# Search Heuristics, Case-Based Reasoning and Software Project Effort Prediction

**Colin Kirsopp, Martin Shepperd, John Hart**

Empirical Software Engineering Research Group
School of Design, Engineering and Computing
Bournemouth University
Royal London House
Bournemouth,
Dorset, UK
Email: {ckirsopp, mshepper, jhart}@bournemouth.ac.uk

## Abstract

This paper reports on the use of search techniques to help optimise a case-based reasoning (CBR) system for predicting software project effort. A major problem, common to ML techniques in general, has been dealing with large numbers of case features, some of which can hinder the prediction process. Unfortunately searching for the optimal feature subset is a combinatorial problem and therefore NP-hard. This paper examines the use of random searching, hill climbing and forward sequential selection (FSS) to tackle this problem. Results from examining a set of real software project data show that even random searching was better than using all available for features (average error 35.6% rather than 50.8%). Hill climbing and FSS both produced results substantially better than the random search (15.3 and 13.1% respectively), but FSS was more computationally efficient. Providing a description of the fitness landscape of a problem along with search results is a step towards the classification of search problems and their assignment to optimum search techniques. This paper attempts to describe the fitness landscape of this problem by combining the results from random searches and hill climbing, as well as using multi-dimensional scaling to aid visualisation. Amongst other findings, the visualisation results suggest that some form of heuristic-based initialisation might prove useful for this problem.

## 1 BACKGROUND TO PROJECT EFFORT PREDICTION

An important problem in the field of software engineering is to be able to make predictions concerning size of, and effort required for, software development projects. These predictions must be made at an early stage during a project, working primarily from feasibility and requirements specification documents. Despite a significant amount of research effort over the past 30 years, no one method has been found to be consistently effective.

Early prediction techniques such as Boehm's COCOMO prediction system (Boehm 1984) (and more recent variants) were algorithms that sought to relate predicted source code size together with a large number of cost drivers to effort and nominal duration. Unfortunately, there is little independent evidence that this type of universal approach yields consistently useful results. It would seem that despite the cost drivers and various parameters COCOMO is over adapted to the data set from which it was developed. Alternative approaches include the use of simple statistical techniques such as stepwise regression to develop models that have local predictive value only. The MERMAID project is a good example of this approach (Kok, Kitchenham et al. 1990). More recently there has been considerable interest in a variety of machine learning (ML) methods that are trained on local data. This has included work with artificial neural nets, for example (Finnie, Wittig et al. 1997), rule induction algorithms (Mair, Kadoda et al. 2000) and genetic programming systems to search for functions that fit the data (Burgess and Lefley 2001; Dolado 2001). Whilst some quite accurate results have been reported, all these techniques suffer from the problem of poor explanatory value. In other words they are able to provide a prediction but not necessarily offer a justification that is helpful for a project manager who will need, in some sense, to trust the prediction if these are techniques are to be deployed in practice. Another ML technique is case-based reasoning (CBR). This has an advantage in that there is substantial evidence (Klein 1998) that humans make use of analogies or prototypes when solving problems.

The remainder of this paper is organised as follows. The next section reviews the use of CBR for effort prediction. We then address the feature subset selection problem which causes particular difficulties for problems characterised by large numbers of features, few cases and limited domain understanding. We briefly review the range of approaches to feature subset selection and in particular how it may be viewed as a search problem. We then turn to our effort prediction case study and show that feature subset selection contributes significantly to prediction accuracy for our CBR approach. We compare three search techniques: random, steepest ascent hill climbing and forward sequential selection. Next we try to explain our results in terms of the fitness landscape and the problems of visualisation since there are 43 binary dimensions. We conclude with a discussion of the significance of our results, the extent to which they might be generalised and areas requiring further investigation.

