entities’ waypoints and the corresponding time for each entity to reach each of the waypoints. This representation is simple, easy to understand and combine successfully with military simulation tools.

A suitable approach for Case Base visualisation is proposed. We use it for mental simulation to help diagnose a situation, to identify whether or not the situation is familiar, and the outcome is desirable. Meanwhile, visualisation can also offer powerful analytical means to uncover patterns and trends in the case base. It can indicate the two most important relationships in CBR community: problem-solution regularity and problem-distribution regularity, thus it can be applied in order to automatically monitor the appropriateness of the case base with the current problem. Our proposal has many advantages when compared to the alternative current visualisation approaches.

A method for assessing the quality of cases by correlation. This method allows us to filter out low quality cases and achieve better adaptation result. The larger the correlation of problem space distances between a case and other cases and their solution space distance, the more this case follows “similar problems have similar solutions”; thus, the better quality this case is. Case filtration can not only help increase the quality of the case base, but also the prediction accuracy of case adaptation. Meanwhile, we also propose a method to measure ‘Problem-Solution Regularity’ of case base. ‘Problem-Solution Regularity’ represents how well the similarities between problems approximate the similarity between according solutions in practice. There are many related discussions in CBR community, but how to calculate the ‘Problem-Solution Regularity’ of a given case base is still unknown. In our research, as both the case problem space and case solution space already have been visualised and transformed in two dimensional space first, their topological relations in original high-dimensional space are kept as good as possible in the two dimensional space. Thus the ‘Problem-Solution Regularity’
can be considered as the similarity between the case problem map and case solution map.

- A new adaptation method for case bases with high dimensional solution space is presented. Case adaptation is a very difficult task, especially for high dimensional data with only a limit number of cases. More importantly, there are always interactions between features in the case base, thus it is virtually impossible to treat every dimension separately. Currently there are no available approaches in CBR area to solve this problem. First, we propose to map the case problem and solution spaces in two different unsupervised maps. Second, analyse the mapping relations between these two maps by another neural network. This idea is very similar to the natural decision making process of matching solutions according to the problem descriptions.

1.4 Organisation of the Thesis

The remainder of the thesis is structured as follows:

- Chapter 2 provides a critical review of the literature on decision making, including decision theory, military decision making and the RDP model. In the same chapter our model to achieve NDM is proposed.
- Chapter 3 gives a detail description of the CBR, including CBR cycle, each phase in the cycle, CBR applications etc.
- In Chapter 4, we introduce different approaches for case representation, and propose our solution for military scenario formulation and corresponding COA representation.
- Chapter 5 discusses different clustering algorithms to achieve more efficient retrieval of cases. Following an evaluation, the most appropriate one for our research problem is selected.
- Chapter 6 discusses visualisation, where different approaches to visualise high dimension data are described, and the most appropriate one for CBR is chosen.
- The basic foundation of CBR is "similar problems have similar solutions". In Chapter 7, we propose a case quality assessment approach to evaluate how
well the case follows this foundation. Experiment results are presented and discussed.

- Chapter 8 considers case adaptation. Different adaptation approaches for CBR are discussed. How to adapt cases whose solution is high dimensional with interactive features is proposed. An example is used to demonstrate the result.

- In Chapter 9, a simple military scenario is employed as a case study. This is used to demonstrate and validate the NDM model proposed in Chapter 2, including case representation, case clustering, case filtration, case visualisation, case adaptation, etc.

- Finally in Chapter 10 we present the conclusion of the thesis where we reflect on our findings and consider the implications for future research.
Chapter 2

Decision-Making

Decision-making can be divided into analytical and intuitive. Analytical decision-making is based on logical analysis of the decision situation while intuitive based decisions rely upon pattern recognition and experience. A good example illustrating both methods was the chess matches between chess master Gary Kasparov and the computer Deep Blue (2005b). While the computer used detailed and exhaustive option analysis to decide on each move, the chess master decided his moves based largely on his knowledge, experience and recognition of patterns.

In Section 2.1, we introduce classic decision theory. Naturalistic Decision Making is described in Section 2.2. In Section 2.3, we discuss the Recognition Primed Decision making model. Section 2.4 presents related military background knowledge. Section 2.5 discusses Artificial Intelligence in military decision systems, while related Artificial Intelligence researches about NDM is discussed in Section 2.6. How to implement NDM model with CBR is discussed in Section 2.7. Finally in Section 2.8, we conclude the whole chapter.

2.1 Classical decision theory

Early research in decision-making was focused on analytical decision-making. Classical decision theory has been the dominant framework (von Neumann & Morgenstern 1953). Classical decision theory is the collection of axiom-based models of uncertainty, risk, and utility that provide a method to make an optimal decision from among a list of choices.
It was the work of von Neumann and Morgenstern in the 1940s (von Neumann & Morgenstern 1947) which laid the foundations for the classical decision theory. Their theory of utility sought to model rational preferences between simple randomizations over a consequence space. The objects over which the decision maker was required to express a preference were simple probability distribution. Their Subjective Expected Utility (SEU) Theory includes both subjective probabilities about the uncertainty of an outcome and a decision marker’s personal risk tolerance for that outcome.

Classical decision theory states that humans always make decisions in a logical manner which maximizes the decision outcome value. It focuses on the decision event, prescribing the optimal choice from a fixed set of known alternatives where optimality is defined by the underlying models and choice is dictated by an explicit rule (Beach & Lipshitz 1995; Orasanu & Connolly 1995). It focuses on the decision outcome and provides a means to calculate decision outcomes in terms of probabilities of risk and uncertainty. It normally contains two main parts: First is about uncertainty and risk, it uses probability theory such as Bayesian theory to draw inferences about any situation in any domain. Second is about utility, it uses multi-attribute utility theory to select an optimal action in the any domain. Bayesian probability theory requires that decision makers consider a set of mutually exclusive and exhaustive hypotheses, each of which is assigned a probability. Each potential observation that might bear on those hypotheses is assigned a diagnostic strength. Multi-attribute utility theory is an analogous method for choice. Choices are made based on a combination of the probability of each uncertain state, the importance of each evaluative dimension, and the score of each action-state combination on every evaluative dimension.

Classical decision theory describes the choices of an ideal hypothetical decision maker, who is a computationally omnipotent economic man. It came into question when research showed that humans do not necessarily make decisions in a logical manner (Kahneman & Tversky 1979). Few people spend time performing decision optimization calculations and many decisions can not be formulated in mathematical terms. The insufficiency of decision maker’s time and resources during the operations in practice make it difficult to rationalise (Bentham 1970). Human decision making
consist of many tasks that are quite different from the gambling task for which classical decision theory was designed (Bentham 1970).

Personal biases also influence decisions and tend to drive humans away from the purely optimal choice because of many competing factors. This led researchers to investigate more thoroughly how humans actually make decisions. As a result, the theory of Naturalistic Decision Making was developed (Zsambok 1997b).

### 2.2 Naturalistic Decision Making (NDM)

Since the mid 1980's, research attention has been turned to the more intuitive approach towards decision making, or naturalistic decision making. This happened because it is not feasible to apply classical decision making research analysis to many real-world decision making practice, and classical decision making fails to account for the decision maker's experience, task complexity and the demands of the naturalistic environment.

The NDM approach was formally launched in 1989 at a conference in Ohio (Klein, Orasanu, Calderwood, & Zsambok, 1993). It is a theory that models a person's mental decision process in his natural environment. NDM has been formally defined as the way people use their experience to make decisions in field settings (Zsambok 1997b). It is based on the intuitive steps a person follows in reaching a decision rather than on a mathematical process for computing optimal outcomes. It accepts a satisfactory choice but not necessarily optimal. Nobel Prize winner Herbert Simon found that successful experienced businessmen chose satisfactory solutions instead of the optimal one, because the latter one was unlikely to be achieved (Simon 1957).

Researchers have identified eight factors that most often appear in naturalistic decision settings (Orasanu & Connolly 1993). A decision maker is likely to employ the naturalistic process to arrive at a decision when one or more of these factors are present. These factors are:

- Ill-structured problems.
- Uncertain dynamic environments.
- Shifting, ill-defined, or competing goals.
- Action/feedback loops.
- Time stress.
- High stakes.
- Multiple players.
- Organizational goals and norms.

The more experience a decision maker has in a particular decision domain, the more likely he/she is to employ NDM since his/her experience provides a significant intuitive feel of which COA should be chosen. Several models of NDM have been proposed, such as Image Theory (Beach 1990), Noble’s Model of Situation Assessment (Noble 1989), Model of Explanation-Based Decisions (Pennighto & Hastie 1993) and Dominance Search Model (Montgomery 1989). However, not a single one of them has all the characteristics of naturalistic decision (Zsambok, Beach, & Klein 2002). In (Lipshitz 1993), nine different types of models were discussed. Most models assume some level of expertise in the field.

### 2.3 Recognition Primed Decision making (RPD)

Klein’s RPD model (Klein 1989a) has been describe as the prototypical NDM model. It is indeed the most often cited and best researched of the NDM models. The origins of the RPD model are set in command and control performance, the earliest work focussed on observing and obtaining protocol from urban fire-ground commanders about emergency events they had handled. The RPD model was formulated to instantiate NDM in a formal manner and represents the decision process of an experienced decision maker. The RPD model explains how people use their experience to achieve decisions without comparing the strengths and weakness of the courses of actions. So the key feature of RPD is the emphasis on situation assessment, not the generation and comparison of alternatives, because the suitable course of action will emerge when the situation is understood. Experience is the source of the ability to recognize problems and their solutions.

According to (Klein 1993), there are some features that RDP model is different from classic decision models.
• RPD focuses on situational assessment rather than comparing several decision options.
• RPD describes how people use their experience to arrive at a decision.
• RPD asserts that an experienced decision maker can identify a satisfactory COA as the first one he considers rather than treating option generation as a random process.
• RPD relies on satisfying rather than optimizing finding the first COA that works rather than the optimal one.
• RPD focuses on sequential evaluation of COAs rather than on the simultaneous comparison of several options.
• RPD asserts that experienced decision makers use mental simulation to assess a COA rather than comparing the strengths and weaknesses of several COAs.
• RPD allows the decision maker to be more quickly prepared to initiate action by committing to a COA being evaluated rather than waiting until all COAs are compared.

Interestingly, the more experience the decision marker has with the problem, the more likely he/she will employ RPD to achieve the decision. On the contrary, a decision maker who is not very familiar with the problem domain usually prefers using analytical methods to compare possible alternative instead of RPD. According to (Drillings & Serfaty 1997; Kaempf et al. 1996; Klein 1989b), military decision makers employ RPD in at least 60% of decision situations. The naval officers employ RPD in about 95% of their decision situations (Kaempf, Klein, Thorsden, & Wolfe 1996). The RPD model does provide an explanation as to why military commanders are able to make decisions faster than what would be considered normal using the rational choice model. RPD focuses on assessing the situation rather than considering multiple courses of action, and the decision-makers do not generate a list of options.

2.3.1 Situation awareness (SA)

SA is the perception of the decision maker for the corresponding decision situation. This perception normally is directly related with experience, because the more effort
the decision maker tries on situational awareness, the more efficient he/she can match the current decision situation to previous experience. According to (Endsley & Kiris 1995), a model of SA has three levels:

- **Level 1 SA**: Perception of the elements in the environment. The first step to achieve SA is to perceive the status, attributes, and dynamics of relevant elements in the environment.

- **Level 2 SA**: Comprehension of the current situation. Comprehension of the current situation is based on a fusion of the separate level 1 elements. Level 2 SA involves understanding the significance of objects and events in the environment and combining this data to form a holistic picture of the environment in light of one’s goals.

- **Level 3 SA**: Projection of future status. The highest level of situation awareness is to project the future actions of elements in the environment. This is achieved through knowledge of the status and dynamics of elements in the environment and comprehension of the situation (level 1 and level 2 SA).

Kaempf, Klein, Thorsden, and Wolfe (1996) studied US Navy offices who had been involved in anti-air operations, and found that most decisions concerned the nature of the situation. For those decision about adopting a COA, fewer than 5% involved comparisons between alternatives.

Different approaches have been suggested as explanations for how decision-makers achieve SA. (Kaempf, Klein, Thorsden, & Wolfe 1996) found that military officers identified the following methods of arriving at SA:

1. 87% of participants used a feature matching strategy. This is where the decision-maker sees the situation as familiar, and arrives at situation awareness through a series of recognised cues.
2. 12% of participants used story generation. When the environment does not provide enough information to be recognised as familiar, the decision-maker constructs a story to explain the information and to arrive at greater situation awareness.

3. Only 1% of cases did not fit into either of these main categories.

In the RPD model, when situational recognition occurs, there will be four by-products: cues, goals, actions and expectancies. These four elements come from experience and describe the cognitive concepts on which a decision maker operates. Cues represent those physical and mental elements on which a decision maker relies in order to understand and to monitor a situation. The decision maker will be aware of cues on which to focus so that information overload does not occur. Cues are often made up of aggregated pieces of information that a decision maker assembles in his or her mind (Sokolowski 2002; Sokolowski 2003b). The decision maker should also be aware of goals that he/she must attain as part of the decision. He/she will know what to expect next as the situation unfolds and the decision is implemented. Finally, the decision maker will know from previous experience what actions worked in this type of situation.

### 2.3.2 Mental Simulation

The decision maker use mental simulation to help diagnose a situation. By mentally examining various aspects of the elements in the situation, he/she can decide whether the situation is familiar or not. Normally, the decision maker in RPD is only looking at one COA at a time. Once an option is considered, he/she will use mental simulation to work it through at a deeper level, looking for pitfalls and opportunities. If the primary action under consideration does not quite solve the problem, then the decision maker may employ mental simulation to consider different aspects of the action to modify so that it can achieve a satisfactory result. At some point the decision maker may decide that the primary action will not work. He/she must then choose another action for consideration.
2.3.3 Different variations of RPD

There are three different variations of RPD model. Fig 2.1 illustrates the first level of decision making in RPD. A situation is experienced and the products of this recognition are expectations, knowledge of relevant cues, plausible goals and typical actions. Implicit within this recognition is an appropriate COA for the current situation.

This normally happens in routine situations, the expert comes to recognize a situation as typical and acts accordingly, especially under time pressure where there is no time for deliberation. Once the situation is recognized as familiar, a single COA is ‘primed’ and implemented. This process has been replicated across a wide range of samples including fire-fighters, ship and tank commanders, aviation pilots, offshore oil managers.

The second level of decision-making, Diagnose the Situation, is shown in Fig 2.2. This level captures an important characteristic distinguishing expert and novice decision-makers. During situation assessment, experts are adept at realizing when they do not have sufficient information to adequately assess a situation and the COA is not obvious. The situation is diagnosed using techniques such as feature matching.
and story building, each of which requires that more information be extracted from the situation. Experts are also adept at recognizing anomalies between current and past situations.

The third level of RPD model, Evaluate COA, occurs when there is uncertainty concerning a COA, as shown by Fig 10.1. It should not be inferred that the decision-maker compares alternative courses of action. Instead, a COA is evaluated using mental simulation to confirm that it will work. If there is any doubt, the COA is modified, or when necessary, the situation is reassessed. (Klein 1993)

2.4 Related Military Background

In this section, we introduce the related military background, including a review of the military decision making process. Reference is also made to military decision support systems.

2.4.1 Military Command and Control (C2)

Military decision making is very complex and difficult. Commanders must make decisions under quite difficult circumstances that may have enormous consequences.

There are many different definitions of C2. According to (2005a), Command and Control are the exercise of authority and direction by a properly designated commander over assigned and attached forces in the accomplishment of the mission. C2 are performed through an arrangement of personnel, equipment, communication, facilities, and procedures employed by a commander in planning, directing, coordinating, and controlling forces and operations in the accomplishment of the mission. Thus, C2 is the term that describes the job of the military commander. It is characterised by ill-structured problems, changing conditions, high stakes and time demands.

In different armies, there are different command structures. In top-down order specific oriented command style, the command is issued by the supreme commander of the force, and the subordinates are not allowed to make any changes. In objective specific oriented command style, the subordinated units are allowed some freedom to decide
for themselves how to achieve the objective. Mission specific oriented command structure is a more open command style: subordinated units have full autonomy to synchronize with other units in order to achieve the global goal and thus may comprise a number of objectives necessary, without the interference from the supreme commander (Mason & Moffat 2000). This is schematically shown in Figure. 2.3. UK doctrine has moved recently more towards the bottom, i.e. towards mission specific style.

<table>
<thead>
<tr>
<th>Top Down</th>
<th>Bottom Up</th>
</tr>
</thead>
<tbody>
<tr>
<td>-Soviet Union</td>
<td></td>
</tr>
<tr>
<td>-Chinese army</td>
<td></td>
</tr>
<tr>
<td>Objective Specific</td>
<td></td>
</tr>
<tr>
<td>-UK/US</td>
<td></td>
</tr>
<tr>
<td>Mission Specific</td>
<td></td>
</tr>
<tr>
<td>-WW2 Germany</td>
<td></td>
</tr>
<tr>
<td>-Israeli army</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 2.3 Military command styles**

In the past, commanders were often trained to develop a few COA for a tactical scenario. Then they were required to estimate the advantage and disadvantages of each COA and weigh those evaluations to reach a decision. In this rational choice model, the commander was seen as one who performs the calculations that will lead to optimal decisions. Klein (1988) observed that commanders in difficult situations and under time stress did not appear to use the classical approach, even when they were trained in this approach. Rational choice is a useful model to apply under certain conditions. For example, it is better when there is sufficient information about the situation available and the alternative COAs are well understood. Unfortunately, in a battle situation, even in the rare event that such complete information is available, the time demands may be too difficult to fully apply the ration choice model.
2.4.2 Military decision-making process

The Army’s doctrine provides a seven-step decision-making process (FM 5-0 2004).

1. Problem Definition: Determine the scope, limitation and the cause of the problem. Consider the desired end state. Then prepare a problem statement.
2. Information Gathering: Learn facts and assumption associated with the problem. Commanders must ensure the relevant information is accurate.
3. COA Development: Develop various plans to solve the problem and determine which one suits.
4. COA Analysis: With an appropriate evaluation criterion, analyse alternatives.
5. COA Comparison: List advantage and disadvantage of each COA.
6. Decision: Select a COA or modify or modify it. Commanders may implement a completely different COA.
7. Execution and Assessment: monitor the execution to ensure meeting the success criteria and the desired end-state.

The problems of this process are its rigidity, and its large time requirements and resources to execute the whole process correctly. Meanwhile, it is not very suitable for ill-structured problems, because it best supports a process with a clearly defined objective and end-state.

2.4.3 OODA Loop (Observe, Orient, Decide, Act)

John Boyd’s OODA Loop, illustrated in Fig. 2.4, was introduced in 1987 as a way to model military decision making and has been accepted as a valid representation (Boyd 1987; Joint Staff 1996). The basic notion is that commanders observe, orient, decide, and act. They then observe the outcome and begin the cycle over again.

During the first stage of the OODA loop, the decision-maker must observe what is happening around him/her. He/she collects and synthesizes available data from a variety of sources to obtain situational awareness, which occurs in the second stage, the orientation stage of the loop. After the data is collected, it must be synthesized into information by the decision-maker. The decision-maker orients himself/herself to
the information he/she observed by creating a mental picture of the world around him/her, which is also supported by the RPD model. It is in this stage that a decision-maker uses the information and his/her own knowledge to recognize a situation as typical. The RPD model emphasizes the importance of situation assessment in expert decision-making. The mental image, formed during orientation, is influenced by the decision maker’s experience or recognition and it serves as the foundation for the decision making. At the third stage, the decision-maker weighs the information gathered, considers the alternative courses of action and makes a decision. This is consistent with the Rational Model. In stressful, complex, dynamic situations the element of time criticality is one of the most distinctive features of decision making.

They make a decision as soon as the minimum information is acquired. The last stage in the OODA loop is the action stage. The decision maker initiates some action based on the three previous stages. The action may include the decision not to act. Observation of the actions or inactions starts the cycle over again.

![Fig. 2.4 OODA loop](image)

The amount of time used by a decision-maker to cycle through the OODA loop is related to the cycle speed or size of the loop. Although there are some similarities between the OODA model and the RPD model, for example, both of them begin with observing the situation, both of them try to relate to past experience and both of them depend on feedback, RPD goes further: it also includes mental simulation.
2.5 Artificial Intelligence (AI) in Military decision aid systems

As a branch of computer science, AI is concerned with automation of intelligent behaviour. It tries to build machine that can act and react appropriately to its environment. It can be decomposed into a number of sub disciplines such as game playing, reasoning, search, expert systems, natural language understanding, computer vision, agents, modelling human performance, planning and robotics, machine learning etc. AI has been applied to many decision support systems, such as in the medical and engineering areas. However, it is not so easy for military applications. There are some obvious reasons, as follows (Dupuy 1988):

- The complexity of the scenario the human commander must face is high.
- No battle is ever exactly the same as previous combat examples.
- There is no living expert personally familiar with all or even most of various military scenarios.
- The weapons and equipment of today’s battlefields are different from those of past battlefields.
- The doctrine must change as the components change, but the validity can not be determined correspondingly.

In spite of this, the use of artificial intelligence in the military decision aid systems is quite wide spread. War gaming has been applied in the military for decision making operations for a long time. Germany used war games to plan the invasion to France successfully in 1940. Japan used war games to plan its attack on Pearl Harbour in 1941 (Peralta 1990). Besides war gaming, AI is also very popular in building military decision systems. Graph theory, Petri Nets, game theory and data mining have all been applied in this area (Boukhtouta et al. 2004). A good military decision support system must have quick response, and be adaptive to the changing situations. Several computational methods exist for implementing a military decision process (Boukhtouta, Bedrouni, Berger, Bouak, & Guitouni 2004). Neural networks, fuzzy logic, and CBR are techniques that have been employed to increase the robustness of military decision models and have succeeded in varying degrees. Multi-agent system simulation has just begun to be used to implement decision making in the military
domain. There are many decision support or decision making systems in the military area, some examples are as follows:

Dynamic Analysis and Re-planning Tool (DART) (BBN Systems and Technologies 1992) is a user interactive information system for a military human commander to deploy a large number of troops and equipment. It won the DARPA’s outstanding performance award for the rapid modification and transportation feasibility analysis.

FOX Genetic Algorithm (FOX-GA) based on the Genetic Algorithm technology (Schlabach, Hayes, & Goldberg 1998), generates a large number of potential COAs through crossover and mutation when there is not a sufficient number of COAs for the uncertain, dynamic battlefield. Then it uses a war game based on coarse grained representation to evaluate rapidly the fitness of the generated COAs. Since the standard Genetic Algorithm will group very similar solutions together, a niching strategy is applied in order to ensure diversity in this approach. The input to FOX-GA are avenues of approach, tactical assembly areas, terrain objectives, the forward edge of the battlefield and lines of defensive terrain.

OWL is a decision analytic COA war gaming to predict alternative outcomes of battle based on uncertain information available about friendly and enemy forces. It is a post -processor for FOX-GA. It executes the same war-game scenario repeatedly with randomly generated inputs which are derived from a defined probability distribution function (Uckun et al. 1999).

Modular Semi-Automated Forces (ModSAF) is a set of software modules and applications used to construct Advanced Distributed Simulation (ADS) and Computer Generated Forces (CGF) applications. It is perhaps the most widely used behavioural simulator in military synthetic environments. By using a finite state machine implementation it can generate a plan at the Company echelon level. Skeletal plans are input which represent the higher level orders to be transformed into a plan (LaBoissonniere 1999).

The Anticipatory Planning Support System (APSS) has been developed by the department of computer science at Texas A&M University to provide decision support for military operations (Hill, Surdu, & Pooch 2001; Hill, Surdu, & Pooch
In a traditional decision support system, only one single COA will be chosen for execution. In APSS, a genetic algorithm is applied to generate as many reasonable solutions as possible and dynamically modify and update them during execution. Invalid solutions will be pruned according to the actual state, and new ones will be developed and predicted by simulation before their execution state.

Most of these systems are very complicated and cost many man years of work. Furthermore, the military decision model used in the majority of these systems is rule-based model. Rule based knowledge representation has had significant success in addressing the needs of the military, especially in simulation community. However, these systems draw heavily upon a static set of preset rules lack any underlying model of human behaviour (Klein & Crandall 1996).

Responses to situations are limited to what has been incorporated in the rule sets, and are difficult to make realistically flexible when novel situations are encountered. Meanwhile, it is very difficult to define a set of rules that account for all decisions that a simulated military commander must make, models based on this approach tend to be too predictable, complex to understand, difficult to modify and inflexible to migrate from one scenario to another (Sharma 1996). For our project, it is impracticable to develop a huge system like these, and more importantly, we are more interested in an approach which can mimic the human learning process. Thus, in the following section, we shall review some related research about NDM, especially in military area.

2.6 Related AI researches about NDM

There are many researches about implementing NDM models with AI. Some examples are as follows:

2.6.1 RPDAgent

John Sokolowski and researchers at Old Dominion University's Virginia Modeling, Analysis & Simulation Center used multi-agent system to mimic the cognitive process identified by RPD, which is called RPDAgent. Using an operational military decision scenario to test model validity, decisions produced by RPDAgent were compared
against decisions made by military officers. It was found that RPDAgent produced
decisions that were equivalent to its human counterparts. RPDAgent's decisions were
not optimum decisions, but decisions that reflected the variability inherent in those
made by humans in an operational military environment.

In RPDAgent, the frame was chosen as the data structure to hold the major portion of
RPDAgent's experience (Sokolowski 2002). Each frame corresponds to a single
experience that holds all the cues, goals and actions describing that experience.
Environmental variables are assigned numeric values that represent a variable's
characteristic. The higher the value, the more favourably the variable influences the
cue. The cue value is calculated by summing the values of its associated
environmental variables. RPDAgent generates fuzzy values for each cue based on the
cue value. Each cue has three fuzzy sets associated with it, an unsatisfactory fuzzy set,
a marginal set, and a satisfactory fuzzy set. The higher the cue value, the more likely
it is to fall in the satisfactory range. Goals are another element of experience over
which the model user has control. Goals also have fuzzy sets associated with them.
These fuzzy sets define how well a goal is being satisfied. Because very rarely can a
decision maker find an action that will completely satisfy all his or her goals,
experience tells a decision maker how far he or she can compromise on a specific goal
and still arrive at a satisfactory decision. RPDAgent simulates this process. It
instantiates a reactive agent for each goal in the frame. Each reactive agent is
responsible for evaluating the attainment of its assigned goal. When one or more goals
are not evaluated as satisfactory, the reactive agents try to negotiate a compromise by
lowering the standards by which they evaluate their goals (Sokolowski 2003b).

The shortcoming of the RPDAgent is that it only has a fixed set of actions, so it only
can choose the actions from that set. Also, RPDAgent should have the potential to
learn based on its decisions, but it still lacks a proper learning methodology.

2.6.2 Adaptive BDI

Belief-Desire-Intention (BDI) is a model about human practical reasoning, developed
by Michael Bratman as a way of explaining future-directed intention (Bratman 1987).
It has been popular used in military applications. In many military simulation systems,
BDI agents were used not only for operations analysis, but also for simulating the
roles of team members or opponents of the human operators. However, it generally used a fixed plan library, so it is not adaptive, can not support agent learning, in order to improve the adaptive ability, researchers try to propose some update version (Norling 2003).

Emma Norling and other researchers in the University of Melbourne’s Intelligent Agent Laboratory try to use BDI to model some expert player of Quake 2. They adapt Applied Cognitive Task Analysis (ACTA) to gather the knowledge that the character need to operate in the world. One of the advantages of using the BDI framework for modelling human is that the domain experts could naturally express their knowledge in a way that can map to the framework, so the plan library can be built up very easily, and the experts can understand the plans very easily. They focus on the belief part of BDI agent by looking at various methods to interpret subtle contexts among decision situations, the system is developed based on JACK (Norling & Ritter 2001).

There are also other related research projects, such as OASIS air traffic management system prototype by Lucas, Ljungberg and others for Airservices Australia (Ljungberg & Lucas 1992) and the SWARM, air mission simulation system by Tidhar, Lucas, Rao and others for DSTO Australia (Lucas et al. 1992; Rao et al. 1993; Tidhar, Selvestrel, & Heinze 1995).

