A Knowledge Discovery Approach to Understanding Relationships between Scheduling Problem Structure and Heuristic Performance

Kate A. Smith-Miles[1], Ross J.W. James[2], John W. Giffin[2], and Yiqing Tu[1]

[1] School of Mathematical Sciences, Monash University, Melbourne, Australia
{kate.smith-miles,ytu}@sci.monash.edu.au
[2] Department of Management, University of Canterbury, Christchurch, New Zealand
{ross.james,john.giffin}@canterbury.ac.nz

Abstract. Using a knowledge discovery approach, we seek insights into the relationships between problem structure and the effectiveness of scheduling heuristics. A large collection of 75,000 instances of the single machine early/tardy scheduling problem is generated, characterized by six features, and used to explore the performance of two common scheduling heuristics. The best heuristic is selected using rules from a decision tree with accuracy exceeding 97%. A self-organizing map is used to visualize the feature space and generate insights into heuristic performance. This paper argues for such a knowledge discovery approach to be applied to other optimization problems, to contribute to automation of algorithm selection as well as insightful algorithm design.

Keywords: Scheduling, heuristics, algorithm selection, self-organizing map, performance prediction, knowledge discovery.

1 Introduction

It has long been appreciated that knowledge of a problem's structure and instance characteristics can assist in the selection of the most suitable algorithm or heuristic [1, 2]. The No Free Lunch theorem [3] warns us against expecting a single algorithm to perform well on all classes of problems, regardless of their structure and characteristics. Instead we are likely to achieve better results, on average, across many different classes of problem, if we tailor the selection of an algorithm to the characteristics of the problem instance. This approach has been well illustrated by the recent success of the algorithm portfolio approach on the 2007 SAT competition [4].

As early as 1976, Rice [1] proposed a framework for the algorithm selection problem. There are four essential components of the model:

- the problem space P represents the set of instances of a problem class;
- the feature space F contains measurable characteristics of the instances generated by a computational feature extraction process applied to P;

T. Stützle (Ed.): LION 3, LNCS 5851, pp. 89–103, 2009.

- the algorithm space A is the set of all considered algorithms for tackling the problem;
- the performance space Y represents the mapping of each algorithm to a set of performance metrics.

In addition, we need to find a mechanism for generating the mapping from feature space to algorithm space. The Algorithm Selection Problem can be formally stated as: For a given problem instance $x \in P$, with features $f(x) \in F$, find the selection mapping $S(f(x))$ into algorithm space A, such that the selected algorithm $\alpha \in A$ maximizes the performance mapping $y(\alpha(x)) \in Y$. The collection of data describing $\{P, A, Y, F\}$ is known as the meta-data.

There have been many studies in the broad area of algorithm performance prediction, which is strongly related to algorithm selection in the sense that supervised learning or regression models are used to predict the performance ranking of a set of algorithms, given a set of features of the instances. In the AI community, most of the relevant studies have focused on constraint satisfaction problems like SAT, QBF or QWH (P, in Rice's notation), using solvers like DPLL, CPLEX or heuristics (A), and building a regression model (S) to use the features of the problem structure (F) to predict the run-time performance of the algorithms (Y). Studies of this nature include Leyton-Brown and co-authors [5-7], and the earlier work of Horvitz et al. [8] that used a Bayesian approach to learn the mapping *S*. In recent years these studies have extended into the algorithm portfolio approach [4] and a focus on dynamic selection of algorithm components in real-time [9, 10].

In the machine learning community, research in the field of meta-learning has focused on classification problems (P), solved using typical machine learning classifiers such as decision trees, neural networks, or support vector machines (A), where supervised learning methods (S) have been used to learn the relationship between the statistical and information theoretic measures of the classification instance (F) and the classification accuracy (Y). The term meta-learning [11] is used since the aim is to learn about learning algorithm performance. Studies of this nature include [12-14] to name only three of the many papers published over the last 15 years.

In the operations research community, particularly in the area of constrained optimization, researchers appear to have made fewer developments, despite recent calls for developing greater insights into algorithm performance by studying search space or problem instance characteristics. According to Stützle and Fernandes [15], "currently there is still a strong lack of understanding of how exactly the relative performance of different meta-heuristics depends on instance characteristics".

Within the scheduling community, some researchers have been influenced by the directions set by the AI community when solving constraint satisfaction problems. The dynamic selection of scheduling algorithms based on simple low-level knowledge, such as the rate of improvement of an algorithm at the time of dynamic selection, has been applied successfully [16]. Other earlier approaches have focused on integrating multiple heuristics to boost scheduling performance in flexible manufacturing systems [17].

For many NP-hard optimization problems, such as scheduling, there is a great deal we can discover about problem structure which could be used to create a rich set of features. Landscape analysis (see [18-20]) is one framework for measuring the characteristics of problems and instances, and there have been many relevant developments in this direction, but the dependence of algorithm performance on these measures is yet to be completely determined [20].