## 2 CASE-BASED REASONING

A number of research groups including ourselves have been investigating applying CBR to software project prediction since the mid 1990s (Prietula, Vincinanza *et al*. 1996; Shepperd, Schofield *et al.* 1996). The basic approach is that each completed project is considered as a separate case and added to a case base. Each case is characterised by $n$ features which might be continuous, discrete or categorical. Example features might include the number of interfaces, the level of code reuse and the design method employed. Clearly, a restriction is that these features must be known (or reliably estimated) at the time of prediction. A new project, for which a prediction is required (known as the target case), is also characterised by the same feature set and plotted in standardised $n$-dimensional feature space. Distance, usually a modified form of Euclidean distance is used to identify the most similar cases to the target and these, since they have known values for effort, etc., are used as the basis of the prediction. For a thorough review of CBR the reader is referred to (Kolodner 1993).

There have been some differences in approach, for instance Prietula *et al.* make substantial use of adaptation rules whilst our work is closer to a $k$ nearest neighbour ($k$-NN) method. We believe our approach to have the advantage of being more flexible since we are not restricted to a particular set of features which is a requirement for adaptation rules. This flexibility enabled us to develop ANGEL, a software estimation CBR tool that has a shell structure that can deal with arbitrary sets and types of features. Features are re-scaled so that the influence of a feature is not related to the choice of unit. This is achieved by normalising the using the difference between maximum and minimum observed values as a denominator. For more details see (Shepperd, Schofield *et al.* 1996). In general the results have been sufficiently encouraging — we found that ANGEL performed as well or better than a stepwise regression model across 9 data sets (Shepperd and Schofield 1997) — to generate significant interest.

## 3 FEATURE SUBSET SELECTION PROBLEM

Searching for useful feature subsets has been recognised as a challenge for the ML community as a whole for a number of years. This is because all techniques — and not just CBR — are potentially vulnerable to erroneous, irrelevant or redundant data. Approaches to searching for subsets fall into two categories: filters and wrappers (Kohavi and John 1997). Filters operate independently of the ML algorithm reducing the number of features prior to training. By contrast, the wrappers use the ML algorithm itself on some sample of the data set in order to determine the fitness of the subset. This tends to be computationally far more intensive, but can find better subsets than the filter methods. In this paper we focus on wrappers. This is because our goal is prediction of a continuous variable rather than classification and filter methods look for features strongly correlated with the dependent variable and orthogonal to the independent variables. Given our need to re-scale features to overcome problems of differing units, orthogonality is not necessarily a desirable characteristic of a feature subset.

Various wrapper methods have been investigated by a number of researchers. The original version of ANGEL addressed the problem of searching for the optimal feature subset by an exhaustive search using a jack knife on the case base in order to determine fitness. However, the search is $O(2^n)$ so when $n$ exceeds 15-20 this becomes computationally intractable. Other approaches have included different variants of hill climbing algorithms (Skalak 1994), simulated annealing algorithms (Debuse and Rayward-Smith 1997), sequential feature selection algorithms, both forward and backward (Aha and Bankert 1996) and genetic algorithms (Whitley, Beveridge *et al.* 1997). These have generally been reported to lead to improvements in accuracy without the prohibitive computational cost of an exhaustive search.

Essentially all these methods have a search component to generate candidate subsets from the space of all possible subsets and a fitness function which is a measure of the error deriving from the ML algorithm using the subset, trained on a sample from the data set and validated on a holdout sample. Typical sampling techniques are the jack knife and $n$-fold validation. The fitness function is generally a measure of error and as such is a cost that should be minimised. The exact nature of the measure will depend upon the nature of what is being predicted but is usually either based on the cost of misclassifications or the sum of absolute residuals.

# 4 PREDICTION CASE STUDY

This case study examines the use of different search algorithms to optimise the feature subset used to build effort prediction systems for software development projects. We look at the accuracy of the prediction systems built using different feature subsets selected using these algorithms. The results discussed in this study were generated by the ArchANGEL CBR tool[1]. The same tool settings (except feature subset selection algorithm) were used for all runs. Predictions were made using an inverse distance weighted average of the 3 nearest neighbours, i.e. $k$=3. The data set was jack-knifed to produce a prediction for each case and the sum of absolute residuals for these predictions was used as the accuracy indicator for each prediction system.