BDI is a powerful model about human reasoning. However, in the BDI model, an agent will only focus on a subset of desires that are consistent and commit resources to achieving them. The concept of a BDI agent refers to a single agent that is controlled by its BDI rules. BDI does not enter in to the individual behavior of its component agents. Rather, it governs the overall behavior of the MAS as a meta agent. However, the real military circumstances are much more complex, so the update more detail version of BDI is needed.

2.6.3 Composite Agent (CA)

Researchers in Naval Postgraduate School have developed Composite Agent (CA) (Hiles 2001). It is composed of a combination of cognitive Symbolic Constructor Agents (SCA), which is sensing and interpreting the environment, then build a symbolic inner environment, and Reactive Agents (RA), which select action based on inner environment. Each one of them is responsible for one particular CA behaviour.
CA can simulate an individual decision maker with a personality. SCAs sense the CA’s external environment and convert it to an internal representation of the decision situation. RAs then act on this internal representation to choose a set of actions consistent with furthering the goals that the CA has selected. These goals are based on the characteristic behaviours of the CA. Through the interactions of the RAs while pursuing their goals, a complex pattern of behaviour emerges that represents decision-making in a descriptive form (Hiles 2001).

RPD tries to determine whether a situation closely matches past experience or not, which is same as the function of SCA. Also SCA control and filter information in order to avoid the overwhelming sensory input for CA, which is very similar to the function of cues in the RPD model. Goals guide CA to achieve the desired end state by choosing the suitable action. It also has a simple reactive learning process which match the components of the RPD model. There appears to be a significant overlap between the characteristics of the RPD and CA. CAs perform situational recognition and use cues to filter environmental inputs. Clearly, CAs are goal driven with goal satisfaction playing a central role in determining the set of actions to be carried out in support of a decision. CAs monitor expectancies to ensure that they can adapt to any changes in the decision situation. CAs focus on determining one set of actions rather than analyzing multiple COAs. These points are the key characteristics of the RPD model. They are embodied by the CA, which makes CA technology a viable choice to implement and mimic the RPD decision process in a computational form (Hiles 2001).

2.6.4 Long Time Memory (LTM)

Because RPD is a theory of experienced decision making, Water Warwick (Warwick et al. 2001) employs a computational analogue of long-term memory directly from Hintzman’s multiple-trace memory model. The basic idea is that each experience an agent has leaves behind its own trace, even if that experience happens to be exactly like another experience the agent has had. A multiple trace memory is a collection of episode tokens rather than a store of episodic types. Recognition in a multiple trace model is a process of comparing a given situation against every trace in memory, computing a similarity value for each trace and then using these values to form an “echo.” Each row in LTM contains a bit string and a floating point number. For example, in their model, the bit string is 23 bits long, the first 14 bits (0-13) encode
features of the situation, the next four bits (14-17) encode expectancies the agent has about the situation and the last five bits (18-22) encode the associated course of action.

Recognition begins when a probe is sent to LTM. Intuitively, the probe is a snapshot of the current situation. When the probe is sent to LTM, three things happen. First, the probe is compared to each row in LTM. This comparison yields a similarity value for each row in LTM. Second, they use the similarity value to compute an activation value for each row in LTM and use those values to construct an echo. Third, the echo is analyzed to determine what has been recognized.

Let $p$ be a probe to LTM and let $t$ be a single row of LTM where $p_i$ and $t_i$ are the $i^{th}$ bits of the probe and row respectively. Suppose a situation is encoded by $k$ bits, then the similarity value, $s_{p,t}$, between $p$ and $t$ is given by:

$$s_{p,t} = \frac{1}{k} \sum_{i=0}^{k-1} p_i t_i$$ (2.1)

The echo itself is initially formed as a vector of nine floating point values which are then coerced into integer values after some analysis. The first four values represent expectancies while the last five values encode the COA. Each row in LTM contributes to the echo according to its activation value, which is simply the cube of the similarity value. The larger the exponent, the larger the relative contribution of the most similar rows in LTM.

The computation of a particular value in the echo vector, $e_i$, depends on whether the value represents an expectancy or whether it is used to encode the COA. Suppose $e_i$ represents an expectancy. Let $t_{ji}$ be the $i^{th}$ bit of row $j$ in LTM and let $a_j$ be the activation value for row $j$. If there are $n$ rows in LTM, then:

$$e_i = \sum_{j=1}^{n} t_{ji} a_j$$ (2.2)

If $e_i$ is one of the values used to encode the COA, then its value is given by:

$$e_i = \sum_{j=1}^{n} t_{ji} a_j r_j$$ (2.3)
where $r_j$ is the success value of the situation-COA pair represented in the $j^{th}$ row of LTM

However, it is not reasonable to model the military situation with the LTM, it seems too simple and abstract, difficult to scale to large knowledge bases. Given the complexity of military scenarios, a much more detail model is needed.

Other related research including Robichaud (Robichaud 2001) and Liang (Liang 2001) work about developing a general-purpose computational fuzzy RPD model that utilizes fuzzy sets, fuzzy rules, and fuzzy reasoning. A fuzzy interpretation of the external environment was added since humans internalize their external environment in a personal way. Also, perfect battlefield information is rarely available and must often be interpreted for its meaning. This enhancement added another layer of humanism to this implementation of RPD.

2.7 Implement NDM Model with CBR

RPD model provides an example of NDM. Although not synonymous with NDM and certainly limited in what it attempts to explain, the research on RPD did demonstrate that people can make reasonable decisions without having to perform extensive analysis and compare the strengths and weakness of the COAs. People use their experience to recognize the situation that they have previously encountered and evaluate the COA by mental simulation.

Although RPD model has many advantages, there are multiple limitations associated with it as well. RPD model does not address all the concerns of NDM, such as the influence of team and organizational constraints. It is lack of empirical base, the neglect to specify cognitive process, its lack of prescriptive guidance as a descriptive account and finally its inherent inability to distinguish between good and poor decisions or to identify errors (Klein 1997) and in the event of adoption of a complex strategy, the RPD model also refrains from specifying what happens when people do compare COA and more importantly, it does not take into account the generation of new COA. There are similarities between CBR and RPD, however RPD has some special characteristics, and pure CBR cannot capture all the aspects of the RPD
process. Therefore, in order to solve these problems, in this section, we propose to apply CBR to implement a NDM model, which is shown by Fig.2.5.

There must be a representation of the environmental cues required for situation assessment, a pattern matching mechanism that allows situations to be recognized on the basis of environmental cues, and a collection of situations that embody the knowledge gained through accumulated experience.

**Collection of Situations:** The collection of situations represents the memory of previous experiences. Each scenario and its COA, represented by environmental cues, as well as the corresponding output of it are recorded in the case base. The output of each case is described by a winning value $W$ as how successful the COA is.

**Environmental Cues:** Military scenarios are modelled as continuously processing environmental data. At each time unit, situations are recognized on the basis of patterns present within combinations of these cues. Environmental cues can include variables explicitly represented within the simulation (e.g., $x$, $y$, $z$ coordinates of entities) and variables derived through computational processes. Those environmental cues should be continuously monitored, using pattern matching to recognize the situation, as well as changes to the situation. In Chapter 4, scenario representation is discussed in detail.

**Pattern Matching:** Recognition of situations is a pattern matching process. Each known situation has a pattern of environmental cues associated with it. Pattern matching involves a continuous process of matching environmental cues with known situations. Where environmental cues form a pattern resembling that of a known situation, the corresponding COA is chosen.

Therefore, in our project the case base will be clustered first. Detail clustering approaches are discussed in Chapter 5. The clusters might be arbitrary shape, not necessarily spheric, each with fixed boundary surfaces which will be defined by cluster algorithm. Therefore it is unavoidable that some parts of the case space remain uncovered by any cluster. There are then three possibilities, as schematically shown in Figure. 2.6 for a 2-dimensional case space.
In the first situation, as shown by Figure 2.6 a, the new case C is in one of the clusters. So the case is recognized as an example of one of the prototypes. As the first hint, the system returns the case which is the most representative of this cluster, namely the most centrally located case in this cluster. The winning value $W$ of this central case is chosen as the mental simulation result for this new case, so the commander gets an immediate idea of the chances of success in such a situation. Then adaptation will work out the proposed solution for the new case C.

In the second situation, as shown by Figure 2.6 b, the new case C is not in any cluster, in order to simulate the process how the human commander applies to choose a satisfactory COA, a singular evaluation value $S$ is defined, which is a function of the winning value $W$ of the central case of each cluster and the distance between the corresponding central case and the target case. Each cluster is evaluated in this way. Once an $S$ bigger than a threshold is found, we consider that the commander’s expectation is achieved, and the corresponding cluster is chosen as a satisfactory prototype, not necessarily the optimal one. Meanwhile, this target case is included in the corresponding cluster, and the size of this cluster increases. This process would be akin to undertaking Level 2 recognition wherein there is insufficient familiarity with the pattern of environmental cues to recognize the situation through Level 1 Recognition. The commander have never met this problem, but his expectation can be achieved by one of the prototypes, thus this new problem will be treated as an instance for this prototype in the future.

The third situation, as shown by Fig 2.6c, is similar to the second one. However, now none of the singular evaluation $S$ is bigger than the predefined threshold of acceptability after the mental simulation. This means that the commander’s expectation can not be achieved by current experience, as he faces a problem which he has not met before. As we mentioned, RPD has advantages which avoid having to generate novel responses from scratch. But in the third situation, logical deliberation is needed. The only option we have is to create new cases to populate this void in the case space.
Mental Simulation: In NDM, the decision maker uses mental simulation to help diagnose a situation, to identify whether or not their outcome is desirable. By mentally examining various aspects of the elements in the situation, he/she can decide whether the situation is familiar or not. In the first situation of our cluster result, we propose to use the winning value $W$ of this central case is chosen as the mental simulation result for this new case, so the commander gets an immediate idea of the chances of success in such a situation. Meanwhile, we use case visualisation in our project to give commanders intuitive insight about their case base, to examine various aspects of the cases. Case visualisation is discussed in detail in Chapter 6.

Our three variations are similar to the three level of RPD, however there are differences. In first RPD level, when the expert recognises the situation as typical, it assumes the single course of action is obvious. However, RPD does not consider adaptation. In real-life scenario, once those relevant experiences are retrieved, none of them may match the current situation exactly. But they can give the decision maker a schema that indicates which aspects of the current situation are most important. The decision maker can then proceed to adaptation based on this schema. In the event that the situation is not immediately recognised, then the expert actively seeks information to find cues and features that may reveal the nature of the situation. In the third RPD level, it still does not allow the decision-maker compares alternative COA. It evaluates using mental simulation to confirm that it will work. If there is doubt, the COA is modified, or reassesses the situation. However, RPD is not always the best model for all decision environments. There are situations require an analytical approach.

In our project, because of limited time and resource available, only the first situation will be discussed in the rest of the thesis. Actually, once the case base is large enough, the target case usually occurs in the first situation. This is similar to a very experienced human commander, facing those familiar situations, so he does not worry about unexpected new problems.

Fig 2.5 describes our proposal to apply CBR to implement a NDM model. When new case occurs, it will be clustered first, and then different approaches are processed according to the result. If it is not a typical cluster, and singular evaluation $S$ is bigger
than the predefined threshold of acceptability, then the situation is recognised, then case will go to next step, same as those case who is recognised as typical after clustering. If none of the singular evaluation $S$ is bigger enough, then this case is a situation the commander never met before, analytical approach is needed. Case filtration can help us to discard those low quality cases. It can be processed before the case clustering actually. However, because cases in each cluster are more similar, it can achieve a better result if applied on clustered cases. The detail of the case filtration approach is discussed in Chapter 7.

![Diagram](image)

**Fig. 2.5 Implementing NDM with CBR**

Case visualisation is applied, not only can it help us to maintain a better quality case base but it will also provide a detail mental simulation approach for the decision maker. After case adaptation, the adapted result is evaluated and then added to the original case base.
2.8 Conclusions

Researchers have recognised that environments such as the military, fire fighting, and emergency services present a specific set of conditions relevant to the generation of appropriate decision theories. These environments have been described as naturalistic environments. Naturalistic environments are complex and dynamic, with time pressure, high stakes, and uncertainty. It has been recognised that the specific conditions require a new theory of decision-making, which has been termed naturalistic decision-making. Within the relatively new field of NDM several decision-making models have been presented.

The British military has been conducting experiments and demonstrated its validity (Blendell et al. 2008; Pascual et al. 2008). Through various researches about military C2 decision making, The UK Defence Research Agency (DRA) found NDM are applied dominantly (87%) over classic (2%), hybrid (3%) and other (8%), when the commanders are familiar with the scenarios. It is believed that a more novel scenario would result in much greater utilization of classical or hybrid strategies, due to the lack of experience with the situation (Pascual & Henderson 1997a).

The most commonly cited model proposed to describe decision-making in naturalistic environments is the RPD model. It is a descriptive model that explains how experienced decision makers work problems in high stress decision situations. For military RPD, a commander’s knowledge, training, and experience generally help him/her correctly assess the situation and develop, mentally war game a plausible COA, rather than taking time to deliberately and methodically compare it with
alternatives by a common set of abstract evaluation approaches (Fallesen & Pounds 2001; Klein 1998b; Pascual & Henderson 1997b).

From the perspective of effectiveness, however, RPD is not always the best model for all decision environments, since it has some disadvantages. Therefore, we propose a model to implement NDM with CBR. Each single situation is represented as a case. The case base is clustered, filtered, visualised. Adaptation is applied of generating new COA based on the experience. Once evaluated, the result is put back into the case base, thus demonstrating it has the ability to learn.

However, we are aware, there are still situations that require the careful deployment of resources and analysis of abstract data, such as anticipating an enemy’s COA, still require an analytical approach. If there is time for analysis, a rational process normally provides a better solution for these kinds of problems. The NDM can complement the analytical approach. Neither is appropriate for all decision problems, both of them have their own virtue.
Chapter 3

Case-Based Reasoning

In this thesis, we propose a model to implement NDM with CBR (Case-Based Reasoning). Each single situation is represented as a case. Cases in the case base are clustered, filtered, visualised. Meanwhile, adaptation is applied to generate new COA based on the experience. Once evaluated, the result then would be included in the case base. The whole process includes the CBR cycle. Thus in this chapter, we shall introduce some background about CBR.

CBR is a problem-solving paradigm which originated in the US. It has become an increasingly popular methodology over the recent decades. CBR methodology aims at enhancing knowledge acquisition, knowledge maintenance, efficiency of problem solving, quality of solutions, and user acceptance (Leake, Kinley, & Wilson 1996). In recent years, CBR has made great progress in both theoretical researches and practical applications. Great success has been achieved in the many domains, such as Medical Diagnosis, Help Desks, System testing, Electronic Commerce, Decision Support to Argumentation, Legal Reasoning. These applications can be classified as shown by Fig.3.2. However, the most successful application of CBR technology has been in the field of “help desk” applications (Kriegsman & Barletta 1993). These applications act as intelligent retrieval systems for information which is grouped together as a case. It relies on the ability of the CBR system to effectively and efficiently retrieve cases.

In Section 3.1, we introduce what is CBR. CBR cycle is described in Section 3.2. In Section 3.3, we discuss the classification of CBR applications. Section 3.4 describes the related CBR military applications. Finally, in Section 3.5, we summarise the whole chapter.
3.1 Introduction

The origins of CBR come from both cognitive science and artificial intelligence. It was stimulated by research about how people can solve problems by remembering the way they solved similar problems in the past. Similarly, CBR was developed as a methodology that judges a new problem by comparing and adapting relevant cases in a case base, in which each case describes a problem and a solution to that problem.

In order to conduct the research to follow the model of human reasoning in cognitive science, Roger Schank and Robert Abelson at Yale University formalise the idea of human problem solving by introducing a notion referred as a script, which is defined as a structure used in the conceptual memory that holds information about stereotypical situations (Schank 1982b). The general human knowledge about situations is organised in the form of scripts, depend on what humans found their expectations and draw conclusions. These considerations form the origins of CBR. Schank later proposed a dynamic memory model in which reminding has a significant role in problem solving and learning. It has been noted that people analyse the problems and create solutions in the context of prior experiences. Instead of dealing with the problem in an isolate manner, people rather place a new problem in a similar context previously experienced and construct the solution based both on the current problem specification and useful information extracted from prior experiences that can facilitate finding a solution to the new problem. This is how the idea behind CBR originated.

The first CBR system based on the Schank’s dynamic memory model was developed in 1984 by Janet Kolodner, which is called CYRUS (Computerised Yale Retrieval and Update System). It stored and retrieved events such as travels and meetings of Cyrus Vance during the period in which he was the US secretary of state (Kolodner 1993). Kristian Hammond developed a CBR system called CHEF in 1986, whose task was to create recipes (Hammond 1986).
Apart from being an approach to problem solving, CBR may be also regarded as an approach of machine learning. Thus, it is also regarded a sub-field of machine learning, learning in CBR is a derivative of the actual problem solving. Every attempt to solve a new problem represents a new experience. If the attempt is successful, then the experience can be incorporated into the CBR system so it could be reused for future similar problems, otherwise the explanation of the failure can be stored in order to overcome similar obstacles in the future. In this manner, a CBR system can learn from both success and failure. This form of learning is easier to implement than is the generalisation of concrete experiences. (Aamodt & Plaza 1994b; Hunt 1999).

CBR is fundamentally different from other major approaches in the field of Artificial Intelligence (AI) in the way in which the problems are solved. Traditionally, AI methods tackle the problems in a generalised manner by trying to provide a theoretical framework, which would be usually applied in the form of rules. CBR aims to counter the disadvantages of problem solving in traditional knowledge-based systems. For example, rule-based approaches lack robustness and flexibility, they are restricted to narrow problem domains and difficult to maintain and update throughout their life cycle. Meanwhile, rule acquisition is usually time consuming, and sometimes can be unreliable. Instead of relying on general knowledge of a problem domain, CBR solves new problems based on the previous experiences. Furthermore, another important difference between CBR and other AI approaches is the ability of incremental sustained learning. When a problem is solved, the solution and problem description will be added to the case base, thus making it available for solving future problems. Also, when an attempt to solve a problem fails, the reason for the failure is identified and recorded in order to avoid the similar future occurrences, which is why we think CBR has the ability to learn from the previous experience. As noted by Watson (Watson 1999), case-based reasoning is a general methodology, as distinguished from rule-based reasoning and neural networks, which are more correctly viewed as technologies. The distinction is based on the observation that case-based systems can utilize many different AI technologies in both the case retrieval and case adaptation phases of the CBR process.

Previous work has shown that CBR provides a number of advantages over alternative approaches (Pal & SHIU 2004):
• CBR doesn’t require extensive analysis of domain knowledge. In model or rule-based systems, a model or a set of rules need to be extracted. In CBR, the main task becomes the collection of relevant existing cases and their representation and storage.

• CBR permits problem solving even if domain knowledge is incomplete, or not fully understood, defined. Due to their rigidity in problem formulation and modelling, model-based systems sometimes can not solve a problem that is on the boundary of their knowledge or scope or when there is missing or incomplete data. In contrast, CBR systems can use the previous experience as the domain knowledge, and provide reasonable solutions. The most important thing is to know how to compare two cases. If insufficient knowledge exists to build a model or to derive a set of heuristics, CBR system can still be developed by using only a small set of cases from the domain. The fundamental theory of domain knowledge does not have to be quantified or understood entirely. While in a problem domain where only a few cases are available, CBR system can start with these few known cases and build its knowledge incrementally as cases are added.

• CBR allows shortcuts in reasoning. In CBR, if a suitable case is found, a solution can be proposed quickly. For problem domains that require significant processes to create a solution from scratch. The alternative approach of modifying a previous solution can reduce the processing requirement significantly. Additionally, reusing the earlier solution also allows the actual steps taken to reach that solution to be reused for solving other problems.

• CBR can lead to improved explanation capability in situations where the most comprehensible explanations are those that involve specific instances. In most domains there will be occasions when a user wishes to be reassured about the quality of the solution provided by the system. By explaining how a previous case was successful in a situation, using the similarity between the cases and
reasoning involved in the adaptation, the CBR system can explain its solution to the user.

- CBR can help avoid past errors and learn from the errors and success. In CBR, the system keeps a record of each situation that occurred for future reference. It records failures as well as success, and perhaps the reason for those failures, information about what caused failures in the past can be used to predict potential failures in the future.

For CBR, knowledge acquisition is made easier as there is no need to transform experience into rule. It boils down to simple recording previous experience. Knowledge maintenance is also simplified as new cases can be added any time. The addition of new case can be also automatised as CBR systems have the ability to learn gradually from new experiences. The efficiency of problem solving is improved because the result of prior reasoning is taken as a starting point for solving a new problem, avoiding the computational expense of prior reasoning that lead to the result. The quality of solutions improves significantly in domains that are ill-structured in which no complete models or rules can be easily defined. Users more readily accept CBR systems because of the fact that cases provide an explanation for a particular choice of solution by presenting a context in which a similar solution produced satisfactory result.

There are several distinctive subtypes of the generic CBR approach. In exemplar-based reasoning, a concept is defined by means of its representative examples. Solving a problem means to classify an unclassified exemplar. No adaptation is needed. Instance-based reasoning is syntactically oriented. An instance is required to redeem the semantics of the problem. In memory-based reasoning, memory is modelled as a collection of cases. The reasoning process involves an introspection of the memory. In a more specific sense, Case-based reasoning is characterised by an adaptation of a retrieved solution for which a general knowledge about the problem domain is needed. Finally, analogy-based reasoning is similar to case-based reasoning. The difference is that it crosses the domain boundaries, however case-based reasoning normally is restricted to single domain cases.
3.2 CBR cycle

The fundamental assumptions of CBR are (Leake, Kinley, & Wilson 1996):

- Regularity: similar problems tend to have similar solutions;
- Recurrence: types of problems tend to reappear, either frequently or only periodically.

As discussed, the advantage of CBR over rule-based systems refers to easier knowledge acquisition and the ability to learn from experience. Unlike model-based systems, CBR systems exploit not only general but the specific knowledge as well. The specific knowledge can be gained through experience and is not restricted only to the development stage of a system. New experience may be gained at each attempt to solve a new problem. The specific knowledge is stored in the form of cases. Cases are stored in a manner that supports future exploitation of the lessons learned when solving the corresponding problems.

CBR may be viewed as a multi-stage cycle which involving following four "re-" processes (Aamodt & Plaza 1994b), as shown in Fig. 3.1.

1. RETRIEVE the most similar cases.

During this process, CBR application will search the case base to find the most approximate cases to the current situation. One or several cases that are considered useful for solving the new problem are selected. Cases are selected based on the similarity of the problem description between the new problem and the cases contained in the case base. It is generally assumed that similar problems should have similar solutions. When the case representation is complicated, similarity assessment can be a very difficult and computationally expensive task. In addition, when the size of case base increases, the efficiency of retrieval decreases because an increasing number of cases must be taken into account to find the most similar cases from the case base. Therefore, many CBR systems apply indexing in order to retrieve cases more efficiently.
Case retrieval is a process in which a retrieval algorithm is used to retrieve the most similar cases for the solution of a current problem. How to efficiently retrieve the most similar case to a current problem from the case base usually is composed of two problems:

- How to retrieve a set of similar cases from the case base;
- How to find the most similar one in this set.

In the Case Retrieval step, for each field there is a local similarity measure determining the similarity between two fields. Each case will have a global similarity measure determining the similarity between two cases based on local similarities of the fields.

\[
\text{Distance (} T, C \text{)} = \sqrt{\sum_{i=1}^{n} [T(x_i) - C(x_i)]^2}
\]  \hspace{1cm} (3.1)

Where  
- \( n \) is the number of features,
- \( x_i \) is a feature,
- \( T(x_i) \) is the value of feature \( x_i \) for the target case,
- \( C(x_i) \) is the value of feature \( x_i \) for the query.

In many situations, the different parts of the case can be weighted to indicate their relative significance to the overall match. Here is a typical evaluation function used to compute nearest-neighbor matching

\[
\text{similarity}(T, C) = \frac{\sum_{i=1}^{n} w_i \times \text{sim}[T(x_i), C(x_i)]}{\sum_{i=1}^{n} w_i}
\]  \hspace{1cm} (3.2)

Where \( w_i \) is the importance weight of feature \( x_i \), and \( \text{sim} \) is the similarity function of target feature and the compared case.

Similarity assessment is one of the key issues in CBR (Leake & Wilson, 1998). The task of similarity assessment is to select cases based on the resemblance existing between two cases, and depends on the problem domain and a chosen case.
representation. The methodology is based on the hypothesis that the degree of similarity reflects the extent of adaptation needed to produce a solution to the new problem by transforming a solution to an old similar problem. Generally, a collection of initially match case rather than a single case results form retrieval (Admodt 1994). The elaborate matching is then performed to provide more thorough evaluation of these cases with respect to the new case. The result of elaborate matching is the ranking of initially matched cases.

Algorithms like fuzzy logic can be used as a basis for the similarity measure. Other approaches may include Nearest Neighbour, Decision tree or Neural Networks etc. The accuracy of the extraction process of similar cases to a given problem determines the success of the CBR system. If the system cannot find the best matching cases, the solution suggested by these cases will not be the most appropriate one for the given problem. The Nearest Neighbour Algorithm is extensively used in CBR systems for robust case extraction. It measures the Euclidean distance between attributes of the Target case and cases in the case memory. In some cases nearest neighbour matching can be improved further by utilizing domain knowledge. However, straight nearest neighbour matching is only really feasible on relatively small case memories, because this approach requires that every case is compared with the input scenario. It is infeasible for large case memories. After a matching case is retrieved, we can adapt the solution of the retrieved case to fulfil the needs of the current case. After evaluation, the current case with the new solution will be stored in the case base.

Using the most similar case to solve a problem can often cause CBR systems susceptible to noisy cases. This problem is overcome to a certain extent by generating a solution based on a number of similar cases rather than just one. Another approach is to remove the problem cases from the case base. There are some researches about case editing (Brighton & Mellish 2002; Delany & Cunningham 2004; McKenna & Smyth).

2. REUSE the cases to attempt to solve the problem.

Once a matching case is a retrieved, CBR system attempts to reuse the solution of retrieved case in order to derive the solution for the new case by adapting the
retrieved case. This process includes using the retrieved case and adapting it to the new situation. The simplest way to use a retrieved case is just simply copying the unchanged solution of that case. However, sometimes the solution which had been applied to the cases should be adjusted accordingly through an adaptation process. The reuse phase can become computationally expensive, especially when the case representations are complex and the solution part of the case requires detail description. Generally, the effort involved for adaptation depends on the similarity between the retrieval cases and the new case. The more similar the cases are, the less effort the adaptation should cost. On the other hand, Derivational Adaptation (Wilke & Bergmann 1998), instead of applying rules or formulas directly to the solution stored in cases, try to generate the original solution in order to produce a new solution to the problem.

Most commercial systems simply provide solutions from retrieved cases and present to the user. The user then, extracts the relevant parts of the solution, manually, to form the new case.

3. REVISE the proposed solution if necessary.

There is not so much discussion about this step in the CBR literature. In many practical applications, the reuse and revise stages are sometimes difficult to distinguish, thus many researches may use ‘adaptation’ to replaces and combines them. Since the proposed solution could be inadequate, we need revise the solution, which usually evaluate the case solution generated by reuse. If it is successful, then learning from the success, it is retained in the case base. Otherwise, an opportunity for learning from failure arises, and then this process can correct the first proposed solution. The revision typically is not the task of the CBR systems. Because the solution determined by the CBR system is verified by the real world and possibly revised by the human domain experts, then the revised case is re-enter into the CBR system for its use in the subsequent retain phase.