Clearly, Rice's framework is applicable to a wide variety of problem domains. A recent survey paper [21] has discussed the developments in algorithm selection across a variety of disciplines, using Rice's notation as a unifying framework, through which ideas for cross-fertilization can be explored. Beyond the goal of performance prediction also lies the ideal of greater insight into algorithm performance, and very few studies have focused on methodologies for acquiring such insights. Instead the focus has been on selecting the best algorithm for a given instance, without consideration for what implications this has for algorithm design or insight into algorithm behaviour. This paper demonstrates that knowledge discovery processes can be applied to a rich set of meta-data to develop, not just performance predictions, but visual explorations of the meta-data and learned rules, with the goal of learning more about the dependencies of algorithm performance on problem structure and data characteristics.

In this paper we present a methodology encompassing both supervised and unsupervised knowledge discovery processes on a large collection of meta-data to explore the problem structure and its impact on algorithm suitability. The problem considered is the early/tardy scheduling problem, described in section 2. The methodology and meta-data is described in section 3, comprising 75,000 instances (P) across a set of 6 features (F). We compare the performance of two common heuristics (A), and measure which heuristic produces the lowest cost solution (Y). The mapping *S* is learned from the meta-data {P, A, Y, F} using knowledge derived from self-organizing maps, and compared to the knowledge generated and accuracy of the performance predictions using the supervised learning methods of neural networks and decision trees. Section 4 presents the results of this methodology, including decision tree rules and visualizations of the feature space, and conclusions are drawn in Section 5.

2 The Early/Tardy Machine Scheduling Problem

Research into the various types of E/T scheduling problems was motivated, in part, by the introduction of Just-in-Time production, which required delivery of goods to be made at the time required. Both early and late production are discouraged, as early production incurs holding costs, and late delivery means a loss of customer goodwill. A summary of the various E/T problems was presented in [22] which showed the NP-completeness of the problem.

2.1 Formulation

The E/T scheduling problem we consider is the single machine, distinct due date, early/tardy scheduling problem where each job has an earliness and tardiness penalty

and due date. Once a job is dispatched on the machine, it runs to completion with no interruptions permitted. The objective is to minimise the total penalty produced by the schedule. The objective of this problem can be defined as follows:

$$\min \sum_{i=1}^{n} \left(\alpha_i \left| d_i - c_i \right|^+ + \beta_i \left| c_i - d_i \right|^+ \right) . \quad (1)$$

where n is the number of jobs to be scheduled, c_i is the completion time of job i, d_i is the due date of job i, α_i is the penalty per unit of time when job i is produced early, β_i is the penalty per unit of time when job i is produced tardily, and $|x|^+ = x$ if $x > 0$, or 0 otherwise. We also define p_i as the processing time of job i. The decision variable is the completion time c_i of job i, derived from the optimal starting sequence of jobs and their processing times.

The objective of this problem is to schedule the jobs as closely as possible to their due dates; however the difficulty in formulating a schedule occurs when it is not possible to schedule all jobs on their due dates, which also causes difficulties in managing the many tradeoffs between jobs competing for processing at a given time [23]. Two of the simplest and most commonly used dispatching heuristics for the E/T scheduling problem are the Earliest Due Date and Shortest Processing Time heuristics.

2.2 Earliest Due Date (EDD) Heuristic

The EDD heuristic orders the jobs based on the date the job is due to be delivered to the customer. The jobs with the earliest due date are scheduled first, while the jobs with the latest due date are scheduled last. After the sequence is determined, the completion times of each job are then calculated using the optimal idle time insertion algorithm of Fry, Armstrong and Blackstone [24]. For single machine problems the EDD is known to be the best rule to minimise the maximum lateness, and therefore tardiness, and also the lateness variance [25]. The EDD has the potential to produce optimal solutions to this problem, for example when there are few jobs and the due dates are widely spread so that all jobs may be scheduled on their due date without interfering with any other jobs. As there are no earliness or tardiness penalties, the objective value will be 0 and therefore optimal.

2.3 Shortest Processing Time (SPT) Heuristic

The SPT heuristic orders the jobs based on their processing time. The jobs with the smallest processing time are scheduled first, while the jobs with the largest processing time are scheduled last; this is the fastest way to get most of the jobs completed quickly. Once the SPT sequence has been determined, the job completion times are calculated using the optimal idle time insertion algorithm [24]. The SPT heuristic has been referred to as “the world champion” scheduling heuristic [26], as it produces schedules for single machine problems that are good at minimising the average time of jobs in a system, minimising the average number of jobs in the system and minimising the average job lateness [25]. When the tardiness penalties for the jobs are similar and the due dates are such that the majority of jobs are going to be late, SPT is

likely to produce a very good schedule for the E/T scheduling problem, as it gets the jobs completed as quickly as possible. The "weighted" version of the SPT heuristic, where the order is determined by p_i/β_i, is used in part by many E/T heuristics, as this order can be proven to be optimal for parts of a given schedule.

2.4 Discussion

Due to the myopic nature of the EDD and SPT heuristics, neither heuristic is going to consistently produce high quality solutions to the general E/T scheduling problem. Both of these simple heuristics generate solutions very quickly however and therefore it is possible to carry out a large sample of problems in order to demonstrate whether or not the approach proposed here is useful for exploring the relative performance of two heuristics (or algorithms) and is able to predict the superiority of one heuristic over another for a given instance.

3 Methodology

In this section we describe the meta-data for the E/T scheduling problem in the form of {P, A, Y, F}. We also provide a description of the machine learning algorithms applied to the meta-data to produce rules and visualizations of the meta-data.