The data used for the case study is the so-called 'Finnish dataset'. This dataset contains 407 cases described by 90 features. The features are a mixture of continuous, discrete and categorical. However, there are a number of missing data values and also some features that would not be known at prediction time and so should not be included in a prediction system. Removing features with missing values or after-the-event data, leaves a subset of 44 features that are actually used in the case study. The data set also exhibits significant multi-collinearity, in other words there are strong relationships between features as well as with the feature to be predicted, namely effort. Software project effort data sets are characterised by relatively few cases (almost invariably under 500 and typically less than 50). Therefore the data set used in this paper is at the large end of this spectrum.

The number of possible combinations of feature subsets is $2^n$ where $n$ is the number of features. In this case study there are 44 features, however, one is used as the target feature (what we are trying to predict). This leaves 43 features as the input to the subset search, giving 8.8 $e^{12}$ possible combinations. Clearly an exhaustive search is not feasible so other search strategies must be considered.

This case study examines three alternative strategies:

1. Random feature subset selection
2. Multi-start steepest ascent hill climbing
3. Forward sequential selection

We restricted our choice for two reasons. First, other groups have had some success with these algorithms for finding good feature subsets. Second, there seems no purpose in examining more complex search strategies if the problem can be effectively solved using a hill-climber (Juels and Wattenberg 1994). Also, in order to analyse the value of feature subset selection we used the accuracy from using all features as a comparative baseline.

As a search problem we need to consider two additional issues: representation of solutions and measurement of fitness. Fortunately, for feature subset selection problems the set of candidate features can simply represented as a bit string where 1 denotes selected and 0 excluded. This also provides a view of neighbourhood which is defined as any move derived from mutating a single bit, in other words moving any one feature into, or out of the selected set. Fitness, or strictly speaking cost[2], is a little more complex. As stated in the previous section we are concerned with wrappers. Informally we prefer a feature subset that leads to more accurate project effort prediction. This is defined as jack-knifing across the entire data set so that we obtain a predicted effort value $\hat{e}$ for every case. This can be compared with the true effort value e, in order to derive an absolute residual $r$ which is $|\hat{e}-e|$ since we are indifferent to the direction of error. For the entire data set we sum the absolute residuals. Note that this is a more neutral view of prediction error compared with, say, the sum of the squares of the residuals which adopts a more risk averse stance since a few extreme errors will dominate the fitness measure.

## 4.1 RANDOM FEATURE SELECTION

A simple approach to finding a suitable feature subset in such a large search space is to collect results from a large number of randomly generated feature subsets. Table 1 shows the summary of results from 4028 randomly sampled feature subsets.

Table 1: Summary statistics for random feature selection

| | |
|---|---|
| Count | 4028 |
| Mean | 91272 |
| Median | 91831 |
| Min | 56988 |
| Max | 135586 |

It is worth noting that using all features gives a result close to the mean of the random feature subsets. This implies that a randomly chosen subset of the features is likely to be just as good as using all available features. The best solution found has an accuracy value of 56988 compared with 88522 using all features. This is a significant improvement (recall that low values of sum(|r|) are preferred), however we next turned to a more systematic search strategy in order to seek better results.

## 4.2 HILL CLIMBING

A commonly used search strategy is hill climbing. In this case study the algorithm used was a multi-start steepest

[1] ArchANGEL is the most recent version of the ANGEL software tool for project prediction. It may be downloaded from http://dec.bmth.ac.uk/ESERG/ANGEL/

[2] Whilst we actually wish to minimise the fitness function we will retain the usual terminology of hill-climbing, peaks and so forth.

ascent hill climbing where each climb has a new, randomly selected starting point (or initial feature set). The algorithm is steepest ascent because the entire neighbourhood is evaluated and the move with the best result is used as the next base position. The neighbourhood is defined as any new feature set that can be obtained from toggling a single bit. This definition of the neighbourhood means that any feature set has 43 neighbours that must be evaluated for each step in the climb.