4. RETAIN the new solution as a part of a new case.
This process enables CBR to learn and create a new solution and a new case that should be added to the case base in order to be used for future problem solving. The retain phase is the learning phase of the CBR system. The learning from success or failure of the proposed solution is triggered by the outcome of the evaluation and possible repair. It consists of selecting which information from the case to retain, in what form to retain it, and how to index the case for later retrieval from similar problems, also including how to integrate the new case in the memory structure. By adding the revised case the case base, the new problem solving experience becomes available for reuse in the future problem solving. Machine learning methods can be used in CBR system about the similarity measure and the solution transformation as well as, techniques from statistics and information theory also can be integrated. The more cases installed in the case base, the easier should the future case retrieval and adaptation should be, some first CBR systems stored every newly solved case ((Leake, Kinley, & Wilson 1996). However, there are also drawbacks as well. It may lead to the explosive growth of the case base. The continuous growing of the case base will result in decreasing retrieval efficiency. On the other hand, it is not necessary to store each solved cases in order to provide adequate case coverage. Moreover, the quality of case base is not linked only to its size. It also depends on the diversity of the cases stored in the case base, which in turn depends on the similarities that exist between them (Aamodt & Plaza 1994a). To avoid this problem, strategies are required for selectively adding new cases to the case base, and forgetting case that have been already stored in the case base. It should be noted that the revise and retain stage maybe performed anywhere from moments to many months after the start of the cycle. This is due to a possible time delay for the effects of applying a solution to take effect. For example in a medical diagnosis task, the success or failure of a particular treatment may not be known for months afterwards.

In summary, the CBR cycle starts with the description of a new problem. This new problem is matched against previously experienced cases by retrieving one or more similar ones from case memory, then reusing the solution part of the retrieved cases, and giving a suggested solution or revising the solution, retaining the repaired case and incorporating it into the existing case base. However, this cycle rarely occurs without human intervention that is usually involved in the RETAIN step. Not all
solved cases need to be stored in the case base. For example, if the case is redundant or if the process providing its solution was a trivial one. In addition to a solution to the new problem, any other information that may facilitate future reasoning about similar cases may be retained as well.

Fig. 3.1 CBR Cycle (Aamodt & Plaza 1994b)

Reinartz et al. propose to extend two additional "re-" steps for the original four steps of Aamodt and Plaza’s CBR cycle (Reinartz, Iglezakis, & Roth-Berghofer 2001). A step called review to monitor the quality of system knowledge, and a restore step which selects and applies maintenance operations. Their revised model, emphasizes the important role of maintenance in modern CBR and indeed proposes that the concept of maintenance encompass the retain, review and restore steps (Iglezakis, Reinartz, & Roth-Berghofer 2004).
3.3 A Classification of CBR Applications

CBR can be applied to extremely diverse application domains, because there are many different ways to represent, index, retrieve and adapt in CBR system. Generally, CBR applications can be classified into classification and synthesis two categories.

![Classification Hierarchy of CBR Applications](image)

**Fig. 3.2** A Classification Hierarchy of CBR Applications (Althoff et al. 1995)

Classification tasks are very common. A new case is matched against those in the case-base from which an answer can be given. The solution from the best matching case can be reused. Most commercial CBR tools support classification tasks.

Synthesis tasks attempt to get the new solution by combining previous solutions and usually many constraints will be applied during synthesis. Therefore they are more difficult to implement compared with classification counterpart. CBR systems that perform synthesis tasks must make use of adaptation and are usually hybrid systems combining CBR with different AI techniques (Watson 1997)

3.4 CBR in Military applications

As discussed, rule based systems have drawbacks such as difficulties of knowledge acquisition. The knowledge representation of CBR consists of cases. It is easier to
acquire cases than to obtain rules. The intensive development of CBR in the USA can be attributed to the massive investments in CBR. Defence Advanced Research Projects Agency (DARPA) motivated the development of CBR starting in 1980s by organising a number of conferences and providing financial support. Thus, some systems which have tried to apply CBR in the military area appears (Dupuy 1988; Goodman 1989). Dupuy(1988) thinks only military history can provide the cases for a CBR military decision making system. Only the military history can provide the richness of experience necessary for situation assessment and testing. They propose a methodology called ACEDACED, which is composed as follows

- Assessment of current situation: friendly, enemy, environment
- Cases from Historical combats
- Experts’ Distilled Wisdom Review
- Doctrine Review
- Analysis
- Courses of Action Possible
- Evaluation
- Decision

Their data come from the Land Warfare Data Base (LWDB) of 605 battles in engagements between 1600 and 1973, which was developed by the HERO Division of DMSI. They tested their system by using 30 modern military scenarios.

The Battle Planner was built using a frame-based CBR Shell (Goodman 1989). They also used the Land Warfare Database as source data. It provides two main interfaces for knowledge engineering domain knowledge. The first is used to create symbolic hierarchies; the second is used to derive new features from existing features. The first version of Battle Planner used a nearest-neighbour algorithm for case retrieval, but they applied inductive discrimination analysis similar to the ID3 algorithm later (Quinlan 1986).

To validate the system, a randomly selected 10% of the available cases were set aside before indexing. Then they were treated as hypothetical battle situations. If the
predictions were accurate the cases were scored as hits, otherwise, they were scored as misses. Battle Planner is 81.3% accurate at predicting the victor of a case from the LWDB.

Joint Assistant for Deployment and Execution (JADE) (Mulvehill & Carioli 1999) is a planning tool that applies to military force deployment in order to retrieve and reuse force modules from previous plans whose force capabilities and composition satisfy the current situation. It is developed by BBN Technologies under contract to the Air Force Research Lab and the DARPA.

Keirsey et al. (Keirsey et al.) describe how CBR can be integrated with behaviour based control techniques to maintain and execute a large variety of manoeuvre tactics for computer generated F-14 entities. They use cases to achieve knowledge acquisition. Using behaviour expressed as concurrent control laws, knowledge acquisition may be performed at a high level of tactical manoeuvres rather than at a lower level of determining heading, velocity and altitude commands.

HICAP (Munoz-Avila et al. 1999) is a case-based tool to assist a military commander with planning NEOs (Noncombatant Evacuation Operations). It integrates a hierarchical task editor, HTE, with a conversational case-based planner, NaCoDAE/HTN. The former allows users to edit doctrine tasks and select tasks to operationalize, while the latter allows users to interactively refine HTN plans. While the doctrine describes general aspects of planning, experiences from previous operations can give more detailed information suitable for the current situation.

3.5 Summary

In this chapter, we introduce the background knowledge about CBR. There are various advantages of the CBR approach. Knowledge modelling, knowledge acquisition and learning are all become easier in CBR. The fact makes CBR suitable for problem domains for which rule-based approach is not appropriate due to the difficulty of formulating the rules or the quantity of the rules. In CBR domain experts explicitly present specific examples. Thus no intermediates are needed between the
domain expert and system designer. Explaining the results is clearer, since the suggested solution is explained by means of a similar example contained in a case whose solution is adapted to fit the new problem description (Riesbeck, 1988). A CBR system gradually improves its competence over time as new experiences are gained. This is convenient for problem domains for which the existing theoretical knowledge is not certain or complete, such as military domain. In addition, a human can understand a CBR system's reasoning and explanations and are able to be convinced of the validity of the solutions provide by the system.

As previously mentioned, in our research, we are more interested in applying CBR to military decision support area, which is very demanding and ill-structured, complicated area. In particular, our main concern is how to use CBR to find the suitable COAs for a specific scenario. In Chapter 4, we discuss how to represent a military scenario as a case. There we will find it is a very different case base as not only the case problem space is high dimensional but the case solution space is high dimensional as well. Especially the individual dimension which cannot be treated separately because they might relate to each other. This makes case adaptation very difficult, normal approaches cannot be used directly on each individual dimension separately. One simple solution might be to collect as many cases as possible, then for each target case, a nearest neighbour case can be found. Its solution can then be used for the target case without any adaptation. Obviously, it is not possible in many condition, such as, there may be time limitation or other restrictions to collect so many cases, we might have difficulties to fully understand the scenario, the scenario is too complicated, or even the data is incomplete itself. Therefore, a novel adaptation approach is needed, which is discussed in Chapter 8.

Furthermore, in our research, as discussed in Chapter 2, our case base is clustered, visualised and filtered first to achieve better adaptation results, the detail of the technologies are discussed in the latter part of this thesis as well. In Chapter 9, as a case study, a simple scenario is used for demonstration.
Chapter 4

Case Representation

In order to implement NDM with CBR, each scenario we are facing will be represented as a case. In this chapter, we shall discuss how to represent scenario, the COA, as well as the structure of case base. Case representation is the most important and fundamental part of CBR systems. It will significantly affect the way in which the similarity between cases is evaluated as well as the efficiency of the retrieval. Case representation should provide support to a full range of data types. It should contain sufficient information to solve new problems. In this chapter, all the possible representations of cases are discussed and how to represent military scenario efficiently and practically is described as well.

In Section 4.1, we introduce generic case representation and possible data representation methods. Scenario representation is described in Section 4.2. In Section 4.3, we discuss about COA formulation. Finally, conclusions drawn from the critical review are presented in Section 4.4.

4.1 Case representation

The most important element in a CBR system is the case base itself. A case is defined as a unit encapsulating knowledge relevant to a particular experience. The information that is stored about the experience will depend on the domain as well as the purpose for which the case will be used. For any CBR systems, the details will usually include specification of the problem and the relevant attributes of the environment that describe the circumstances of the problem. Another crucial part of the case is a
description of the solution that was used on a previous occasion when a similar situation was encountered. Generally, there are three major parts of a case (Kolodner 1993), as shown by Table 4.1.

The problem: This part contains the state while the case is happening and what problem needs to be solved. It also includes the particular attributes of a case that are required for case retrieval, such as measurements, values or some attributes, usually with weight which can describe the importance of the attribute.

The solution: This part contains the particular attributes of the solution for case adaptation, such as the solution scenario or particular values which describe specific features of the case.

Outcome: the result of the state when the case occurred (This part is optional).

<table>
<thead>
<tr>
<th>Major Parts</th>
<th>Contents</th>
</tr>
</thead>
</table>
| **Problem** | 1. Goals to be achieved  
2. Constraints on the goals  
3. Features of the problem situation and relationship between its parts |
| **Solution** | 1. Solutions  
2. Reasoning steps  
3. The set justifications for decisions |
| **Outcome** | 1. The outcome itself  
2. Explanation of the expectation violation and/or failure  
3. Repair strategy  
4. Pointer to next attempt at solution |

Table 4.1 The Contents of the Major Parts of a Case (Kolodner 1993)
In our research, the case can be represented by a military scenario and a COA, as shown in Table 4.2. In the following section, we shall discuss some common structures for case representation.

<table>
<thead>
<tr>
<th>Problem Description</th>
<th>Case Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario representation</td>
<td>COA</td>
</tr>
</tbody>
</table>

Table 4.2 Our case representation

Case representation is concerned about the content of a case, as discussed in Table 4.1. Apart from the content, case representation also deals with the structure of cases, and the organisation of the case base that facilitates search and retrieval. The representation will significantly affect the way in which the similarity between case will be assessed and efficiency of the retrieval.

In the simplest representation, cases are represented as feature vectors. A variety of other representational formalisms may be used to represent cases, such as frames, objects, predicates, semantic nets, rules etc. Case representation is suitable for a specific implementation within a particular problem domain if it contributes to the functionality of the whole system and facilitates further knowledge acquisition.

A case base can often be represented as a table whose rows correspond to individual cases, while its columns correspond to their attributes, described by a finite number. Each attribute represents one dimension in the n-dimensional case representation space. Thus a case can be represented as a point in this n-dimensional space. This linear (or flat) case memory does not impose any organisational structure on the cases. Each case is considered to be as potentially significant as all the other cases. New cases may then be added to the front, back or the middle of the list. Usually new cases will add to the front of the list. That is, the more recent cases are likely to be found at the front of the list. The flat case representation is suitable for knowledge poor domains or domain knowledge is too complicate or impossible to obtain, and it is the most common structure for case representation.
Plaza (Plaza, 1995) suggested to use structured case representation. It is a natural way to represent composite cases and its offers a possibility to treat case components as cases as well. Additionally, it provides richer expressiveness in comparison to the traditional attribute-value representation. A case represented by a tuple is said to be represented structurally if the value of an attribute can be another tuple. Such tuples represent a generalisation of the first order terms, and name by Plaza as feature terms. Each argument is referred through its identifier instead of its position. In this approach, case can be represented as collections of objects, each of which is described by an objects class that defines the set of attributes together with a type for each attribute. Object classes are arranged in a class hierarchy. Meanwhile, the relational attributes can represent a directed binary relation. Such as ‘a part of’ relation between the object that defines the relational attribute and the object to which it refers.

Sanders et al. (Sanders et al. 1997) argued for the necessity of encoding information about the relations between attributes by means of a suitable case representation. Case representations not having the ability to express the relations between case components are called feature-based representation. Such case representation may lead to incomplete case descriptions due to the omission of potentially useful information. In order to improve the usability of individual cases the case descriptions should be as complete as possible. However, it is not always possible to predict all future uses of cases.

Trees and graphs can also be used to represent cases. This can be appropriate if the knowledge experience to be represented has an inherent natural graph or tree structure. Nodes and edges of trees or graphs can be labelled or described in an attribute value manner.

Hierarchical structure is a common case base structure, which stores the cases by grouping them into appropriate categories to reduce the number of cases that have to be searched during retrieval. There are many different ways to achieve this, such as putting the most important features at the top of the hierarchy so that they are considered first. In Decision tree oriented memories, nodes do not represent cases, instead they are decision points such as "female or male?" The system uses these nodes to partition the case memory up into relatively small groups of cases. These
groups of cases often organised in a linear manner. The advantage of this approach is that it is only necessary to follow the branch indicated by each decision point to find a small group of cases. In general such an organisation will exploit some decision tree generating algorithm (such as ID3 or C4.5). Such algorithms are often available in a number of commercial CBR toolkits. However, a problem common to all induction systems, is over-fitting of the system. This can result in the system requiring apparently irrelevant information during the case retrieval process. When new cases are added to the case memory as they may change the structure of the decision tree used to organise the cases.

In Knowledge Guided Indexing, the case base is organised by indexes which effectively subdivides the cases into groups. By following the indexes, we can obtain subsets of the case base, and then selects suitable case of these subsets. The indexes used generally take advantages of domain specific and search specific knowledge. The problem is that manual defined indexes are time consuming and indexes may also be difficult to maintain.

The dynamic memory model was developed by Schank (Schank 1982a) based on his Memory Organisation Packet (MOP) theory. The case memory in this model is a hierarchical structure of MOP, it organises specific cases which share similar properties under a more general structure called generalised episode (GE). A GE contains three different types of objects: Norms, cases and indices, Norms are features common to all cases indexed under a GE. Indices are features which discriminate between a GE’s cases. An index may point to a more specific generalised episode or directly to a case. When a new case is given, the search will start at the root node. If during the storage of a case, two cases end up under the same index, a new GE is automatically created, so the memory structure is dynamic. A case is retrieved by finding the GE with most norms in common with the problem description. Indices under that GE are then traversed in order to find the case which contains most of the additional problem features. However, this approach is computationally more complex and will cause difficulty when new case added.

Linear or so called attribute-value representation is easy to understand and implement, with simple and efficient retrieval. Thus we propose to represent case in a flat way,
and then partition them later by suitable clustering algorithms. In attribute-value representation, all information contained is represented through sets of attribute values. These values can be fixed or vary from case to case. To each attribute a certain type is assigned. The type represents the value range for allowed values. In order to find the suitable representation, we shall consider some possible attribute types and their representation:

4.1.1 Representation of Symbolic attributes

The attributes of a case may take symbolic values. For example, if we have features “Route to objective”, “Perception”, “Quality”, their values maybe:

- Route to objective (adequate, marginal, inadequate)
- Perception (unimportant, important, vital)
- Quality (excellent, good, poor)

A particular situation when the Route to objective is adequate, Perception is important, Quality is poor, maybe represented by:

<table>
<thead>
<tr>
<th>Route to objective</th>
<th>Perception</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>adequate</td>
<td>important</td>
<td>poor</td>
</tr>
</tbody>
</table>

**Table 4.3 Symbolic representation**

The case is composed of the symbolic values of all features. For example in the Table 4.3, Case = {adequate, important, poor}.

4.1.2 Representation of Binary (Nominal) attributes

In this approach, each possible value of a feature is a binary variable, i.e. when the particular value of the feature is true, the binary variable has value 1 in the case representation. Otherwise, it has value 0. For example, when the Route to objective is adequate, Perception is important, and Quality is poor, as shown by Table 4.4.
Table 4.4 Binary representation

The case is composed of the binary values of all the features. For example in Table 4.4, Case = \{10010001\}.

4.1.3 Representation of Ordinal attributes

In this approach, we number sequentially all values of each feature. For example, when the Route to objective is adequate, Perception is important, and Quality is poor, we have:

<table>
<thead>
<tr>
<th>Route to objective</th>
<th>Perception</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>adequate</td>
<td>marginal</td>
<td>inadequate</td>
</tr>
<tr>
<td>inadequate</td>
<td>unimportant</td>
<td>important</td>
</tr>
<tr>
<td>important</td>
<td>vital</td>
<td>excellent</td>
</tr>
<tr>
<td>good</td>
<td>poor</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.5 Ordinal representation

The case is composed of the ordinal values of all the features. For example in Table 4.5, Case = \{1,2,3\}

Fuzzy logic, which is a mathematical method for expressing imprecise knowledge by using a membership function, can be applied to measure the degree by which a given element belongs to a set. Here is an example for fuzzy ordinal representation:
Table 4.6 Fuzzy ordinal representation

The case is composed of the fuzzy ordinal values of all the features. For example in Table 4.6, Case = \{1.3, 2.2, 1.9\}

On the other hand, fuzzy logic can also help us to convert continuous values into categories. For example, if we represent time as “immediate”, “limited” and “unlimited”, based on expert knowledge, we may set up a different membership function for each category. Therefore, time may also be represented by a three dimensional vector. Fig. 4.1 shows an example, where the horizontal axis represents the time available for the battle, while the vertical axis represents the value of the membership function.

![Membership function plots](image)

**Fig. 4.1** Fuzzy logic representation of time

(Where the horizontal axis represents the time available for the battle)
4.1.4 Representation of Numerical attributes

In this approach, we can use the actual values of a feature to represent it, if it is measurable in numerical. For example, if one of the features is “Humidity”, instead of being characterised as being high, moderate or low, we could represent it by its exact value. Then a case could be represented by listing the exact values of its features.

Each of these discussed types of attributes can exist in a case, even at same time. In the following section we shall explain our approach to represent case in our case base.

4.2 Scenario representation

In related military CBR systems, cases can be made up of war stories, prior experience, tactics and doctrine (Pratt 2001). According to domain knowledge, METT-T (Mission, Enemy, Terrain, Troops, and Time) are factors human commanders usually consider in the real battlefield. Therefore, it is very convenient for us to use these parameters to represent a scenario. Below we briefly outline each factor in turn:

- Mission: The specific task assigned to a unit or individual which commander pass to their subordinates. It is the duty or task together with the purpose that clearly indicates the action to be taken. It can be divided according to type of operation, such as offence or defence. Offence operation usually includes Movement to Contact, Attack, Exploitation and Pursuit etc, while defence usually includes Mobile Defence, Position Defence etc.

- Enemy: A command should consider the enemy’s strength, armament type, location, doctrine, capabilities, equipment and probable COA to be faced in the mission. To simplify the problem, an enemy entity may be represented as a symbolic object, while its position can be represented by the corresponding location. Its organisation may be section, platoon, or company. And its combat effectiveness may be characterised as being inoperable, degraded or in full capability.
• Terrain: The geographical location where the mission will take place is important, so the commander conducts a map reconnaissance to determine key terrain, obstacles, cover and concealment, and troop movement, such as line of departure and phase lines. Key terrain is any area whose control affords a marked advantage to the force holding it. Some types of key terrain are high ground, bridges, towns, and road junctions. Obstacles are natural or man-made terrain features that stop, slow down, or divert movement. Consideration of obstacles is influenced by the unit’s mission. An obstacle may be an advantage or disadvantage, depending upon the direction of attack or defence. Obstacles can be found by conducting a thorough map reconnaissance and study of recent aerial photographs. Cover and concealment are determined for both friendly and enemy forces. Concealment is protection from observation; cover is protection from the effects of fire. Most terrain features that offer cover also provide concealment from ground observation. There are areas that provide no concealment from enemy observation. These danger areas may be large or small open fields, roads, or streams. During the leader’s map reconnaissance, he determines any obvious danger areas and, if possible, adjusts his route.

In order to simplify the problem, we may assume that there is one single attribute for the terrain feature, taking values Open Wooded passable, Wooded impassable, Rolling, Hilly passable, Hilly impassable, River passable, River impassable etc. For example, we may divide the whole battle field into the left, centre and right field, usually according to the main axis. In each part, the terrain feature is assumed to have one uniform value, chosen from the \( n \) possible kinds of different terrain value, then the total number of possible different composite terrain will be \( n^3 \).

• Troops: Consideration of friendly troops is equally important. Because units are employed according to their capabilities, the size and type of the unit to be moved and its capabilities, physical condition, status of training, and types of equipment assigned, all affect the selection of routes, positions, fire plans, and
the various decisions to be made during movement. We may consider the representation of the friendly troops to be same as that of the enemy.

- Time: The time that has been allotted to troops to reach an objective or move from one point to another is also important. The commander must conduct a map reconnaissance to determine the quickest and reasonable route to the objective; this is not always a straight route. As we discussed before, it can be divided into immediate, limited and unlimited, and be treated as a fuzzy variable. For example Fig. 4.1 shows a membership function of Time, where the horizontal axis represents the time available for the battle. According to it, the time value of a mission which should be finished in 1.5 hour must be “limited”. However, we need to depend on expert knowledge to set up the correct membership function.

Therefore, the scenario in our research will be represented by Mission, Enemy, Terrain, Troops, and Time. A detail example is discussed in Chapter 9.

4.3 COA formulation

A COA is a potential solution to an assigned mission. The development of COA is to generate options that satisfy the mission, commander’s intent and guidance of the commander. A COA consists of a sketch and a textual statement. During the development process, the COA graphic sketch and narrative statement will help the commanders to understand how the organization will accomplish the mission.

The graphic sketch clearly portrays the scheme of manoeuvre of the main and supporting efforts and critical manoeuvre and fire support control measure, such as objectives, boundaries, phase lines, and fire support coordination lines. For example, it includes a depiction of what terrain features are considered important. The results of analyzing terrain, such as possible paths for movement and good locations for different kinds of operations are identified. The disposition of troops and equipment, both friendly and enemy forces is shown by means of unit symbols, a vocabulary of graphical symbols defined by the military doctrine. This graphical vocabulary also
includes symbols for tasks, such as destroy, defend, attack, and so on (MCWP 5-1 2001).

The COA narrative statement provides the purpose and tasks of the main and supporting efforts, the reserve and the sequencing of the operation. For example, why units are being assigned the tasks that they are and timing information that would be difficult to express in the sketch (MCWP 5-1 2001).

Once COAs have been developed, each prospective COA will be examined with following criteria (MCWP 5-1 2001).

- Suitability: Does the COA accomplish the purpose and tasks?
- Feasibility: Does the COA accomplish the mission within the available time, space and resources?
- Acceptability: Does the COA achieve and advantage that justifies the cost in resources?
- Distinguishability: Does the COA differ significantly from other COAs
- Completeness: Does the COA include all tasks to be accomplished? Does it describe a complete mission?

COA formulation is a very difficult and important problem for our research. When a military commander faces a tactical mission, he/she will develop a set of COAs to achieve it and inform their staff and units what to do. A COA details a mission occurring over terrain with an objective to achieve under unit/equipment and time constraints. Generally, it is composed of commands for each entity in the troops. It is difficult to transform these narrative statements and graphic sketches into a form which the computer can process.

MAK VR-FORCE (MAK Technologies 2003), which is discussed in detail in the latter part of this thesis, is the scenario simulation environment of our research. Actually no matter how complicated a COA is, hasty attack, deliberate attack, withdraw, maintain position or occupy etc, all of these will eventually be transformed to movements of entities in VR-FORCES. So why not represent a COA by a series of positions of entities, and once the entities reach the suitable waypoint, they will fire
automatically. Therefore, we can represent a COA by entities’ waypoints and the corresponding time for each entity to reach each of the waypoints.

In other words, the COAs formulation will be a matrix, quite similar to the *synchronization matrix* which human commanders apply to decide COAs. The matrix is composed of entities’ location waypoints at different time steps during the scenario. Each row is corresponding to one entity and each column is one time step. Table 4.7 shows an example.

<table>
<thead>
<tr>
<th>Entity name</th>
<th>Time 1</th>
<th>Time 2</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entity 1’s route</td>
<td>Entity 1’s waypoint 1</td>
<td>Entity 1’s waypoint 2</td>
<td>...</td>
</tr>
<tr>
<td>Entity 2’s route</td>
<td>Entity 2’s waypoint 1</td>
<td>Entity 2’s waypoint 2</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

*Table 4.7 COA representation matrix*

An example is shown by Table 4.8, in which, entity A move from waypoint $a$ to waypoint $a'$, entity B move from waypoint $b$ to waypoint $b'$ while entity C move from waypoint $c$ to waypoint $c'$.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>b</td>
<td>..</td>
<td>..</td>
<td>..</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$a'$</td>
<td>$b'$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>c'</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 4.8 An example of waypoint movement*

Using the column and row index as their corresponding waypoint locations, the COA representation matrix of this example is as follows.
Table 4.9 COA representation matrix for the example in Table 4.8

<table>
<thead>
<tr>
<th>Entity Name</th>
<th>Time 1</th>
<th>Time 2</th>
<th>Time 3</th>
<th>Time 4</th>
<th>Time 5</th>
<th>Time 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>(3,2)</td>
<td>(3,3)</td>
<td>(3,4)</td>
<td>(3,5)</td>
<td>(3,6)</td>
<td>(3,7)</td>
</tr>
<tr>
<td>B</td>
<td>(5,4)</td>
<td>(6,4)</td>
<td>(7,4)</td>
<td>(7,5)</td>
<td>(7,6)</td>
<td>(7,7)</td>
</tr>
<tr>
<td>C</td>
<td>(9,3)</td>
<td>(9,4)</td>
<td>(9,5)</td>
<td>(9,6)</td>
<td>(9,7)</td>
<td>(10,7)</td>
</tr>
</tbody>
</table>

4.4 Summary

In this chapter, we discussed different approaches to represent the case. Case normally has two components: problem specification and solution. Generally, the problem specification consists of a set of attributes and values which can define the case uniquely and should sufficient to predict a solution for that case. Military scenario is very complicated therefore difficult to describe. COA normally consist of a narrative statement and a graphic sketch, thus it is very complex to formulate. We propose to formulate scenario by ‘METT-T’, and represent the COA by entities’ waypoints and the corresponding time for each entity to reach each of the waypoint.

In our project, the case base is organized into case group so that only a small subset of cases needs to be searched during retrieval, although linear or so called attribute-value representation is used for each single case. This subset should contain the best matching or most useful cases. Clustering algorithms are generally used to partition the cases. Thus in the next chapter, we shall discuss the related clustering algorithms, especially the one suitable for achieving pattern recognition of our NDM model.
Chapter 5

Case Clustering

Pattern Recognition is an important part to implement NDM. In which, each scenario we are facing will be compared to the pattern of known scenario in the case base. Once a match is found, the recognition is occurred. Therefore, the cases in our case base are clustered first to implement the NDM model. This process provides the important pattern recognition part in NDM. Meanwhile, when compared to linear case memory, cases organised as clusters can also help the retrieval and adaptation much more efficiently. This chapter presents different cluster algorithms. Subsequent to evaluation between alternatives, the Self-Organised Map (SOM) is chosen as the most appropriate and viable clustering solution for our research. Not only can it group similar cases together but it may also provide a visualisation interface which can help us understand the structure of case base, including the relationships between cases. The visualisation of case bases will be discussed in detail in Chapter 6. More importantly, SOM provides the appropriate clustering for our NDM model.

In Section 5.1, we introduce clustering and many cluster approaches. The basic Self-Organized Map is described and discussed in Section 5.2. Section 5.3 explains clustering by SOM. Section 5.4 discusses the shape of clusters. In Section 5.5, we review some further topics concerning SOM. Finally in Section 5.6, we present our conclusions.

5.1 Clustering

In traditional case retrieval, cases usually are indexed manually first, which costs much time and energy. Indeed when the case base becomes larger, the retrieval process will become more time-consuming. It is assumed that similar cases are
positioned close to one another in the case space. Indeed rather than being scattered uniformly over the case space, cases tend to form clusters corresponding to certain types of problems. The space between the clusters may be less dense because the corresponding problems occur less often. The retrieval task may be defined as the task of finding the position of a new case in the case space and fetching the cases from its immediate surrounding (Coenen & Watson 1999). Yang and Wu (Yang & Wu 2000) partition cases into clusters where the case in the same cluster are more similar than cases in other clusters. Cluster can be converted to new smaller case bases as well.