3.1 Meta-data for the E/T Scheduling Problem

The most critical part of the proposed methodology is identification of suitable features of the problem instances that reflect the structure of the problem and the characteristics of the instances that might explain algorithm performance. Generally there are two main approaches to characterizing the instances: the first is to identify problem dependent features based on domain knowledge of what makes a particular instance challenging or easy to solve; the second is a more general set of features derived from landscape analysis [27]. Related to the latter is the approach known in the meta-learning community as landmarking [28], whereby an instance is characterized by the performance of simple algorithms which serve as a proxy for more complicated (and computationally expensive) features. Often a dual approach makes sense, particularly if the feature set derived from problem dependent domain knowledge is not rich, and supplementation from landscape analysis can assist in the characterization of the instances. In the case of the generalised single machine E/T scheduling problem however, there is sufficient differentiation power in a small collection of problem dependent features that we can derive rules explaining the different performance of the two common heuristics. Extending this work to include a greater set of algorithms (A) may justify the need to explore landscape analysis tools to derive greater characterisation of the instances.

In this paper, each n-job instance of the generalised single machine E/T scheduling problem has been characterized by the following features.

1. Number of jobs to be scheduled in the instance, n
2. Mean Processing Time $\overline{p}$: The mean processing time of all jobs in an instance

3. Processing Time Range p_σ: The range (max – min) of the processing times of all jobs in the instance
4. Tardiness Factor τ: Defines where the average due date occurs relative to, and as a fraction of the total processing time of all jobs in the instance. A positive tardiness factor indicates the proportion of the schedule that is expected to be tardy, while a negative tardiness factor indicates the amount of idle time that is expected in the schedule as a proportion of the total processing time of all jobs in the sequence. Mathematically the tardiness factor was defined by Baker and Martin [29] as: $\tau = 1 - \frac{\sum d_i}{n \sum p_i}$
5. Due Date Range factor D_σ: Determines the spread of the due dates from the average due date for all jobs in the instance, normalized by the size of processing times. It is defined as $D_\sigma = \frac{(b-a)}{\sum p_i}$, where b is the maximum due date in the instance and a is the minimum due date in the instance, and is a fraction of the total processing time needed for the instance
6. Penalty Ratio ρ: The maximum over all jobs in the instance of the ratio of the tardy penalty to the early penalty.

Any instance of the problem, whether contained in the meta-data set or generated at a future time, can be characterized by this set of six features. It is not the only possible set of features but, as the results presented later in this paper demonstrate, it captures the essential variation in instances needed to accurately predict heuristic performance. Since we are comparing the performance of only two heuristics, we can create a single binary variable to indicate which heuristic performs best for a given problem instance. Let $Y_i=1$ if EDD is the best performing heuristic (lowest objective function) compared to SPT for problem instance i, and $Y_i=0$ otherwise (SPT is best). The meta-data then comprises the set of six-feature vectors and heuristic performance measure (Y), for a large number of instances, and the task is to learn the relationship between features and heuristic performance.

In order to provide a large and representative sample of instances for the meta-data, an instance generator was created to span a range of values for each feature. Problem instances were then generated for all combinations of parameter values. Note that these parameters are targets for the instances and the random generation process may create slight variation from these target values. The parameter settings used were:

- problem size (number of jobs, n): 20-100 with increments of 20 (5 levels)
- target processing time range p_σ: processing times randomly generated with a range ($p_{max} - p_{min}$) of 2-10 with increments of 2 (5 levels).
- target due date range factor D_σ as a proportion of total processing time: ranges from 0.2 to 1 in increments of 0.2 (5 levels)
- target tardiness factor τ as a proportion of total processing time: ranges from 0 (all jobs should complete on time) to 1 (all jobs should be late) in increments of 0.2 (6 levels)
- penalty ratio ρ: 1-10 with increments of 1 (10 levels)

From these parameters the following instance data can be generated:

- processing times p_i : calculated within the processing time range.
- processing time means $\overline{p}$: calculated from the randomly generated p_i
- due dates d_i : due dates randomly generated within the due date range and offset by the tardiness factor.

To calculate the actual p_σ, actual D_σ and actual τ we use the actual p_i, d_i of the problem rather than the target values. Ten instances using each parameter setting were then generated, giving a total of 5 (size levels) x 5 (processing time range levels) x 6 (tardiness factor levels) x 5 (due date range factor levels) x 10 (penalty ratio levels) x 10 (instances) = 75,000 instances.

A correlation analysis between the instance features and the Y values across all 75,000 instances reveals that the only instance features that appear to correlate (linearly) with heuristic performance are the tardiness factor (correlation = -0.59) and due date range factor (correlation = 0.44). None of the other instance features appear to have a linear relationship with algorithm performance. Clearly due date range factor and tardiness factor correlate somewhat with the heuristic performances, but it is not clear if these are non-linear relationships, and if either of these features with combinations of the others can be used to seek greater insights into the conditions under which one heuristic is expected to outperform the other.

Using Rice's notation, our meta-data can thus be described as:

- P = 75,000 E/T scheduling instances
- A = 2 heuristics (EDD and SPT)
- Y = binary decision variable indicating if EDD is best compared to SPT (based on objective function which minimizes weighted deviation from due dates)
- F = 6 instance features (problem size, processing time mean, processing time range, due date range factor, tardiness factor and penalty ratio).

