<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper published in *Expert systems with applications*. This paper has been peer-reviewed but does not include the final publisher proof-corrections or journal pagination.

Citation for the original published paper (version of record):

Bandaru, S., Ng, A H., Deb, K. (2017)

Data mining methods for knowledge discovery in multi-objective optimization: Part B - New developments and applications.

Expert systems with applications, 70: 119-138

<https://doi.org/10.1016/j.eswa.2016.10.016>

Access to the published version may require subscription.

N.B. When citing this work, cite the original published paper.

This is an article distributed under the terms of Creative Commons Attribution Non-Commercial No Derivatives license: <https://creativecommons.org/licenses/by-nc-nd/4.0/>

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:his:diva-13266>

Data Mining Methods for Knowledge Discovery in Multi-Objective Optimization: Part B - New Developments and Applications

Sunith Bandaru^{a,*}, Amos H. C. Ng^a, Kalyanmoy Deb^b

^a*School of Engineering Science, University of Skövde, Skövde 541 28, Sweden*

^b*Department of Electrical and Computer Engineering, Michigan State University,
East Lansing, 428 S. Shaw Lane, 2120 EB, MI 48824, USA*

Abstract

The first part of this paper served as a comprehensive survey of data mining methods that have been used to extract knowledge from solutions generated during multi-objective optimization. The current paper addresses three major shortcomings of existing methods, namely, lack of interactiveness in the objective space, inability to handle discrete variables and inability to generate explicit knowledge. Four data mining methods are developed that can discover knowledge in the decision space and visualize it in the objective space. These methods are (i) sequential pattern mining, (ii) clustering-based classification trees, (iii) hybrid learning, and (iv) flexible pattern mining. Each method uses a unique learning strategy to generate explicit knowledge in the form of patterns, decision rules and unsupervised rules. The methods are also capable of taking the decision maker's preferences into account to generate knowledge unique to preferred regions of the objective space. Three realistic production systems involving different types of discrete variables are chosen as application studies. A multi-objective optimization problem is formulated for each system and solved using NSGA-II to generate the optimization datasets. Next, all four methods are applied to each dataset. In each application, the methods discover similar knowledge for specified regions of the objective space. Overall, the unsupervised rules generated by flexible pattern mining are found to be the most consistent, whereas the supervised rules from classification trees are the most sensitive to user-preferences.

Keywords: Data mining, knowledge discovery, multi-objective optimization, discrete variables, production systems, flexible pattern mining

1. Introduction

Multi-objective optimization (MOO) concerns the simultaneous minimization or maximization of two or more objective functions. When these objectives are conflicting with

*Corresponding author

Email addresses: sunith.bandaru@his.se (Sunith Bandaru), amos.ng@his.se (Amos H. C. Ng), kdeb@egr.msu.edu (Kalyanmoy Deb)

respect to each other, there exist multiple solutions to the MOO problem. A common example of conflicting objectives is cost versus performance. Reducing cost often leads to deterioration of performance, whereas improving performance requires additional cost. Thus, the minimization of cost and the maximization of performance cannot be simultaneously achieved, which results in a trade-off between the two. Formally, MOO problems are represented as,

$$\begin{aligned} &\text{Minimize} && \mathbf{F}(\mathbf{x}) = \{f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_m(\mathbf{x})\} \\ &\text{Subject to} && \mathbf{x} \in S \end{aligned} \quad (1)$$

where $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ are m (≥ 2) conflicting objectives that have to be simultaneously minimized¹ and the variable vector $\mathbf{x} = [x_1, x_2, \dots, x_n]$ belongs to the non-empty feasible region $S \subset \mathbb{R}^n$. The feasible region is formed by the constraints of the problem which include the bounds on the variables. A variable vector $\mathbf{x}_1$ is said to *dominate* $\mathbf{x}_2$ and is denoted as $\mathbf{x}_1 \prec \mathbf{x}_2$ if and only if the following conditions are satisfied:

1. $f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2) \forall i \in \{1, 2, \dots, m\}$,
2. and $\exists j \in \{1, 2, \dots, m\}$ such that $f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2)$.