In our research, the cases in the case base will be organised in clusters of similar cases. When a new case comes along, it will be compared with all clusters, after which a suitable cluster then will be chosen. Clustering belongs to unsupervised learning, and it can help us group all similar cases together. Existing clustering methods can be applied for case clustering because of the way cases are represented. Thus the notion of similarity/distance is inherent in CBR (Yang & Wu, 2001). To apply clustering, we need a measure of distance between the objects to be clustered. We shall discuss such measures first.

5.1.1 Distance Measures

The distance or similarity measure is processed normally according to the 'local-global' principle. The definition of global measures is defined on the whole case while the definition of local measures is defined on the level of the attributes. Different approaches are used to calculate distances between different types of attribute.

For nominal or categorical objects, such as terrain, we may measure the distance for two categories by using simple matching:

$$d_{ij} = \frac{p - m}{p}$$

where $p$ is the number of total possible feature values and $m$ is the number of values which are the same in object $i$ and object $j$. If there is more than one nominal
feature, p is the number of total features and m is the number of features whose values are the same. We may also give different weights to different features.

The Jaccard coefficient (Han 2000) can be used for the binary representation:

\[ d_{ij} = \frac{q + r}{v} \]  

(5.1)

where v is the total number of digits, q is the number of digits that are 1 for object i and 0 for object j, and r is the number of digits that are 0 for object i and 1 for the object j.

For ordinal data, we need to transform the numbers into ranks first, if they are not in ranked form (r = 1 to R). Then normalize the rank into a standard value \( x \in [0,1] \) by:

\[ x = \frac{r-1}{R-1} \]  

(5.2)

After that, we can treat them as numeric objects.

For numeric objects, various metrics are normally used to measure the similarity or dissimilarity between two objects. Examples are the Canberra distance, Chebyshev norm, Minkowski distance, Manhattan metric etc (Teknomo 2005). The most popular one is the Euclidean metric:

\[ Euclid(X,Y) = \sqrt{\sum_{i=1}^{v} (x_i - y_i)^2} \]  

(5.3)

where \( X = (x_1, x_2, ..., x_v) \) and \( Y = (y_1, y_2, ..., y_v) \) are the representation of two objects.

The Cosine coefficient is to represent each model as a vector in a multi-dimensional space and calculate the angles between the vectors. The direction of a vector can be viewed as the representation of model template, therefore a high value of the Cosine similarity measure indicates models that are expected to belong to the same template.
\[
Co \sin e(X, Y) = \frac{\sum (x_i \cdot y_i)}{\sqrt{\sum (x_i)^2 \cdot \sum (y_i)^2}}
\] (5.4)

If \( X = (x_1; \ldots; x_v) \) and \( Y = (y_1; \ldots; y_v) \) are vectors representing the two cases, where \( v \) is the feature size, the dot product can give an indication of the inter-model similarity:

\[
X \cdot Y = \sum_{i=1}^{v} (x_i \cdot y_i)
\] (5.5)

Generally the dot product is normalized to be a value between 0 and 1 to represent the similarity, which 0 indicates no similarity while 1 indicates complete similarity.

Different types of data also have different units and take different range of values, so in order to avoid the effect of units and range of value influencing the result, we need to normalize the data into the range \([0, 1]\):

\[
\delta = \frac{d - d_{\text{min}}}{d_{\text{max}} - d_{\text{min}}}
\] (5.6)

where \( \delta \) is the value after normalization, \( d_{\text{min}} \) is the minimum value of the datum before normalization, and \( d_{\text{max}} \) is its maximum value.

Generally we could have case of mixed type of attribute, which are composed of nominal, ordinal and numeric. In order to measure their similarity, we need to normalize all the variables first, and assign a different weight to each feature. The aggregated similarity is the simple weighted average of distance matrices of each variable.

\[
d_{ij} = \frac{\sum_{k=1}^{n} w_{ijk} \cdot \delta_{ijk}}{\sum_{k=1}^{n} w_{ijk}}
\] (5.7)
where $\delta_{ij}^k$ is the dissimilarity between object $i$ and object $j$ for attribute $k$, while $w_{ij}^k$ is the corresponding weight.

5.1.2 Clustering methods

There is a variety of classifications for clustering methods (Can & Ozkarahan 1990; Sneath & Sokal 1973). In order to choose a clustering method, the following criteria must be considered:

- The method is independent of the initial ordering of the subjects.
- The method should be stable, either the new item or changes of the old one should only cause minor alternation in the clustering.
- The classification should be well distinct, a given data should produce a single cluster or at least one of small set of compatible clustering.
- Strong for the noise, small errors can only cause minor alternation in the clustering.

A widely adopted definition of optimal clustering is the one which can minimize the distance within the clusters while maximizes the distances between clusters. Generally, clustering methods can be divided into (Han 2000):

- Non-hierarchic clustering or partitioning methods, such as K–means and PAM (Kaufman & Rousseeuw 1990; MacQueen 1967);
- Hierarchical methods, such as Chameleon, BIRCH (Karypis, Han, & Kuanr 1999; Zhang, Ramakrishnam, & Livny 1996);
- Density based methods,

Density-based methods try to discover clusters with an arbitrary shape. They group the neighbouring points of a dataset into clusters based on density condition. Clusters are regarded as dense regions of objects in the data space which are separated by regions of low density. OPTICS (Ordering Points to Identify Clustering Structure) (Ankerst et al. 1999), DENCLUE (DENsity-based CLUstEring) (Hinneburg & Gabriel 2007), DBSCAN (Density-Based
Spatial Clustering of applications with Noise (Ester et al. 1996) are popular density-based algorithms;

- Grid-based methods,

Grid-based methods use a multi-resolution grid data structure that divides the space into a finite number of cells that form a grid structure on which all clustering operations are performed. This approach can process constantly, independently of the number of data objects. Popular algorithms include CLIQUE (Clustering High-Dimensional Space), WaveCluster and STING (Statistical Information Grid) (Wang, Yang, & Muntz 1997);

- Model-based methods,

Model-based methods use mathematical and probability models. Popular algorithms include AUTOCLASS and COBWEB (Fisher 1987);

- Neural networks.

5.1.3 Non-Hierarchic Clustering Methods (NHCM)

Non-hierarchic clustering is also referred as partitioning clustering. It divides a collection of N cases directly into M disjoint clusters and uses heuristics to assign cases to clusters in order to achieve better computational efficiency. It requires a priori decision such as the number of clusters, cluster size, criterion for cluster membership and possibly the cluster seeds. NHCM are better than hierarchic clustering ones in the sense that they do not depend on previously found clusters. On the other hand, NHCM make implicit assumptions on the forms of clusters.

NHCM are considered to perform less well than hierarchic methods in terms of effectiveness or quality of clustering. Though, due to their efficiency for high speed and less memory, and their capacity to organize search results fast (Cutting et al. 1992), non-hierarchic clustering methods still receive a great deal of attention nowadays.
5.1.4 Hierarchic Clustering Methods (HCM)

Hierarchic structures can be found in real life taxonomies and employed by domain experts when manually classifying data collections. In HCM, the input data are not partitioned into the desired clusters in one step. It creates tree-like classifications, which is called dendrogram. Clusters of highly similar cases are nested within larger clusters of less similar cases. By cutting the dendrogram at a desired level, a clustering of items into disjoint groups is obtained. HCM permit a convenient graphical display, in which the entire splitting of clusters is shown. The single cluster containing the entire collection is the root of the tree while the individual cases are leaves. The other nodes correspond to clusters at different levels of similarity. The final clusters are obtained by performing a series of successive process until the final clusters are obtained. One example is the Single Link Method which will be discussed later. Hierarchical clustering tries to merge smaller cluster into large ones, or split larger clusters. The clustering methods differ in the rule by which it is decided which two small clusters are merged or which large cluster is split. It can be classified as divisive and agglomerative (Jardin & Rijsbergen 1971).

Divisive methods build cluster hierarchies from top down by successively dividing the single initial large cluster into smaller and smaller clusters of cases, by finding dissimilarities between cases within clusters. They usually are typically monothetic classifications, in which all cases member in a cluster must contain certain terms in order to belong to it. Divisive methods have received less attention (Voorhees 1986b) and are less commonly used.

Agglomerative methods build cluster hierarchies from bottom up by successively agglomerating smaller clusters to more general large ones. It starts with each case in a single cluster. They usually are polythetic classifications, where cases in a cluster have terms in common, but none are required for cluster membership. The Hierarchic agglomerative clustering algorithm can be described as follows (Voorhees 1986b):

*Each case to be clustered constitutes a single cluster*

*Compute similarity between clusters*
WHILE there is more than one cluster DO

Merge the most similar clusters
Recompute similarities between clusters

ENDWHILE

Most agglomerative clustering methods require the clusters to be compact and well separated. In real life, this is rarely the case. Rather, the gaps between clusters are obscured by noise, the clusters overlap and there are outliers.

5.1.5 Single-link clustering method

Single-link clustering is one of the most widely used and simplest HCM. The method is known as single-link because clusters are joined at each stage by the single shortest or strongest link between them. The inter-cluster similarity is defined as the similarity between the most similar pair of cases, one from each cluster. There are several theoretical advantages (Jardine & van Rijsbergen 1971) over other methods:

- The clustering obtained only depends on the rank-ordering of similarity values, not on their absolute values.
- It is stable under small errors in similarity values.
- It is stable under update: the cluster hierarchy is unlikely to change drastically when further objects are incorporated.
- The order of input is not significant. A given set of data should define exactly one hierarchy.

It is one of very few clustering techniques which can outline nonellipsoidal clusters. However, it is incapable of delineating poorly separated clusters. Also, it does have some disadvantages (Jardine & van Rijsbergen 1971); namely:

- It tends to form long, loosely bound clusters with little internal cohesion, where cases in the same cluster not necessarily more similar to each other when compared to cases not in the cluster.
The information of the clusters depends on the order which items are examined. Therefore it is difficult to reproduce the clustering results. It produces a high number of aberrant cases, cases not similar to other cases, isolated at highest levels in the hierarchy.

There are some other HCM, such as Complete-link clustering method, in which the inter-cluster similarity is defined as the similarity of the least similar pair of cases, one from each cluster, when cluster are merged, the similarity between cases is at least equivalent to the similarity of the cluster which results in tighter clusters. Group-average clustering method based on the mean of similarities between all pairs of cases, one from each cluster (Voorhees 1986a). While Ward method joins cluster whose fusion result in the least increase in the sum of distance from each case to the centroid (mean of the cluster) of its cluster, and results in homogeneous, spherical cluster with symmetric hierarchy (Voorhees 1986a) (Kaufman 1990).

The disadvantages of most HCM are in the calculation which is very costly. Furthermore, HCM makes no provision for the reallocation of cases that may have been poorly classified at an early stage in the analysis. Thus it is impossible to correct a poor initial partition. Also, it is not possible to undo any merge or split operation.

5.1.6 K-Means clustering

A commonly used NHCM is K-Means clustering (MacQueen 1967). In K-Means clustering, the algorithm tries to find a partition in which objects within each cluster are as close to each other as possible, and as far from objects in other clusters as possible. More specifically, K-Means aims to minimize the mean square error of each point in a cluster, with respect to its cluster centroid. In other words, the centroid of each cluster is the mean of each cluster, it is a point to which the sum of distances from all objects in that clusters is minimised. K-Means Algorithm can be described as follows:

Given $u^i \in R^N, i = 1,...,P$

- Select randomly $k$-points as the means for cluster $C^k$
• Assign each object to nearest mean according to
\[ u^i \in C^k \text{ where } k = \arg \min_j \| u^i - m^j \| \] (5.9)

• Compute the new mean for each cluster to replace the means for cluster \( C^k \)

• Repeat until there are no changes in any mean \( m^k \), \( k = 1,...,K \)

\[
m^k(t+1) = \frac{1}{\text{card}(C^k(t))} \sum_{u \in C^k(t)} u^i
\] (5.10)

Where \( \text{card}(C^k(t)) \) is number of element in the cluster \( C^k \) at iteration \( t \)

K-Means is a very efficient algorithm. It scales linearly with \( O(nkt) \), where \( n \) is the number cases in the case base and \( k \) is the chosen number of clusters and \( t \) is the number of iterations.

Though simple, K-Means can produce satisfactory clustering result by reaching its local optima. However, it does have its weakness. First of all, a good value of \( k \) is usually not known beforehand. Secondly, it depends on the availability of the entire input data set for batch processing which could be memory intensive.

Thirdly, it is susceptible to the initial choice centroids. Under certain circumstances, an unfortunate selection of initial centroids may prevent us from finding the best possible clustering. Furthermore, K-Means is very sensitive to noise and outliers. Because a cluster is always represented by its arithmetic mean, a noise or outlier may cause the cluster centroid to move outside of the main cluster. Moreover, if these cases is one of selected initial centroids, it might turn out that no other cases are mapped to this cluster, which will cause isolated cluster. In order to avoid this, some variations of it appeared, such as PAM (partitioning around Medoid) (Kaufman & Rousseeuw 1987), which uses the most centrally located object in a cluster instead of the mean, thus reduce the sensibility while some values that distort data distribution are found. It is essentially an existing object form the cluster, and it is the closest to the corresponding mean. PAM is found to be more robust to noise and outliers, but it is usually more computationally expensive.
Meanwhile, K-Means is not capable of detecting clusters with non-convex shapes. The clusters K-Means can find are always of a spherical shape, which may not faithfully represent the actual shape of the natural cluster in the case base.

5.2 Self-Organising Map (SOM)

SOM (Kohonen 1982) is an unsupervised neural network, which was developed by Professor Teuvo Kohonen in the early 1990s; hence, it is also known as a “Kohonen network” (Freeman & Skapura 1991). SOM has been successfully used in a wide variety of applications, such as pattern recognition, image analysis and fault diagnosis. It provides a "topological" mapping from the input space to the clusters (Kohonen 2004). The basic algorithm was inspired by the way in which various human sensory impressions are neurologically mapped into the brain such that spatial or other relations among stimuli correspond to spatial relations among the neurons. This is called competitive learning. It is an adaptive process in which the neurons in a neural network gradually become sensitive to different input categories of samples in a specific domain of the input space (Amari 1980; Didday 1970; Grossberg 1976; Kohonen 1982). A kind of division of labour emerges in the network when different neurons specialize to represent different types of inputs. The specialization is enforced by competition among the neurons. When an input arrives, the neuron that is best able to represent it wins the competition and is allowed to learn it even better.

If there is an ordering between these neurons, for example, then the neurons are located on a discrete lattice. This characterizes the relative position of neurons with respect to their neighbours. These are topological properties instead of exact geometric ones. Fig. 5.1 shows an example of SOM neighbourhoods. The SOM may be generalized, if not only the winning neuron but also its neighbours on the lattice are allowed to learn. Neighbouring neurons will gradually specialize to represent similar inputs and the representations will become ordered on the map lattice. SOM consists of two layers of neurons, an input layer with n input nodes, which correspond to the n dimensions of the input vectors, and N output nodes, which correspond to the N decision regions, with every input node connected to every output node. All the
connections are weighted. The lattice type of the array can be defined to be rectangular, hexagonal or even irregular.

A SOM forms a nonlinear projection from a high dimensionality data manifold onto a low-dimensionality grid. The basic steps are outlined as follows:

1. Initialization: choose random values for the weight vectors of all neurons
2. Similarity matching: using the angular distance or the Euclidean distance to measure the similarity between two cases, represented by vector in the case space.

The angular distance between vector $x$ and $y$ is defined as

$$\cos \theta = \frac{x^T y}{\|x\| \|y\|}$$  \hspace{1cm} (5.11)

The Euclidean distance $d_j$ is calculated by summing the squared distance of the input values $x_i$ and the corresponding neural weights $w_{ij}$

$$d_j = \sqrt{\sum_{i=1}^{k} (x_i - w_{ij})^2}$$  \hspace{1cm} (5.12)

These neighborhoods could be written as

$N_{13}(1) = \{8, 12, 13, 14, 18\}$ and

$N_{13}(2) = \{3, 7, 8, 9, 11, 12, 13, 14, 15, 17, 18, 19, 23\}$

Fig. 5.1 Neighbourhood examples of SOM (Mathworks 2002)
\[ c = \arg \min_j \| x - W_j \|, \quad j = 1, 2, \ldots, N \] (5.13)

where \( W_j = (w_{1j}, w_{2j}, \ldots, w_{nj}) \)

3. Updating: adjust the synaptic weight vectors of all neurons, using the update formula

\[
W_j(t+1) = \begin{cases} 
  w_j(t) + \alpha(t) \eta(t) [x(t) - w_j(t)] & j \in N_c(t) \\
  w_j(t) & j \notin N_c(t)
\end{cases}
\] (5.14)

Here \( \alpha(t) \) is the learning rate, a scalar usually decreasing monotonically in time (\( 0 < \alpha(t) < 1 \)).

A neighbourhood \( N_c \), is selected around the winner node \( c \) which has a minimum Euclidean distance. There are different ways to define neighbourhood, such as examples shown by Fig 5.1 and 5.2.

\( \eta(t) \) is called neighbourhood function. The winning unit and its neighbours adapt to represent the input by modifying their reference vectors towards the current input. The amount the units learn is governed by this neighbourhood kernel. It is a decreasing function of the distance of the units from the winning unit on the map lattice. In other words, the largest weight adjustment which is positive occurs for the winner, and smaller positive changes are made to adjacent neuron, and so on until at some distance the weight adjustment goes to zero.

For convergence it is necessary \( \eta(t) \to 0 \) when \( t \to \infty \). Generally, two kinds of \( \eta(t) \) are often used.

The simple one, for nodes within neighbourhood \( N_c \), \( \eta(t) = \alpha(t) \), and \( \eta(t) = 0 \) if node is outside \( N_c \). Thus, at each learning step, all the nodes within \( N_c \), are updated, whereas the nodes outside \( N_c \), are left intact.

Gaussian function is commonly used as the smoother neighbourhood kernel,

\[
\eta(t) = \exp\left(-\frac{\| r_c - r_i \|^2}{2\sigma(t)^2}\right)
\] (5.15)
Where \( r_c \in \mathbb{R}^2 \) and \( r_j \in \mathbb{R}^2 \) are the location vectors of nodes \( c \) and \( j \), respectively in the array.

\( \sigma(t) \) is the variance parameter specifying the spread of the Gaussian function, it corresponds to the radius of \( N_c \), which is decreasing as the training progresses.

4. Continuation. Continue with step 2, both \( \alpha(t) \) and \( N_c \) are dynamically during the training, until no noticeable changes are observed.

At the beginning of the learning process the radius of the neighbourhood is fairly large, but it is made to shrink during learning. This ensures that the global order is obtained at the beginning is not destroyed later; whereas towards the end, as the radius gets smaller, the local corrections of the prototype vectors in the map are more specific. This allows the topological order of the map to be formed.

![Diagram of topological neighbourhood](image)

**Fig. 5.2** Examples of topological neighbourhood

Therefore those nodes that are topographically close in the array up to a certain prespecified geometric distance will activate each other to learn something from the same input. This will result in a local smoothing effect on the weight vectors of neurons in the neighbourhood, which in continued learning leads to global ordering. This is called topology preservation. The more similar two inputs are, the closer their best matching units are likely to be on the final map. Topology preservation refers not only to intra-cluster relationships but also to the inter-cluster relationships. In other words, not only the distances of cases within a cluster are meaningful but also the
distances between clusters, which means two nearby clusters should be more similar than two distant clusters.

5.3 Clustering by SOM

SOMs have very fuzzy borders around their clusters, whereas K-Means and HCM have crisp ones. Thus visual inspection is often required for clustering with SOM. Distance matrix techniques such as the Unified Distance Matrix (U-Matrix) method (Kraaijveld 1992; Ultsch & Siemon 1989) are often used to draw the boundaries between different clusters. The average distances between neighbouring prototype vectors are represented by shades in grey scale. Normally dark shade indicates a large distance between adjacent map units while light shade represents small distance. In this way, a cluster landscape is formed over the SOM.

Another visualisation method is to display the number of hits in each map unit. Training of the SOM positions the interpolating map units between clusters and thus obscures cluster borders. The Voronoi sets of such map units have very few samples or may even be empty. Thus zero-hit units can be used to indicate cluster border (Zhang & Li 1993).

By using the U-Matrix method, clusters can be automatically acquired from the trained SOM. Nevertheless, without any a priori knowledge, it is still difficult to mark the regions based on the grey scale map. This is because not all the clusters that were marked had clear boundaries to separate them from the other clusters; plus, some small clusters might be skipped or merged into the more predominant clusters. Therefore, traditional clustering algorithm such as K-means can be applied to the trained SOM to further clarify the boundaries between clusters. Sammon mapping can be applied on prototype vectors too. A special technique is to project the prototype vectors into a colour spaces so that similar map units are assigned similar colors (Kaski, Venna, & Kohonen 1999; Varfis 1993).

SOM is a good tool for mapping high-dimensional data into a two-dimensional feature map. It has the ability to deal with high-dimensional data naturally. We can see the scalability of SOM in (Kohonen et al. 2000), which can deal with 6.8 million document collections. Similar samples in the input space turn out to be neighbours in
SOM. Thus we can easily find interesting patterns from visual inspection, which is another good property SOM has. SOM is very popular in the application of data mining and is a good visualization method for complex data sets. Application areas include image processing, speech recognition, process control, economical analysis, diagnostics in medicine, etc.

5.4 Shape of the cluster

Consider a case as a point in an n-dimensional space, with each of the n variables being represented by one of the axis of this space. The variable values of each case now define an n-dimensional co-ordinate in this space. Clusters can be described as continuous regions of this space containing a relatively high density of points, separated from other such regions by containing a relatively low density of points.

An advantage of considering clusters in this way is that it does not restrict the shape of clusters rigidly. If cluster is defined as cases in the cluster should be closer to each other than to cases in other clusters, then it is restricted to be spherical clusters. Majority of clustering techniques only can find clusters of a particular shape. However, there is no priori reason to consider any clusters are of one particular shape. Although spherical clusters might be expected to be the most common, many dataset are likely to contain clusters of other shapes.

As stated above, SOMs have very fuzzy borders around their clusters, without crisp ones which can be found with K-Means and HCM. Thus visual inspection is often required for clustering with SOM. Of course, personal visualisation can be employed to select clusters based on ones own subjective judgement. This is another reason for selecting SOM as our clustering algorithm. In Chapter 2 we discussed that in order to achieve pattern recognition of NDM model. There are three different situations to consider. We need to judge whether the new case belongs to the original cluster or not. Normally it depends on the prototype vector of the new case. The border of this arbitrary shape cluster may not be crisp, thus most common cluster algorithms cannot achieve it. However, SOM can provide the visualisation as well as the clustering at same time, which enables us to judge the relation between the new case and the
original clusters. This process can also be set up automatically if we combine the SOM results with normal clustering ones, such as K-Means and Sammon mapping etc.

5.5 Some further topics about SOM

The choice of parameters will affect the SOM result. Those parameters include the map width and height, the number of iterations, the size of the initial radius of the neighbourhood kernel, and the initial value of the learning rate. There are no strict guidelines for the right choices. Trial and error is necessary to determine the most suitable values. As a rule of thumb, it has been suggested to use rectangular maps and an initial radius equal to the height of the map. As for the value of the initial learning rate, 0.05 has been suggested (Kohonen 2004). Fig 5.3 shows how the learning rate decreases linearly as the number of cycle increases when the total iteration number is 1000, initial radius is 10 and initial learning rate is 0.05. Under the same conditions, Fig. 5.4 shows how the radius decreases as the number of cycles increase.

Fig. 5.3 Learning Rate decrease monotonically while the number of cycle increases
Fig. 5.4 Radius decrease monotonically while the number of cycle increases.

Fig. 5.5 shows the neighbourhood function. The adjustment factor decreases monotonically with a Gaussian shapes while the distance from Best Matching Unit (BMU) increases. As the number of the cycles increases, both the height and the width of the neighbourhood kernel decrease.

Fig. 5.5 The neighbourhood function (for changing distance and cycle number)

Choosing the initial values for SOM is always a complicated process, and there are some rules and procedures that should be followed. There is not a standard number of input vectors that should be provided, however, generally the more data provide to the SOM for training, the better cluster result will be for the new input.
Random initialization can be used, even if we start from an arbitrary initial state. Eventually these unordered vectors will be ordered in the long run. However, if the initial values are selected as regular, the process can be made to converge much faster. Random initialization assumes a map, knowing nothing about the input data. Thus it requires a number of additional training iterations until the map can roughly represent the training data. In practical applications one can also start from an initial state that is already ordered and roughly complies with the input density function. Under these circumstances the learning process converges rapidly.

Using random samples from the input training dataset can save some time. When the training begins, the map is already in a state in which it represents at least some of the input data. This obviously reduces the number of training cycles. However, the choice of the input samples for initialization is random, and the number of map nodes is very small compared to the number of the training cases. Thus the initial map is unlikely to be a faithful representation of the given dataset.

We can also use Principal Component Analysis (PCA) to reduce the dimensionality of the input dataset while still retaining most of the original variability in the dataset. PCA is based on the concepts of Eigenvectors and Covariance.

If $A$ is a square matrix, and there is a vector $x \neq 0, x \in \mathbb{R}^n$, we have:

$$Ax = \lambda x$$

then scalar $\lambda$ is called the eigenvalue of $A$ with corresponding eigenvector $x$. Consider matrix $A$ ($k \times k$) as a linear transformation in the $k$-dimensional Euclidean space. An eigenvector of $A$ is a vector that gives a direction in which that transformation is simply a scaling. The eigenvalue is the amount of scaling.

The covariance is a measure of the tendency of two features to vary together. It can indicate the mutual dependence of the dimensions of the data. The covariance matrix is a square, diagonal matrix contains covariance of pairs of features. By sketching the eigenvector of the covariance matrix on top of a scatter plot of the dataset, we can find the eigenvectors indicate the directions in which the cloud of data points in
stretched. The larger the eigenvalue is, the greater the variation along its associated eigenvector. This is shown by Fig. 5.6. $w_1$ is the eigenvector corresponding to the largest eigenvalue, while $w_2$ corresponds to the second largest eigenvalue.

Linear initialization is the most popular initialization for the SOM. After calculating the eigenvectors and eigenvalues of the training dataset, the map is initialized along the $mdim$ greatest eigenvectors of the training data, where $mdim$ is the dimension of the map grid.

Research has been undertaken in the area of dimensionality reduction of the input space, in order to reduce the possibility of overfitting. Overfitting basically forces the neural network to learn unnecessary data that most of the time confuses the trained network for clustering new data. Similarly the number of the training cycles and the initial values of the neural network are significant to the performance of the SOM. A large number of cycles usually guarantees a better training result, but it slows down the training process and increases the risk of overfitting.

Another important problem for the SOM is the size of the map. The number of map units, which typically varies from a few dozen up to several thousand, determines the accuracy and generalization capability. Meanwhile, the map should be capable of representing the input data. The bigger the map usually means better training, because
the training data can be spread into a long area, providing more detailed analysis. However this is costly because of the longer training time. Thus the size of the map should depend on the training data and the number of clusters.

5.6 Conclusions

Clustering is an approach to identify natural groups without any a priori knowledge. In addition, it is an efficient approach to organise cases in the case base. Many clustering algorithms prefer certain shapes of clusters, and the algorithms will always assign the data to clusters of such shapes even if there were no clusters in the data. Another problem is the choice of the number of clusters in some algorithms; for example in K-Means a different K will cause quite different kinds of clusters. Good initialization of the cluster centroids is also important. Otherwise some clusters may even be empty if their centroids lie initially far from the distribution of data.

SOM are trained in such a way that there is no need of previous knowledge about the structure of the data, thus the organization of high dimensional input space into a two dimensional plane is unsupervised. This is an advantage as well as disadvantage. As the distance between input data are evenly distributed within a constrained plane, SOM cannot be a standalone clustering method. In order to detect and display the hidden structure, it should be combined with U-Matrix. Ultsch and Vetter have also argued that SOM clustering (with U-Matrix) performs better than standard statistical clustering methods such as hierarchical clustering and K-Means, especially when the clustering is applied to complex feature vectors (Ultsch & Vetter 1994). Furthermore, SOM has been used successfully in many other cluster applications, such as CBIR (Content-Based Image Retrieval) (Nishikawa, Horiuchi, & Kotera 2005) and WebSOM (2008).