Additional features could undoubtedly be derived either from problem dependent domain knowledge, or using problem independent approaches such as landscape analysis [28], landmarking [28], or hyper-heuristics [30]. For now though, we seek to learn the relationships that might exist in this meta-data.

3.2 Knowledge Discovery on the Meta-data

When exploring any data-set to discover knowledge, there are two broad approaches. The first is supervised learning (aka directed knowledge discovery) which uses training examples – sets of independent variables (inputs) and dependent variables (outputs) - to learn a predictive model which is then generalized for new examples to predict the dependent variable (output) based only on the independent variables (inputs). This approach is useful for building models to predict which algorithm or heuristic is likely to perform best given only the feature vector as inputs. The second broad approach to knowledge discovery is unsupervised learning (aka undirected knowledge discovery) which uses only the independent variables to find similarities and differences between the structure of the examples, from which we may then be

able to infer relationships between these structures and the dependent variables. This second approach is useful for our goal of seeking greater insight into *why* certain heuristics might be better suited to certain instances and, rather than just building predictive models of heuristic performance.

In this section we briefly summarise the machine learning methods we have used for knowledge discovery on the meta-data.

Neural Networks

As a supervised learning method [31], neural networks can be used to learn to predict which heuristic is likely to return the smallest objective function value. A training dataset is randomly extracted (80% of the 75,000 instances) and used to build a non-linear model of the relationships between the input set (features F) and the output (metric Y). Once the model has been learned, its generalisation on an unseen test set (the remaining 20% of the instances) is evaluated and recorded as a percentage accuracy in predicting the superior heuristic. This process is repeated ten times for different random extractions of the training and test sets, to ensure that the results were not simply an artifact of the random number seed. This process is known as ten-fold cross validation, and the reported results show the average accuracy on the test set across these ten folds.

For our experimental results, the neural network implementation within the Weka machine learning platform [32] was used with 6 input nodes, 4 hidden nodes, and 2 output nodes utilising binary encoding. The transfer function for the hidden nodes was a sigmoidal function, and the neural network was trained with the backpropagation (BP) learning algorithm with learning rate = 0.3, momentum = 0.2. The BP algorithm stops when the number of epochs (complete presentation of all examples) reaches a maximum training time of 500 epochs or the error on the test set does not decrease after a threshold of 20 epochs.

Decision Tree

A decision tree [33] is also a supervised learning method, which uses the training data to successively partition the data, based on one feature at a time, into classes. The goal is to find features on which to split the data so that the class membership at lower leaves of the resulting tree is as "pure" as possible. In other words, we strive for leaves that are comprised almost entirely of one class only. The rules describing each class can then be read up the tree by noting the features and their splitting points. Ten-fold cross validation is also used in our experiments to ensure the generalisation of the rules.

The J4.8 decision tree algorithm, implemented in Weka [32], was used for our experimental results, with a minimum leaf size of 500 instances. The generated decision tree is pruned using subtree raising with confidence factor = 0.25.

Self-Organizing Maps

Self-Organizing Maps (SOMs) are the most well-known unsupervised neural network approach to clustering. Their advantage over traditional clustering techniques such as the k-means algorithm lies in the improved visualization capabilities resulting from the two-dimensional map of the clusters. Often patterns in a high dimensional input space have a very complicated structure, but this structure is made more transparent

and simple when they are clustered in a lower dimensional feature space. Kohonen [34] developed SOMs as a way of automatically detecting strong features in large data sets. SOMs find a mapping from the high dimensional input space to low dimensional feature space, so any clusters that form become visible in this reduced dimensionality. The architecture of the SOM is an multi-dimensional input vector connected via weights to a 2-dimensional array of neurons. When an input pattern is presented to the SOM, each neuron calculates how similar the input is to its weights. The neuron whose weights are most similar (minimal distance in input space) is declared the winner of the competition for the input pattern, and the weights of the winning neuron, and its neighbours, are strengthened to reflect the outcome. The final set of weights embeds the location of cluster centres, and is used to recognize to which cluster a new input vector is closest.

For our experiments we randomly split the 75000 instances into training data (50000 instances) and test data (25000 instances). We use the Viscovery SOMine software (www.eudaptics.com) to cluster the instances based only on the six features as inputs. A map of 2000 nodes is trained for 41 cycles, with the neighbourhood size diminishing linearly at each cycle. After the clustering of the training instances, the distribution of Y values is examined within each cluster, and knowledge about the relationships between instance structure and heuristic performance is inferred and evaluated on the test data.

4 Experimental Evaluation

4.1 Supervised Learning Results

Both the neural network and decision tree algorithms were able to learn the relationships in the meta-data, achieving greater than 97% accuracy (on ten-fold cross-validation test sets) in predicting which of the two heuristics would be superior based only on the six features (inputs). These approaches have an overall classification accuracy of 97.34% for the neural network and 97.13% for the decision tree. While the neural network can be expected to learn the relationships in the data more powerfully, due to its nonlinearity, its limitation is the lack of insight and explanation of those relationships. The decision tree's advantage is that it produces a clear set of rules, which can be explored to see if any insights can be gleaned. The decision tree rules are presented in the form of pseudo-code in Figure 1, with the fraction in brackets showing the number of instances that satisfied both the condition and the consequence (decision) in the numerator, divided by the total number of instances that satisfied the condition in the denominator. This proportion is equivalent to the accuracy of the individual rule.