Table 2: Summary statistics for 113 hill climbs

| | |
|---|---|
| Count | 113 |
| Mean | 47803 |
| Median | 51909 |
| Min | 29916 |
| Max | 61049 |

Table 2 gives the summary statistics for the accuracy levels achieved by the 113 hill climbs. We see that the best feature subset found by hill climbing gave a result of 29916. This is a significant improvement over both using all features and random searching (88522, 56988), see also Table 3. Also worthy of note is that the maximum (worst) value of a peak found by hill climbing was 61049. This is only slightly worse than the best of the random results (56988). In fact, all but 8 of the 113 climbs produced results better than all 4028 random selections.

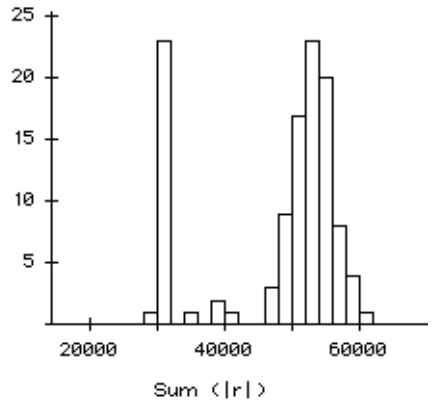


Figure 1: Distribution of Results from Hill Climbing

Figure 1 shows the distribution of results from Table 2. Note the bi-modal nature of this hill climbing results. There are number of significantly better solutions that are clearly separated from the main distribution of results. When the distinction between this set of outlying good solutions and the other results was investigated it was found that the best solutions used fewer features.

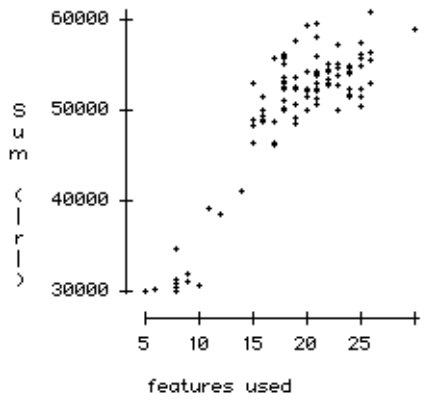


Figure 2: Scatter plot of accuracy against number of features used

Figure 2 shows a scatter plot of sum(|r|) against the number of features included in the feature subset. A slight trend is notable in the main cloud of results but it is the lower valued outliers that dominate, in other words no good solution was found that contains more than 15 features (out of 43).

### 4.3 FORWARD SEQUENTIAL SELECTION

Another form of search that can be used in this situation is a sequential feature selection algorithm. We used a forward sequential selection (FSS) algorithm due to the observation that the best solutions from hill-climbing were relatively sparse (see Figure 2)[3]. FSS works in a similar way to steepest ascent hill climbing. The first difference is that it starts from having no features selected rather than a random selection. Secondly, features are only added (never removed). The best single feature for prediction is found first and added to the feature set. An attempt is then made to add a second feature. This is done by evaluating the combination of the selected feature with each of the other features. The best of these combinations becomes the current feature set and an attempt is made to add another feature. This process is repeated until none of these combinations yields a better result than the current feature set.

[3] An alternative variant would be the backward sequential search (BSS). In this algorithm all features are initially selected and features are removed one by one. Normally, BSS is recommended rather than a FSS. This is because BSS evaluates features to remove in the presence of all the other features that may be included in the final solution. This allows it to take advantage of any interaction between the features when making the decision on which feature to remove. However, it has been suggested that BSS 'is more easily confused by large numbers of features' and than FSS 'is preferred when the optimal number of selected features is small', see for example (Aha and Bankert 1996). Since this search has a large number of features and the results from hill climbing suggest that the better results have a small number of features we chose FSS.

The FSS result was 30202. This is only slightly worse than the result from the best hill climb (by less than 1%) and this result was reached far more quickly. The FSS result was obtained by evaluating just 243 feature combinations. A single hill climb required an average of 688 evaluations and only 3 of 113 hill climbs were equal to or better than the FSS result. This means that hill climbing would on average require around 19000 evaluations to find an equal or better result.

