# Case-Based Adaptation for Product Formulation

Diana Marcela Segura Velandia, a A West, Chris J. Hinde


# Case-Based Adaptation for Product Formulation

| | |
|---|---|
| Journal: | *International Journal of Computer Integrated Manufacturing* |
| Manuscript ID: | TCIM-2008-IJCIM-0034.R1 |
| Manuscript Type: | Original Manuscript |
| Date Submitted by the Author: | 04-Sep-2008 |
| Complete List of Authors: | Segura Velandia, Diana; Loughborough University, Mechanical and Manufacturing Engineering<br>West, A A; Loughborough University, Department of Manufacturing Engineering<br>Hinde, Chris; Loughborough University, Computer Science |
| Keywords: | NEURAL NETWORK APPLICATIONS, ARTIFICIAL INTELLIGENCE, DECISION SUPPORT SYSTEMS |
| Keywords (user): | Case-based reasoning, adaptation knowledge |
| | |

"Adaptation" *International Journal of Computer Integrated Manufacturing*
*Vol. X, No. X, Month 200X, 000–000*



REVIEW ARTICLE

# Case-Based Adaptation for Product Formulation

Diana Segura Velandia[*] [a], Andrew A. West[a], Christopher J. Hinde[b]
[a]Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, Loughborough, LE11 3TU, United Kingdom
[b]Department of Computer Science/Research School of Informatics, Holywell Park, Loughborough University, Loughborough, LE11 3TU, United Kingdom
Telephone +44 (0) 1509 227 677
Fax: +44 (0) 1509 227 648

Abstract

The fact that Case-based reasoning (CBR) adaptation in design domains is knowledge-intensive is one of the major factors that has limited the industrial application of CBR systems. Nevertheless, inductive techniques can ease the adaptation knowledge acquisition bottleneck by enabling useful knowledge to be elicited from the case-base (CB). Application of neural networks that use the knowledge available in the CB to (i) generate a desired mapping from differences between a query and retrieved cases, (ii) to minimise those differences and hence (iii) to adapt retrieved cases so that an optimal solution to a query is found is studied in this paper. This adaptation method is suitable for CBR systems that use numerical-valued attributes for describing a case.

Keywords: Case-based reasoning, adaptation knowledge, artificial neural networks, polyurethane manufacture, polyurethane formulation.



[*] Corresponding author Email: d.segura@lboro.ac.uk

"Adaptation" *International Journal of Computer Integrated Manufacturing*
*Vol. X, No. X, Month 200X, 000–000*

(verso) *at al*
(recto) Journal title International Journal of Computer Integrated Manufacturing

REVIEW ARTICLE



# Case-Based Adaptation for Product Formulation

Abstract

The fact that Case-based reasoning (CBR) adaptation in design domains is knowledge-intensive is one of the major factors that has limited the industrial application of CBR systems. Nevertheless, inductive techniques can ease the adaptation knowledge acquisition bottleneck by enabling useful knowledge to be elicited from the case-base (CB). Application of neural networks that use the knowledge available in the CB to (i) generate a desired mapping from differences between a query and retrieved cases, (ii) to minimise those differences and hence (iii) to adapt retrieved cases so that an optimal solution to a query is found is studied in this paper. This adaptation method is suitable for CBR systems that use numerical-valued attributes for describing a case.

Keywords: Case-based reasoning, adaptation knowledge, artificial neural networks, polyurethane manufacture, polyurethane formulation.

## Introduction

The pressures to reduce time to market and generate cost effective, high quality products that satisfy customer demands can be supported by the development of CBR systems in design domains i.e. systems that not only serve as retrieval systems such as those found in help-desk applications but also serve as problem solvers. Progress has been made to the point that small CBR prototypes are available for various design tasks (e.g. see (Watson 1997)). The majority of these still rely on a human-based approach to adaptation in which a domain expert proposes the changes that are required to adapt the retrieved case to solve a problem (Craw, Wiratunga et al. 1998; Khemani, Selvamani et al. 2002).
Most researchers seem to agree that adaptation is one of the challenges in the development of useful CBR systems. Adaptation becomes essential especially in domains such as design because solutions never match completely past solutions and frequently two or more previous solutions must be combined to solve a new problem (Börner 2001). A review by (Wilke and Bergmann 1998) also suggests that the adaptation stage is still considered to be the least developed in the implementation of CBR systems. The findings of studies examining approaches to adaptation suggest that adaptation is a difficult task and hence the least developed stage in CBR largely because of the complexity in eliciting or acquiring adaptation knowledge and further compounded by the fact that this task requires a deep understanding (at expert level) of the problem domain

(Bandini and Manzoni 2001; Craw, Jarmulak et al. 2001). It is not surprising that existing commercial CBR tools do not include a well developed adaptation framework.
Studies by (Hanney and Keane 1997) and (Craw, Jarmulak et al. 2001) indicate that the adaptation acquisition problem can be reduced by using the knowledge that may be already present in the case-base. Like inductive learning programs, artificial neural networks (ANN) can be used to learn domain knowledge from examples. However, the ANN approach has received little attention as a tool for CBR adaptation and more studies need to be conducted to study how ANNs could induce knowledge from a CB to reduce the adaptation knowledge acquisition task and perform adaptation of cases.
The purpose of this paper is: (i) to study how the ANN approach could induce knowledge from a CB to adapt cases and (ii) to discuss various issues regarding neural system design, implementation and operation in CBR systems for adaptation of cases in formulation applications.

***Adaptation using the case-base: related work***

The aspects of how to extract knowledge that might already exist in a CB (or case-repository) to adapt cases represented as feature vectors using an ANN approach are covered in this section. First, the problem of adaptation of cases represented as feature vectors is reviewed, followed by the motivations for the use of an ANN approach to solve this problem. Classical reviews and introductions on alternative adaptation strategies (e.g. transformational, generative, null) are reported elsewhere (Kolodner 1993; Wilke and Bergmann 1998; Fuchs, Lieber et al. 1999). Information about formal adaptation frameworks not discussed in this paper can also be obtained from a number of other sources such as the papers by Bergmann and colleagues (Bergmann and Wilke 1998; Wilke and Bergmann 1998), and the surveys by Smyth and co-workers in (Hanney, Keane et al. 1995; Smyth and Keane 1996; Hanney and Keane 1997; Smyth and Keane 1998).
Different case representations require different adaptation strategies (Wilke and Bergmann 1998). For cases that are represented as attribute-value vectors, adaptation involves to replace a value(s) corresponding to the solution in the retrieved case with a value that can fit the desired solution (Figure 1). The adaptation goal is to find the solution feature values for a given a query **q**. A feature case vector **c** is composed of both a case problem $\{\mathbf{c}_p\}$ and a case solution $\{\mathbf{c}_s\}$. For a PU case, the case solution is the set of features that describe a PU solution i.e. the formulation ingredients for a given foam. In retrieval, **q** is matched against cases stored in the CB, so that the closest match is retrieved, i.e. $\mathbf{c}_j^{(r)}$. Its solution set is then adapted so that an optimal solution $c^{(a)}$ to **q**, can be found.

Insert Figure 1 Here.
Figure 1. Adaptation of cases represented as feature vectors.