The results allow us to state a few rules with exceptionally high accuracy:

1) If the majority of jobs are expected to be scheduled early (tardiness factor <= 0.5) then EDD is best in 99.8% of instances
2) If the majority of the jobs are expected to be scheduled late (tardiness factor > 0.7) then SPT is best in 99.5% of instances

3) If slightly more than half of the jobs are expected to be late (tardiness factor between 0.5 and 0.7) then as long as the tardiness penalty ratio is no more than 3 times larger than the earliness penalty ($\rho \leq 3$), then EDD is best in 98.9% of the instances with a due date range factor greater than 0.2.

The first two rules are intuitive and can be justified from what we know about the heuristics - EDD is able to minimise lateness deviations when the majority of jobs can be scheduled before their due date, and SPT is able to minimise the time of jobs in the system and hence tardiness when the majority of jobs are going to be late [25]. The third rule reveals the kind of knowledge that can be discovered by adopting a machine learning approach to the meta-data. Of course other rules can also be explored from Figure 1, with less confidence due to the lower accuracy, but they may still provide the basis for gaining insight into the conditions under which different algorithms can be expected to perform well.

```
If (τ <= 0.7) Then
   If (τ <= 0.5) Then EDD best (44889/45000 = 99.8%)
   If (τ > 0.5) Then If (D_σ <= 0.2) Then If (ρ <= 3) Then EDD best (615/750 = 82.0%)
                                                 Else SPT best (1483/1750 = 84.7%)
                                    Else If (ρ <= 3) Then EDD best (5190/5250 = 98.9%)
                                       Else If (τ <= 0.6) Then EDD best (8320/8750 = 95.1%)
                                          Else If (p̄ <= 2) Then EDD best (556/700 = 79.4%)
                                             Else If (n <= 60) Then SPT best (1150/1680 = 68.4%)
                                                Else EDD best (728/1120 = 65%)
Else SPT best (9950/10000 = 99.5%)
```

Fig. 1. Pseudo-code for the decision tree rule system, showing the accuracy of each rule

4.2 Unsupervised Learning Results

After training the SOM, the converged map shows 5 clusters, each of which contains similar instances defined by Euclidean distance in feature space. Essentially, the six-dimensional input vectors have been projected onto a two-dimensional plane, with topology-preserving properties. The clusters can be inspected to understand what the instances within each cluster have in common. The statistical properties of the 5 clusters can be seen in Table 1. The distribution of the input variables (features), and additional variables including the performance of the heuristics, can be visually explored using the maps shown in Figure 2. A k-nearest neighbour algorithm (with k=7) is used to distribute additional data instances (from the test set) or extra variables (Y values) across the map.

Looking first at the bottom row of Table 1, it is clear that clusters 1, 2 and 3 contain instances that are best solved using EDD (Y=1). These clusters are shown visually in the bottom half of the 2-d self-organizing map (see Figure 2a for cluster boundaries, and Figure 2b to see the distribution of Y across the clusters). These three clusters of instances account for 70.2% of the 75,000 instances (see Table 1). The

remaining clusters 4 and 5 are best solved, on average, by SPT. The maps shown in Figure 2c – 2h enable us to develop a quick visual understanding of how the clusters differ from each other, and to see which features are prominent in defining instance structure.

Table 1. Cluster statistics for training data (test data in brackets) - mean values of input variables, and heuristic performance variable Y, as well as cluster size

	Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**	**All Data**
instances	17117 (8483)	10454 (5236)	7428 (3832)	8100 (4000)	6901 (3449)	50000 (25000)
instances (%)	34.23 (33.93)	20.91 (20.94)	14.86 (15.33)	16.2 (16.0)	13.8 (13.8)	100 (100)
n	60.65 (61.03)	59.73 (59.73)	58.73 (58.96)	57.8 (57.7)	63.39 (61.56)	60.0 (59.97)
$\bar{p}$	2.77 (2.76)	5.24 (5.22)	5.08 (5.07)	5.12 (5.11)	2.70 (2.71)	4.0 (3.99)
p_σ	3.54 (3.52)	8.48 (8.45)	8.16 (8.13)	8.24 (8.21)	3.41 (3.41)	6.0 (5.99)
τ	0.31 (0.31)	0.36 (0.35)	0.21 (0.21)	0.72 (0.73)	0.72 (0.72)	0.43 (0.42)
$D\sigma$	0.70 (0.70)	0.88 (0.88)	0.38 (0.38)	0.40 (0.39)	0.40 (0.40)	0.6 (0.59)
ρ	5.89 (5.88)	4.93 (4.99)	5.37 (5.41)	5.24 (5.19)	5.87 (5.72)	5.5 (5.49)
Y	1.00 (0.99)	1.00 (1.00)	0.99 (0.99)	0.36 (0.36)	0.42 (0.41)	0.82 (0.82)

By inspecting the maps shown in Figure 2, and the cluster statistics in Table 1, we can draw some conclusions about whether the variables in each cluster are above or below average (compared to the entire dataset), and look for correlations with the heuristic performance metric Y. For instance, cluster 2 is characterized by instances with above average values of processing time mean and range, below average tardiness factor, and above average due date range factor. The EDD heuristic is always best under these conditions (Y=1). Instances in cluster 3 are almost identical, except that the due date range factor tends to be below average. Since cluster 3 instances are also best solved by the EDD heuristic, one could hypothesize that the due date range factor does not have much influence in predicting heuristic performance. An inspection of the maps, however, shows this is not the case.