## 5 FITNESS LANDSCAPE

The close relationship between the form of the fitness landscape and the performance of search algorithms has been noted by many researchers, e.g. (Crisan and Muhlenbein 1998; Reeves 2000). Since the 'no free lunch' theorem (Wolpert and Macready 1997), researchers realise that is not possible to say that a particular search algorithm is always better than another. Research now concentrates on trying to show that a particular search algorithm is better than another for a restricted class of search problem. Difficulties with this approach include trying to define a class of search problem and trying to identify that class of problem *a priori*.

A step towards identifying which search algorithms are best for which class of problem is to identify characteristics of the fitness landscape that aid a particular algorithm. Work has been done towards this on a theoretical basis (Sharpe 1998). This section attempts to describe the landscape for this particular search problem and to comment on how this may have affected the results from the search algorithms used. However, there are several challenges. First, and obviously, there is the sheer size of the landscape. Second, there is a visualisation problem. It is convenient to think in terms of a landscape in which the x and y co-ordinates represent the search space and the height the quality or fitness of the solution. Thus we use such abstractions as hill-climbing and peaks. For our problem we must confront 43 dimensions each of extreme coarseness i.e. binary. Consequently simple representations are inapplicable.

The random search can be thought of as sampling from the entire landscape. The results from this sampling could therefore be used to make inferences about the underlying population where the underlying population is the set of results obtained from an exhaustive search of the feature space (if that were computationally possible).

The results shown in Figure 3 show the distribution of 'heights' in the fitness landscape (but not their positions). Values follow an approximately normal distribution centred on a mean of just over 91000 with approximate maximum and minimum values of 135000 and 57000. This gives some idea of typical values, although clearly the whole point of a hard search is that we are interested in the extreme outliers. So we have some idea of the proportions of the fitness landscape that lie at particular heights, but no information on how individual points are positioned in the fitness landscape. The landscape could be entirely chaotic or organised as a single smooth peak. There is no way of knowing the structure of the landscape from these results.

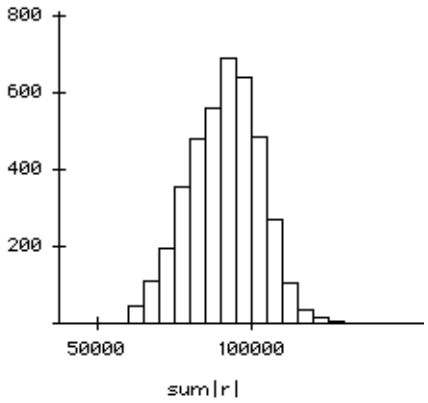


Figure 3: Distribution of results for randomly selected feature sets

The results from the hill climbing also help provide information concerning the fitness landscape. Hill climbing algorithms depend on certain assumptions about the nature of the fitness landscape for their operation. They exploit local correlations in the structure of the search space, i.e., they assume that there is likely to be better point near to a good point. A hill climbing algorithm would not work in a chaotic landscape. The significantly better results produced by the hill climbing (over the random selection) suggest that there is structure in the landscape that is exploited by hill climbing. The length of the climbs also supports this view. The less structured the landscape the shorter individual climbs would be. The median climb length was 15 steps and the maximum climb length was 31 steps. Compare this with the 43 steps that is needed to traverse the entire breadth of the state space and there are clearly large-scale structures in the fitness landscape.

What else can the hill climbing results tell us about the structure of the fitness landscape? Firstly, there are a large number of distinct peaks in the landscape (highly multi-modal). The 113 hill climbs found 98 distinct peaks - only 15 climbs found a previously visited peak. Interestingly, all of the peaks that were visited more than once were one of the ten highest peaks. This suggests that the better peaks in this landscape have larger basins of attraction.

If we combine the hill climbing results with the random results another observation can be made. The best of

4028 random selections was better than only the worst 8 hill climbs. There is very little chance of finding a good result by random searching. This implies that little of the feature space lies near the top of the hills, i.e., the peaks are very sharp.