For many real-world applications, variables describing a set of objects are of different types. For instance a PU foam formulation can be described in terms of compression set (quantitative variable), amount of wear (ordinal), type of polyol (nominal) and presence or absence of voids (binary variable). Heterogeneous distance functions are used to handle mixed variables. Reviews of some other distance functions that handle qualitative attributes and heterogeneous attributes include: (Michalski, Stepp et al. 1981) review some distance functions for qualitative binary attributes are given; (Giraud-Carrier and Martinez. 1994; Gordon 1999) review distances that can handle variables having more than two states or categories i.e. multi-state (e.g. linear if the states are ordered (they can be discrete or continuous) or nominal if they are not). A general measure, which can be used to compare objects described by mixed variables, is described in (Gower 1971). Several heterogeneous distance functions are reviewed in (Wilson and Martinez 1997). The authors presented three distance functions namely, the Heterogeneous Value Difference Metric (HVDM), the Interpolated Value

Difference Metric (IVDM), and the Windowed Value Difference Metric (WVDM). The HVDM distance function uses, the Euclidean distance on linear attributes, and the Value Difference Metric (VDM), which was introduced by Stanfill & Waltz (Stanfill and Waltz 1986), on nominal attributes. The IVDM and the WVDM handle quantitative variables in a similar fashion to VDM.

In the development of complete CBR systems (i.e. systems including adaptation), (Hanney and Keane 1997) have reported on the use of learning adaptation rules from cases represented as nominal feature vectors. The antecedent part of an adaptation rule consists of the feature differences between each pair of cases, with the differences between the solutions provided as the consequent part. Adaptation is carried out by applying these rules to deal with feature differences that are found between a query and a retrieved case.
In particular, for the formulation domain, (Jarmulak, Craw et al. 2001) recently proposed a "knowledge light" approach for learning adaptation knowledge from the cases in the CB. In a fashion similar to the approach used by (Hanney and Keane 1997), constructed adaptation cases are derived by comparing differences between the problem features. For numerical attributes, these differences correspond to feature differences between a probe case and the corresponding retrieved case, and the difference between their solutions (refer to Figure 2). This adaptation approach, deals with feature differences that are found between a query and a retrieved case, by using single or combined suggested corrections.

Insert Figure 2 Here
Figure 2. Constructing adaptation cases. *Source:* (Jarmulak, Craw et al. 2001).

### *A CBR framework for polyurethane formulation: CBRPUR*

In the PU domain, only a few experts have mastered PU formulation knowledge after years of experience hence the design of new formulation is expert-dependent. Several reasons account for this dependency on experts' knowledge such as (i) the complex chemistry of the PUs (Saunders and Frisch 1962; Saunders and Frisch 1962), (ii) the lack of understanding of the underlying principles that govern the formulation process (e.g. PU structure-property relationships, kinetics) (Abouzahr, Ophir et al. 1982; Dounis and Wilkes 1997; Sykes 1999) and (iii) the difficulties of knowledge transfer in this domain. The use of the CBR and ANN paradigms enable the support of PUs formulation tasks by providing a framework for the collection, structuring, and representation of real formulating knowledge.
By organising the problem-specific knowledge and the information on previous cases, the CBR system for PU formulation (CBRPUR) provides non-specialised users with the necessary guidance for solving PU foam formulation problems. The CBRPUR system consists of two main modules that are activated to perform specific tasks of the case-based problem solving process. The case-based (CB) module allows the user to translate formulations into cases (case description) and it also allows the retrieval of similar solutions to a query (case retrieval). For describing a case, the user enters the feature values of a formulation. The retrieval process begins with the search engine selecting from the system's CB only those cases for which the current problem has been previously solved. For each of these past cases, the retrieval mechanism then computes similarity ratings and the user is presented with a list of past cases graded with respect to a similarity metric. Once an appropriate formulation(s) has been selected, this is presented to the ANN module (i.e. an ANN trained on data associated with input changes in mechanical parameters and changes in output formulations) which is responsible for the adaptation of the formulation in order to meet the query's specifications.

## Artificial Neural Network Based Adaptation

The ANN-based adaptation method is developed by training an ANN to generate a mapping between case-problem feature differences (i.e. delta properties) and case-solution feature differences (i.e. delta ingredients) using the back-propagation algorithm. The trained network is a representation of how changes between pairs of property features affect formulation ingredient features. For the purposes of this research, the goal was to find an ANN that can prove that experimental PU formulation data contained in the CB could be used to guide CBR adaptation.

### *Implementation of an artificial neural network to guide CBR adaptation*

#### *Polyurethane formulation data*

Data from a statistical experiment was used for a case study. Thirty four formulations for a Combustion Modified High Resilient (CMHR) flexible foam made using conventional PU hand mixing methods were used as the sample data. The properties measured for these formulations include density (Herrington and Hock 1991), hardness , tensile strength (the maximum tensile stress applied during stretching to rupture) and elongation at break (the percentage of elongation at rupture) and compression set. The formulation recipes and corresponding mechanical properties are given in Figure 3.

Insert Figure 3 Here
Figure 3. PU formulation data.

#### *Methodology and Experimental Setup*

The application of a feed-forward, back-propagation ANN for adaptation of retrieved cases has three phases, namely, *pre-treatment, training and testing*. Phase one, the *pre-treatment* task, as its name suggests involves pre-processing of the input data pattern so that the quality of the data can be determined. Some basic procedures include the elimination of outliers, normalisation, standardisation, transformation and data reduction. Phase two is the *training* process, during which a set of training examples are presented to the network. The parameters of the network are iteratively adjusted to enable input and output relationships to be learned. Phase three is a *testing* process, during which a known input pattern is presented to the trained network and this is required to discover the possible output so that a validation of the answer can be made. Phase one is important to utilise ANNs effectively (and in general any computation technique) to support the analysis and modelling of complex problems. Pre-treatment involves knowing as much as possible about the reliability of the network's input pattern and the transformation of data to support the networks training phase. Phase two is usually a lengthy task and can require many iteration steps to satisfy the training conditions (defined in the following sections). Most ANN application studies usually concentrate on this phase, where the network architecture and input variables can intentionally be changed to optimise the network's performance. Once the network's parameters have been adjusted, the network is ready for testing. The testing phase can be very simple since it only requires the evolution of the performance of the trained ANN when predicting an output appropriate for new inputs.

*Selection of the neural input patterns.* The data sets that were used to choose the training and validation sets to implement the ANN as well as the normalisation that the input and output patterns received are described in this section.

Test cases were randomly selected and correspond to cases No. 11, 12, 22 and 25 (refer to Figure 3). Each input pattern $\delta \boldsymbol{p}$ was obtained by taking the differences between the case-problem attributes for the cases stored in the CB. The difference between the problem part of two cases i.e. the delta property, is an exemplar of an input pattern for the ANN and a number of researchers (Hanney and Keane 1997; Jarmulak, Craw et al. 2001) have suggested the use of these differences between the case-problem features to build adaptation cases. In order to obtain a large set of input patterns for the network, delta property values were calculated using all but the probe cases of the CB.

Input patterns = $\delta p = c^{(i)} - c^{(j)} = \left\{ case\ i_{\substack{problem \\ attribute\ k}} - case\ j_{\substack{problem \\ attribute\ k}} \right\}$, for $i \neq j$.

where $\boldsymbol{c}^{(i)}$ is the i$^{th}$ feature vector which contains only the problem attributes of a case i.e. the mechanical properties.

In a similar fashion, the outputs of the network are obtained by calculating the differences between the solution attributes of a case; this is between the formulation ingredients.

Targets Network = $\delta f = c^{(i)} - c^{(j)} = \left\{ case\ i_{\substack{solution \\ attribute\ k}} - case\ j_{\substack{solution \\ attribute\ k}} \right\}$, for $i \neq j$.

A second data set was obtained to train the network by taking only the differences between a probe case and its first five closest retrieved cases. This will result in a smaller data set than the one previously described but it would include significantly smaller differences that the network could be able to represent more easily. As in the previous example, test cases correspond to cases No. 11, 12, 22 and 25. The total number of input patterns $\delta \boldsymbol{p}$ was obtained by (i) calculating the first five closest cases when each case is a query and (ii) calculating the differences between the case-problem attributes for each query and its first five retrieved cases (Figure 4).