If neither $\mathbf{x}_1 \prec \mathbf{x}_2$ nor $\mathbf{x}_2 \prec \mathbf{x}_1$, then $\mathbf{x}_1$ and $\mathbf{x}_2$ are said to be non-dominated with respect to each other (provided $\mathbf{x}_1 \neq \mathbf{x}_2$) and denoted as $\mathbf{x}_1 \parallel \mathbf{x}_2$. A vector $\mathbf{x}^* \in S$ is said to be Pareto-optimal (Deb, 2001; Miettinen, 1999), if there does not exist any $\mathbf{x} \in S$ such that $\mathbf{x} \prec \mathbf{x}^*$. The n dimensional space formed by the variables is called the *decision space*, while the m dimensional space formed by the objectives is called the *objective space*. Each point $\mathbf{x}$ in the decision space maps to some point in the objective space based on the values of the objective functions at $\mathbf{x}$. Figure 1 shows such a mapping for a MOO problem with $n = 2$ variables and $m = 2$ conflicting objectives that are to be minimized. According to the aforementioned definition of dominance, $\mathbf{a} \prec \mathbf{b}$, $\mathbf{c} \prec \mathbf{b}$, $\mathbf{d} \prec \mathbf{a}$, $\mathbf{d} \prec \mathbf{b}$, $\mathbf{d} \prec \mathbf{c}$ and $\mathbf{a} \parallel \mathbf{c}$. All points on the curve $\mathbf{ef}$ are Pareto-optimal solutions, because there are no feasible points that dominate any of them.

Evolutionary algorithms (EAs) have become increasingly popular for solving optimization problems due to their global search characteristics. EAs emulate the processes of reproduction and natural selection, to evolve a population of randomly generated solutions over several generations (iterations). Reproduction of new solutions (children) from current solutions (parents) is simulated using special operators called *crossover* and *mutation*. When only a single objective function is to be minimized, selection of good solutions for the next generation is performed by simply ranking the parent and child solutions in the ascending order of their objective function values, and choosing the required number of solutions from the top. This is similar to natural selection, in which only the fittest individuals survive. The crossover-mutation-selection process is repeated until a specified number of generations or performance target or function evaluations is reached.

Multi-objective evolutionary algorithms (MOEA), such as the Non-Dominated Sorting Genetic Algorithm or NSGA-II (Deb et al., 2002) used in this study, differ mainly with respect to the selection operator. NSGA-II uses a procedure called *non-dominated*

¹Maximization of an objective function f_i is achieved by minimizing $-f_i$.

Figure 1: Mapping of solutions from a two-dimensional decision space to a two-dimensional objective space. The curve **ef** represents the Pareto-optimal front because there are no feasible points that dominate any of the solutions on it.

sorting that takes the dominance relationships into consideration when ranking solutions. To achieve a good spread of Pareto-optimal solutions, NSGA-II uses a diversity preserving operator that promotes the selection of solutions that are least crowded by other solutions.

The solutions generated by the MOEA may together, or in part, hold vital knowledge that can give a better understanding of the problem. Data mining of these solutions can reveal interesting properties about the Pareto-optimal solutions and, in general, can help discover domain-specific knowledge (Deb & Srinivasan, 2006; Deb et al., 2014). The knowledge, thus obtained, can also be integrated into an expert system for reformulating the MOO problem or for improving the MOEA as explained later in Section 4.

In this paper, we collectively refer to these solutions as the *MOO dataset*. It contains the decision variable values and the corresponding objective function values for all solutions evaluated during the entire optimization process. Part A of this paper serves as a comprehensive review of several existing methods. Based on the representation of the knowledge gained, these methods were categorized as:

1. Descriptive statistics: These are simple univariate and bivariate measures that summarize various aspects of variables in the form of numbers. The representation of the knowledge is therefore *explicit*, meaning that it is compact and can be easily stored, transferred or parsed programmatically. They are further categorized as:
 - (a) Measures of central tendency
 - (b) Measures of variability/dispersion
 - (c) Measures of shape distribution
 - (d) Measures of correlation/association
2. Visual data mining methods: These methods express knowledge visually through graphs, heatmaps, colored clusters, etc. These representations are *implicit*, meaning that they lack a formal notation and, hence, are prone to subjectivity in interpretation. Visual data mining methods are further categorized as:
 - (a) Graphical methods

- (b) Clustering-based visualization methods
- (c) Manifold learning
- 3. Machine learning methods: They are capable of learning or discovering knowledge from data in both implicit and explicit forms. For example, support vector machines and neural networks generate implicit knowledge models, whereas decision trees result in explicit rules. Machine learning methods are further categorized as:
 - (a) Supervised learning
 - (b) Unsupervised learning
 - (c) Hybrid learning (proposed in this paper)

Readers are referred to Part A for a discussion on the above categories and their relative merits. For the purposes of this paper, we summarize the three main limitations of existing methods in the following section.