As well as the local structures within the landscape that aid individual hill climbs there could also be larger scale structures or trends. To try to identify any such trends requires a way of visualising the position and height of points in 43 dimensional binary space. A multi-dimensional scaling (MDS) algorithm was used to achieve this visualisation. MDS is used to compress high dimensional data into lower dimensional data while attempting to retain the relative distances between the data points. We used the MDS algorithm provided by the statistics package SPSS to convert the similarity in feature subsets corresponding to each peak into co-ordinates in 2 dimensional space. These points were then plotted and their height or fitness denoted by different symbols. Note that we have no data on intervening points so we cannot draw contours.

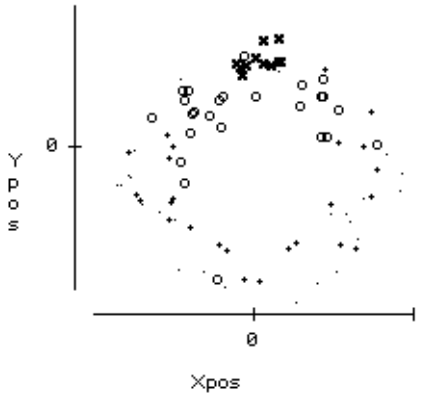


Figure 4: MDS of hill climbing results

The results for the best 100 hill climbs are shown in Figure 4. There is a general tendency for the better peaks to be in the upper part of the figure and the worse peaks to be in the lower part of the figure. More notable is the tight cluster of Xs representing the 25 best climbs (10 best peaks). These observations show that there are some global trends within the feature space. This may be a trend in just the height of the peaks or a general raising of the landscape. To investigate this issue a second random sampling was done that was constrained to the region of the highest 10 peaks. Figure 5 shows the results from this localised random sampling.

From the clear differences in central tendency between the constrained and the global random samples, there appears to be a tendency for the average height to increase towards a particular region in the feature space. To continue the landscape analogy, although there are many local peaks throughout the landscape, in general the landscape rises towards a localised massif. This massif has a higher average level of fitness as well as the highest peaks. It does, however, still contain some deep sinkholes (see Table 3 where the lowest fitness was in excess of 143000).

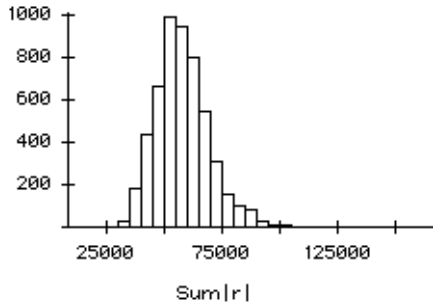


Figure 5: Distribution of restrained random search

Table 3: Summary statistics for restrained random search

| | |
|---|---|
| Count | 5358 |
| Mean | 58095 |
| Median | 56714 |
| Min | 31353 |
| Max | 143824 |

In some ways FSS can be thought of as a single steepest ascent hill-climb with a fixed starting point and some moves made taboo, since once features are included in a subset they cannot subsequently be removed. The best hill climbs are all in a region of sparse feature subsets and the FSS path also operates in this region of the fitness landscape. The value of a good starting point can be seen in the short length of the path (5 steps) and consequent efficiency gains.

To summarise, we know the landscape is multi-modal, that the peaks have steep slopes and small summits and somewhat surprisingly there are some very large scale structures as evidenced by the large basins of attraction for the better solutions. This last observation was unexpected. What we do not know is whether there exists some higher peak, or peaks, but with a very small basin of attraction.

## 6 DISCUSSION

This paper has looked at the performance of a number of search techniques to solve a real world software engineering problem. The problem was the optimisation of the feature set used by a *k*-NN system to predict software project effort. Table 4 shows a summary of the results for these techniques.. The table shows accuracy of the best feature set found by each method (measured as the sum of the absolute residuals). It also shows the maximum and mean values for techniques where multiple results were produced. To provide an intuitive feel for the practical value of the prediction systems produced, an alternative measure of accuracy — mean absolute relative error — is also included[4]. Finally, as a means of indicating the efficiency of the techniques we provide the number of different feature sets evaluations required by each technique.