Insert Figure 4 Here

Figure 4. Datasets for the ANN study.

*Data pre-treatment.* The calculated the differences between attribute values, (i.e. the input patterns for the network) were normalised using the min-max normalisation in the interval [-1, 1]. Three types of outputs were used in training: (i) real-valued outputs normalised in the interval [0..1], (ii) binary-valued outputs (i.e. 0,1) and (iii) bipolar-valued outputs (i.e. the binary values corresponding to zero were changed to -1). The binary- and bipolar-valued outputs were obtained by codifying each difference in formulation ingredients between two cases into a triad of ones and zeros according to either a negative, positive or null difference. An example is given in Figure 5. Differences between the formulation ingredients between two cases (e.g. case *i* and case *j*) were calculated. This difference was codified into binary values depending on whether the change in ingredients was negative (case's ingredient should be increased), positive (case's ingredient should be decreased), or zero (leave a case' ingredient as it is). The normalised real-valued outputs were used first in preliminary experiments to narrow down the search space for the network adequate architecture; therefore these results are presented first.

Insert Figure 5 Here

Figure 5. A network training sample.

*Network data sets.* Without enough formulations to justify splitting the data into training and test sets, the *cross-validation* approach was adopted. By following this approach the training set was partitioned into five distinct segments. Each sample in the training set completely specifies all the inputs as well as the outputs that are desired when those inputs are presented. Five networks therefore were trained, each time using random chosen *training sets* with 792 exemplars and *training-test sets* with 20 exemplars (refer to Table 8). The network with the lowest error with respect to the training test set was selected. The performance of the selected network was confirmed by measuring its performance on a third independent set of data called a *test* set. If the error in the test set is not acceptable, a common practice is to joint the training and training test set and re-train (Masters 1993 p.183).

Insert Table 1 Here
Table 1. Networks trained.

*Network architecture.* There has been much debate about how to find the correct network architecture for a particular problem. A large body of research exists which suggest several rules of thumb that can be applied to the problem at hand (Hecht-Nielsen 1990; Masters 1993; Bishop 1995; Hagan, Demuth et al. 1996). In considering how to find the correct network architecture that can map the feature sets studied in this section, the first point to consider is what is meant by the term *network architecture*.
The network architecture is the description of the number of the layers in an ANN, each layer's transfer function, the number of neurons per layer, and the connections between layers. Finding the correct network architecture is a not an easy task. Aside from the number of neurons in a network's output layer, the number of neurons in each layer can be found by trial and error. One rough guideline for choosing the number of hidden neurons in many problems is the geometric pyramid rule (Masters 1993 p. 176). For a one-hidden layer feed-forward network, with n input neurons and m target neurons the rule states that the hidden layer would have sqrt (mn) neurons. However, the problem studied in this study is highly complex and this rule may underestimate the number of hidden neurons required. Therefore, the systematic change of the number of number of neurons in each layer to find an optimum network's architecture was the approach used in this study (Figure 6).

Insert Figure 6 Here
Figure 6. Methodology followed to find an optimum network architecture.

*Algorithm in the MATLAB Artificial Neural Network Toolbox.* To implement a two-hidden layer feed-forward ANN that could guide CBR adaptation the MATLAB 7 Artificial Neural Networks Toolbox was used. The systematic change in the number of neurons in each layer was performed using MATLAB (Figure 6). The pseudo-code to create, train and simulate the ANN using the MATLAB 7 ANN Toolbox used in this project to assist CBR adaptation is presented in Figure 7. The network's functions and training parameters used in MATLAB for the feed-forward back-propagation networks are presented in Table 2.

Insert Figure 7 Here
Figure 7. Pseudo-code to create and train the ANN using MATLAB.

Insert Table 2 Here
Table 2. Network functions and training parameters used in MATLAB.

*Algorithm in GHOST*. Experiences using the MATLAB 7 ANN Toolbox indicated that when the number of iterations increases, the computing time arises leading to a lengthy training process. This is due to the fact that the TRAINLM is suited only for networks with a small number of weights . An alternative ANN software programme, GHOST was used in parallel to MATLAB. GHOST is ANN software in development that is being written in C++ at Loughborough University (Hinde 2004). The network functions and training parameters available in GHOST and used for the feed-forward back-propagation networks are presented in Table 10. Initially, MATLAB was used to find the optimum architecture when real-valued outputs were used. Once an idea of the optimum architecture was found, further training was carried out using GHOST. Several networks were trained with different training and test sets as mentioned formerly.

Insert Table 3 Here
Table 3. Network functions and training parameters used in GHOST.

***Using an artificial neural network to adapt cases***

By training an ANN, as it was mentioned previously, a mapping between the network's input and the output patterns is obtained. The trained network was used to adapt cases in the following manner (refer to Figure 8). First, the problem features *difference* between a retrieved case and a query is presented to the network. The trained network having learnt that for various delta properties, certain changes in the output variables result, reports on how the retrieved case solution features (i.e. formulation ingredients) should be adapted.

Insert Figure 8 Here
Figure 8. Artificial neural network for adaptation of PU retrieved cases.

In order to adjust the amount of ingredients that need to be changed, a "vote mechanism" that uses ranked retrieved cases and their distances to the query was used. The contribution of the retrieved cases to the answer is determined by a relevance factor. The definition of this factor allows to take into account the similarity between the query and cases i.e. the similar the case to the query is (smaller the distance), the higher the contribution of the case to the answer should be.
The relevance factor for each case is defined in terms of the distance between the retrieved case and the query by the following relation:

$$f_i = 1 - \frac{d_{ij}}{\sum_{i,j=1}^{r} d_{ij}} \qquad (1)$$

where $r$ is the number of retrieved relevant cases; and $d_{ij}$ is the distance function used in the similarity assessment between a case $i$ and a query $j$.
For instance, the steps for using this vote mechanism using the first two ranked cases include to: (i) present the difference between the closest retrieved case and the query to the trained artificial neural network, (ii) determine relevance factors, (iii) obtain an adapted solution case by using the relevance factor for each solution feature, according to the algorithm shown in Figure 9.

Insert Figure 9 Here
Figure 9. Algorithm to adapt retrieved cases using the trained ANN output.

Note: For the case when two ranked retrieved cases are used, each network's output attribute is compared. If the network's output is the same, a New_Adjusted case is generated by taking the contribution of each ranked case to the response. If they are different, the New_Adjusted case attributes are the same as the closest retrieved case.

An example of the adaptation of cases using this vote mechanism is shown in
. The relevance factor is calculated as:

$$f_1 = 1 - \frac{0.142}{0.142 + 0.189}, \cdots f_2 = 1 - \frac{0.189}{0.142 + 0.189}$$

Insert Figure 10 Here
Figure 10. Examples of adaptation of cases by means of a vote mechanism and a trained ANN.

## Results and discussion

The results of the training of several feed-forward ANNs and the use of a trained network with the lowest training error to adapt cases are presented in this section. Three types of outputs were studied i.e. real-valued, binary-valued (0,1) and bipolar-valued (-1,1) outputs as it was described previously. The normalised real-valued outputs preliminary experiments served to narrow down the search space for the network adequate architecture, therefore these results are presented first.