An advantage of SOM is its ability to handle very large data sets and works well even when the original space has a very high dimensionality. It can create a set of prototype vectors representing the dataset and carries out a topology preserving projection of the prototypes from the high dimensional input space onto a low dimensional grid, which can be used as a convenient visualisation surface for showing different features of the data at the time of clustering. Thus, A SOM not only can help cluster the case base
but can also be used as an important visualisation aid by providing a complete picture of the case base. When good visualisation support is available, clustering can provide a helpful impression of the distribution in the case base. It is very useful as an exploratory exercise before further working. It can help us to determine whether the data are sufficient or not, or, whether we need to select proper pre-processing or re-model the case base.

Meanwhile, in order to achieve the NDM model discussed in Chapter 2. Our clusters can be arbitrarily shaped and their borders may not be crisp, thus most common cluster algorithms cannot successfully cluster unless they are dealing with convex shapes. However, SOM can provide the visualisation as well as clustering at same time. Once the BMU of the new case is chosen on the map, decision makers can judge the relation between the new case and the original clusters based on visualisation using their own subjective judgement. This process can also be set up automatically once SOM results are combined with normal clustering approaches such as K-Means.

For SOM, different initial values, different sequence of the training vectors and different learning parameters will have different results. If the SOM network is not very large, in the order of a few hundred nodes for example, then the selection of process parameters is not very crucial. However, for large maps, it may be important to minimize the total learning time. Effective choices for those functions and their parameters should be determined by experiments. The optimal map for the same input dataset is expected to yield the smallest average quantization error.
Chapter 6

Case Base Visualisation

In order to apply CBR to implement the model for NDM, we propose to use visualisation to achieve mental simulation of NDM. In NDM, the decision maker uses mental simulation to help diagnose a situation, to identify whether or not their outcome is desirable. By mentally examining various aspects of the situation, he/she can then decide whether the case is familiar or not. With the help of different visualisations for different aspect of the case base, the decision maker can observe different facets of the elements in the case base. This process will help him/her to decide whether the case is recognizable or not. By comparing different visualisations, he/she may also consider whether the suggest COA can achieve a satisfactory result. It can help the decision marker to decide whether the COA need modified or not.

The success of CBR systems depends on the quality of the case base and the problem solving coverage it provides. However, case bases nowadays have become larger and the systems are more complex. Visualisation can help users to maintain the case base, understand its structure, and the relationships between cases. Not only can visualisation give users intuitive insight but it can be used to identify regions in the case base which are either over or under populated. In addition, it may be used to uncover hidden interactions between the features that are used to describe the cases, and thus provide guidance for improving the quality of the case base.

In Section 6.1, we discuss different visualisation approaches. Section 6.2 introduces visualisation in related CBR systems. Section 6.3 proposes to use ViSOM as a visualisation tool. In Section 6.4, we discuss similarity assumption. Section 6.5 presents a simple example. Finally we conclude this chapter in Section 6.6.
6.1 Visualisation

The dimensionality of most case bases are more than two, sometime can even be very high. In order to visualise them, high dimensional case bases should be projected to a low dimensional space. Data projection is widely used in many areas such as decision support, bioinformatics, financial analysis and knowledge management, etc. The goal of the projection is to represent high dimensional data in a lower dimensional space in such a way that certain properties of the structure of data are preserved as faithfully as possible. The projection can be used to visualize the data if a sufficiently small output dimensionality is chosen.

6.1.1 PCA

There are various projection methods that can be used with different strengths. Classic projection methods, such as PCA are computationally light (Bishop 1995). PCA projects the data onto its principal directions, which are represented by the orthogonal eigenvectors of the covariance matrix of the data. It can effectively reduce the data variables and is the optimal linear projection because it minimizes the mean square error between the original data and the projection on the principal subspace. But it may lose useful information when dealing with highly nonlinear data, as it cannot capture nonlinear relationships which are higher than second order statistics. When the dimensionality is much higher than two, it only can provide limited visualisation power (Wen 1998).

It cannot take into account nonlinear structures, structures consisting of arbitrarily shaped clusters or curved manifolds, since it describes the data in terms of a linear subspace. Because high dimensional data are usually located in low dimensional nonlinear manifolds, nonlinear projection often provides a more accurate representation of the data. There are many extensions to nonlinear PCA, such as principal curves (Hastie & Stuetzle 1989) and principal surfaces (Kraaijveld, Mao, & Jain 1995), but a valid algorithm is required for a good implementation.

6.1.2 Multidimensional scaling (MDS)

MDS project data onto a two dimensional subspace by preserving as close as possible the inter-point metrics. It can preserve the pair-wise distances between data points...
thus they are proportional in both the mapping space and the original input space. The mapping is normally nonlinear and can reveal the overall structure of the data. A general fitness function can be described as

\[ S = \frac{\sum_{i,j} (d_{ij} - D_{ij})^2}{\sum_{i,j} D_{ij}^2} \]  

(6.1)

where \( d_{ij} \) represents the proximity of \( i \) and \( j \) in the original space, \( D_{ij} \) represents the distance between \( i \) and \( j \) in the projected space.

MDS depends on an optimization algorithm to search for the configuration which gives the best value of the fitness function. A gradient method is normally used. However, it is computationally intensive and may have problems such as local minima.

### 6.1.3 Sammon Mapping

Sammon Mapping is a well-known example of MDS, it tries to optimize a fitness function that describes how well the pairwise distances in the dataset are preserved. A general fitness function can be described as

\[ S_{\text{Sammon}} = \frac{\sum_{i<j} \frac{(d_{ij} - D_{ij})^2}{d_{ij}}}{\sum_{i<j} d_{ij}^2} \]  

(6.2)

where \( d_{ij} \) represents the proximity of \( i \) and \( j \) in the original space, \( D_{ij} \) represents the distance between \( i \) and \( j \) in the projected space.

Sammon mapping uses the Newton optimization method for the optimal configuration. It converges faster than the simple gradient methods, but the computation is even more complex. Like other MDS methods, it is a point to point mapping, which does not provide any explicit mapping function and cannot accommodate new data. For any additional data, the projection has to be recalculated from scratch based on all
data. This makes it computationally intensive; hence, it is very difficult to apply Sammon mapping when the dataset is large and the memory space is limited.

6.1.4 SOM

For visualisation, in principle, cases close to each other in the projection should also be close to each other in the input space. However, this does not always hold; the projection often folds badly due to the low dimensionality of the output space. While simplicity is gained by reducing dimensionality, information is effectively lost when the data item is simply represented by a dot. There are many methods available to enhance projection visualization. For example, selected objects can be connected by lines or represented by different colours. Here we review some common approaches used for SOM visualisation.

Honkela (Honkela, Kaski, & Kohonen 1996) suggested to plot a diagram for each node on the map, inside each node, for every high dimension vector, using a two dimensional coordinate plane and to incorporate further axes for each additional dimension. Thus every vector can be plotted as a curve connecting the axes. The intercept at axis $i$ corresponds to the value of the high dimensional data in dimension $i$ as shown in Fig. 6.1.

![Fig 6.1 Honkela’s suggestion for visualization.](image)

There are a large number of visualization techniques map each dimension of the input space onto a certain feature of the icon, such as the famous Chernoff’s Faces (Chernoff 1973). It uses simple cartoon like faces to represent multidimensional data. Each dimension of the input vector is assigned to a facial characteristic, such as nose
size, eye spacing, mouth width etc. This approach depends on human's ability to recognize small difference in facial characteristics.

Instead of faces, many methods use geometric figures, each feature of vector corresponds to the length of an edge, the radius of a circle, the colour of a hexagon or another property of the icon (Pickett & Grinstein 1998). Colour as a means of representing dimensionality is used in numerous SOM visualizations. Since every colour has a Red, Green and Blue component, it is capable of displaying three dimensions at once. Meanwhile, it appears to be very easy for the human eyes to spot similarities in colour. In (Kaski, Venna, & Kohonen 1999), the perceptual differences of the colours reflect the distance relations within the cluster.

When the correlation between different input sample features is of the interest, "Component Planes" can be used. It shows each weight vector component on a separate plane. Users can examine which features are distributed in a similar way. An example is shown by Fig 6.8.

Fig 6.2 Squares and Hexagons (Simula et al. 1999)

Fig.6.2 shows a single comment plane with a Hit Map on top of it. The size of the red square indicates the number of items in the dataset that map to the associated node. If a node represents the best matching unit for a large number of dataset items then the square will be large, otherwise it will be small.

SOM is a very good tool for non-linear smooth mapping high-dimensionality data into a low dimensionality feature map, typically of one or two dimensions. It creates a set of prototype vectors which represent the data set and achieve a topology preserving projection of a high dimensionality input space onto a low dimensionality grid. The
accuracy of the maps in preserving the topology or neighbourhood relations of the input space has been measured in various ways.

The Average Quantization Error is the average distance between the vector data from their prototypes.

\[ AQE = \frac{1}{N} \sum_{x} \| x - w_{c(x)} \| \]  \hspace{1cm} (6.3)

The goodness of a map can be derived from this as

\[ \frac{1}{1 + AQE} \]  \hspace{1cm} (6.4)

Generally, when the size of the map increase, there are more units to represent the data, therefore each data vector will be closer to its best matching unit, thus the \( AQE \) will be smaller.

Topographic product introduced by Bauer and Pawelzik (Bauer & Pawelzik 1992) try to find folds on the maps. Since the SOM approximates the higher dimension of input space by folding itself, the topographic product can be an indicator about the topographic error. However, it does not differentiate the correct folds following a folded input space and the actually erroneous folds. Kohonen himself proposed another approach to measure the proportion of all data vectors whose first and second best matching units are not adjacent (Kohonen 2001). This is called topographic error. The smaller the topographic error is, the better the SOM preserves the topology. Generally, the higher the dimensionality of the input space, the larger the topographic error is. This is because the increasing difficulty to project units in right order as the dimensionality of the prototype grows.

There have been some research efforts to enhance the topology preservation of SOM. In (Kirk & Zurada 2000), a SOM was trained to minimise the quantization error first, and then minimise the topological error in the second stage. A Double SOM (DSOM) uses dynamic grid structure instead of a static structure, together with the classic SOM learning rules to learn the grid structure of the input data (Su & Chang 2001). The Expanding SOM (ESOM) preserves not only the neighbourhood information but also
the ordering relationships, by learning the linear ordering through expanding (Jin et al. 2004).

SOM can be used as a visualisation map to show the relationship between data and clusters. Data points that are close by are usually projected to nearby nodes. However, SOM can only preserve topological structures of input data on the output space. It cannot faithfully portray the distribution of the data and its structure. It does not directly apply to scaling, which aims to reproduce proximity in distance on the lower visualization space. The map does not show the distance between the neurons. A colouring or grey tone scheme is applied to the map for marking relative distances between the neurons expressed by the U-matrix.

6.1.5 Visualisation induced SOM (ViSOM)

ViSOM (Yin 2001) uses a similar grid structure as the normal SOM. The difference is how it updates the weight of neurons. In SOM, the weights of neurons in the neighbourhood of the winner are updated by

\[ w_k(t+1) = w_k(t) + \alpha(t) \eta(v,k,t)[x(t) - w_k(t)] \]

(6.5)

where \( \eta(v,k,t) \) is the neighbourhood function.

The second half of it, especially the updating force \( x(t) - w_k(t) \) can be rearranged as,

\[ F_{xv} = x(t) - w_k(t) = [x(t) - w_v(t)] + [w_v(t) - w_k(t)] = F_{vx} + F_{kv} \]  

(6.6)

\( F_{xv} \) represents the updating force from the winner \( v \) to the input \( x \). It adapts the neurons toward the input in the direction which is orthogonal to the tangent plane of the winner.

\( F_{kv} \) is a lateral force which brings neuron \( k \) to the winner \( v \). This contraction force brings neurons in the neighbourhood toward the winner and thus forms a contraction around the winner on the map at each time step. The scale of this force is controlled by the normalized distance between these two weights. As shown by the following figure.
Fig. 6.3 Decomposition of the SOM updating force (adopted from Yin 2002)

ViSOM updates the neighbourhood according to

\[ w_v(t+1) = w_v(t) + \alpha(t) \eta(v, k, t) \left( [x(t) - w_v(t)] + [w_k(t) - w_v(t)] \right) \left( \frac{d_{vk}}{\Delta_{vk}\lambda} \right) \]  \hspace{1cm} (6.7)

where \( w_v(t) \) is the weight of the winning neuron at time \( t \),

\( d_{vk} \) is the distance between neurons \( v \) and \( k \) in the input space,

\( \Delta_{vk} \) is the distance between neurons \( v \) and \( k \) on the map, and

\( \lambda \) is a positive pre-specified resolution parameter; the smaller it is, the higher resolution the map provides.

In the ViSOM, the distances between the neurons on the map are analogous to the distances of their weights in the data space by adjusting the inter-neuron distances on the map in proportion to those in the data space. The distance between neurons is proportional to that of the original data space, subject to the quantization error. This is very similar to Sammon mapping to achieve the proportionality. For SOM, when the size of the map increases, the average quantization error will be smaller. For ViSOM, the quantization errors also reduce while the number of nodes increases. When the resolution of the map is high enough, the distance between projected data on the map will reflect those in original space. The map preserves the inter-neuron distances as well as topology as faithfully as possible. The SOM and ViSOM are similar only when the data are uniformly distributed, or, when the size of the map is very large.

In Fig. 6.4, 6.5, 6.6 and 6.7, PCA, Sammon mapping, SOM and ViSOM respectively are applied to Iris data (Fisher 1936). From the results, Sammon mapping is better
than PCA, as it reveals the structure details better. The ViSOM result closely resembles that of the Sammon mapping except that the data points are more separated. Each individual data point can be seen more clearly, instead of overlapping as would occur in Sammon mapping. Therefore, along with maintaining the intercluster structures more details of intra-cluster and inter-point distribution is provided by ViSOM. Similar experiments about other datasets can be found in (Sarvesvaran & Yin 2004; Yin 2002).

Because of the absence of an explicit projection function, MDS such as Sammon mapping is not suitable for visualising CBR; particularly, where new cases are continually added. The dimensionalities of most case bases are much higher than two, thus PCA cannot provide reasonable visualisation result for CBR neither.

Fig. 6.4. PCA for Iris data

Fig. 6.5. Sammon mapping for Iris data

Fig. 6.6. SOM of Iris data

Fig. 6.7. ViSOM for Iris data (Yin 2001)
The ViSOM is as simple as SOM, the only difference is whether your visualisation should consider the scaling or not. ViSOM can provide visualisation with similar capabilities as the Sammon mapping as it allows the new data projected on the map without the need to recalculate from scratch. Most algorithms need to be re-run when the dataset changes. With SOM and ViSOM, the incremental updates can be carried out at relatively low computational cost. When a new case is added into the map, we can simply carry out a single iteration of the algorithm to accommodate it.

Another benefit of using SOM for case base visualization is the convenience. SOM is a very popular technology for clustering and visualization. There are many ready toolboxes and software available, which means that ViSOM can be easily updated. Therefore employing SOM and ViSOM for case base visualisation offers great advantages based on the evidence of the research results from other related areas.

### 6.2 Visualisation in related CBR systems

Many CBR applications use graphical representations to present the solutions. In (Mileman et al. 2000), a system for retrieval of complex 3D shapes for metal casting industry is described. Retrieved and adapted prototyped casting is presented by a set of 2D projections of major components. In (Silva Garza & Maher 2001), a system providing aid in house design use more extensive visualization method to illustrate all stages of adaptation. Genetic algorithms are used for adaptation and the visualisation clarifies the evolutionary process of solution generation. Hua and Faltings developed an architectural design tool, CADRE, to create new designs based on old ones (Hua & Faltings 1993). Visualisation can also offer powerful analytical means to uncover patterns and trends in the case base. Mullins and Smyth developed a CBR shell called CASCADE (Mullins, Smyth, & McKenna 2000) which can maintain a real time visualisation of the case base to help the user improve the competence of the case base. A Forced-Directed Graph-Drawing algorithm was used to represent a case base on a two dimensional screen whilst preserving the similarity relationships between cases as on-screen distances.
6.3 Visualisation for CBR

CBR systems undergo continuous transformation grow in size and complexity rapidly. Visualisation can provide significant support for CBR applications. In this section, we will discuss how visualization can achieve this.

One of disadvantage of CBR is that the case base must cover the majority of the problem space; otherwise the system will face the problem that a new case cannot find any match in the case base. In which case, the system has to adapt the solution of a retrieved case with very low similarity to the target case. Therefore in order to achieve success with CBR systems then the case base should be composed of problems which tend to recur. Hence, future problems are likely to be similar to current problems and case base will contain cases similar to the new problems it encounters. The relationship between old and new problems is called ‘problem-distribution regularity’ (Leake & Wilson, 1998).

Once the map of the case base is set up, it may suggest that some regions in the case base are over or under populated. Thus we can adjust the case base to achieve better coverage and distribution. It is relatively straightforward to demonstrate whether the target case is close to the previous cases or not. If most new cases are far away from original cases, there is insufficient case coverage of current problems or even the suitability to apply CBR to find solution for new cases is doubted. The map can also help us to identify the “hot spot” in case base, thus it is possible to reorganize the case base to speed case retrieval or to delete cases which have never been retrieved.

Meanwhile, visualisation with SOM can also give us an intuitive understanding of the case base structure. It can help us discover patterns and trends in case base. In this, component planes play the key role. Each component planes consists of the value of a single case feature in all map units. Simple inspection of a component plane provides an idea of the spread of values of this case feature. The component planes can also help us find the correlations between a pair of features in the cases. Human eyes are good at pattern matching. Especially with SOM that have a regular shaped grid map. For example, Fig. 6.8 shows every feature dimension of the Iris data. It is easy to find the last two features of these case bases since they have a similar pattern.
Further, it is easy to select interesting feature combinations for future investigation. For example, visualisation can be set up for a function of each other of the two different features. Furthermore, once the visualisation of case base is set up, and new case occurs, SOM can help classify and examine whether the new data belongs to same data distribution as the original map. It is very useful to investigate which part of the case map distribution best corresponds to the new case, where on the map the new case located, how accurate is that correspondence and how similar are two cases in terms of the map, etc.

![Feature values of Iris data](image1)

![U-Matrix](image2)

**Fig. 6.8.** Feature values of Iris data  
**Fig. 6.9.** U-Matrix shows the solution values

Instead of simply pointing out the Best Matching Unit (BMU), how well the prototype vectors reflect to the original case can be visualised. A simple approach is to use the quantization error as an indicator. Fig 6.10 shows one example. In a poor match, there are large areas on the map which correspond to the case as well as the BMU, depicted with large dark areas. In a good match, a clear cluster can be seen.
Another approach is to position the cases in the visualisation so that the accuracy is apparent from either the size or the position of the sample marker. Fig. 6.11 shows an example. In Fig. 6.11(a), a kernel density estimator has been implemented based on the map. The upper grid marks the 50% probability boundary of the Gaussian kernel associated with each of the map units. Each bar represents one case, the higher the bar, the less probable such a case is. In (b), the interpretation is more straightforward, the diameter of the circle is scaled by the average distance of each map unit to its neighbour. If the circle is smaller than the hexagon, the BMU is closer to the case than to its neighbour.
Additionally, the success of CBR systems also depends critically on the case base itself, which is the quality of the individual cases. We will discuss how visualisation can help us uncover the quality of case base.

6.4 Similarity assumption

One of the major assumptions in Artificial Intelligence is that similar experiences can guide future reasoning, problem solving and learning. This is known as the similarity assumption. “Similar problems have similar solutions” is stated for almost all problem solving paradigms and representation, such as supervised learning, rule-based induction, and decision tree induction etc.

In CBR, the similarity assumption plays a central role. The success of any CBR systems is contingent on retrieving cases that are relevant to the target problem. One can argue that if the system cannot fit “similar problems have similar solutions” then CBR is not suitable, or, there is noise in the case base. In fact, it is not always the truth. In real life, irrespective of the similarity metric selected, for many cases, the problem and solution are disproportionately distant from the target case. How well the distance of the problem space approximates to the distance of the solution space is called ‘problem-solution regularity’. A function can be applied to measure the probability that a case returned as optimal by similar function will actually be an optimal case (Leake & Wilson, 1998). Bergmann et al. have also addressed this problem in that the similarity of cases in the problem space does not always correspond to the usefulness of the cases to solve the problem (Bergmann et al. 2001).

Smyth and Keane proposed adaptation-guided retrieval to augment traditional measures of similarity with adaptation knowledge about whether a case can be easily modified to fit a target problem (Smyth & Keane 1999). In (Premraj, Shepperd, & Shepperd 2005), the author regards these cases as unreliable and not suitable to be reused to deliver solutions. They should be dealt with extreme caution. In order to solve this problem, cases can be assessed and discriminated, while poor cases are overlooked to enhance solution quality. In (Woon et al. 2005), the problem space and solution space are treated as a unifying space. Interpolation is worked on candidate
cases which are not only close in the problem space but are also close in the solution space. In (McDonnell & Cunningham), the authors use linear models to judge the quality of cases in order to find the best difference cases by providing case gradients and identifying noisy cases. However, most of these approaches are only suitable for low dimensional solution space dataset.

Visualisation can help us evaluate how well a case base follows the similarity assumption. For a case base whose solution space is single dimensional, if the solution is categorical, such as Iris data, ViSOM can be used directly, as shown by Fig.6.5. If the solution is numerical, a colouring or grey tone scheme can be applied to demonstrate the solution. ViSOM can be used to describe the case problem space, because the inter neuron distances can reflect the original data distances. In addition, colour or grey tone can be applied to express the solution as well. For example, in Fig.6.9, the colour of each neuron demonstrates the value of solution of the corresponding case. In this way, how close a case is to its neighbours, can be represented by the location distances of the corresponding neurons. Meanwhile, the proximity of their solutions can be represented by the colour of these neurons.

For case base whose solution space is multi-dimensional, colour or grey tone can be used to express the solution difference between the case and its closest neighbour case. Similar to Fig. 6.9, but this time the colour demonstrates the difference between solutions instead of the value of solution itself. Specifically, the larger the solution distance is then the lighter the colour of the neuron. Hence a case base with many dark colour neurons is more suitable for CBR because it follows the similarity assumption. If necessary, two maps can be set up for visualisation purposes; namely, one map to portray the case problem space and the other to represent the case solution space. Such an approach will not only help in the analysis of the quality of the case base but will also help evaluate the difficulty of adaptation of the previous case solution to the new target case.

6.5 A simple example

An example is discussed in this section in order to demonstrate how visualisation can provide an insight to the case base including its problem-solution regularity. This is a
very simple example with high dimensional problem space and solution space, thus two maps can be set up for comparison. Another reason for using this simple artificial dataset is that if some unexpected result occurs, we can easily find where the problem is. This dataset is referred to as “dataset1”, as shown by Table 6.1.

![Table 6.1. Dataset1](image)

Imagine it as a simple terrain, which is represented by a discrete grid of square cells with binary values, where 1 represents the presence of an obstacle, and 0 represents flat terrain. A tank is moving from the left to the right. Our task here is to find the tank’s route.

The case problem space is the representation of the terrain, which is described by a vector created by reading the values of the terrain representation from top to bottom, left to right. For example, the corresponding representation of Table 6.1 is 01 10 10 01 10 10 10 10. The case solution space is the representation of the route through the grid, giving the sequence of row indexes traversed. In other words, to find the index of the row which is free of an obstacle. If the top row is free of an obstacle, it is represented by 0. If the bottom row is free of an obstacle, it is represented by 1. For example, the corresponding solution representation of Table 6.1 is 01101101.

The value in each cell is randomly chosen, and the corresponding solution is recorded. In this way, 254 different cases are collected for different possible terrains. For each case, the case problem space has 16 dimensions while the case solution space has 8 dimensions. This dataset is so simple that it is easy to tell the solution space is highly related to the problem space. Actually, once two maps of this dataset are set up, the case problem space map is exactly same as the case solution space map, which is shown by Fig 6.12 and Fig 6.13. In which, each neuron is represented by a dot while the labels are correspond to those cases projected on the neuron. In fact, the case problem space map is exactly same as the case solution space map, all the cases projected on the same neuron of the problem space maps also projected on the same
neuron of the solution space map. Thus it is the perfect dataset for CBR, which totally follows: similar problems always have similar solutions.

Fig. 6.12. Case problem space for dataset1
However, for the many domains, especially real life problems, similar problems do not always have similar solutions as discussed. In this case, solving a new problem hugely rely on adaptation. A consideration is whether the initial retrieved solutions offer any advantages, or, the adaptation is so demanding that it would be better to solve the problem from scratch without employing CBR.

For example, “dataset1” can be modified by simply adding another row of cells or by changing values of some columns. Fig. 6.14 and Fig.6.15. show the visualisation of “dataset2”, which is modified by adding another row on “dataset1”. There are obvious differences between the case problem space map and case solution map. The more we add the input dimensions or modify the values of the columns, the more obvious the difference between the two maps is. This means the CBR adaptation will be commensurately more difficult.
Fig. 6.14. Case problem space for dataset2
Fig. 6.15. Case solution space for dataset2

6.6 Conclusions

The usefulness of graphical displays arises from the power of the human visual system in detecting patterns. For CBR, visualisation can provide an intuitive insight to the case base and help us analyse the case base. However, there is so much information in the case that is impossible to show it all in a single figure or visualisation. The number of visual dimensions normally determines how many different kinds of information can be efficiently inserted into one visualisation. Typical visualisation dimensions include position, size, color etc. Due to limited number of visual dimensions, multiple visualisations can be used instead of one. That is why we propose to use SOM or ViSOM for case base visualisation. Not only can it conveniently project high dimensional case into a two dimensional figure, giving an idea about the overall shape and possible cluster structure of the case base, but also it can provide many other visualisation to help us analyse the characteristics of features.
in each case. Moreover, it can be used to examine each case for classification and novelty detection purposes.

In our NDM model, we propose to use visualisation to achieve mental simulation of NDM. With different visualisations for different aspect of the case base, the decision maker can examine various aspects of the elements in the case base. He/she can then decide whether the case is familiar or not. By comparing different visualisations, he/she may also consider whether the suggest COA can achieve a satisfactory result. Different aspects of the action can be analysed to help the decision marker to decide whether he/she needs to modify the COA or choose another action for consideration.

Visualisation can also indicate the two relationships in CBR community: problem-solution regularity and problem-distribution regularity, thus it can be applied to automatically monitor the appropriateness of the case base when new problems occur. In this chapter, we use a simple example to demonstrate how visualisation can help us to investigate the problem-solution regularity. However, the demonstration is too simple and the result is still vague. In order to achieve a better result, a more specific measurement is needed to give more accurate evaluation about the quality of case base, which will be discussed in Chapter 7.
Chapter 7

Case Quality Assessment

In our research, a case is represented by the scenario and the corresponding COA. Both are composed of complicated and high dimensional data. Thus the case base we are concerned with is far more complex than that encountered in the most common CBR systems. In order to achieve better adaptation result, we need measure the quality of cases in our case. Thus in this chapter, we shall discuss some case quality assessment approaches.

Nowadays, large case libraries are increasingly prevalent. Some case library sizes can reach thousands or even millions of cases. While new cases are constantly added to the case base, the growing cost of case retrieval outweighs the efficiency benefits from additional cases. As a result the research topic “Case-Based Maintenance” (CBM) has become very popular in the CBR area. It implements policies for revising the organization or contents of the case base in order to facilitate future reasoning for a particular set of performance objectives (Leake & Wilson 1998). Case quality assessment is a very important part of CBM, in which, cases are evaluated by the different criteria in order to control the growth of case base. However, it is rare to consider case quality assessment outside CBM.

In Section 7.1, we review related case quality analysis work in CBM area. Section 7.2 introduces Problem-Solution Regularity, together with proposing a new approach to calculate it. Section 7.3 presents a novel method for case quality assessment based on correlation. In Section 7.4, a number of experiments are undertaken to analyse how the case quality assessment can help case adaptation, where normal KNN is employed as a test and simple case deletion is used. Finally in Section 7.5, we conclude this chapter.
7.1 Case Quality Analysis in CBM

As the new case is constantly added to the case base, the problem solving ability of CBR system is enhanced. However, the complexity of systems is also increased and there must be inconsistent redundant cases added to the case base, which will affect the system performance (Leake & Wilson 1998). Intuitively, an effective case base is one which is able to answer as many queries as possible efficiently and correctly. There are many different ways to detect redundant and inconsistent cases in CBM. In (Racine & Yang 1996), some criteria for evaluating case bases are proposed as follows.