1.1. Limitations of Existing Data Mining Methods for Knowledge Discovery

The survey in Part A concludes that while several data mining methods already exist for numerical data, most of them are not tailored to handle MOO datasets, which come with inherent properties that distinguish them from ordinary datasets. The three main limitations are:

1. The presence of two different spaces, objective space and decision space, adds a certain degree of difficulty in discovering knowledge of relevance. Existing data mining methods are not designed to work with these spaces. Since decision makers are often interested in the objective space, it would be beneficial to develop data mining methods that operate in the decision space and, at the same time, take the structure of solutions in the objective space into consideration. Furthermore, decision makers usually prefer certain regions of the objective space. Data mining methods for MOO datasets should be able to take these preferences into account, even if they are often vague in the real world.
2. Discrete variables occurring in optimization problems can be of three main types:
 - (a) Integers: Variables that are by their very nature, integers.
 - (b) Practically discrete: Variables that are forced to take only certain real values, due to practical considerations.
 - (c) Categorical: Variables for which the numerical value is of no significance, but only a matter of convenience. They can be further classified as,
 - i. Ordinal: Variables that represent positions on a scale or sequence. For example, a variable that can take ‘Low’, ‘Medium’ or ‘High’ as options can be encoded with numerical values 1, 2 and 3 or 10, 20 and 30, respectively. The values themselves are of no importance, as long as the order between them is maintained.
 - ii. Nominal: Variables that represent unordered options. For example, a variable representing the choice between ‘x’ and ‘o’ in a game of tic-tac-toe is nominal. Programmatically, they can be represented as either 0 and 1 or 1 and 0, respectively.

MOO datasets containing categorical variables cannot be easily handled using existing data mining methods, most of which rely on some kind of distance measure for clustering. Since the numerical values of categorical variables are of no importance, any distance measure that uses these values will be arbitrary.

3. Most existing data mining methods represent knowledge in implicit forms. However, decision makers usually prefer explicit forms of knowledge because they are easy to understand and interpret. Explicit forms are also useful when the knowledge is expected to be a part of an expert system that aids in decision making (Liao, 2003). In Part A, we identify three explicit forms, (i) analytical relationships, (ii) decision rules, and (iii) patterns or association rules, and discuss their merits. We also note that while analytical relationships are well-suited for continuous variables, knowledge involving discrete variables is better captured through decision rules and patterns.

The paper is organized as described in the following paragraphs.

In Section 2, we develop four new data mining approaches by extending existing methods. The proposed methods overcome the three major issues described above in the following ways:

1. Firstly, all four methods treat the decision space and the objective space separately. Knowledge obtained about the variables from the decision space is visualized in the objective space to reveal its structure. Three of the methods are capable of taking the decision maker's preference into account and find knowledge that is specific to the preferred solutions.
2. Secondly, none of the proposed methods use any distance measure in the decision space and thus are capable of dealing with all types of discrete variables. Clustering, when used, is only performed in the objective space.
3. Thirdly, all methods generate knowledge in explicit forms due to the underlying methods. The nature of these forms is described in Section 2 for each method.

In Section 3, after a brief discussion on the practicality of multi-objective optimization of production systems, each method is applied to three realistic problems from the production engineering domain. These problems have been specifically chosen to highlight the importance of using discrete variables in production systems and to show the efficacy of the proposed methods. The application problems are:

1. Flexible Machining Cell (FMC) design problem in Section 3.1: 3 objectives, 2 nominal variables, 5 practically discrete variables and 2 integers.
2. Balanced Unpaced Flow Line (BFL) problem in Section 3.2: 2 objectives, 1 constraint, 13 practically discrete variables and 12 integers.
3. Automotive Component Manufacturing (ACM) problem in Section 3.3: 3 objectives, 9 ordinal variables and 31 integers.