### *Artificial Neural Network Implementation*

*Real-valued outputs.* A systematic change in the number of neurons in two layers was used in this study to train a feed-forward ANN with real-valued outputs normalised in the interval [-1..1] in MATLAB. Early tests of the perceptron network (5–8) yielded high mean square errors. Figure 11 a) shows the performance (i.e. mean square error) of the network against the number of hidden neurons in the first layer. It can be seen that as the number of neurons are increased, the error of the network decreases monotonically. Figure 11 b) shows the computing effort (training time in seconds) versus the number of hidden neurons in the first layer. By increasing the number of neurons the training time is also increased in a linear fashion for the chosen interval. In addition, it can be observed that by using more than 80 neurons the error does not decrease significantly but it results in a high computation training time.

Insert Figure 11 Here
Figure 11. Artificial neural network performance.
a) Mean square error as a function of number of neurons in the hidden layer.
b) Training time as a function of the number of neurons in the hidden layer.

In order to find a balance between computational effort (i.e. low training time), good generalisation (i.e. not many hidden neurons) and good modelling ability (i.e. enough hidden neurons), 80 hidden neurons were chosen in each of the two hidden layers. A partial justification for choosing this number of neurons relies also in the fact that the two-hidden layer architecture is known to approximate any continuous mapping provided that the number of hidden neuron is sufficiently large (Bishop 1995). Additionally, it is expected that if the network with this size shows good performance, the network's configuration can be later optimised by using growing and pruning algorithms if required, see for instance (Bishop 1995 p.353).

*Binary-valued output results.* After 20 thousand epochs, the measure of network's performance (the mean square error) for the five trained networks using binary-valued outputs is high when compared to the performance goal of 0.001 indicating that convergence was not reached (Table 11). For instance, for NET 1 the minimum error reached was 4.90 after twenty thousand iterations which is about 4.90 thousand times the performance goal. In addition, the error after twenty thousand epochs was about the same for all networks (approximately 5.0 in average) indicating that these errors may not depend of the data subdivision. It is probable that either the training set or the training-test set were not representative for the relationship to be modelled i.e. they did not contain important information the network should have learnt. Especially for the training-test sets that contained only twenty samples, a high chance of selecting a non-representative subset was considerable. In addition, these results are not enough evidence for concluding that the networks' low performance is due to (i) cross-validation subset selection alone, (ii) strong noise in the data and/or (iii) an error optimisation problem (i.e. a local minimum instead of the global minimum was incidentally reached). It was mentioned in the preceding sections that if the cross-validation error is not acceptable, a common practice is to train using the training and testing set together (Masters 1993 p.183). This assumes the training set being representative of the population. It was assumed that by using the whole data set, a set representative of the population could be obtained. The results of the re-trained network using training and training-test sets merged are shown Table 12. The results with an independent test set are shown in Table 13. An independent test set composed of the differences between the problem part (measured properties) of four probe cases and their retrieved cases using the Euclidean distance was presented to the trained network to assess how well the network performs with real data. It can be seen that there is only one minor error indicating good network's performance on the test set.

Insert Table 4 Here
Table 4. Results on the 20$^{th}$ thousand iterations

Insert Table 5 Here
Table 5. Results for trained network using merged training and test set

The results described above indicate that a trained ANN is able to recognise important patterns in formulation data and therefore can be used to diagnose formulation changes required for adaptation of retrieved cases. From Table 13 it can be seen that there is only one error when the test set is presented. However, this is not enough evidence that the network is able to generalise when presented with more data independent from the original data set. Given the large number of hidden units, and the small training set the network is likely over-fitting the data.

Insert Table 6 Here
Table 6. Testing results of best performance network.

The results shown in Table 13 indicate that the performance goal was mainly limited by the ability of collecting new data and by the computational resources available. More PU formulation data are required if a trained network free of generalisation issues is required. In addition, these results reiterate the importance of the relationship between (i) the number of hidden neurons and (ii) the training set, in the implementation of ANNs. This relationship can lead towards two extremes as it has been illustrated. First, the network can learn

both the data and the noise present in the data and hence it does not generalise well. Second, the scarcity of the training set is partially compensated by limiting the number of hidden neurons but this prevents the network from learning as it should.
*Training with CBR retrieved differences.* A second data set obtained by taking only the differences between probe cases and its first five closest retrieved cases was used to train a two-hidden layer feed-forward back-propagation network. The idea behind this approach is that because smaller differences are present when closer cases are compared, the network could learn more rapidly compared to the case when higher differences are present even though the training set is reduced significantly from 812 to 145 exemplars. After 20 thousand epochs, the trained network (NET 8) using binary-valued outputs the error did not decrease (Table 14). This suggests that the data presented to the network are not representative of the population and confirms the conclusions from the preceding paragraphs in which it was suggested that collection of more independent formulation data is necessary.

Insert Table 7 Here
Table 7. Results for the network NET 8

***Adaptation using the Artificial Neural Network***

Despite the fact that the NET 6 is not able to generalise, it is a good example that the trained ANN can learn important patterns in PU formulation data as a PU expert would do. It can remember relationships within the data to diagnose formulation changes required for adaptation of retrieved cases but it is not able to apply its learning capabilities to other sets of data accurately.
For the purposes of this research, the main goal was to find a network that can do well enough to prove that experimental formulation data contained in the CB can be used as a knowledge container to guide CBR adaptation. Despite the fact that NET 6 do not generalise accurately, it can prove that it is capable of use PU real formulation data to guide CBR adaptation. The results from the validation test shown in Table 13 were used to find the formulation ingredients when the first two retrieved cases using the Euclidean distance were used.

## Conclusions

An ANN methodology designed to carry out the adaptation task in a CBR system for supporting PU formulation is presented in this paper. It is well known that adaptation of cases is a difficult task that has made difficult the deployment of complete CBR systems. For this reason, in order to solve this problem, several researchers have proposed to exploit the knowledge that might be already contained in the CB by means of inductive techniques that elicit knowledge from the CB and ease the adaptation knowledge bottleneck by learning from cases. However, it is not clear (i) what cases need to be compared to generate adaptation rules and cases and (ii) what rules or cases need to be applied to make the algorithm effective and efficient.
Research has highlighted the potential of ANNs an inductive technique to be used in product formulation application to map formulation ingredients or processing conditions to final product properties. It this paper, a feed-forward back-propagation neural network to adapt retrieved PU cases can be implemented as well as a method to use the knowledge learnt by the network to adapt retrieved PU cases.
A two-layer feed-forward network trained using a modified back-propagation algorithm was found to be able to generate adaptation knowledge from a CB composed of real PU formulation data. The network was trained to learn a relationship between the change in formulation properties and formulation ingredients. When a new query exists, the CBR system retrieves cases that partially match the query and the difference between the

query and the closest retrieved case is presented to the trained network for it to propose what ingredients of the retrieved formulation need to be adjusted to compensate for the differences found.
In order to know how much to change the formulation ingredients, a vote mechanism was proposed. This method worked by taking into account the "weight" or contribution to the answer by defining a relevance factor based on the measure of distance used in the retrieval algorithm in a way that the similar the case to the query is (smaller the distance), the higher the contribution of the case to the answer should be.
Although the network has a low global error (4%), the network does not have good generalisation capabilities. This maybe due to the fact that the network was presented with a very small data set and hence it is not able to discover the relationships between the large number of variables involved.



## Acknowledgements

The authors wish to express their gratitude to Dr Richard Heath and Elastogran BASF, UK for their help and cooperation with the PU formulations. This work was supported financially by the Engineering and Physical Sciences Research Council (EPSRC).