The distribution of Y across the map (Figure 2b) shows a clear divide between the clusters containing instances best solved using EDD (bottom half) and the clusters containing instances best solved using SPT (top half). Inspecting the distribution of features across this divide leads to a simple observation that, if the tardiness factor τ is below average (around 0.5 represented by white to mid-grey in Figure 2f), then EDD will be best. But there are small islands of high Y values in clusters 4 and 5 that overlay nicely with the medium grey values of due date range factor. So we can observe another rule that EDD will also be best if the tardiness factor is above average and the due date range factor is above average. Also of interest, from these maps we can see that problem size and the penalty ratio do not influence the relative

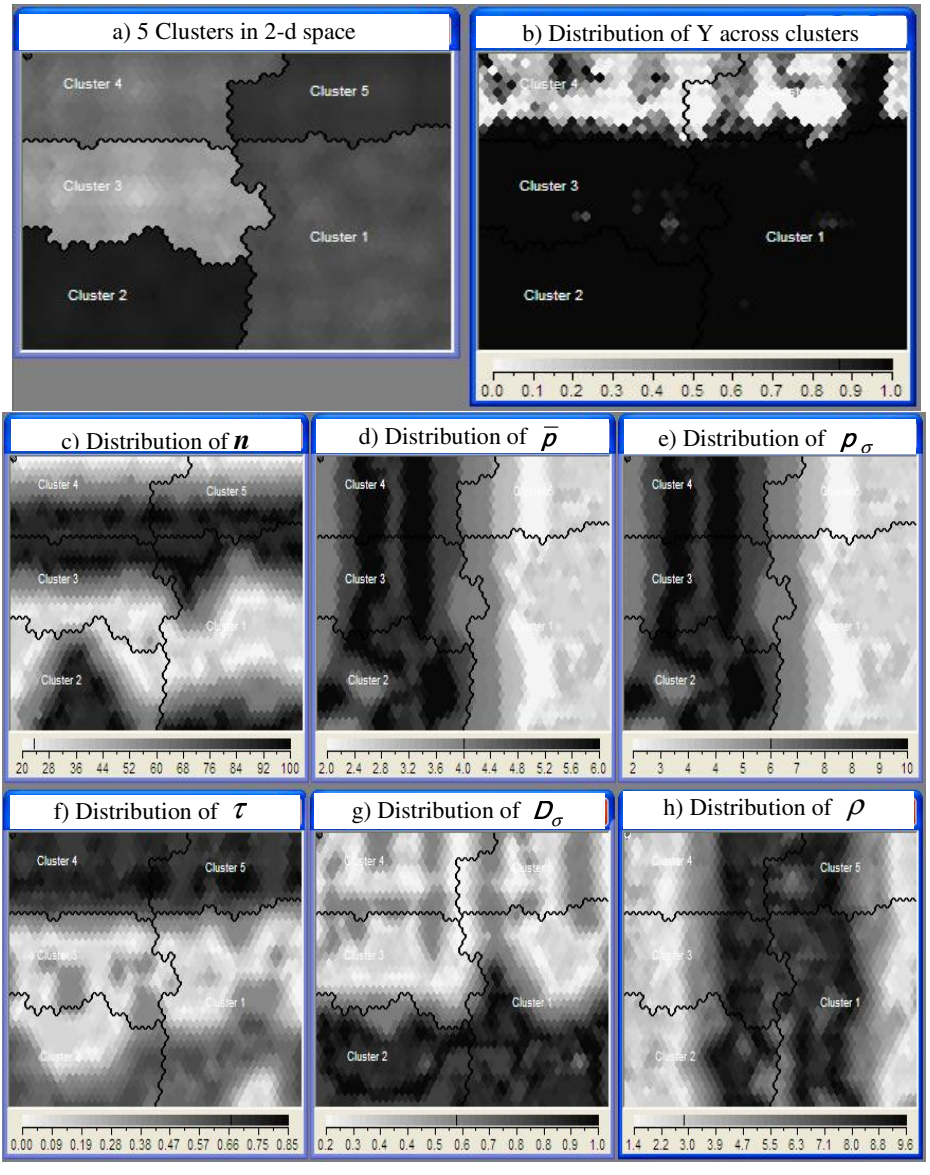

Fig. 2. Self-Organizing Map showing 5 clusters (fig. 2a), the heuristic performance variable Y (fig 2b), and the distribution of six features across the clusters (fig 2c – fig 2h). The grey scale shows a minimum value as white, and maximum value as black.

performance of these heuristics. As neither heuristic considers the penalty ratio (it is used within the optimal idle time insertion algorithm [24], common to both heuristics, but not used by the EDD or SPT heuristics themselves), its not being a factor in the clusters is not surprising.