Table 4: Summary of results

| | All | Random | Hill | FSS |
|---|---|---|---|---|
| Evaluations | 1 | 4028 | 74433 | 243 |
| Min sum\|r\| | 88522 | 56988 | 29916 | 30202 |
| Max sum\|r\| | | 135586 | 61049 | |
| Mean sum\|r\| | | 91272 | 47803 | |
| % Error | 50.8 % | 35.6 % | 15.3 % | 13.1 % |

The simplest and quickest technique is to use all of the features, as this requires a single evaluation. However the accuracy results obtained are very poor. Prediction made using this system had an average error of 50.8%. This is intended as a base line against which we see all other techniques yield accuracy benefits.

The results show that a random search is clearly better than simply using all features. The best feature set found with this method resulted in a prediction system with an average error of 35.6%. Longer runs of random trials could bring this value down further but more intelligent search techniques would probably be preferred.

Hill-climbing found the best solution known to us (as measure by sum|r|[5]). This result was equivalent to an average error on predictions of about 15%. However, hill climbing is very computationally intensive. The table shows that a total of 74433 different feature sets were evaluated during the 113 hill climbs. This form of multiple-start hill climbing (like the random search) has no in-built end point. It can be run indefinitely but is likely to show diminishing returns as it converges on an exhaustive search.

Forward sequential selection yielded a result only fractionally worse than the best hill climbing result in terms of residuals and slightly better in terms of relative error (13.1 %). The major advantage of the FSS technique was only 243 evaluations were required to reach this result

Following this study the authors would make the following recommendations to anyone trying to optimise a feature subset for case-based prediction of software development effort.

1. If the feature set is small (<15-20 features) use an exhaustive search. This will always find the optimum feature subset.
2. If the feature set has more than 20 features an exhaustive search will not be computationally possible. Based on the work presented in this paper (albeit a single dataset) the authors would recommend trying FSS since it is significantly more efficient than a hill-climber and there was little difference in accuracy.

It is worth noting that the inherently noisy nature of such real world data means that there may be little scope for further improvement. Also, for our problem we are only seeking good engineering approximations. Thus we can view a solution as good enough. Any "real world" project effort prediction will be vulnerable to changes in environment, requirements, staff and so on, thus in practice ultra-high levels of accuracy are somewhat illusory. If readers are considering applying more complex algorithms to tackle this type of problem they should consider whether it would be worthwhile in the light of the relatively small improvement that might potentially be yielded.

As well as applying a set of search techniques to a particular software engineering problem, we have also attempted to describe the fitness landscape of the problem. Given that the fitness landscape is comprised of 43 binary dimensions describing the landscape is a difficult task. A number of techniques were employed to help describe and visualise the landscape. Firstly, the result from the random search were interpreted as samples from the fitness landscape and used to find how much of the landscape was at various heights. Secondly, hill climbing was used to look for the presence of structure in the landscape and to gauge the number and height of peaks in the landscape. Multi-dimensional scaling was used as a means of visualising the relative position of the various peaks found by hill climbing. The results from the MDS helped to identify global trends in the landscape. Localised exhaustive or random searches can also be used

[4] We did not use mean absolute relative error (sometimes referred to as MMRE in the software engineering literature) as our fitness function since it is asymmetric and heteroscedastic.

[5] From table 3 it can be seen that although hill climbing has a lower sum|r| than FSS, it has a higher % error. This rank reversal is due to different accuracy indicators measuring different aspects of error.

to further investigate areas of interest within the fitness landscape.

The result of the investigation of the fitness landscape for this problem showed it to be highly multi-modal. The landscape contained sharp peaks and troughs and rose towards a multi-peaked 'massif' that contained all of the better results found.

To conclude, we have described the successful use of search techniques on a real world software engineering problem. By means of the search for better feature subsets, prediction error has been reduced from 50.8 % to 13.1 %. We have also suggested techniques to help visualise fitness landscapes and used them on a real example. Lastly we have noted that applying search techniques to engineering problems is not necessarily the search for an optimum value. A good enough result reached in a timely fashion may be better than a prolonged search for an optimum result. The successful use of simple search techniques such as FSS and hill climbing suggest that researchers should try such methods before resorting to more complex techniques and that as a side effect these techniques also reveal useful information about the nature of the fitness landscape.