- Consistency includes intra-case consistency and inter-case consistency. The former refers to whether the case is consistent with the background knowledge while the latter is that two cases must be consistent with each other when both are used in a composite solution.
- Correctness of a case base is measured by how often the case that is retrieved is the case in the case base that answers the query most effectively.
- Redundancy is to determine if the incoming case is subsumed by other cases in the case base or if it subsumes existing cases. If two or more cases in a case base are very similar and are retrieved for the same queries, it is unnecessary to keep both in the case base and may degrade the efficiency.
- Revision Effort is defined as the cost associated with revising the retrieved case to answer the query.
- Coverage is the size of problem set that can be solved by a certain case. It provides a precise measure of competence contributions for individual cases.
- Reachability is the size of the case base that can provide solution to the current problem.
- Retrieval Cost is the cost to retrieve the correct case from the case base, given the problem description.
- Relevancy is how the case present to the user relevant to the problem at hand.
- Abstractness is how well the case can be generalized.

In CBM, cases are classified according to these criteria, mostly according to two concepts: coverage and reachability, as shown by Fig. 7.1. Set A and B represent the coverage and reachability of case e, respectively. Set C, which is the intersection of A
and B represents those cases that can be solved by case e, or reciprocally, those cases in the intersection can be used to solve case e. Those cases with larger coverage have higher contribution to the competence of the case base. On the other hand, cases with large reachability have lower contributions to the competence of the case base, because these cases can be solved by many other existing cases.

Fig 7.1 Case coverage and reachability

Cases that are the only case that can answer a specific query are pivotal cases. Cases which are completely subsumed by other cases are auxiliary cases. Between them are the spanning cases. In addition, support cases exist in groups which support an idea. It is easy to see that pivotal cases are the most unique, spanning cases are less, but auxiliary cases are not unique. Then according to different cases, appropriated maintenance technique such as deleting and sieving are applied (Racine & Yang 1996), (Smyth & Keane 1995). However, some of the criteria are very difficult to measure, especially when they are employed before the case adaptation in the CBR cycle. Most CBM criteria are only suitable for case maintenance where the system has been used for a period of time. Therefore these criteria can be evaluated according to the performance of each individual case during this time. In our case, we need to a criterion to work before case adaptation or even the case retrieval stage. A simple and straightforward case quality analysis approach is preferred in to assist us in order to increase the accuracy of solution prediction.
As discussed in Chapter 5, in an ideal CBR system, the case with closest problems would also have the closest solutions. In practice, the similar problems may not have similar solutions. In (Faltings 1997), the author uses probability theory to prove that the assumption that a problem with similar features to an earlier one is likely to have a similar solution is guaranteed to be true on average. In the real world, the similarity of cases in the problem space does not always correspond to the usefulness of their associated solutions to solve the problem. In (Leake & Wilson 1998), the authors use 'problem-solution regularity' to represent how well the similarity between problems approximate the similarity between according solutions in practice. They follow the notion of precision in the information retrieval area, measure the probability that a case returned is an optimal case. However, it is not easy to apply this measurement in practice; especially, when it is used as a criterion for case filtering before any case retrieval occurs.

In our research, as both the case problem space and the case solution space already have been visualised and transformed into two dimensional space first, their topological relations in original high-dimensional space are kept as good as possible in the two dimensional space. Thus the problem-solution regularity can be considered as the similarity between the case problem map and case solution map. For example, Fig. 7.2 shows an example of three cases A, B, C, in the case problem space while Fig. 7.3 shows their corresponding solutions A', B', C' in the case solution space. Therefore, the problem-solution regularity, we call it PSR here, can be regarded as the similarity of these two triangles, such as

\[
\text{PSR of case A, B, C} = |\angle a - \angle a'| + |\angle b - \angle b'| + |\angle c - \angle c'| \quad (7.1)
\]

\[
\cos \angle a = \frac{AB \cdot AC}{|AB| \cdot |AC|} \quad \text{and} \quad AB = B - A \text{ while } AC = C - A \quad (7.2)
\]
In this way, we can calculate the difference between angles in case problem space and those in case solution space, thus the problem-solution regularity is acquired. Similarly, if all the cases are connected one by one to become a polygon, such as Case1 connects to Case2, Case2 connects to Case3, ..., Case N-1 connects to Case N, case N connects to Case 1. Suppose the angle points to Case 1 in case problem space is called $\angle C_1$ while that of case solution space is called $\angle C_1'$; the angle points to Case 2 is called $\angle C_2$ while that of case solution space is called $\angle C_2'$; ...; the angle points to Case N is called $\angle C_N$ while that of case solution space is called $\angle C_N'$. Therefore, the problem-solution regularity of this case base can be regarded as the similarity of two polygons, one for the case problem space while the other is for the case solution space.

$$PSR = |\angle C_1 - \angle C_1'| + |\angle C_2 - \angle C_2'| + ... + |\angle C_N - \angle C_N'|$$

(7.3)

This is a shape comparison or matching problem. There are many research about shape matching in computer vision area, for example turning function can be used to measure the similarity of two polygonal shapes (Arkin et al. 1991). However, in our research we are more interested in how to find the specific poor quality cases among the case base. 'problem-solution regularity' can be used as a measurement to evaluate the quality of whole case base instead of individual case, the detail of calculating it is not the focus of this research.
7.3 Correlation for case quality measuring

In our research project, in order to apply CBR more efficiently, cases that do not follow similarity assumption need to be filtered at first. In this section, a new case quality measure based on correlation is proposed.

In multivariate statistics, the Pearson correlation coefficient is often used to measure the linear dependency between two random variables.

\[ r = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{(n-1)s_x s_y} \]  

(7.4)

where \( \bar{x} \) and \( \bar{y} \) are the expected values and \( s_x \) and \( s_y \) are standard deviations for \( X \) and \( Y \), \( n \) is the size of sample. The value of the correlation coefficient ranges from -1 to 1. The larger the absolute value of \( r \), the greater the linear dependence between \( X \) and \( Y \). Positive values indicate that \( X \) increases with \( Y \) while negative values indicate that \( X \) decreases with \( Y \). A zero value indicates that there is no linear dependency between \( X \) and \( Y \) (Hoel 1984).

In our system, the distance between each case pair in the problem space and the solution space are calculated first. For case 1, C1-C2, C1-C3,... C1-Cn in the problem space will be regarded as the value of variable \( X \) in equation (7.4); C1-C2, C1-C3,... C1-Cn in the solution space will be regarded as the value of variable \( Y \) in equation (7.4).

Then the correlation is acquired, the larger the correlation of problem space distances between case 1 and other cases and their solution space distance, the more the case 1 follows "similar problems have similar solutions", thus the better quality case 1 is. Hence, correlation of case 1 is regarded as a statement of the quality of case 1.

Generally, we are only interested in cases which are close to the target case. For those cases which follows the similarity assumption, but have large difference in both problem and solution space, we will discard them as well. Thus, only cases in the
limited neighbourhood $h$ are used to calculate the correlation, where $h$ is a neighborhood size defined according to size of case base.

Similarly, we can determine the quality of case2, 3, ..., n, for every case in the case base. This measure can also help us maintain the case base, thus increasing the quality of case base and its accuracy for solution prediction. Once cases with good correlation between the problem space and the solution space are acquired, the likelihood of a successful adaptation is high.

Similar to case addition, case filtering is very important for case maintenance nowadays. There may be redundant cases and unreliable cases in the case base. We could locate cases that represent or cover other cases in this situation, and delete those redundant cases. For case adaptation, cases with high quality should have priority over low quality cases.

### 7.4 Experiment result

First, we use Boston housing database from UCI Machine Learning Repository (Blake & Merz 1998) as our experiment dataset. It has 506 instances, 12 numeric and 1 binary attributes. The correlation calculation is based on a limited neighbourhood ($h=40$). This means only the closest 39 neighbours of each case will be used to determine the correlation.

Generally, when data is composed of mixed types such as binary, ordinal and numeric etc, in order to measure their similarity, we need to normalize all the variables first, and assign a different weight to each feature if necessary. The aggregated similarity is the simple weighted average of distance matrices of each variable.

\[
d_{ij} = \frac{\sum_{k=1}^{n} w_{ik} \delta_{ij}}{\sum_{k=1}^{n} w_{ik}}
\]

where $\delta_{ij}$ is the dissimilarity between object $i$ and object $j$ for feature $k$.
For this dataset, the correlation of most case are less than 0.6, there are even some negative values. In order to analyse how the case quality measure can help case adaptation, normal KNN is employed as a test, and simple case deletion is used. The following figures show the results. When K=7, normal KNN has the best result. After cases with low correlation are filtered, the result improves. However, if the threshold of correlation is set too high, such as 0.3, the result is not as good.

![Graph showing KNN results for Boston housing database](image)

**Fig. 7.4.** KNN result for Boston housing database (The horizontal axis is the K of KNN, which is from 1 to 15. The vertical axis is the mean absolute error)

As discussed, cases in our research are composed of complicated and high dimensional data. In order to evaluate whether our approach can be applied to other similar data, we need to find a public dataset of case bases with high-dimensional solution space. However, it is difficult to find such kind of case base. We choose Water Treatment Plant Database, also from UCI Machine Learning Repository, as our experiment dataset. There are 38 numeric attributes, 527 instances. After cases with missing values are discarded, there are 380 cases. We divided the first 30 attributes as the problem space while the rest 8 attributes as the solution space (In a dataset, attributes with known values are normally regard as problem part while attributes with unknown values are referred to solution part. Thus problem part and solution part can be divided by a human. Different dividing methods are tested on this dataset with similar results). The problem space need to be normalized first, otherwise the distance
calculation will be dominated by some dimensions. The calculation is based on a limited neighbourhood (\( h= 30 \)). This means only the closest 29 neighbours of each case will be used to determine the correlation. For this dataset, the correlation of most case are less than 0.45, there are even some negative values. In order to analyse how the case quality measure can help case adaptation, normal KNN is employed as a test, and simple case deletion is used. The following figures show the results. Error is calculated as the mean Euclidean difference between the output and the corresponding real case.

Fig. 7.5. KNN result for Water Treatment Plant (K=1)
(Horizontal axis is the No. of evaluation while Vertical axis is the value of error)

Fig. 7.6. KNN result for Water Treatment Plant (K=3)
(Horizontal axis is the No. of evaluation while Vertical axis is the value of error)
Fig. 7.5 shows when K=1, the result of KNN with no threshold applied, KNN with threshold of correlation=0.1, KNN with threshold of correlation=0.15, KNN with threshold of correlation=0.2. We can find that deleting the low quality cases makes the result worse. This is because the closest neighbour of target cases could be deleted. The higher the correlation threshold used on the correlation means more cases are discarded. For this reason, the result will be even worse.

Fig. 7.6 shows the result of same approaches while K=3. After low quality cases are deleted, the adapted ability becomes better than normal KNN, esp. KNN with threshold of correlation=0.1. However, the result of KNN with threshold of correlation=0.2 is worse than that of KNN with threshold of correlation=0.1, this is because the higher the correlation, the better the case quality is, but more cases are deleted as well. There are possibly not enough high quality cases in the case base, thus cases which are far from target case but with high quality may be retrieved.

Fig. 7.7 shows the result of same approaches while K=5. K=5 is the best choice for normal KNN with the best result. After low quality cases are deleted, the result are much better then normal KNN, especially KNN with threshold of correlation=0.1. Fig. 7.7 shows the result of same approaches while K=7, the results become worse than normal KNN after the cases are deleted.

In summary, when K=5, KNN with threshold of correlation=0.1, the result is best. The correlation can help us to choose the better quality cases, but if the threshold is too high, many close cases will be filtered; thus, cases which are far away from target but with high quality will be retrieved. This will make the result worse. Therefore it is essential to define a suitable threshold according to each individual case base specifically.

Another interesting finding is, for some datasets, such as "dataset2" discussed in Chapter 6, the case correlations of this dataset are mostly larger than 0.7, which means case quality is very good, in this situation, any case deletion will affect the case adaptation ability. This is because even the worst quality case in this dataset has very high correlation and it is good enough to predict its close target case. Any case
filtering will cause the distant cases to be retrieved and make the result worse. Fig. 7.9 shows result of KNN with threshold of correlation=0.75, with best choice of K (K=3).

Fig. 7.7. KNN result for Water Treatment Plant (K=5)

(Horizontal axis is the No. of evaluation while Vertical axis is the value of error)

Fig. 7.8. KNN result for Water Treatment Plant (K=7)

(Horizontal axis is the No. of evaluation while Vertical axis is the value of error)
Fig. 7.9. KNN result for dataset2 (K=3)

(Horizontal axis is the No. of evaluation while Vertical axis is the value of error)

7.5 Conclusions

In this chapter, we analysed and discussed how to filter low quality case before case adaptation using an approach based on case quality assessment with correlation. Meanwhile, we also introduced how to measure the Problem-Solution Regularity of a case base. Some experiments are applied to demonstrate how the case quality assessment can help the case adaptation. It proves case quality assessment can help us implement NDM model more efficiently and effectively.

Meanwhile, we found that if the threshold of case filtration is too high, many close cases will be filtered. As a result, cases which are far away from target but with high quality will be retrieved. This will make the result worse. Therefore defining a suitable correlation threshold is the key and needs further discussion. It should be a function of the size of case base, the size of neighbourhood, and it should relate to the specific adaptation algorithms applied. Furthermore, how the low quality cases are treated will also affect the definition of this threshold.
Chapter 8

Case Adaptation

To implement NDM with CBR, the adaptation is needed to adjust the COA for the new case. Even the pattern is recognized, the mental simulation is successful, normally the stored case is not the same with the target case, thus it is necessary to adapt the case. One of advantages of CBR is that it can avoid solving problems from the scratch. However Adaptation is the most challenging issue in the CBR cycle as it is the most difficult part to handle (Leake, 1996; Bergmann & Wilke, 1998). Case adaptation is the process of operation on the solution of similar case so as to make it suitable for the target problem. Ideally, the adaptation mechanism of CBR system should produce a solution from scratch in case no similar case can be found (Waston & Marir 1994).

In this chapter, we shall discuss how to adapt case solution, especially for high dimensional solution space. The chapter is organised as follows. In Section 8.1, different adaptation models are discussed. Related works are described in Section 8.2. In Section 8.3, we use neural network to adapt case solution which is multi-dimensional, and discuss its related problem. Section 8.4 introduces an improved approach. In Section 8.5, we propose how to acquire the solution of target cases. Evaluation and example results are reported in Section 8.6. Finally, our discussion and conclusions are in Section 8.7.

8.1 Introduction

There are many different adaptation techniques available, but all having different requirements on available adaptation knowledge. It is often easy to acquire the cases, the required adaptation knowledge is often very hard to get. That is why adaptation is
considered as the most difficult step in CBR. In Richter (1995), the author describes four containers to store CBR system knowledge. They are vocabulary knowledge which is used to describe the whole domain, the case knowledge of the cases in the case base, the retrieval knowledge which is used for the retrieval of similar cases, and the adaptation knowledge which is used to transform the retrieved solution for solving the target problem.

Wilke and Bergmann classify adaptation into three main types (Wilke & Bergmann 1998): Null adaptation, transformation adaptation and generative adaptation. Null adaptation simply applies the solution from the retrieved cases to the target case. It is the approach adopted by a simple Nearest Neighbor (NN) technique and maybe combined with taking the inverse distance weighted mean for K Nearest Neighbours (KNN) when K>1. This technique is suitable for problem domains which incur complex reasoning but typically simple solutions. Transformation adaptation modifies the old solution derived from the retrieved cases. There are structural transformations, which are based on some function of the target and retrieved case feature vectors, and rule-based transformations. The rules are either elicited from the domain experts or learnt using an induction algorithm. According to whether the structure of solution is changed, it can be classified into substitution adaptation and structural adaptation two categories. In substitution adaptation, only the parameter of solution is changed. The structure of solution is not changed. Generative adaptation entails deriving the solution to the problem from scratch. The derivation is handled by the case based system, largely independent of the case base. In practical application, all the above adaptation could be combined. A general framework of case adaptation is proposed in (Chang et al. 2004), which can be easily expanded if necessary. In addition to these adaptation models, some researchers apply compositional adaptation in which the newly adapted solution components from multiple cases are combined to produce a new composite solution (Redmond 1990; Sycara & Navinchandra 1991).

8.2 Related Work

Creating an automatic adaptation mechanism that is able to determine how the solution needs to be modified to fit the new circumstances is a complex affair. Case adaptation generally requires detailed knowledge of both the task and domain at hand.
CBR is often used for situations where a theory or model to construct the solution cannot be defined due to lack of knowledge. However, in these situations, adaptation knowledge is not always accessible and available.

The simplest adaptation strategy consists of using adaptation rules to resolve differences and possible conflicts between the old case and the new problem. Unfortunately, because CBR is often applied to domains which are poorly understood or difficult to codify, it is particularly difficult to develop adaptation rules. In order to overcome the difficulties and limitations of rule-based adaptation, Leake (Leake, Kinley, & Wilson 1996) proposed a hybrid case-adaptation process combining memory of previously applied adaptations with rules that are able to find in the system's memory the appropriate information for guiding and implementing the necessary transformation. The system’s memory retains not only the transformation operation during any adaptation process, but also a trace of the steps taken during the memory’s search. Although considered powerful (Jarmulark, Craw, & Rowe 2001), Leake’s approach is limited by the need to consider only one adaptation target at any time. In addition, this approach is not appropriate for CBR systems that have a modest knowledge acquisition capability. This is because the method relies on the availability of substantial adaptation knowledge and its explicit representation. Finally, this method also relies on the intensive involvement of the user whenever the system’s adaptation knowledge is insufficient.

Hanney and Keane (Hanney & Keane 1997) proposed building adaptation rules directly from the case-base by analysing the differences between cases and their corresponding solutions, and identifying, if possible, a plausible pattern. Jarmulak et al. (Jarmulark, Craw, & Rowe 2001) also developed an adaptation method based on the use of the CBR knowledge content. Each case in the system’s memory is used as a test case and compared with the others in the case-base that are most similar to it. For each comparison made, an adaptation case is constructed. This contains information about the differences between the problems and solutions of the test case and the retrieved cases, as well as the description of the test case and the proposed solution. As a new problem arises, the adaptation cases are utilised to estimate the correctness of the proposed solution and suggest the necessary adjustments.
The above mentioned methods are referred to as "knowledge-light methods" (Wilke et al. 1997). These methods learn adaptation knowledge from the CBR system's own cases and treats them as sources of knowledge. They initially pre-process the information extracted and, afterwards, pass it to a learning algorithm. The learning algorithm must be designed with respect to the problem domain under investigation and the adaptation goal considered. It transforms the pre-processed knowledge to obtain the required adaptation solution. Knowledge-light adaptation methods must be supported by a significant amount and variety of knowledge contained in the CBR system. Insufficient knowledge can badly affect its performance. Furthermore, the adaptation knowledge obtained from a learning algorithm must be correctly and properly combined with knowledge already stored in the adaptation module, resolving, if necessary, possible contradictory and incompatible situations.

Many CBR systems' solution spaces are atomic with only one dimension, such as the price of a property or the classification problems. Adaptation for solution space which has multiple properties is much more challenging. However, in the application we consider here, CBR is applied to find the suitable COA for a military scenario. A COA is represented by the participating entities' waypoints at the corresponding times. Therefore, our case solution space is multi-dimensional. We could simply treat it as several single dimensions, apply the same methodology on each individual dimension and then combine the results. This, however, would treat a COA as a collection of independent decisions. This clearly is not correct because there are interactions between the features of the COA. That is why we propose another approach to solve case adaptation in this situation.

8.3 A direct approach and its problems

For our system, the easy and direct adaptation approach is based on the domain knowledge. We could ask human commanders to revise the suggested solution directly, or adapt retrieved cases using the adaptation rules, based on the army doctrine or other domain knowledge acquired from human commanders. Gradually the performance of the system could be improved by adding new cases.
When domain knowledge is not available, a neural network can be used to adapt the retrieved cases automatically. This is the approach we adopt in our work. In particular, we set up a three layers Back-Propagation (BP) network. The network is trained by using as input the problem space differences between all pairs of cases, while the solution space differences between the corresponding cases are the target output for each pair. For example, suppose there are five cases in the case base (C1, C2, C3, C4, and C5). Then the input of the BP network consists of the problem space differences C1-C2, C1-C3, C1-C4, C1-C5, C2-C3, C2-C4, C2-C5, C3-C4, C3-C5 and C4-C5. The target outputs are the solution space differences between the same pairs of cases. Because CBR is based on the idea that similar problems have similar solutions, in this way we could analyse how similar these cases are, and how similar their solutions are. Once the BP network is trained, the problem space difference between the target case and its most similar case is input into the network, and then the solution space difference between these two cases is obtained. Thus the solution of the target case can be achieved. Other approaches such as genetic algorithm may also be applied.

8.3.1. BP network

A BP network is supervised, multilayer, feed-forward network. It is probably the most well researched training algorithm in neural networks. It uses the delta rule, which is a Gradient Descent algorithm, to find the optimal solution, which minimises the square error between the observed output and the desired output:

$$E_d (\hat{\omega}) = \frac{1}{2}(t_d - o_d)^2$$

(8.1)

where $t_d$ is the desired target and $o_d$ is the real output.
$E_d(\vec{w})$ is called cost function, it provides a quantitative measure of the difference between the actual output and the target output. The gradient descent algorithm then tries to minimize the squared error by searching the weight space.

When interpreted as a vector in weight space, the gradient specifies the direction that produces the steepest increase. So the negated gradient indicates the direction which produces the steepest descent along the error space (Mitchell 1997).

Training begins with random weights. The training examples are iteratively presented as input and the weights are modified until all training examples can be classified correctly. The weight updating formula is:

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

\(\eta\) is the learning rate, a positive constant. Its choice is very important. If it is too large, the gradient descent will overstep the minimum and oscillate in the solution space. If it is too small, the computational cost will be very high. In most cases, we often choose a value which we gradually reduce as the iterations proceed. We can work out the gradient of the error function with respect to weight \(w_i\) as follows:
The BP network learns a predefined set of input-output example pairs by using a two phase propagate adapt cycle. After an input pattern has been applied as a stimulus to the first layer of network units, it is propagated through each upper layer until an output is generated. This output pattern is then compared with the desired output. An error signal is computed for each output unit. The error signals are then transmitted backward from the output layer to each node in the hidden layer that contributes directly to the output. This process is repeated, layer by layer, until each node in the network has received an error signal. And its connection weights have been updated. The BP network supports a wide range of applications, especially classification and prediction. Fig 8.2 shows a typical BP network.

\[
\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2
\]
\[
= \frac{1}{2} \sum_{d \in D} 2(t_d - o_d) \frac{\partial}{\partial w_i} (t_d - o_d)
\]
\[
= \sum_{d \in D} (t_d - o_d) \frac{\partial}{\partial w_i} (t_d - \bar{w} \cdot \bar{x}_d)
\]
\[
\frac{\partial E}{\partial w_i} = \sum_{d \in D} (t_d - o_d)(-x_{id})
\]

So at last we can get

\[\Delta w_i = \eta \sum_{d \in D} (t_d - o_d)x_{id} \quad (8.3)\]

Fig. 8.2 BP Network
8.3.2. Hidden Units

The number of neurons in the hidden layers is not known in advance, and usually is estimated by a trial and error approach. One approach is to set up a BP network with an excessive number of hidden units, which is larger than required then some redundant units are removed during the learning process. In order to find which hidden units can be removed, the output of all the hidden units is monitored and analysed across all the training examples after the network achieves convergence. If the output of certain hidden unit is approximately constant for all training examples, then this unit can be removed because it does not contribute essentially to the solution. If two hidden units give approximately the same or opposite output for all training examples, only one of the hidden units is needed, because both neurons convey the same information. This process is called pruning (Sietsma & Dow 1989).

Similarly, we can gradually increase the number of hidden units. Because the learning process may become trapped in a local minimum or a very flat plateau, adding extra hidden nodes may help it escape from the local minimum. Only after achieving the desired convergence, it is then possible to remove some of them in order to find the minimal size of the network which can achieve the desired result (Hirose, Yamashita, & Hijiya 1991).

However, when the case problem space and the case solution space both are of high dimensionality then the construction of the neural network under these circumstances must be complicated and a large number of cases are required to train the neural network. This is particularly difficult for cases when only a small number of training samples are available. To solve this problem, we propose to employ a SOM to project the case problem space and the solution space first, to reduce the size of the BP network. As indicated in Chapter 6, SOM and ViSOM can dramatically reduce the data dimensionality. It will help us to visualise the case base as well. In this chapter, we will also employ SOM and ViSOM for case adaptation.
8.4 Improved approach

The key feature of ViSOM is that the distances between the neurons on the map can reflect the corresponding distances in the original data space. The map preserves the inter-neuron distances as well as topology as faithfully as possible.

We employ ViSOM on both the case problem space and the case solution space. Once two ViSOM are set up, the location of cases in the case problem space ViSOM can be used as input while the location of the corresponding case in the case solution space ViSOM as output. Because these locations are only two dimensional, the structure of the BP network is much simpler than one created from directly inputting the original dataset. This approach tries to mimic the case problem space and the case solution space as input and output patterns respectively, and map the problem to the solution by weighting the connections.

Because the ViSOM can preserve the inter-point distances of the input data on the map, the located nearest case to the target case can be adapted to the target case solution. In Fig 8.3 and Fig. 8.4, a 3-layer BP network was used for demonstration purposes. The input vector has 2 elements. There are 5 neurons in the hidden layer while 2 neurons in the output layer, which are applied in our experiment. The structure of the BP network may vary.

![Diagram of BP network with SOM data input and output](image)

**Fig. 8.3** The structure of the BP network with SOM data input and output
The other approach is, instead of the location, the location difference between each case pair of case problem space ViSOM is used as the input while the location difference between the same case pair of case solution space ViSOM as the output. The difference between target case and its nearest case is input to the trained network after the network is trained. In this way the target case location on case solution space ViSOM is acquired.

![Diagram](image)

**Fig. 8.4** Another approach to train the BP network

### 8.5 How to find the target case solution

After the target case location on the case solution space ViSOM is acquired, if there is a previous case projected on the same location, the solution of this case will be chosen as the target solution. If there are more than one previous cases projected on the location then the mean of these case solutions will be used as the solution for the target case. However, if there is no previous case projected on this location, how can we find the corresponding high dimensional solution for this exact location? There are several possible solutions as follows.

First, the prototype vector of the corresponding node of the case solution space ViSOM can be used. Once the solution space ViSOM is trained, each node has its corresponding prototype vector. As the location of the target case in the solution space
ViSOM is known, the corresponding node can be regarded as the winning node for the target case solution, therefore its weight can be regarded as the output suggestion.

Second, KNN with inverse distance weighting can be used as well. However, instead of distances in the problem space, the distances between the target case location and its neighbours in the solution space map are used.

Third, an approach called Kriging (Armstrong 1998) which usually provides better interpolation results can be used. This will be discuss it in detail in Chapter 10 within the context of future work.

8.6 An Example

In this section, a simple example is presented to demonstrate that how our approach can adapt high dimensional cases. In order to simulate the problem we may face, a suitable dataset is needed. First, it must be relatively easy to determine the reason in the event of an unexpected result occurring. Meanwhile, it should have high dimensional problem space as well as high dimensional solution space. Furthermore, each individual dimension of its solution may be related to each other. Suppose we have a case base whose problem space dimension is 6, which are \( x_1, x_2, x_3, x_4, x_5, \) and \( x_6 \). While the solution space dimension is 6, too, which are \( y_1, y_2, y_3, y_4, y_5 \) and \( y_6 \). And their relations are show as follows.

\[
\begin{align*}
y_1 &= x_1 + x_2 + x_3 + x_4 + x_5 + x_6; \\
y_2 &= x_1 \cdot x_2 \cdot x_3 \cdot x_4 \cdot x_5 \cdot x_6; \\
y_3 &= x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2; \\
y_4 &= y_1 + y_2; \\
y_5 &= y_1 \cdot y_2; \\
y_6 &= y_3 / y_1.;
\end{align*}
\]

To simplify it, we randomly choose 50 training cases, in which, the value of \( x_1, x_2, x_3, x_4, x_5, \) and \( x_6 \) range from 0 to 1, and the corresponding value of \( y_1, y_2, y_3, y_4, y_5 \) and \( y_6 \) are calculated based on the formula above. Meanwhile, 30 evaluation cases are also chosen randomly from the same range, their corresponding value of \( y_1, y_2, y_3, y_4, y_5 \) and \( y_6 \) are also calculated to evaluate the predict result. Figure 8.5 to 8.10 shows the problem space ViSOM and solution space ViSOM with different map size.
The evaluation process is repeated 10 times, each time different evaluation cases are chosen, and Figure 8.11 shows the result, the horizontal axis is the number of the
evaluation while the vertical axis is the mean absolute error. We can find ViSOM with size 30X15 has the best result.