In Section 4, the knowledge gained from different methods is compared and discussed in relation to each MOO problem. The obtained knowledge is also visualized in the objective space. For problems with three objectives, either perspective plots or combination of two-objective plots are used.

2. Data Mining Methods for MOO Datasets

Table 1 shows a summary of the four improved data mining methods developed in this study. The type of learning involved and the form of knowledge representation are also shown. The following sections describe each method in detail.

Table 1: Summary of data mining methods developed in this work.

Type of Learning	Method	Type of Knowledge
Unsupervised	Sequential Pattern Mining	Patterns
Supervised	Classification Trees	Decision Rules
Hybrid	Clustering + Classification Trees	Unsupervised Rules
Unsupervised	Flexible Pattern Mining	Unsupervised Rules

2.1. Sequential Pattern Mining

The term *sequential pattern* was first used in the context of market basket data analysis (Agrawal & Srikant, 1995). An instance in such a dataset typically consists of the transaction ids and the items purchased together by a customer in each transaction, as shown in Table 2. An itemset is a non-empty set of items and is denoted as (item1, item2, ...). A sequence is an ordered list of itemsets, typically enclosed within angled brackets, for example, $\langle\langle\text{item1, item2}\rangle\langle\text{item3}\rangle\rangle$. Thus, each row in Table 2 is a customer’s complete sequence. A customer is said to *support* a sub-sequence if it is contained within that customer’s complete sequence. The goal of sequential pattern mining is to generate an exhaustive list of *all* the sub-sequences that are supported by at least a predefined number of customers called ‘minimum support’. For example, the following patterns, represented by superscript numbers in Table 2, have a minimum support of 3 customers,

1. $\langle\langle\text{bread}\rangle\langle\text{cheese}\rangle\rangle$: Support = 5 (Customers 1, 2, 3, 5 and 8)
2. $\langle\langle\text{juice}\rangle\langle\text{apples}\rangle\rangle$: Support = 4 (Customers 1, 4, 6 and 9)
3. $\langle\langle\text{milk}\rangle\langle\text{tissues}\rangle\rangle$: Support = 3 (Customers 1, 4 and 10)
4. $\langle\langle\text{diapers, tissues}\rangle\langle\text{beer}\rangle\rangle$: Support = 4 (Customers 4, 6, 9 and 10)

Note that when comparing sub-sequences, the order of items within an itemset is not important, but the order of itemsets within the sequence is. Further, the sub-sequence $\langle\langle\text{milk}\rangle\langle\text{tissues}\rangle\rangle$ is not the same as $\langle\langle\text{milk, tissues}\rangle\rangle$, since the former implies that the items, milk and tissues, come from different transactions.

A typical discrete optimization dataset is shown in Table 3. The entry d_{ij} represents the j -th discrete option of the i -th variable. A sequential pattern mining algorithm can be applied to such a dataset, by considering each solution vector as a customer sequence, each variable as a separate transaction, and each discrete variable option as a different item. The only fundamental difference between the datasets in Tables 2 and 3 is that the former can have multiple items per transaction and a different number of transactions for each customer. In discrete optimization datasets, itemset and item are equivalent, because for all customers (solutions) each transaction (variable) can have only one item (discrete value). Also, the sequence length is fixed for all customers (solutions) and is equal to the number of transactions (variables).

2.1.1. Apriori Algorithm

The Apriori algorithm Agrawal & Srikant (1994) is one of the earliest and most popular association rule mining techniques in the literature. Given a dataset and a minimum support value, the algorithm first generates a list of all one-itemset sequences. Those sequences that meet the minimum support condition form the set L_1 which stands for *frequent one-itemset sequences*. Next, the algorithm makes use of the *downward closure property* or *Apriori principle*, which states that

Table 2: An example of customer sequences in market basket data.