## References

Flexible cellular polymeric materials —Determination of compression set. British Standards Institution.
Flexible cellular polymeric materials —Determination of hardness (indentation technique). British Standards Institution.
Flexible cellular polymeric materials —Determination of tensile strength and elongation at break. British Standards Institution: UK.
Abouzahr, S., Z. Ophir, et al., 1982. Structure Property Behavior of Segmented Polyether-MDI-Butanediol Based Urethanes-Effect of Composition Ratio. *Polymer Preprints*, 23: p. 1077.
Bandini, S. and S. Manzoni. CBR Adaptation for Chemical Formulation. *In:* D. W. Aha and I. Watson, Editors *4th International Conference on Case-Based Reasoning, ICCBR 2001*. 2001. Vancouver, Canada: Springer-Verlag GmbH, 634-647.
Bergmann, R. and W. Wilke. Towards a new formal model of transformational adaptation in case-based reasoning. *In: 6th German Workshop on CBR*. 1998. University of Rostock, 43-52.
Bishop, C., 1995. *Neural Networks for Pattern Recognition*. Oxford: Clarendon Press.
Börner, K., 2001. Efficient Case-Based Structure Generation for Design Support. *Artificial Intelligence Review*, 16(2): p. 87-118.
Craw, S., J. Jarmulak, et al. Learning and Applying Case-Based Adaptation Knowledge. *In:* D. W. Aha and I. Watson, Editors *Proceedings of the 4th International Conference on Case-Based Reasoning (ICCBR 01)*. 2001. Vancouver, Canada: Springer-Verlag, 131-145.
Craw, S., N. Wiratunga, et al., 1998. Case-Based Design for Tablet Formulation. *Lecture notes in computer science*, (1488): p. 358.
Demuth, H. and M. Beale, 1998. *MATLAB'S Neural Networks User's Guide*: The MathWorks, Inc., Natick, MA, United States.
Dounis, D. V. and G. L. Wilkes, 1997. Structure-Property Relationships of Flexible Polyurethane Foams. *Polymer Preprints*, 38(11): p. 2819-2828.

Fuchs, B., J. Lieber, et al. Towards a Unified Theory of Adaptation in Case-Based Reasoning. *In:* K. D. Althoff and L. K. Bergmann, Editors *3rd International Conference on Case-Based Reasoning, ICCBR 1999*. 1999. Seeon Monastery, Germany: Springer-Verlag GmbH, 104-117.
Giraud-Carrier, C. and T. Martinez. An Efficient Metric for Heterogeneous Inductive Learning Applications in the Attribute-Value Language. *In:* E. A. Yfantis, Editor *Intelligent Systems, Proceedings of GWIC'94*. 1994. Las Vegas, NV, United States: Kluwer Academic Publishers: Dordrecht, 341-350.
Gordon, A. D., 1999. *Classification*: Chapman & Hall/CRC.
Gower, J. C., 1971. A General Coefficient of Similarity and Some of its Properties. *Biometrics*, 27: p. 857-874.
Hagan, M. T., H. B. Demuth, et al., 1996. *Neural Network Design*: PWS Publishing Company.
Hagan, M. T. and M. B. Menhaj, 1994. Training feedforward networks with the Marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6): p. 989 - 993.
Hanney, K. and M. Keane. The Adaptation of Knowledge Bottleneck: How to Ease it by Learning from Cases. *In:* D. B. Leake and E. Plaza, Editors *Case-Based Reasoning Research and Development: Second International Conference on Case-Based Reasoning, ICCBR-97*. 1997. Providence, RI, United States: Springer, 359-381.
Hanney, K., M. T. Keane, et al. Systems, tasks and adaptation knowledge: revealing some revealing dependencies. *In: 1st International Conference on Case-Based Reasoning*. 1995. Sesmira, Portugal: Springer Verlag, 461-470.
Hecht-Nielsen, R., 1990. *Neurocomputing*. Wokingham: Addison-Wesley.
Herrington, R. and K. Hock, eds. *Flexible Polyurethane Foams*. First ed. 1991, Dow Chemical: Midland, MI.
Hinde, C., ed. GHOST. 2004: Loughborough University.
Jarmulak, J., S. Craw, et al. Using Case-Base Data to Learn Adaptation Knowledge for Design. *In: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI 01)*. 2001. Seattle, WA: Morgan Kaufmann, 1011-1016.
Khemani, D., R. B. Selvamani, et al. InfoFrax: CBR used in Fused Cast Refractory Manufacture. *In:* S. Craw and A. Preece, Editors *6th European Conference on Case-Based Reasoning, EWCBR 2002*. 2002. Scotland, UK: Springer-Verlag GmbH, 560-574.
Kolodner, J. L., 1993. *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann Publishers.
Masters, T., 1993. *Practical neural network recipes in C++*. Boston London: Academic Press.
Michalski, R., R. E. Stepp, et al., 1981. A Recent Advance In Data Analysis: Clustering Objects Into Classes Characterised By Conjunctive Concepts, *In:* L. N. Kanal and A. R. Kanal, Editors *Progress in Pattern Recognition*. Amsterdam; New York, Oxford: North-Holland Pub. Co.
Saunders, J. H. and K. C. Frisch, 1962. *Polyurethanes Chemistry and Technology*. New York: Interscience Publishers.
Saunders, J. H. and K. C. Frisch, 1962. *Polyurethanes Chemistry and Technology*. New York: Interscience Publishers.
Smyth, B. and M. T. Keane, 1996. Using adaptation knowledge to retrieve and adapt design cases. *Knowledge-Based Systems*, 9(2): p. 127-135.
Smyth, B. and M. T. Keane, 1998. Adaptation-Guided Retrieval: Questioning the Similarity Assumption in Reasoning. *Artificial Intelligence*, 102(2): p. 249-293.
Stanfill, C. and D. Waltz, 1986. Toward memory-based reasoning. *Communications of the ACM*, 29(12): p. 1213-1228.
Sykes, P. A., 1999. *Structure-property relationships of chain-extended thermoplastic polyurethane elastomers*. PhD Thesis. Loughborough University.

Watson, I., 1997. *Applying case-based reasoning: techniques for enterprise systems*. San Francisco, Calif: Morgan Kaufmann.
Wilke, W. and R. Bergmann. Techniques and Knowledge Used for Adaptation During Case-Based Problem Solving. *In: 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA-98-AIE): Tasks and Methods in Applied Artificial. Lecture Notes In Computer Science*. 1998. Benicassim, Castellon, Spain: Springer-Verlag, London, UK, 497 - 506.
Wilson, D. R. and T. R. Martinez, 1997. Improved Heterogeneous Distance Functions. *Journal of Artificial Intelligence Research*, 6: p. 1-34.



## Tables

Table 8. Networks trained

| Network Output Data sets | Values | Training set | Training-test set | Cross-validation | Network name |
|---|---|---|---|---|---|
| real-valued output | Continues in the interval [0..1] | 792 exemplars | 20 exemplars | | NET 0 |
| binary-valued output | [0, 1] | 792 exemplars | 20 exemplars | X 5 times | NET 1 |
| | | | | | NET 2 |
| | | | | | NET 3 |
| | | | | | NET 4 |
| | | | | | NET 5 |
| | | 125 exemplars | 20 exemplars | | NET 6 |
| bipolar-valued output | [-1,1] | 125 exemplars | 20 exemplars | | NET 7 |

Table 2. Network functions and training parameters used in MATLAB

| *Function* | *Matlab reference* | *Name* |
|---|---|---|
| Training Function | trainlm | Levenberg-Marquardt back-propagation algorithm (Hagan and Menhaj 1994) |
| Performance function | mse | Mean Squared Error |
| Transfer Function | tansig | Hyperbolic tangent sigmoid transfer function |
| Learning Function | learngdm | Gradient descent with momentum weight and bias learning function |
| *Network Parameters* | *Matlab name* | *Value* |
| Maximum number of epochs to train | net.trainParam.epochs | 100 |
| Performance goal | net.trainParam.goal | 1e-5 |
| Maximum validation failures | net.trainParam.max_fail | 5 |
| Minimum performance gradient | net.trainParam.min_grad | 1e-10 |