Within Viscovery SOMine, specific regions of the map can be selected, and used as the basis of a classification. In other words, we can define regions and islands to be predictive of one heuristic excelling based on the training data (50,000 instances). We can then test the generalization of the predictive model using the remaining 25,000 instances as a test set, and applying the k-nearest neighbour algorithm to determine instances that belong to the selected region. We select the dark-grey to black regions of the Y map in Figure 2b, and declare that any test instances falling in the selected area are classified as Y=1, while any instances falling elsewhere in the map are classified as Y=0. The resulting accuracy on the test set is 95% in predicting which heuristic will perform better. The self-organizing map has proven to be useful for both visualization of feature space and predictive modeling of heuristic performance, although the accuracy is not quite as high as the supervised learning approaches.

5 Conclusions and Future Research

In this paper we have illustrated how the concepts of Rice's Algorithm Selection Problem can be extended within a knowledge discovery framework, and applied to the domain of optimization in order that we might gain to insights into optimization algorithm performance. This paper represents one of the first attempts to apply this approach to understand more about optimisation algorithm performance. While only two very simple heuristics have been used to illustrate the approach, we expect full generalization of the methodology to consider a broader range of complex heuristics and meta-heuristics. A large meta-data set comprising 75,000 instances of the E/T scheduling problem has been used to explore what can be learned about the relationships between the features of the problem instances and the performance of heuristics. Both supervised and unsupervised learning approaches have been presented, each with their own advantages and disadvantages made clear by the empirical results. The neural network obtained the highest accuracy for performance prediction, but its weakness is the lack of explanation or interpretability of the model. Our goal is not merely performance prediction, but to gain insights into the characteristics of instances that make solution by one heuristic superior than another. Decision trees are also a supervised learning method, and the rules produced demonstrate the potential to obtain both accurate performance predictions and some insights. Finally, the self-organizing map demonstrated its benefits for visualization of the meta-data and relationships therein.

One of the most important considerations for this approach to be successful for any arbitrary optimization problem is the choice of features used to characterize the instances. These features need to be carefully chosen in such a way that they can characterize instance and problem structure as well as differentiate algorithm performance.

There is little that will be learned via a knowledge discovery process if the features selected to characterize the instances do not have any differentiation power. The result will be supervised learning models of algorithm performance that predict average behaviour with accuracy measures no better than the default accuracies one could obtain from using a naïve model. Likewise, the resulting self-organizing map would show no discernible difference between the clusters when superimposing Y values (unlike in Figure 2b where we obtain a clear difference between the top and bottom halves of the map). Thus the success of any knowledge discovery process depends on

the quality of the data, and in this case, the meta-data must use features that serve the purpose of differentiating algorithm performance. In this paper we have used a small set of problem-dependent features, related to the E/T Scheduling Problem, which would be of no use to any other optimization problem. For other optimization problem like graph colouring or the Travelling Salesman Problem, recent developments in phase transition analysis (e.g. [35]) could form the foundation of the development of useful features. Landscape analysis [20, 27] provides a more general (problem independent) set of features, as do ideas from landmarking [28] and hyper-heuristics [30]. It is natural to expect that the best results will be obtained from a combination of generic and problem dependent features, and this will be the focus of our future research. In addition, we plan to extend the approach to consider the performance of a wider variety of algorithms, especially meta-heuristics, where we will also be gathering meta-data related to the features of the meta-heuristics themselves (e.g. hill-climbing capability, tabu list, annealing mechanism, population-based search, etc.). This will help to close the loop to ensure that any insights derived from such an approach are able to provide inputs into the design of new hybrid algorithms that adapt the components of the meta-heuristic according to the instance features – an extension of the highly successful algorithm portfolio approach [4].

References

1. Rice, J.R.: The Algorithm Selection Problem. Adv. Comp. 15, 65–118 (1976)
2. Watson, J.P., Barbulescu, L., Howe, A.E., Whitley, L.D.: Algorithm Performance and Problem Structure for Flow-shop Scheduling. In: Proc. AAAI Conf. on Artificial Intelligence, pp. 688–694 (1999)
3. Wolpert, D.H., Macready, W.G.: No Free Lunch Theorems for Optimization. IEEE T. Evolut. Comput. 1, 67 (1997)
4. Xu, L., Hutter, F., Hoos, H., Leyton-Brown, K.: Satzilla-07: The Design and Analysis of An Algorithm Portfolio For SAT. In: Bessière, C. (ed.) CP 2007. LNCS, vol. 4741, pp. 712–727. Springer, Heidelberg (2007)
5. Leyton-Brown, K., Nudelman, E., Shoham, Y.: Learning the Empirical Hardness of Optimization Problems: The Case of Combinatorial Auctions. In: Van Hentenryck, P. (ed.) CP 2002. LNCS, vol. 2470, pp. 556–569. Springer, Heidelberg (2002)
6. Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., Shoham, Y.: A Portfolio Approach to Algorithm Selection. In: Proc. IJCAI, pp. 1542–1543 (2003)
7. Nudelman, E., Leyton-Brown, K., Hoos, H., Devkar, A., Shoham, Y.: Understanding Random SAT: Beyond the Clauses-To-Variables Ratio. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 438–452. Springer, Heidelberg (2004)
8. Horvitz, E., Ruan, Y., Gomes, C., Kautz, H., Selman, B., Chickering, M.: A Bayesian Approach to Tackling Hard Computational Problems. In: Proc. 17th Conf. on Uncertainty in Artificial Intelligence, pp. 235–244. Morgan Kaufmann, San Francisco (2001)
9. Samulowitz, H., Memisevic, R.: Learning to solve QBF. In: Proc. 22nd AAAI Conf. on Artificial Intelligence, pp. 255–260 (2007)
10. Streeter, M., Golovin, D., Smith, S.F.: Combining multiple heuristics online. In: Proc. 22nd AAAI Conf. on Artificial Intelligence, pp. 1197–1203 (2007)
11. Vilalta, R., Drissi, Y.: A Perspective View and Survey of Meta-Learning. Artif. Intell. Rev. 18, 77–95 (2002)
12. Michie, D., Spiegelhalter, D.J., Taylor, C.C. (eds.): Machine Learning, Neural and Statistical Classification. Ellis Horwood, New York (1994)