**Acknowledgments**

The authors are indebted to STTF Ltd for making the Finnish data set available.

**References**

Aha, D. W. and R. L. Bankert (1996). A comparative evaluation of sequential feature selection algorithms. *Artificial Intelligence and Statistics V*. D. Fisher and J.-H. Lenz. New York, Springer-Verlag.

Boehm, B. W. (1984). "Software engineering economics." *IEEE Transactions on Software Engineering* **10**(1): 4-21.

Burgess, C. J. and M. Lefley (2001). "Can genetic programming improve software effort estimation? A comparative evaluation." *Information & Software Technology* **43**(14): 863-873.

Crisan, C. and H. Muhlenbein (1998). The frequency assignment problem: A look at the performance of evolutionary search. *Artificial Evolution: Lecture Notes in Computer Science, Vol. 1363:***:** 263-273.

Debuse, J. C. W. and V. J. Rayward-Smith (1997). "Feature subset selection within a simulated annealing data mining algorithm." *J. of Intelligent Information Systems* **9**: 57-81.

Dolado, J. J. (2001). "On the problem of the software cost function." *Information & Software Technology* **43**(1): 61-72.

Finnie, G. R., G. E. Wittig and J.-M. Desharnais (1997). "A comparison of software effort estimation techniques using function points with neural networks, case based reasoning and regression models." *J. of Systems Software* **39**: 281-289.

Juels, A. and M. Wattenberg (1994). Stochastic hillclimbing as a baseline method for evaluating genetic algorithms, Technical Report, Univ. of California at Berkeley.

Klein, G. (1998). *Sources of Power : How People Make Decisions*. Cambridge, Ma, MIT Press.

Kohavi, R. and G. H. John (1997). "Wrappers for feature selection for machine learning." *Artificial Intelligence* **97**: 273-324.

Kok, P., B. A. Kitchenham and J. Kirakowski (1990). The MERMAID approach to software cost estimation. *Esprit Technical Week*.

Kolodner, J. L. (1993). *Case-Based Reasoning*, Morgan-Kaufmann.

Mair, C., G. Kadoda, M. Lefley, K. Phalp, C. Schofield, M. Shepperd and S. Webster (2000). "An investigation of machine learning based prediction systems." *J. of Systems Software* **53**(1): pp23-29.

Prietula, M. J., S. S. Vincinanza and T. Mukhopadhyay (1996). "Software Effort Estimation With a Case-Based Reasoner." *J. Experimental & Theoretical Artificial Intelligence* **8**: 341 - 363.

Reeves, C. R. (2000). Fitness landscapes and evolutionary algorithms. *Artificial Evolution: Lecture Notes in Computer Science, Vol. 1829***:** 3-20.

Sharpe, O. (1998). Beyond NFL: A few tentative steps. *3rd Conf. on Genetic Programming (GP'98)*, Madison, Wisconsin., Morgan Kaufman.

Shepperd, M. J. and C. Schofield (1997). "Estimating software project effort using analogies." *IEEE Transactions on Software Engineering* **23**(11): 736-743.

Shepperd, M. J., C. Schofield and B. A. Kitchenham (1996). Effort estimation using analogy. *18th Intl. Conf. on Softw. Eng.*, Berlin, IEEE Computer Press.

Skalak, D. B. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithms. *11th Intl. Machine Learning Conf. (ICML-94)*, Morgan Kauffmann.

Whitley, D., J. R. Beveridge, C. Guerra-Salcedo and C. Graves (1997). Messy genetic algorithms for subset feature selection. *International Conference on Genetic Algorithms, ICGA-97.*

Wolpert, D. H. and W. G. Macready (1997). "No Free Lunch Theorems for Search." *IEEE Transactions on Evolutionary Computation* **1**(1): 67-82.