Table 10. Network functions and training parameters used in GHOST

| *Function* | *Name* |
|---|---|
| Training Function | Back-propagation algorithm |
| Performance function | Mean Squared Error |
| Transfer Function | Logistic sigmoid transfer function |
| Learning Function | Generalised delta function |
| *Network Parameters* | *Value* |
| Maximum number of epochs to train | >1000 |
| Performance goal | 1e-3 |
| Maximum validation failures | 5 |

Table 11. Results on the 20th thousand iterations

| Feature | NET 1 | NET 2 | NET 3 | NET 4 | NET 5 |
|---|---|---|---|---|---|
| Max No. of errors | 0.901375 | 0.991220 | 0.991027 | 0.915647 | 0.769905 |
| maxsamp | 4 | 7 | 1 | 14 | 14 |
| maxname | output 19 | output2 | output2 | output24 | output21 |
| Num wrong | 20 | 20 | 20 | 20 | 20 |
| Max sq. errors | 6.251496 | 6.640180 | 6.294469 | 5.834857 | 6.344229 |
| Mean sq. errors | 4.907088 | 5.276645 | 5.362985 | 4.875774 | 4.981290 |
| Av. Absolute errors | 9.764001 | 10.195293 | 10.306092 | 9.769579 | 9.958319 |

Table 12. Results for trained network using merged training and test set

| Feature | NET 6 |
|---|---|
| No. Iterations | 17000 |
| Max No. of errors | 1 |
| maxsamp | 19 |
| maxname | output 11 |
| Num wrong | 5 |
| Max squ errors | 6.577664 |
| Mean squ errors | 1.567303 |
| Av. Absolute errors | 2.239573 |

Table 13. Testing results of best performance network.

| Retrieved | Probe | Ing 1 | Ing 2 | Ing 3 | Ing 4 | Ing 5 | Ing 6 | Ing 7 | Ing 8 |
|---|---|---|---|---|---|---|---|---|---|
| 11 | 24 | decrease | increase | no change | no change | no change | no change | no change | increase |
| 12 | 13 | decrease | no change | increase | increase | no change | no change | no change | no change |
| 22 | 28 | increase | no change | no change | decrease | no change | no change | decrease | decrease |
| 25 | 17 | decrease | no change | increase | no change | increase | increase | no change | no change |

Table 14. Results for the network NET 8

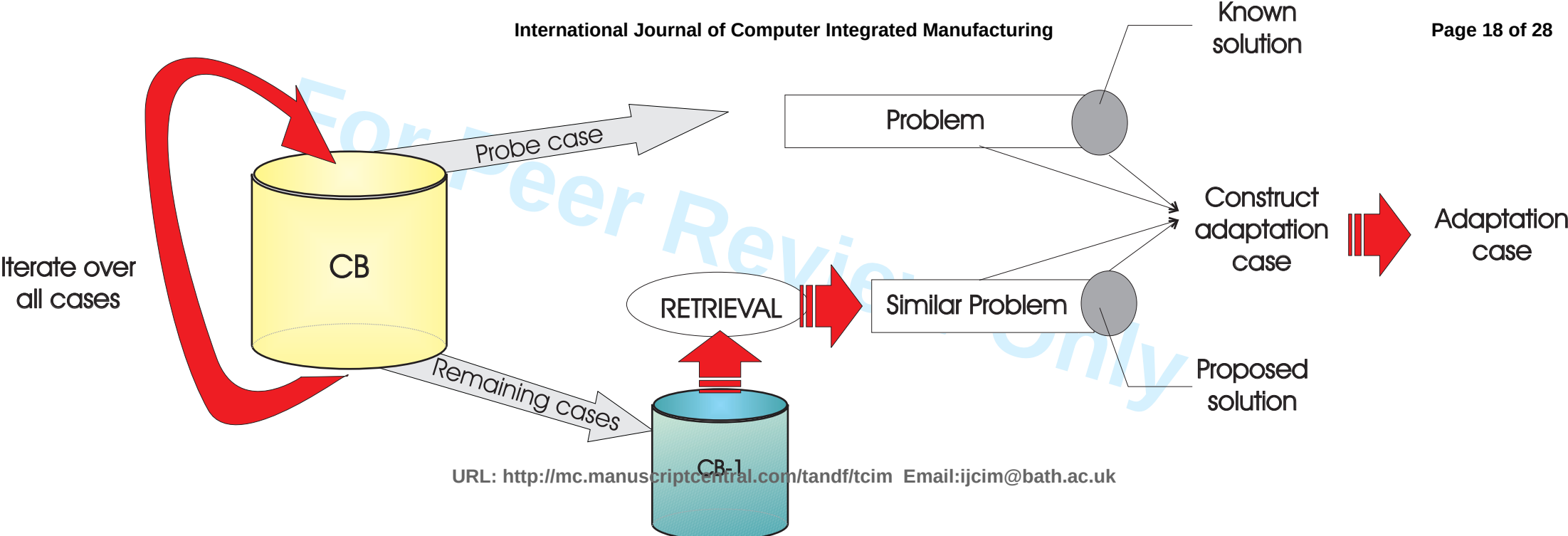
| Feature | NET 8 |
|---|---|
| Max No. of errors | 0.927057 |
| maxsamp | 14 |
| maxname | output9 |
| Num wrong | 20 |
| Max sq. errors | 7.143233 |
| Mean sq. errors | 4.908544 |
| Av. Absolute errors | 9.339203 |



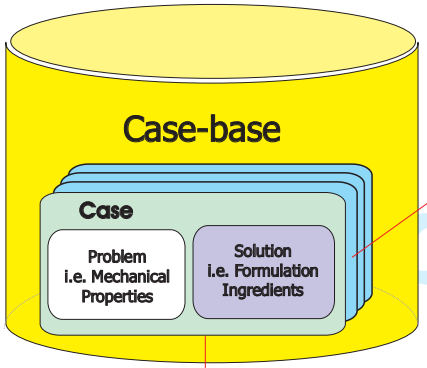
## List of Figure Captions

Figure 11. Artificial neural network performance.

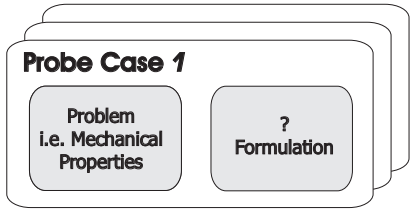
| Task | Result |
|---|---|
| 1. Retrieval<br>a) match problem features between query **q** and cases in the case base<br>$\mathbf{q} = \{c_{p1}, c_{p2}, c_{p3}, \ldots c_p\}$<br><br>b) retrieve *j* closest cases | $\mathbf{c}_1^{(r)} = \{c_{p1}^{(r)}, c_{p2}^{(r)}, c_{p3}^{(r)}, \ldots c_p^{(r)}\}, \{c_{s1}^{(r)}, c_{s2}^{(r)}, c_{s3}^{(r)}, \ldots c_s^{(r)}\}$<br>⋮<br>$\mathbf{c}_j^{(r)} = \{c_{p1}^{(r)}, c_{p2}^{(r)}, c_{p3}^{(r)}, \ldots c_p^{(r)}\}, \{c_{s1}^{(r)}, c_{s2}^{(r)}, c_{s3}^{(r)}, \ldots c_s^{(r)}\}$ |
| 2. Adapt retrieved case<br>a) Adjust solution features<br>b) Propose solution | $\mathbf{c}^{(a)} = \{c_{p1}^{(r)}, c_{p2}^{(r)}, c_{p3}^{(r)}, \ldots c_p^{(r)}\}, \{c_{s1}^{(a)}, c_{s2}^{(a)}, c_{s3}^{(a)}, \ldots c_s^{(a)}\}$ |