![Graph showing evaluation results]

**Fig. 8.11** Result of the evaluation of different map size

Meanwhile, when the size of training dataset increases, the 1-NN will get better result than our approach. The reason is, every target case can easily find a nearest neighbour whose result is good enough without any adaptation. Fig 8.12 shows the result. In which the mean errors of our approach and 1-NN are compared. For calibration, the errors are divided by the norm of their corresponding real result, noted as “error precent”.

It demonstrates our approach is more suitable for case base with limited cases, which is more practical in the real life, because it is not realistic or straightforward to find large number of cases. People may often have difficulty to collect large size of cases available. More importantly, our approach provides an adaptation solution based on the previous cases, while 1-NN is the only other solution which can be applied here because it omits the adaptation process.
Compared to 1-NN, our approach can also acquire result which is outside the training range. Even the interactions between the solution features can be ignored, for approaches like nearest neighbour, the predicted solution will always be limited in the range of the highest and lowest of cases. This ignores the relative position of the target case and its nearest neighbours will sometimes give incorrect results.

Furthermore, nearest neighbour is more susceptible for noisy cases. In nearest neighbour, only those cases which are most similar to the target case will be used to predict solution. However, the case base contains a lot of implicit adaptation knowledge that is potentially useful. Thus we should consider the relative position of the target cases and extract knowledge from other cases as well.

8.7 Conclusions

In this chapter, we discuss how to achieve case adaptation for a case base with high dimensionality solution space. Case adaptation is a very difficult task, especially for high dimensional data with only a limit number of cases. We propose to map the case problem space and the case solution space in two different ViSOMs. Then analyse the mapping relations between these two maps. A simple example is used as
demonstration. Although all the case attributes have numeric values in our example dataset, non-numeric attributes may also be incorporated once they have been reliably converted into numeric form first. Thus our approach has the potential to be applied to other types of datasets as well. In order to evaluate the generality of this approach, other simple direct datasets with high dimensionality solution space will be required. Once suitable datasets are available, additional experiments will be undertaken, analysed and discussed.
Chapter 9

Experimental Design

Scenario representation and the formulation of the COA are difficult and complex tasks. For this reason, there has not been widespread use of CBR in the military decision support area. However, as discussed, CBR is closely aligned with the way military training and planning is conducted. Meanwhile, domain experts are not required to lend their expertise in the form of rules when using CBR, which is a great advantage for many military applications. In this chapter, we will follow the proposed NDM model. A military scenario whose solution part is high dimensional with interactions is used to demonstrate the approach presented in the former part in this thesis.

In Section 9.1, we briefly introduce our methodology. Section 9.2 discusses how the data is collected. Case representations are described in Section 9.3. In Section 9.4, we use SOM to cluster cases and then visualise them in ViSOM. Section 9.5 discusses how we filter the cases and adapt the COA. Evaluation and example results are explained in Section 9.6. Finally in Section 9.7, we summarize the whole chapter.

9.1 Methodology

We begin with a description of our experimental simulation testbed and discuss the data collection process. The use of tactical situations based on actual battlefield scenarios allowed us to capture some of the uncertainty and complexity inherent in combat situations. Then we follow the NDM model proposed in Chapter 2. Cases in our case base are clustered first together with the visualisation. This facilitates better maintenance of case base quality and provides a mental simulation approach for the decision maker. In addition, we use case filtration to discard those low quality cases.
After that, case adaptation is processed for the new case. Finally, the adapted result is used as the COA solution which is implemented in the VR-FORCES synthetic environment (testbed) and subsequently evaluated for success.

The reason why we did not benchmark with other algorithms is that the other algorithms available, considered in the literature review, cannot solve this adaptation problem. For most CBR systems, the solution space is often atomic. However, in some domains such as the military problem we are facing, the solution is characterised by a complex structure and adaptation requires more effort. Meanwhile, interactions between solution parts also have significant impact on the complexity of adaptation. When there are no interactions between solution parts they can be manipulated independently, which simplifies adaptation to a great extent. On the other hand, independent manipulation of highly interacting solution parts may cause ripple effects. For example, resolving a conflict by changing an individual part may result in another part not being applicable under new conditions. This may result in a cycle in the worst case, which prevents a problem from being solved in such a manner. We propose a more sophisticated adaptation method that takes into account interactions between solution parts and deals with them simultaneously.

Different alternative evaluation approaches are discussed because it is difficult and impractical to apply direct benchmark.

9.2 Data Collection

We use MAK VR-FORCES (MAK Technologies 2003) to generate our data source. MAK VR-FORCES is the scenario simulation environment of our research. It is a very powerful and flexible C++ simulation toolkit for generating and executing battlefield scenarios. It has all the simulation features necessary for use as a tactical leadership trainer, threat generator, behaviour model testbed, or Computer Generated Forces (CGF) application. The VR-FORCES application has an intuitive GUI. We can build scenarios by positioning forces, creating routes and waypoints, and assigning tasks or plans with a simple point and click, and interactively add individual entities to a simulation. The entities may include land vehicles, such as T-80 Tank, BTR-80, BMP-2; Air entities such as F-16 Falcon, F/A-18 Hornet; Surface and
subsurface entities such as LCAC (Landing Craft Air-Cushioned). During scenario execution, it interacts with the terrain, detect and engage enemy forces, and calculate damage. A C++ API is also provided to allow us to customise nearly every aspect of the VR-Forces application.

Fig. 9.1 shows a simple breaching scenario. In which, four hostile vehicles (BMP2 1, BMP2 3, T80, and BMP2 2) are arrayed behind a minefield. Three platoons (Blue, Red and White) of four tanks each suppress the hostile vehicles, allowing two engineering vehicles to clear the minefield. The scenario plays out as:

Blue platoon stays in position and fires on hostile forces
Red platoon advances to waypoint red and provides cover for the engineers.
White platoon advances to waypoint white and provides cover for the engineers.
The engineering entities follow behind Red platoon. When Red platoon is at waypoint red and then hostile forces are destroyed, the engineering entities advance on the minefields to clear them.

Fig. 9.1. Breaching Scenario

In order to collect data to populate our case base, we randomly chose the Scaled strength ratio, enemy entities’ locations and friendly troops entities’ locations to generate 300 different scenario variations, which are used as the case description part. Fig 9.3 to Fig 9.8 show some of the examples. Then according to each of the case descriptions, we chose a suitable COA as advised by a military subject matter expert and simulated them in VR-FORCES. We recorded the results, including factors such
as whether the goal was achieved or not, the enemy’s remaining power, the friendly troops’ remaining power, etc. The COA with the highest winning value $W$, which is a weighted function of these factors, will be chosen as the suitable solution:

$$W = w_1 G + w_2 \frac{fr + 1}{(er + 1)f_s}$$

(9.1)

where $w_1, w_2$ are weights defined by domain experts, in our experiment it is assigned 0.8 and 0.2 respectively. $G$ describes whether the goal is achieved or not, $fr$ is the friendly troops’ remain firepower, $f_s$ is the friendly troops’ start firepower, while $er$ is the enemy’s remaining firepower.

Because it is impossible to exhaust all possible enemy plans, with different disposition, their waypoints may also be changed. Thus we need to choose cases which are varied and cover the majority of the variations in the scenario.

In order to evaluate the quality of the suggested COA, it can be compared with the COA suggested by military experts. The other more practical approach is to simulate these generated COAs in VR-FORCES, by testing and recording whether the corresponding COA can help the troops to achieve their mission or not. If the mission is achieved, then the output is positive; otherwise, it is negative. For more accurate evaluation, the cases in the case base are divided into two groups in the ratio of 2:1, which is a common ratio used in artificial intelligence area. One group is for training and the other one is for evaluation. In other words, 100 cases are chosen randomly as the evaluation group, the remaining 200 cases are used for the training group. This process is repeated 10 times.

9.3 Case Representation

Case representation is the most fundamental and important part of CBR. The Case is composed of the case problem part and the case solution part. In our experiment, the case problem part corresponds to the scenario, while the solution part corresponds to the COA.
9.3.1 Scenario Representation

In Chapter 4, the scenario was represented by Mission, Enemy, Terrain, Troops, and Time (METTT). With reference to our simple breaching scenario, the mission, namely to breach the minefield, is fixed. Thus, in order to simplify the scenario representation, it is not necessary to include mission representation here.

Each entity may be represented as a symbolic object, but comparing the effectiveness of the opposing sides is a problem. So instead of representing each troop individually, we combine them together to create an aggregate and use their Scaled Strength Ratio to represent their combat capabilities. According to domain knowledge, T80 is assigned a fire power of 5 while M1A2 is assigned a firepower of 6, and BMP2 is assigned a firepower of 3 (White 2006b).

Estimating an entity’s power is not easy because its power may be affected by other factors, such as the location of the entity (e.g. even if the entity is very powerful, it is useless when it is far away from the target) and the current terrain (e.g. even the most powerful tanks are not a major threat to an enemy on the other side of an impassable river). However, in order to simplify the problem, we currently omit this consideration for this simple scenario. Meanwhile, we may treat Combat Effectiveness as ordinal data instead of symbolic. Such as, “full capability” can be represented by 1 while “Inoperable” can be represented by 0 and “Degraded” can be represented by ½. Then the scaled strength ratio can be defined as:

$$\text{Scaled Strength Ratio} = \frac{\sum_{i=1}^{n} T_i C_i}{\sum_{j=1}^{m} E_j C_j}$$  \hspace{1cm} (9.2)

where $T_i$ is the power assigned to friendly troop $i$ and $C_i$ is its combat effectiveness. $E_j$ is the power assigned to enemy $j$ and $C_j$ is its combat effectiveness.

Each waypoint in VR-FORCES is represented in a coordinate system, such as a geocentric coordinate system; for example waypoint (-2812014, -4332659, 3729407)
as depicted in Fig 9.2. If the number of entities is large then it will increase in the computational cost. How to reduce the required memory and speed up the access of these waypoints becomes an important problem. In VR-FORCES, the whole battle field is covered by a grid. If we assume that the battle field is a chess board and the entities are chess pieces then, instead of using long complicated geocentric locations collected directly from VR-FORCES, we could use the grid information to represent the location of an entity. The entities which are not exactly at the centre of a grid cell will be assigned to the closest grid cell. In this way, each entity location is represented by a corresponding grid cell. Table 9.1 shows an example.

![Table 9.1](image)

**Fig 9.2 An example of entity locations collected from VR-FORCES**

We can store the grid information for all the entities as a matrix. If there is no entity in a grid cell, assign 0 to it. Otherwise, put the entity type in that grid cell. For example, we have entities $a$ and $b$ in the battle field, as shown by Table 9.1.
There are two kinds of enemy entities in the breaching scenario: BMP2 and T80. The numbers of each type of entity may vary in different scenario variations. Suppose we represent BMP2 as 1 and T80 as 2. The enemy force then may be represented by a matrix the elements of which may take values 0, 1 or 2, where 0 means no entities as discussed. We consider this type of representation as categorical data.

Another approach is to store the entity location according to the coordinate system. For example, the entities in Table 9.1 can be represented by:

\[ a = (3, 2) \]
\[ b = (5, 4) \]

This approach is suitable for representing the friendly forces in our experimental scenario. There are four main parts of friendly troops, namely the Blue platoon, the
Red platoon, the White platoon and the Engineers. So we can store their coordinate location. Additionally, because the terrain is fixed, only the X and Y axes values are needed. These are continuous data.

Finally, in order to simplify things even more, the representation of time in our case is omitted as well. For the chosen scenario, the starting disposition implicitly accommodates the consideration of time. In summary, Table 9.2 shows the complete representation of a breaching scenario.

<table>
<thead>
<tr>
<th>Scaled Strength Ratio</th>
<th>Enemy Force Matrix (3 X 9)</th>
<th>Blue Platoon (Xb, Yb)</th>
<th>Red Platoon (Xr, Yr)</th>
<th>White Platoon (Xw, Yw)</th>
<th>Engineers (Xe, Ye)</th>
</tr>
</thead>
</table>

Table 9.2. Scenario Representation

9.3.2 Case solution part

As discussed earlier, COA can be represented by the matrix composed of the entities’ waypoint locations at different time steps during the scenario. Each row corresponds to one entity and each column corresponds to one time step. For the breaching scenario, we use the following four routes to represent the COA, the case solution part: Blue platoon’s route, Red platoon’s route, White platoon’s route and Engineers’ route.

Each route is composed of five waypoints at corresponding time steps, including the start point and the end point, which can be derived from the case description part. In this way, for the solution part we only need to describe another three waypoints, as shown in Table 9.3. In a more sophisticated version of this representation where explicit synchronization of troops is required, one should include the time that each troop should reach their respective waypoints.
Table 9.3. Troops section route representation

<table>
<thead>
<tr>
<th>Route Type</th>
<th>Route Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue Platoon’s Route</td>
<td>((X^b_1, Y^b_1), (X^b_2, Y^b_2), (X^b_3, Y^b_3))</td>
</tr>
<tr>
<td>Red Platoon’s Route</td>
<td>((X^r_1, Y^r_1), (X^r_2, Y^r_2), (X^r_3, Y^r_3))</td>
</tr>
<tr>
<td>White Platoon’s Route</td>
<td>((X^w_1, Y^w_1), (X^w_2, Y^w_2), (X^w_3, Y^w_3))</td>
</tr>
<tr>
<td>Engineers’ Route</td>
<td>((X^e_1, Y^e_1), (X^e_2, Y^e_2), (X^e_3, Y^e_3))</td>
</tr>
</tbody>
</table>

As mentioned, 300 different variations are generated based on the breaching scenario. Fig 9.3 to Fig 9.8 shows some of the examples. In Fig 9.3, the whole White platoon is missing. In Fig 9.4, everything is same as the original scenario except Blue platoon swap the location with White platoon. In Fig 9.5, the friend troops disposition is same as the original one, but the enemy troops all located near the right end of the minefield. In Fig 9.6, the enemy firepower is stronger than that of friend troops. In Fig 9.7, the enemy troops located far away from the minefield. In Fig 9.8, on the friendly side, Blue platoon is missing while on the enemy side, they have extra firepower. In order to demonstrate how to represent the whole case, including the scenario and COA description, the representation of the variation in Fig 9.4 is shown by Table 9.4.

![Fig. 9.3 Breaching exercise variation 1 (White platoon is missing)](image-url)
Fig. 9.4 Breaching exercise variation 2
(White platoon and Blue platoon exchange positions)

Fig. 9.5 Breaching exercise variation 3
(Enemy entities are gathered at the right of minefield)
Fig. 9.6 Breaching exercise variation 4
(Enemy firepower is much stronger than friendly troops)

Fig. 9.7 Breaching exercise variation 5
(Enemy entities located far away from the minefield)
Fig. 9.8 Breaching exercise variation 6
(Blue platoon is missing, while Enemy has extra firepower)

Table 9.4. A case representation example of Fig. 9.4
9.4 Case Clustering

In our experiment, once cases have been collected, they are normalised first, thus each attribute scale such that its variance taken over all the items is unity. They have been linearly initialised in the subspace spanned by the two eigenvectors with greatest eigenvalues computed from the training data. Then cases were clustered by SOM, as shown by Fig. 9.9. The maps were trained in two phases: a rough training with large initial neighbourhood width and a fine tuning phase with small initial neighbourhood width. The commonly used Gaussian neighbourhood function was employed (see Equation 5.15). The neural networks have been created after normalising the variables to avoid any difference in the variables. Different map configurations are executed and the one with best result is selected. The neighbourhood width decreases linearly to 1. The training length of the two phases were 100 and 300 epochs, and the initial learning rates 0.5 and 0.1 respectively, the learning rate decreasing linearly to zero during the training.

![U-matrix SOM of clustering](image)

**Fig 9.9 SOM of clustering**

In the clustering example of Fig 9.9, the 200 training cases are divided into 3 clusters. There are 83 cases in Cluster 1, 67 cases in Cluster 2, and 50 cases in Cluster 3. For
each cluster, two ViSOMs are set up. One is for the problem space, while the other is for the solution space.

For visualisation, the ViSOM is trained using the sequential training algorithm. All maps are linearly initialised and trained in two phases. The neighbourhood width decrease linearly to 1, the neighbourhood function was Gaussian. The training length of the two phases were 50 and 100 epochs, and the initial learning rates 0.5 and 0.05 respectively, the learning rate decreasing linearly to zero during the training. Fig 9.10 to Fig. 9.15 show the visualisation results. The labels shows on the map represent the specific cases which project on the corresponding nodes. Given specific cases, we can observe their relations in the case problem space as well as the case solution space. An advantage of visualisation after the case the cluster is, after clustering, only cases similar enough will be input to the BP network for training; hence, enabling a more accurate result to be achieved.
9.5 Case Adaptation

Referring to Chapter 7, the correlations between the problems and solutions of case pairs are calculated. In our experiment, with a neighbourhood size of 10, correlation of most cases are larger than 0.4, low quality cases are discarded, and the rest are used for adaptation. For example, for Cluster 1, there are 72 cases left; for Cluster 2, there are 53 cases left; for Cluster 3, there are 41 cases left.

For adaptation, 3-layer BP network was used in our experiment. The input vector has 2 elements. There are 5 neurons in the hidden layer while 2 neurons in the output layer. The transfer function for both layers is tan-sigmoid. The training function is ‘trainlm’, the Levenberg-Marquardt algorithm (Levenberg 1944), a very fast training method which is suitable for small networks. Table 9.5 shows an example of the adapted result of a target case. The numbers of the solutions are rounded to match the grid cell location.
Table 9.5. An example of the adapted COA result

9.6 Evaluation

There is a paucity of military CBR decision support systems available. Even though similar projects exist, they are normally based on different scenario data, different case organization, different representation etc, thus it is difficult and impractical to apply benchmarking for our approach with similar ones. Meanwhile, for adaptation, similar approach cannot solve high dimensional solution with interactions. Normally nearest neighbour will be applied in this condition to skip the adaptation. Therefore, alternative evaluation approaches are required.

One direct approach is to depend on domain experts, such as military Subject Matter Experts (SME). We can utilise the Turing Test on the evaluation cases, and compare the adapted COA with the suggestions of SMEs. A more practical approach is to simulate the generated COAs in VR-FORCES, and find out whether the corresponding COAs can help the friendly troops to achieve the goal or not. If yes, how much firepower is used and left can be used to assess how good the COA is. The feedback can be fed back into the system to increase its learning ability for the future.

First, we try to find out whether the suggested COA can help to achieve the goal by assigning the COA to the corresponding scenario and simulate it in VR-FORCES. After clustering, cases are divided into 3 groups which matches the opinion of SME (White 2006a). According to SME, the scenario can be classified according to their Scaled Strength Ratio (SSR). For group1, when SSR is larger than 2.5, the friendly troops have the firepower advantage, thus they can implement a hasty attack. For group2, when SSR is larger than 1.5 but less than 2.5, the friendly troops do not have impressive firepower advantage, thus they should withdraw to the safe distance. The
firing range of M1A2 is larger than that of a T80. Even if the T80 withdraws to a safe distance, the friendly troops can still fire and destroy enemy troops without suffering any damage from the enemy. After most enemy troops are destroyed, friendly tanks can move forward and cover the engineers to clear the minefield. For group3, if the SSR is less than 1.5, the friendly troop should withdraw instead of attacking because of the firepower disadvantage. We randomly choose 100 cases as evaluation cases, and record the number of cases assigned to the right groups, this process is executed 10 times. Fig 9.16 shows the result, the average number is 88.2.

Second, in each group, cases are chosen and their corresponding COAs are input to VR-FORCES to test whether the COA can achieve the goal. For cases in group1 and group2, all the COA can successful achieve the goal, sooner or later, the tanks will cover engineer to breach and clear minefield. Some examples are shown by Fig 9.17 to Fig 9.19. For cases in group3, all the friendly troops will withdraw to safe distance, which also matches the recommendation of the SME.

![Fig 9.16 Number of cases in the right groups](image)

*(Horizontal axis is the No. of evaluation while Vertical axis is the number of cases in the right group)*
Fig 9.17 Test COA in VR-FORCES (SSR>2.5)

Fig 9.18 Test COA in VR-FORCES (SSR>2.5)
Third, we calculate winning value $W$ of COA of each case in the group1 and group2 according to equation (9.1). The remaining firepower of enemy and friendly troops are both recorded. $w_1$ is set to 0.8 while $w_2$ is set to 0.2 based on domain knowledge. Fig 9.20 shows the average $W$ of all the COA of cases in group1 and group2, the evaluation is processed 10 times.

Fourth, in order to further evaluate the quality of the suggested COAs, we compare them with the original COA suggested by domain experts in detail. Generally, the
output COAs of cases in the evaluation group are often not exactly the same as the corresponding COAs stored in the case base. This is because for any given scenario, there could be more than one successful COAs. Thus when COAs are compared, if all the waypoints in the suggested COA are reasonable (difference is within ±2), we consider this COA is good. If only one or two waypoints deviate (difference >2), we think the COA is satisfactory. Otherwise the COA is considered as an error. An example is shown as follows:

Real COA  15,11, 16,9, .......15,14
Good COA  15,12, 15,10, .......15,13
Satisfactory COA  16,7, 10,10, .......15,14
Error  16,7, 9,9, ......... 7,11

Table 9.6 shows the evaluation result. In which, the evaluation is processed 10 times, represented by T1, T2.... T10. The number in the table shows how many of COAs are good, satisfactory or in error.

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
<th>T5</th>
<th>T6</th>
<th>T7</th>
<th>T8</th>
<th>T9</th>
<th>T10</th>
</tr>
</thead>
<tbody>
<tr>
<td>GOOD</td>
<td>73</td>
<td>72</td>
<td>77</td>
<td>79</td>
<td>75</td>
<td>73</td>
<td>83</td>
<td>79</td>
<td>75</td>
<td>84</td>
</tr>
<tr>
<td>SATISFACTORY</td>
<td>11</td>
<td>13</td>
<td>14</td>
<td>14</td>
<td>13</td>
<td>15</td>
<td>9</td>
<td>8</td>
<td>18</td>
<td>11</td>
</tr>
<tr>
<td>ERROR</td>
<td>16</td>
<td>15</td>
<td>9</td>
<td>7</td>
<td>12</td>
<td>12</td>
<td>8</td>
<td>13</td>
<td>7</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 9.6. Evaluation of suggested COAs

Finally, we compare the results of all the approaches proposed for adaptation in Chapter 8. As mentioned, in our evaluation, the cases are divided into two groups in the ratio of 2:1, one for training and the other one for evaluation. Therefore, 200 cases were chose randomly as the training group while the rest 100 cases are used for evaluation. This process is repeated 10 times. The results are shown in Table 9.7. The Mean Error(ME) is the mean Euclidean difference between the predicted COA and the corresponding real COA of these 100 cases. In order to calibrate them, these differences are divided by the norm of their corresponding real COA, their average is then recorded as Mean Percent Error (MPE).
<table>
<thead>
<tr>
<th>Method</th>
<th>ME</th>
<th>MPE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LocKNN</strong> (with K=1)</td>
<td>0.439</td>
<td>6.4142</td>
</tr>
<tr>
<td>K=3</td>
<td>0.435</td>
<td>6.3173</td>
</tr>
<tr>
<td>K=5</td>
<td>0.445</td>
<td>6.573</td>
</tr>
<tr>
<td><strong>LocProto</strong> (with 10 X 10)</td>
<td>0.469</td>
<td>6.772</td>
</tr>
<tr>
<td>20 X 20</td>
<td>0.458</td>
<td>6.204</td>
</tr>
<tr>
<td>10 X 20</td>
<td>0.473</td>
<td>6.808</td>
</tr>
<tr>
<td>30 X 30</td>
<td>0.466</td>
<td>6.647</td>
</tr>
<tr>
<td><strong>DifKNN</strong> (with K=1)</td>
<td>0.432</td>
<td>6.128</td>
</tr>
<tr>
<td>K=3</td>
<td>0.421</td>
<td>5.946</td>
</tr>
<tr>
<td>K=5</td>
<td>0.430</td>
<td>6.437</td>
</tr>
<tr>
<td><strong>DifProto</strong> (with 10 X 10)</td>
<td>0.412</td>
<td>6.119</td>
</tr>
<tr>
<td>20 X 20</td>
<td>0.399</td>
<td>5.874</td>
</tr>
<tr>
<td>10 X 20</td>
<td>0.408</td>
<td>6.107</td>
</tr>
<tr>
<td>30 X 30</td>
<td>0.393</td>
<td>5.475</td>
</tr>
</tbody>
</table>

**Table 9.7.** Evaluation result of different approaches

In Table 9.7, LocKNN shows the ME & MPE results obtained using the location of the map (with different size of map) to train the BP, with KNN (with different K values) to acquire the solution.

LocProto shows the results obtained using the location the map (with different size of map) to train the BP, with the map prototype vector for the solution.

DifKNN shows the results obtained using the location's difference (with different size of map) to train the BP, with KNN (with different K values) to acquire the solution.

DifProto shows the results obtained using the location’s difference (with different size of map) to train the BP, with the map prototype vector for the solution.
Anova: Single Factor

<table>
<thead>
<tr>
<th>Groups</th>
<th>Count</th>
<th>Sum</th>
<th>Average</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loc k=1</td>
<td>10</td>
<td>4.39</td>
<td>0.439</td>
<td>8.89E-05</td>
</tr>
<tr>
<td>Loc k=3</td>
<td>10</td>
<td>4.35</td>
<td>0.435</td>
<td>2.22E-05</td>
</tr>
<tr>
<td>Loc k=5</td>
<td>10</td>
<td>4.45</td>
<td>0.445</td>
<td>1.34E-08</td>
</tr>
<tr>
<td>Loc 10X10</td>
<td>10</td>
<td>4.69</td>
<td>0.469</td>
<td>2.25E-05</td>
</tr>
<tr>
<td>Loc 20X20</td>
<td>10</td>
<td>4.58</td>
<td>0.458</td>
<td>4.45E-05</td>
</tr>
<tr>
<td>Loc 10X20</td>
<td>10</td>
<td>4.73</td>
<td>0.473</td>
<td>1.38E-08</td>
</tr>
<tr>
<td>Loc 30X30</td>
<td>10</td>
<td>4.66</td>
<td>0.466</td>
<td>3.53E-08</td>
</tr>
<tr>
<td>Dif k=1</td>
<td>10</td>
<td>4.32</td>
<td>0.432</td>
<td>4.44E-05</td>
</tr>
<tr>
<td>Dif k=3</td>
<td>10</td>
<td>4.21</td>
<td>0.421</td>
<td>2.27E-05</td>
</tr>
<tr>
<td>Dif k=5</td>
<td>10</td>
<td>4.3</td>
<td>0.43</td>
<td>1.08E-06</td>
</tr>
<tr>
<td>Dif 10x10</td>
<td>10</td>
<td>4.12</td>
<td>0.412</td>
<td>2.29E-05</td>
</tr>
<tr>
<td>Dif 20X20</td>
<td>10</td>
<td>3.99</td>
<td>0.399</td>
<td>3.21E-05</td>
</tr>
<tr>
<td>Dif 10x20</td>
<td>10</td>
<td>4.08</td>
<td>0.408</td>
<td>8.07E-05</td>
</tr>
<tr>
<td>Dif 30x30</td>
<td>10</td>
<td>3.93</td>
<td>0.393</td>
<td>4.89E-08</td>
</tr>
</tbody>
</table>

**ANOVA**

<table>
<thead>
<tr>
<th>Source of Variation</th>
<th>SS</th>
<th>df</th>
<th>MS</th>
<th>F</th>
<th>P-value</th>
<th>F crit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Between Groups</td>
<td>0.087469</td>
<td>13</td>
<td>0.006728</td>
<td>246.4893</td>
<td>7.45E-83</td>
<td>1.798584</td>
</tr>
<tr>
<td>Within Groups</td>
<td>0.003439</td>
<td>126</td>
<td>2.73E-05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>0.090908</td>
<td>139</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 9.8** ANOVA result of different approaches

Table 9.8 shows the ANOVA result of different approaches discussed, in which, $P$-value $<< 0.05$ and $F >> F_{crit}$, thus differences between the results in Table 9.7 are significant.