13. Brazdil, P., Soares, C., Costa, J.: Ranking Learning Algorithms: Using IBL and Meta-Learning on Accuracy and Time Results. Mach. Learn. 50, 251–277 (2003)
14. Ali, S., Smith, K.: On Learning Algorithm Selection for Classification. Appl. Soft Comp. 6, 119–138 (2006)
15. Stützle, T., Fernandes, S.: New Benchmark Instances for the QAP and the Experimental Analysis of Algorithms. In: Gottlieb, J., Raidl, G.R. (eds.) EvoCOP 2004. LNCS, vol. 3004, pp. 199–209. Springer, Heidelberg (2004)
16. Carchrae, T., Beck, J.C.: Applying Machine Learning to Low Knowledge Control of Optimization Algorithms. Comput. Intell. 21, 373–387 (2005)
17. Shaw, M.J., Park, S., Raman, N.: Intelligent Scheduling With Machine Learning Capabilities: The Induction of Scheduling Knowledge. IIE Trans. 24, 156–168 (1992)
18. Knowles, J.D., Corne, D.W.: Towards Landscape Analysis to Inform the Design of a Hybrid Local Search for the Multiobjective Quadratic Assignment Problem. In: Abraham, A., Ruiz-Del-Solar, J., Koppen, M. (eds.) Soft Computing Systems: Design, Management and Applications, pp. 271–279. IOS Press, Amsterdam (2002)
19. Merz, P.: Advanced Fitness Landscape Analysis and the Performance of Memetic Algorithms. Evol. Comp. 2, 303–325 (2004)
20. Watson, J., Beck, J.C., Howe, A.E., Whitley, L.D.: Problem Difficulty for Tabu Search in Job-Shop Scheduling. Artif. Intell. 143, 189–217 (2003)
21. Smith-Miles, K.A.: Cross-Disciplinary Perspectives on Meta-Learning For Algorithm Selection. ACM Computing Surveys (in press, 2009)
22. Baker, K.R., Scudder, G.D.: Sequencing With Earliness and Tardiness Penalties: A Review. Ops. Res. 38, 22–36 (1990)
23. James, R.J.W., Buchanan, J.T.: A Neighbourhood Scheme with a Compressed Solution Space for The Early/Tardy Scheduling Problem. Eur. J. Oper. Res. 102, 513–527 (1997)
24. Fry, T.D., Armstrong, R.D., Blackstone, J.H.: Minimizing Weighted Absolute Deviation in Single Machine Scheduling. IIE Transactions 19, 445–450 (1987)
25. Vollmann, T.E., Berry, W.L., Whybark, D.C., Jacobs, F.R.: Manufacturing Planning and Control for Supply Chain Management, 5th edn. McGraw Hill, New York (2005)
26. Krajewski, L.J., Ritzman, L.P.: Operations Management: Processes and Value Chains, 7th edn. Pearson Prentice Hall, New Jersey (2005)
27. Schiavinotto, T., Stützle, T.: A review of metrics on permutations for search landscape analysis. Comput. Oper. Res. 34, 3143–3153 (2007)
28. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.G.: Meta-Learning by Landmarking Various Learning Algorithms. In: Proc. ICML, pp. 743–750 (2000)
29. Baker, K.B., Martin, J.B.: An Experimental Comparison of Solution Algorithms for the Single Machine Tardiness Problem. Nav. Res. Log. 21, 187–199 (1974)
30. Burke, E., Hart, E., Kendall, G., Newall, J., Ross, P., Schulenburg, S.: Hyper-heuristics: An Emerging Direction in Modern Search Technology. In: Glover, F., Kochenberger, G. (eds.) Handbook of Meta-heuristics, pp. 457–474. Kluwer, Norwell (2002)
31. Smith, K.A.: Neural Networks for Prediction and Classification. In: Wang, J. (ed.) Encyclopaedia of Data Warehousing and Mining, vol. 2, pp. 865–869. Information Science Publishing, Hershey (2006)
32. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2005)
33. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
34. Kohonen, T.: Self-Organized Formation of Topologically Correct Feature Maps. Biol. Cyber. 43, 59–69 (1982)
35. Achlioptas, D., Naor, A., Peres, Y.: Rigorous Location of Phase Transitions in Hard Optimization Problems. Nature 435, 759–764 (2005)