| Run | Chemical Formulae (pphp) | | | | | | | | Mechanical Properties | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Polyether F | Polyether F | Amine Crossli | Amine Cataly | Amine Catal | Silicone | Water | Index | Hardness (N) | Density ($kg/cm_3$) | Tensile Strength (kPa) | Elongation (%) | Compression Set (%) |
| 1 | 91.95 | 1.50 | 1.50 | 0.3 | 0.75 | 0.25 | 3.75 | 110 | 445 | 54.1 | 134 | 73 | 10.1 |
| 2 | 92.70 | 3.00 | 0.00 | 0.3 | 0.75 | 0.25 | 3.00 | 110 | 240 | 48.2 | 86 | 83 | 7.3 |
| 3 | 93.35 | 1.50 | 0.00 | 0.9 | 0.25 | 0.25 | 3.75 | 85 | 384 | 67.6 | 137 | 108 | 12.6 |
| 4 | 89.85 | 3.00 | 1.50 | 0.9 | 0.25 | 0.75 | 3.75 | 85 | 177 | 47.6 | 74 | 83 | 5.8 |
| 5 | 93.60 | 1.50 | 0.00 | 0.9 | 0.25 | 0.75 | 3.00 | 85 | 141 | 59.3 | 62 | 106 | 7.7 |
| 6 | 91.35 | 1.50 | 1.50 | 0.9 | 0.25 | 0.75 | 3.75 | 110 | 454 | 53.1 | 148 | 79 | 8.4 |
| 7 | 91.35 | 3.00 | 0.00 | 0.9 | 0.75 | 0.25 | 3.75 | 85 | 152 | 47.1 | 67 | 89 | 7 |
| 8 | 91.95 | 3.00 | 0.00 | 0.3 | 0.25 | 0.75 | 3.75 | 85 | 235 | 51.2 | 64 | 81 | 6.8 |
| 9 | 91.20 | 3.00 | 1.50 | 0.3 | 0.25 | 0.75 | 3.00 | 110 | 192 | 50.8 | 50 | 62 | 4.9 |
| 10 | 90.45 | 3.00 | 1.50 | 0.3 | 0.75 | 0.25 | 3.75 | 85 | 168 | 48.9 | 56 | 75 | 6 |
| 11 | 90.95 | 3.00 | 1.50 | 0.3 | 0.25 | 0.25 | 3.75 | 110 | 431 | 53.5 | 90 | 59 | 7.7 |
| 12 | 90.85 | 1.50 | 1.50 | 0.9 | 0.75 | 0.75 | 3.75 | 85 | 154 | 48.9 | 61 | 80 | 7.5 |
| 13 | 92.95 | 1.50 | 0.00 | 0.3 | 0.75 | 0.75 | 3.75 | 85 | 149 | 48.2 | 60 | 83 | 8.6 |
| 14 | 91.25 | 2.50 | 1.00 | 0.7 | 0.50 | 0.50 | 3.55 | 100 | 230 | 51.1 | 86 | 81 | 3.9 |
| 15 | 91.10 | 3.00 | 1.50 | 0.9 | 0.25 | 0.25 | 3.00 | 85 | 171 | 61.2 | 65 | 86 | 4 |
| 16 | 89.35 | 3.00 | 1.50 | 0.9 | 0.75 | 0.75 | 3.75 | 110 | 354 | 55.0 | 114 | 70 | 4.6 |
| 17 | 93.20 | 3.00 | 0.00 | 0.3 | 0.25 | 0.25 | 3.00 | 85 | 161 | 59.1 | 55 | 87 | 4.6 |
| 18 | 92.70 | 1.50 | 1.50 | 0.3 | 0.25 | 0.75 | 3.00 | 85 | 175 | 60.1 | 74 | 92 | 5.4 |
| 19 | 92.60 | 1.50 | 1.50 | 0.9 | 0.25 | 0.25 | 3.00 | 110 | 210 | 51.3 | 93 | 87 | 6.9 |
| 20 | 93.10 | 1.50 | 0.00 | 0.9 | 0.75 | 0.75 | 3.00 | 110 | 153 | 48.3 | 66 | 81 | 9.1 |
| 21 | 92.10 | 1.50 | 1.50 | 0.9 | 0.75 | 0.25 | 3.00 | 85 | 159 | 61.1 | 76 | 101 | 6.1 |
| 22 | 94.20 | 1.50 | 0.00 | 0.3 | 0.75 | 0.25 | 3.00 | 85 | 129 | 58.6 | 51 | 95 | 5.6 |
| 23 | 94.70 | 1.50 | 0.00 | 0.3 | 0.25 | 0.25 | 3.00 | 110 | 182 | 50.3 | 67 | 87 | 4.9 |
| 24 | 92.45 | 1.50 | 1.50 | 0.3 | 0.25 | 0.25 | 3.75 | 85 | 301 | 58.6 | 82 | 75 | 7.2 |
| 25 | 90.70 | 3.00 | 1.50 | 0.3 | 0.75 | 0.75 | 3.00 | 85 | 156 | 60.7 | 57 | 85 | 5.3 |
| 26 | 91.25 | 2.50 | 1.00 | 0.7 | 0.50 | 0.50 | 3.55 | 100 | 289 | 51.5 | 104 | 88 | 5.3 |
| 27 | 91.85 | 3.00 | 0.00 | 0.9 | 0.25 | 0.25 | 3.75 | 110 | 469 | 53.6 | 140 | 82 | 7.6 |
| 28 | 92.85 | 1.50 | 0.00 | 0.9 | 0.75 | 0.25 | 3.75 | 110 | 477 | 51.8 | 143 | 81 | 8.9 |
| 29 | 91.60 | 3.00 | 0.00 | 0.9 | 0.75 | 0.75 | 3.00 | 85 | 144 | 58.4 | 70 | 115 | 5.2 |
| 30 | 92.20 | 1.50 | 1.50 | 0.3 | 0.75 | 0.75 | 3.00 | 110 | 275 | 50.5 | 113 | 90 | 4.4 |
| 31 | 91.45 | 3.00 | 0.00 | 0.3 | 0.75 | 0.75 | 3.75 | 110 | 471 | 52.7 | 158 | 86 | 4.1 |
| 32 | 93.45 | 1.50 | 0.00 | 0.3 | 0.25 | 0.75 | 3.75 | 110 | 708 | 62.7 | 217 | 96 | 9.2 |
| 33 | 90.60 | 3.00 | 1.50 | 0.9 | 0.75 | 0.25 | 3.00 | 110 | 273 | 49.8 | 85 | 79 | 9.7 |
| 34 | 92.10 | 3.00 | 0.00 | 0.9 | 0.25 | 0.75 | 3.00 | 110 | 292 | 50.4 | 105 | 83 | 8.6 |