By inspecting Table 9.7, it can be seen that using the location's difference to train the BP and map prototype achieves the best result. Using the location of map to train the BP and KNN for the solution has the worst result. This is because the location's difference of map contains more information than pure location itself. Indeed, using difference also increases the size of training sample. Furthermore, using the map prototype vector for the solution one must consider the whole solution space, not just the nearest neighbours.
9.7 Conclusions

The research and development of military applications is a very demanding area but CBR is very suitable methodological approach to mimic the naturalistic decision making process of human commanders. Therefore, using CBR to suggest possible COA for a military scenario is reasonable assertion. From our experiment, the initial suitability of CBR, as a proof of concept, has been investigated within the context of an established simulation environment VR-FORCES. The results and findings of our experiments have demonstrated that the algorithms developed in this investigation for CBR have the potential to assist human commanders in their decision making under military circumstances when the NDM paradigm is applicable.

However it is recognised that this research is an initial ‘proof-of-concept’ endeavour and there is the need for further work and improvement, which will be discussed in Chapter 10.
Chapter 10

Conclusions and Future Work

In this thesis, we have discussed how the methods developed during this research apply CBR in order to (i) achieve and support a commander's NDM, and (ii) assist the commander in his/her choice of a suitable COA for a given scenario. Previous attempts have mostly focussed on rule-based models within the context of classical decision theory. However, in many decision situations where the commander is operating in fast-moving, fluid situations and under time pressure, the decision process of military commanders cannot be easily represented by the classical decision model. They employ NDM instead. We have proposed to use CBR to achieve a NDM model to capture the decision process of military commanders. Meanwhile, a CBR cycle can gradually improve its competence over time as new experiences are gained. This is convenient for military domains for which the existing theoretical knowledge is often difficult to capture, maintain or complete. Additionally, humans can more readily understand a CBR system's reasoning and recommendations; hence they may be more easily convinced of the validity of the solutions provide by the system. To complete this thesis, Section 10.1 summaries and discuss the contributions of our research. Section 10.2 proposes some further research for improvement. Closing remarks are given in Section 10.3.

10.1 Contributions

Our research is significant in which it produced a number of contributions. These are discussed below.

In Chapter 2, we proposed a NDM model to simulate the decisions making process of domain experts by applying CBR. NDM explains how people use their experience to
achieve decisions without comparing the strengths and weakness of the COAs. CBR mimics how people solve problems based on adapting their best and most similar experience in order to deal with the problem situation at hand. Thus implementing NDM with CBR has great advantages. However pure CBR cannot cover the whole process of NDM. In order to solve this, case clustering is applied for pattern recognition of NDM. There are three different situations after the clustering, which correspond to the different conditions that decision makers may face in the real life. Meanwhile, case visualisation is applied for mental simulation of NDM. With the assistance of visualisation, decision makers can decide whether the situation is familiar or not, whether the target solution is desirable or not, etc. Furthermore, cases are filtered and then adapted to generate new COA for similar but unknown cases. As discussed, RPD is not always the best model for all decision environments. Our model considers conditions when an analytical approach is needed. Additionally, our model can solve the problem that the RPD model has. Namely, RPD lacks an empirical base and more importantly it does not take into account the generation of new COA. In this thesis we apply our methods and algorithms to the CBR cycle, which allows us to model the process of NDM. The result is more understandable for decision maker.

In Chapter 4, we propose a reasonable and practical representation method of military scenario and COA. Military scenario can be complicated and the formulation may contain different types of attributes. COA is normally composed of narrative statement and graph sketch description, which make it extremely difficult to formulate the problem situation its concomitant scenario. Scenarios were formulated according to their military features METT-T first, and then the COA were represented by entities' route points and way points together with the corresponding time for each entity to reach each of the waypoints. This simple representation is relatively easy to understand and combines successfully with our military simulation tools.

In Chapter 6, a proposal of suitable visualisation approach for Case Base is discussed. Case visualisation is discussed in a very limited manner in the CBR literature. It is applied in our model for mental simulation of NDM. It can offer the intuitive insight of the case base and uncover patterns or trends in the case base. Additionally, it can indicate the two most important relationships referred to in the body of knowledge by the CBR community: problem-solution regularity and problem-distribution regularity.
Thus it can be applied to monitor automatically the appropriateness of the case base with respect to the current problem. We have applied SOM and ViSOM in a novel way to visualise the case base from different aspects. This convenient approach provide multiple visualisation, has great advantages than the traditional visualisation approaches. It can greatly assist researchers in the CBR area.

In Chapter 7, a novel assessment approach for the quality of cases by correlation is proposed. There is an assertion in the general CBR literature that the more a case follows “similar problems have similar solutions”, the better quality this case is. Only cases with better quality will help the case adaptation in the latter part of the CBR cycle. Thus, filtering low quality cases can assist to achieve better adaptation results. In our approach, the larger the correlation of problem space distances between a case and other cases and their solution space distance, the better it follows the problem-solution regularity.

A proposal to measure ‘Problem-Solution Regularity’ of case base was presented in Chapter 7. ‘Problem-Solution Regularity’ is used to represent how well the similarities between problems approximate the similarity between corresponding solutions in practice. There are many related discussion in CBR community, but how to calculate the ‘Problem-Solution Regularity’ of a given case base is still unknown. In our research, both the case problem space and case solution space have been already visualised and transformed into two-dimensional space first. In addition, our visualisation approach can ensure the topological relations in original high-dimensional space are kept as good as possible in the two dimensional space. As a result of this, the ‘Problem-Solution Regularity’ can be considered as the similarity between the case problem map and case solution map. Although the proposed method is straight-forward, it nevertheless can be used as an effective benchmark to evaluate the quality of a case base.

In Chapter 8, we introduced an adaptation method for case bases with high dimensional solution space. Case adaptation is a very difficult task, especially for high dimensional data with only a limit number of cases. More importantly, in real life, there are always interaction between those features in the case base, thus it is impossible to treat every dimension separately. Currently there are no available
approaches in CBR area to solve this problem. When facing similar a problem, most researchers and developers will skip this adaptation and choose the nearest neighbour solution, which does not take into consideration any adaptation of tweaking to the solution offered as would happen in NDM. Our research provides a novel approach to this problem. Specifically, the case problem space and the solution spaces are projected in two different unsupervised maps. The mapping relations between these two maps can be used to find to target solution. We achieve this task by a second neural network. This idea is very similar to natural decision making process of matching solutions according to the problem descriptions, hence it can greatly help researchers when they facing similar problems.

In Chapter 9, a military scenario is employed as a case study. Then we follow the NDM model proposed in Chapter 2. Cases in our case base are clustered first and then visualised. Additionally, case filtration is employed to discard low quality cases. After that, case adaptation occurs for the new case. Finally, the adapted result is used as the COA solution which is implemented in VR-FORCES. Different evaluation approaches are used. The suggested COAs are simulated in VR-FORCES to find out whether they can help achieve the corresponding goals or not. The winning value $W$ of COA of each case is calculated too. Meanwhile, suggested COAs are also compared with the original COA recommended by domain experts in detail. The findings demonstrate that our NDM model can imitate the decision process of experienced military human commanders, and identify the fundamental cognitive process, provide pattern recognition and mental simulation and assist human commanders when they are facing a military scenario, as proof of concept. Instead of stereotypical, predictable and doctrine limited recommendation, it can provide a more variable, flexible and adaptable solution. The British military has been conduct experiments and demonstrated the validity of NDM through various researches about military C2 decision making(Blendell et al. 2008; Pascual et al. 2008). Our research not only contributes some interesting novel ideas for CBR community, it also provides an experiment of framework to bring CBR into military NDM. The sheer complexity and dimensionality of the battlefield rarely allow commanders to make decisions using rational, analytical methods in a timely, efficient and effective manner. Applying CBR to achieve NDM can great help us to follow the fundamental cognitive
process and achieve NDM more efficiently and successfully. In addition, our research has opened the door and formed the basis for further research in many areas.

For example, for military training, as pointed out in (Pratt 2001), CBR is closely aligned with the way military training and planning is conducted. The trainee is exposed to a wide range of scenarios with the goal to provide them with some experience. So when the real event happens they will be able to draw on their previous simulated experiences to rapidly come up with the correct COA. Similarly, because different military entities have different power and cost, the approach can be used to suggest commanders for reasonable and effective entities with respect to composition and disposition.

Our approach can also be applied to other application in military area. For example, the prediction of COA of the target case is very difficult. Although we propose a simple solution in this thesis, we are aware the military COA can be far more complicated to formulate by simple mathematic models. Therefore, if the COAs for the target case are already known, we can use the similar approach to predict the victorious probability of the given COA for the target case, thus users can choose the one with the highest victorious probability from a series of COAs.

Our approach can also be incorporated with military simulation platforms, such as VR-FORCES, and its performance can be measured against the simulation tools. Once a suitable case base is set up, our approach can deal with many different scenarios and provide reasonable COAs. More importantly, it has the potential for learning, which can help to improve the decision quality gradually. Therefore, it provides a truly novel approach to represent and maintain the domain knowledge, has great advantages when compared to normal rule-based systems. Moreover, our project has delivered an adaptation approach for CBR. Namely, if no suitable solutions can be found in the original case base, our methodology and methods can be applied to generate the COA, evaluate the recommendation; and, like the human commander gradually learn to achieve the best result.
10.2 Future work

Not only related to military NDM, a very complicated and challenging area, this thesis has addressed many important topics about CBR, including case clustering, visualisation, quality assessment and adaptation etc. In this section, we discuss how to improve our work in the future.

10.2.1 RPD model

![Recognition – Primed Decision Model (Klein 1998a)]

Klein (Klein 1997) suggests that decision making in real-world setting is the result of three processes: situation assessment, formulation of a plausible solution, and mental simulation. Situation assessment is used to generate a plausible course of action while mental simulation is used to evaluate that course of action. It focuses on the task of situation assessment in unfamiliar circumstances in order to formulate a potential solution based on previous experience, simulate the proposed solution, test if it meets the needs of the current problem and adjust it if needed or reject it if it will not do the job.
Fig. 10.1 shows a complete RPD model, it is more complicated then those models we discussed in Chapter 2. When situational recognition occurs, there will be four by-products: cues, goals, actions and expectancies. It fuses the two processes of pattern recognition and mental simulation to optimise decision-making. Through pattern recognition, decision makers recognize the situation as typical and familiar and proceed to take action. If there is no direct recognition, mental simulation is used to evaluate the COA.

In this thesis, we propose a model similar to RPD model and use case clustering to simulate the pattern recognition process, while case visualisation to simulate the mental simulation part. However, it still needs further detailed discussion. In the meantime, while the situation is recognized, how to represent four by-products: cues, goals, actions and expectancies in our model also needs consideration. Furthermore, due to limited time, we only consider to implement the first situation after case clustering in this thesis. To accomplish the other two more complicated situations obviously needs to combine other approaches.

10.2.2 Other possible approaches for adaptation

In this thesis, we propose a novel approach for high dimensionality case space adaptation. A BP network is set up with two ViSOM to adapt previous cases for COA. There are other potential approaches maybe suitable for the current problem as well. Here we shall discuss two of them.

10.2.2.1 Kriging

Kriging is the best linear unbiased estimator. The estimation of an unsampled location is given as the weighted sum of the circumjacent observed points. It is unbiased since it tries to have the mean residual or error equal to 0. It is best because it aims to minimize the variance of the errors. It is a powerful spatial interpolation technique and widely used throughout the earth and environment sciences. It was developed in 1950 by D.G. Krige, a South African mining engineer for predicting gold ore concentrations in mining deposits. Professor G. Matheron improved it and the new method was called kriging. There are many variations of Kriging methods. The most
important ones are ordinary kriging, simple kriging, co-kriging, indicator kriging and universal kriging. The most common versions are simple kriging and ordinary kriging (Isaaks & Srivastava 1989).

Kriging is normally suitable for 2 or 3 dimensional data, so it can not be applied directly on the original scenario dataset. However, after the two ViSOM are set up, our case solution space map is only two dimensional now. In general, in order to solve the problem we must model the covariance matrix of the random variable that represents the data. This is done by modelling the “variogram” of the data. The variogram $\gamma(h)$ is defined as

$$\gamma(h) = 0.5E[(Z(x + h) - Z(x))^2]$$  \hspace{1cm} (10.1)

Where $x$ and $x+h$ are points in the n ($n<4$) dimension space, $Z$ are the corresponding data values and $E[\cdot]$ is the expectation operator.

For a fixed distance $h$, the variogram indicates how different the values are expected to be. In statistics it is common to assume that the variable is stationary, i.e. its distribution is invariant under translation. Thus a stationary random function is homogeneous and self-repeating in space. For any increment $h$, the distribution of $Z(x1), Z(x2), \ldots, Z(xk)$ is same as that of $Z(x1+h), Z(x2+h), \ldots, Z(xk+h)$. Usually only the first two moments, the mean and the covariance are required to be constant. This is called second order stationary. In geostatic, even intrinsic variable whereas it is clearly nonstationary over long distances, it often can be considered as locally stationary (Armstrong 1998).

In earth and environment science, Kriging is normally used to predict single dimensional unknown variable based on the 2 dimensional data. In our case, once the BP network is trained and new target case is input to the network, the location of the target case on the 2 dimensional ViSOM is known, how can we acquire the high dimensional solution projected on this location? The solution is to use the 2 dimensional map location to predict the value of each individual dimension by Kringing, then combine them together. In order to calculate an experimental variogram of a solution space with $m$ dimensions, for each dimension, $S_1, S_2, \ldots, S_m$, a
variogram will be created as a function of the distance in the case solution space. Such as:

\[
\gamma_1(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} (S_{1\alpha}^i - S_{1\beta}^i)^2
\]  

(10.2)

Where \( \gamma_1(h) \) is the variogram of first dimension of the solution.

\( N(h) \) is the total number of pairs of cases in the solution space map which are separated by a distance \( h \).

\( S_{1\alpha}^i \) is the first parameter of solution of case \( \alpha \)

\( S_{1\beta}^i \) is the first parameter of solution of case \( \beta \).

Case \( \alpha \) is \( h \) away case \( \beta \) in the case solution space.

Similarly, for the second dimension of case solution, we have

\[
\gamma_2(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} (S_{2\alpha}^i - S_{2\beta}^i)^2
\]  

(10.3)

And so on.

Actually, instead only used for the solution space map, Kriging can be used directly on the case problem space map as well. The same variogram formula can be used. The only difference is, this time \( h \) is the distance in the problem space map, and \( N(h) \) is the total number of pairs of cases in the problem space map which are separated by a distance \( h \).

Some experiments about Kriging have been done in our research. The following figures show the process by a kriging tool called Easy Kriging. Fig. 10.2 shows how to calculate the Variogram.
Fig 10.2  Variogram calculation with LSQ Fit

Fig 10.3 shows the kriging map of the whole case problem space. Fig 10.4 and Fig 10.5 show the validation with Q1 and Q2 method. In Fig 10.4, the value is out of the optimum range, thus the variogram needs to be adjusted.

Fig 10.3  Kriging map of the whole case problem space
Fig 10.4 Kriging result validated with Q1 (too large, need recreate the variogram)

Fig 10.5 Kriging result validated with Q2

However, the experiment result of Kriging adaptation is not as good as expected. How to configure and apply Kriging to adapt COA, combine it with our current approach to achieve better result still needs more consideration.

10.2.2.2 Hebbian Network

Different parts of the brain use different kinds of neurons and have different connection strategies depending on the function required. When humans are born,
there are a large number of neurons in our brains but majority of them are unconnected. A neuron in the brain consists of a cell body to process signals, dendrites to receive electrical signals, an axon to send outgoing electrical signal, while synapses to send chemical signals to other neurons. The human brain has approximately $10^{11}$ neurons and $10^{14}$ synapses, but the number of synapses is really infinite. Once a connection between neurons is established, it is strengthened each time those neurons are activated, thereby increasing the association between the two. This process is known as "Hebbian learning" (Hebb 1949). In other words, when two neurons on each side of a synapse are activated simultaneously then the synaptic weights are reinforced, but at the same time the synaptic weights could be weakened when two neurons are uncorrelated. It is based on the modification of the connections between neurons, or more specifically is an unsupervised training algorithm that increases the synaptic strength between two neurons that are active at the same time.

This method was one of the first attempts to simulate the process of learning of the biological brain and has been used extensively over the years.

There are some researches has been done about Hebbian network and SOM. Abidi and Ahmad connected a Hebbian network between two pre-trained SOM in a simulation of the development of mappings between concepts and words. The weakness of this approach is the use of pre-trained SOM (Abidi & Ahmad 1997), then applying extra training to the Hebbian network, as the Hebbian network learns only the final condition of the other networks.

In (Ahmad et al. 2003), a modular neural network-based system is presented where the component networks learn together to classify a set of complex input patterns. Each pattern comprises two vectors: a primary vector and a collateral vector. Examples of such patterns include annotated images and magnitudes with articulated numerical labels. The primary and collateral vectors are mapped on a SOM, with the combiner based on a variant of Hebbian networks. Certain features of SOM's allow for one to many mappings between the primary and collateral maps, hence establishing a broader association between the two vectors when compared with the association due to synchrony in a conventional Hebbian association.
The system contains two SOMs, which interlink through a Hebbian network that learns to combine these two networks in a dynamic fashion. Each SOM node is associated with the other SOM's output via the Hebbian network connection. The learning performed within the Hebbian network that interconnects an SOM node pair is bidirectional. This means that the weights change is affected by the winning node activation given from both SOMs. The training takes place in a parallel way, where all the networks, including the Hebbian interconnections, learn synchronously from the beginning of the training to the end. In other words, the project relationship between case problem and case solution can be constructed as a Hebbian link, which evolves over time. This association is bi-directional.

As we proposed, two ViSOM can be set up, one for the case problem space, the other for the case solution space. Our target is to find the relationship between these two maps, like human brain. Our current approach is to use a BP network, and use the location's difference to train the network. Because of the SOM does not proportionally represent the distance between nodes, that is why ViSOM is used here. However, if we can use Hebbian network to find the relation between two SOM, the process are more direct and easy to explain, possibly better result can be achieved. This approach is very interesting and more promising, and it mimics the learning of biology brain and combine the projection between the case problem and case solution space, thus it could simulate a person's mental decision process better, the result maybe more easily understandable for human as well.

10.2.3 Case Representation

As discussed, COA and scenario are very complicated and difficult to formulate. Although we propose a simple solution in this thesis, it is obviously not easy to thoroughly transform COA described by the human language and graph sketch to a mathematical model. Meanwhile, for the current simple experimental scenario, our proposal, as a proof of concept, is sufficient. However, different scenario will have different attributes, how to represent METT-T in more detail still need future discussion. Domain knowledge should be combined when consider future formulation. In fact, the difficulty in determining the current situational awareness, and
formulating of COA, are the main reasons to prevent the wide spread use of CBR in military decision support area.

10.2.4 Problem-Solution Regularity and Case quality assessment

In this thesis, we propose an interesting idea to measure the ‘Problem-Solution Regularity’. It can be used as a benchmark to evaluate the quality of a given case base and help us to maintain a better and efficient case base. It is very different from other approaches in CBR area. How to extend this idea and evaluate it still requires further investigation and experiments. Meanwhile, correlation is proposed to apply to filter the poor quality cases at first, to ensure the better prediction result of case adaptation. Nevertheless, we did not give a detail threshold of filtration to apply for general case base. Is there any relationship between this threshold and the size of case base? How to define this threshold? There are still many work need to be done.

10.2.5 Case Visualisation

In this thesis, we propose to use visualisation to mimic the mental simulation part of NDM. ViSOM is used in our approach. Not only it is convenient, but also it can provide many different kinds of visualisation to demonstrate different aspect of the case base. Both the whole case base and the detail attribute can be shown. It has great advantages than normal visualisation approaches in CBR area. However, how to properly visualise the ‘similarity assumption’ still needs more research. Meanwhile, when two maps are shown, more direct demonstrations are needed to show the suitability of applying CBR.

For example, different colour can be applied to represent different cluster of cases, the same colour will be assigned to the corresponding case solution in the solution map. In this way, the similarity between two maps can be easily recognized, because human eyes are highly sensitive to colours. In order to achieve better visualisation result, some special training methods can be applied. For example, shown in Fig 10.6, the right map is for the case problem space, the left one is for the case solution space. Because the ‘red’ cluster, which may be the cluster we concern about, is on the left top of the problem map, we could use the corresponding solution of those cases in ‘red’ area as the initial weight to train the solution space map. In this way, the ‘red’
area in the solution map will possibly locate in the left top area of the solution map as well. And so on. Of course, all these colour area can not match perfect on their corresponding location and shape, unless the data can 100% follow “similar problems will have similar solutions”.

![Image](image_url)

**Fig 10.6 An example about colour matching**

Other future work would be to transfer the visualisation into 3D. It will give a huge advantage when compared to conventional 2D visualisation. Since position in 3D is much more flexible, powerful indicator than that of 2D. For SOM, additional benefits may result from the fact that SOM can maintain the original information of high dimension space better. Meanwhile, we can also provide multiple viewpoints for the 3D visualisation. For example, the result can be rotated, zoomed, which will greatly help user to investigate the detail of the case base.

### 10.2.6 Map size and shape

In the CBR adaptation part, in order to validate our approach, we use a simple mathematic problem to demonstrate it. Its generalisability still needs further consideration. Meanwhile, we notice map size of ViSOM can affect the result. Does a big map mean a better result? Theoretically, the bigger the map usually means better training, because the training data can be spread into a long area, providing more detailed analysis. However, this is costly because of the longer training time, and based on experiments sometimes the results are not necessarily better. Any kind of shape of map has its advantage? Is there any relationship between them? If the answer is affirmative then how does one choose a suitable map size and shape to achieve the
optimum result? All these problems require further research if one is to obtain an informative answer.

10.2.7 System Construction

All the approaches discussed in this thesis can be integrated with VR-FORCES seamless to achieve the CBR cycle. In this way, given any scenario, the system will automatically collect cases, execute, then monitor and evaluate the result in the testbed as well. If the solution is successful, then store the case COA. Otherwise, discard it. Gradually, the system will learn automatically to achieve increasingly better results with time. The fundamental theory of domain knowledge does not have to be quantified or understood entirely. Even with only a few cases are available, it can accumulate successful cases to build its knowledge base incrementally. The system construct framework is shown in Fig. 10.7. The whole system may include a computer network, with different machines in charge of different tasks. The system can combine with military simulation tools, generate and evaluate COA automatically. It has the potential to significantly help the military accumulate reasonable domain knowledge and become an important component of a military simulation tool.
10.2.8 Application in Other Areas

Many approaches discussed in this thesis can also be applied in other areas too, NDM models like RPD are used in many decision making area, not limit in military domain. Meanwhile, our current approach did not require any prior domain knowledge, which is an advantage for many area whose domain knowledge is too complicated to represent or impossible to acquire. However, supported by domain experience, even partly, can obviously help us achieve better result. For example, for case representation, detail structure or hierarchy can be constructed based on domain knowledge, different weight can be assigned to the features as well. For COA, better formulation combined with expert knowledge will greatly help increase the quality of case base. Meanwhile, rules or constraints can also be applied to improve case clustering, retrieval, adaptation result, and dramatically reduce the difficulty of adaptation.
Furthermore, those technologies we discussed in this thesis can be applied in any CBR systems, not necessarily limited to NDM. Approaches as case visualisation, the calculation of ‘problem-solution regulation’, case filtration and case adaptation for high dimensional data will help future researches in the CBR community.

10.3 Closing Remarks

This thesis discusses how to implement NDM, the naturalistic approach to mimicking the decision making process of human commanders, and has been shown to be a very interesting and challenging problem in what is considered to be a very complicated area.

NDM considers the time demands of the military situation, the lack of adequate information, and the present uncertainty of military scenario. The sheer complexity and dimensionality of the battlefield rarely allow commanders to make decisions using rational, analytical methods in a timely, efficient and effective manner. The NDM theory represents an understanding of military commanders’ decision making because their experience and training, can make important personal contributions to the quality of their decisions. Moreover, NDM is supported by a body of cognitive research on behaviour in command and control situations.

We have discussed how to combine CBR with different technologies in order to achieve the main components of the NDM model. In this thesis, different approaches are presented to improve case quality, to guide users to construct the case base, generate COA for a similar scenario. More importantly, our approach simulates the decision making process of human commanders. It finds the projection of the problem into the solution space. Many of the examples and discussions in this research employ relatively uncomplicated scenarios; nonetheless, the CBR approach to NDM, developed as a proof of concept herein has delivered promising results. As previously noted existing decision support systems place an excessive reliance on doctrine and consequently fail to exhibit sufficient flexibility or adaptability. Features of the NDM model enable development of the system which does not suffer from these shortcomings. Instead of doctrine, the recognition of situations, as well as knowledge derived from this recognition, may be based on SMEs past experiences. A
NDM implementation with CBR can continually learn and adapt as experiences are accumulated. Thus it provides a highly flexible approach that readily adapts in response to varying situational factors.

Perhaps, in near future, given any scenario, no matter how complicated it is, military commanders can sit in front of computers which will provide them with a list of suitable COAs, each with a detailed graph description, explanation and vivid demonstration in the simulation battlefield. A reasonable solution will be automatically chosen and sent out to all entities of the friendly troops. Perhaps then, dare we say that we will only need computer/agent military commanders?
References

Command and Control (C²). 16-6-2005a.
Ref Type: Internet Communication

IBM Deep Blue. 12-6-2005b.
Ref Type: Internet Communication

WebSOM. 22-1-2008.
Ref Type: Internet Communication


Ankerst, M., Breunig, M. M., Kriegel, H. P., & Sander, J. "OPTICS: Ordering Points To Identify the Clustering Structure", Philadelphia PA.


Armstrong, M. 1998, Basic Linear geostatistics Springer Verlag.


Beach, L. R. 1990, Image theory: Decision Making in Personal and Organizational Contexts Wiley, Chichester, UK.


Ref Type: In Press

Ref Type: Electronic Citation

Ref Type: Unpublished Work
Notes: DERA paper (in preparation)


Boyd, J. R. 1987, A Discourse on Winning and Losing MU 43947.


Kolodner, J. 1993, Case Based Reasoning Morgan Kaufmann, San Mateo, CA.


LaBoissonniere, F. 1999, An approach to design Autonomous Agents within ModSAF, Master, Royal Military College of Canada.


Future Directions in Simulation Systems Workshop.


Mileman, T., Knight, B., Petridis, M., Cowell, D., & Ewer, J. 2000, "Case-Based Retrieval of 3-D shapes for the design of metal castings", Journal of Intelligent Manufacturing.


Paplinski, A. P. 2003, Lecture Notes on Neural Networks School of Computer Science and Software Engineering, Monash University, Australia.


Redmond, M. Distributed Cases for Case-Based Reasoning; Facilitating Use of Multiple Cases. 304-309. 1990. Boston, Massachusetts, USA, Morgan Kaufmann. Proceedings of the Annual National Conference on Artificial Intelligence.


Robichaud, F. 2001, "Implementing a Naturalistic Command Agent Design", Proceedings of the Tenth Conference on Computer Generated Forces, Norfolk, VA


Sokolowski, J. 2003b, "Representing Knowledge and Experience in RPDAgent", Scottsdale, AZ, pp. 419-422.


Teknomo, K. Similarity Measurement. 27-4-2005.
Ref Type: Internet Communication


Watson, I. 1999, "CBR is a methodology not a technology", *The Knowledge Based Systems Journal*, vol. 12, no. 5-6, pp. 303-308.


White, G. Discussion with SME. 2-11-2006.
Ref Type: Personal Communication