Probe cases




Remaining cases

2

Total no. of remaining cases = 29

1

**Feature Differences between each case and its first 5 retrieved cases**

for case i ≠ case j

1. Retrieve 5 closest cases for each case i
2. Calculate difference between case i and its retrieved cases

29 X 5= 145 exemplars

**Feature Differences between each case with each other**

for case i ≠ case j

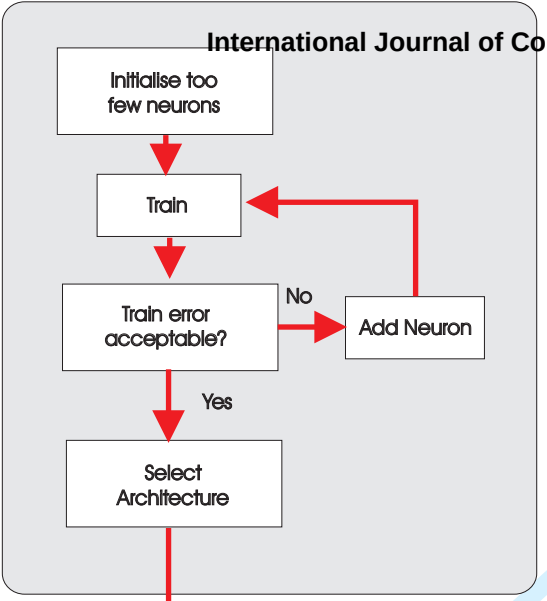
1. Calculate difference between case i and case j

29 X 28= 812 exemplars

**Case-Solution (i.e. Formulation Ingredients)**

| | Ingredient 1 | Ingredient 2 | Ingredient 3 | Ingredient 4 | Ingredient 5 | Ingredient 6 | Ingredient 7 | Ingredient 8 |
|---|---|---|---|---|---|---|---|---|
| Case i | 20 | 60 | 10 | 40 | 110 | 1.5 | 15 | 10 |
| Case j | 30 | 60 | 20 | 35 | 85 | 0 | 15 | 50 |
| (Case i -Case j) | -10 | 0 | -10 | 5 | 25 | 1.5 | 0 | -40 |
| | **1, 0, 0,** | **0, 1, 0,** | **1, 0, 0,** | **0, 0, 1,** | **0, 0, 1,** | **0, 0, 1,** | **0, 1, 0,** | **1, 0, 0** |
| | **Increase,** | **No change,** | **Increase,** | **Decrease,** | **Decrease,** | **Decrease,** | **No change,** | **Increase** |

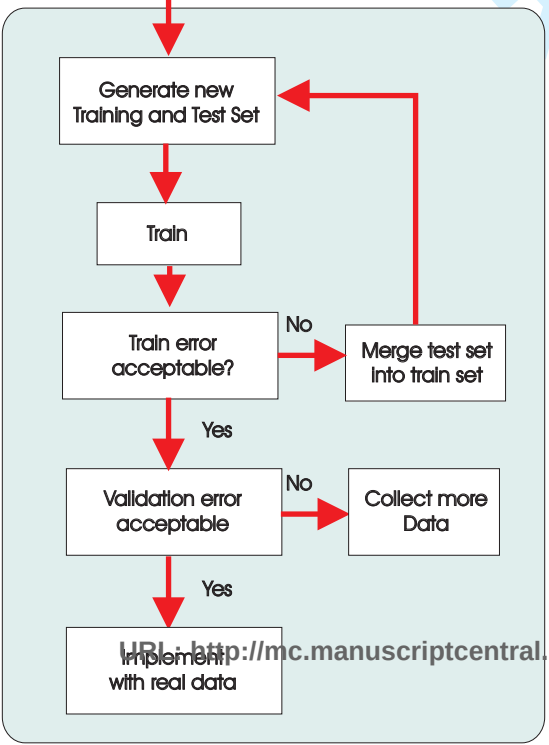
Formulation ingredients in Case i to minimise difference




**Matlab Implementation**

| | |
|---|---|
| Inputs | 5 real valued variables [-1,1] |
| Outputs | 8 bipolar valued variables [1,0,-1] |
| Training Function | Levenberg-Marquardt |
| Transfer Function | Hyperbolic sigmoid |
| No. of epochs to train | 100 |
| Performance goal | 1e-5 |

- Easy implementation
- Slow training
- First tool available




**Ghost Implementation**

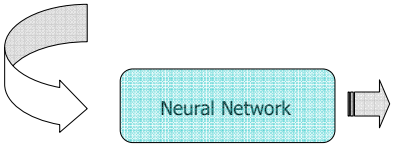
| | |
|---|---|
| Inputs | 5 real valued variables [-1,1] |
| Outputs | 24 binary valued variables [1,0] |
| Training Function | Back-propagation |
| Transfer Function | Sigmoid |
| No. of epochs to train | 20000 |
| Performance goal | 1e-3 |

- Easy implementation
- Faster training
- Limited training

```
Present the input patterns P and targets T
inputs = P
targets = T

Create a two-layer feed-forward network
for iteration =1:50
      for Number of Neurons in Layer2 = 0:100
            for Num Neurons Layer1 = fixed at optimum number
                  net = newff([Inputs range],[No units of each layer],{'transfer function
for                 each layer'});

Network Training
      net.trainParam.epochs = 50;
[net,tr,Yo,E] = train(net,P,T);
end
      end
         end

Network Simulation
Present the input patterns P to simulate the network.
inputs = P
            [Y] = sim(net,P)
```

| **Mechanical Properties** [0,1] | | | | | |
|---|---|---|---|---|---|
| | Hardness | Density | Tensile Strength | Elongation | Compression Set |
| Query | 0.009 | 0.218 | 0.413 | 0.675 | 0.883 |
| Retrieved Case | 0.007 | 0.211 | 0.412 | 0.686 | 0.908 |
| Difference | 0.002 | 0.007 | 0.001 | -0.011 | -0.025 |

| **Chemical Formulae** (pphp) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Polyether Polyol 1 | Polyether Polyol 2 | Amine Crosslinker | Amine Catalyst 1 | Amine Catalyst 2 | Silicone | Water | Index |
| ? | ? | ? | ? | ? | ? | ? | ? |
| 92.95 | 1.5 | 0 | 0.3 | 0.75 | 0.75 | 3.75 | 85 |




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 92.95 | 1.5 | 0 | 0.3 | 0.75 | 0.75 | 3.75 | 85 |
| decrease | no change | increase | increase | decrease | no change | no change | no change |

**But how to apply these changes?**

```
Calculate for each attribute i,

for i=1: p
      IF Adapt1 (i) == Adapt2 (i) // i.e. adapted responses are the same e.g. decrease

                New_Adjusted (i) = Adapt1 (i)*f1 + Adapt1 (i)*f2
      ELSE
                New_Adjusted (i) = Adapt1 (i)

// NOTE
//  Adapt1 (i): NN value for the retrieved case 1; f1= relevant factor for the retrieved case 1.
//  Adapt2 (i): NN value for the retrieved case 1; f2= relevant factor for the retrieved case 2.
//  New_Adjusted (i): adjusted values for attribute i.
```

| | Euclidean distance | Adapt case [neural network output data] | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Polyether polyol 1 | Polyether polyol 2 | Amine Crosslinker | Amine Catalyst 1 | Amine Catalyst 2 | Silicone | Water | Index |
| Retrieved Case 1 | 0.142 | decrease | no change | increase | increase | no change | no change | no change | no change |
| Retrieved Case 2 | 0.189 | decrease | no change | increase | no change | no change | no change | increase | decrease |

| | Relevance factor | Retrieved Formulation [CBR output data] | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Retrieved Case 1 | 0.572 | 92.95 | 1.5 | 0 | 0.3 | 0.75 | 0.75 | 3.75 | 85 |
| Retrieved Case 2 | 0.428 | 93.1 | 1.5 | 0 | 0.9 | 0.75 | 0.75 | 3 | 110 |

| | Predicted Formulation Using Neural Network Ourtput data and Relevance factor f | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Polyether polyol 1 | Polyether polyol 2 | Amine Crosslinker | Amine Catalyst 1 | Amine Catalyst 2 | Silicone | Water | Index |
| Adapted Case | 92.95 | 1.5 | 0 | 0.3 | 0.75 | 0.75 | 3.75 | 85 |