Customer ID	Transaction 1	Transaction 2	Transaction 3	Transaction 4
Customer 1	bread ¹	juice, cheese ¹ , bananas, milk ³	tissues ³ , juice ²	apples ²
Customer 2	rice, flour, bread ¹	sugar, cakemix	cheese ¹	—
Customer 3	grapes, gum	bread ¹ , oranges, juice	chicken, cheese ¹ , juice	—
Customer 4	juice, cheese, milk ³	bananas, diapers ⁴ , tissues ^{3,4}	bread, juice ²	apples ² , beer ⁴
Customer 5	scissors, gift wrap, bread ¹	milk, tissues, juice, glue	bread, milk, cheese ¹	—
Customer 6	flour	juice ² , tissues ⁴ , diapers ⁴	bandaid, apples ²	beer ⁴ , dish soap
Customer 7	milk, sugar, cheese	flour, sugar, apples	—	—
Customer 8	beer, juice, bread ¹ , sugar	dish soap, oranges, chicken	milk, cheese ¹	bread, chips
Customer 9	diapers ⁴ , juice ² , tissues ⁴	apples ² , juice, beer ⁴ , chips	—	—
Customer 10	bulb, batteries	tissues ⁴ , milk ³ , diapers ⁴	steak, chips, beer ⁴	dish soap, tissues ³

Table 3: An example of a discrete optimization dataset. d_{ij} represents the j -th discrete option of the i -th variable. The wildcard * can take any integer between 1 and n_i for the i -th variable. We assume some values for * in solution k .

Solution ID	Variable 1 (x_1)	Variable 2 (x_2)	Variable 3 (x_3)	Variable 4 (x_4)	...	Variable n (x_n)
Solution 1	d_{1*}	d_{2*}	d_{3*}	d_{4*}	...	d_{n*}
Solution 2	d_{1*}	d_{2*}	d_{3*}	d_{4*}	...	d_{n*}
Solution 3	d_{1*}	d_{2*}	d_{3*}	d_{4*}	...	d_{n*}
...	...	...	...	...	...	...
Solution k	d_{12}	d_{21}	d_{33}	d_{42}	...	d_{n2}
...	...	...	...	...	...	...
Solution $N - 1$	d_{1*}	d_{2*}	d_{3*}	d_{4*}	...	d_{n*}
Solution N	d_{1*}	d_{2*}	d_{3*}	d_{4*}	...	d_{n*}

Table 4: The truth vector of solution k from Table 3, as required for flexible pattern mining. The vector is split into multiple lines using $\rightarrow$ symbols.

Solution ID	$x_1 = d_{11}$	$x_1 > d_{11}$	$x_1 < d_{12}$	$x_1 = d_{12}$	$x_1 > d_{12}$	$x_1 < d_{13}$	...	$x_1 < d_{1n_1}$	$x_1 = d_{1n_1}$	$\rightarrow$
Solution k	0	1	0	1	0	1	...	1	0	
$\rightarrow$										
Solution k	1	0	1	0	0	1	...	1	0	$\rightarrow$
$\rightarrow$										
Solution k	0	0	0	0	1	1	...	1	0	$\rightarrow$
...	...	...	...	...	...	...	...	...	...	
$\rightarrow$										
Solution k	0	1	0	1	0	1	...	1	0	$\rightarrow$

Every non-empty subset of a frequent sequence is also frequent.

This also means that frequent sequences with a larger number of itemsets can be formed by combining smaller frequent sequences. The candidate generation step combines various one-itemset sequences to form candidate two-itemset sequences, of which those satisfying minimum support form a new set L_2 . The process repeats until L_k is empty, i.e., no frequent k -itemset sequences can be found. A final step, called the *maximal phase*, prunes all non-maximal frequent sequences from $\bigcup_k L_k$. A sequence is maximal if it is not contained in any other sequence. The maximal sequences that remain after pruning are called *sequential patterns*.

The above described variant of the Apriori algorithm is called AprioriAll and can be applied to discrete optimization datasets to obtain sequential patterns. However, due to the differences with market basket data, a few modifications are required. Firstly, note that frequent sequences can come from any transactions for different customers. For example, the sequence $\langle(\text{diapers, tissues})(\text{beer})\rangle$ comes from the second and fourth transactions for customers 4 and 6, whereas for customer 9 they come from their first and second transactions. Such patterns would not be meaningful for optimization datasets, because the transactions represent variables. To illustrate, the sequence $\langle(x_1 = 10)(x_2 = 15)\rangle$ is different from $\langle(x_1 = 10)(x_3 = 15)\rangle$. This modification can be introduced in the AprioriAll algorithm in a manner similar to the use of time constraints in Srikant & Agrawal (1996), which specify time intervals between adjacent elements in a sequential pattern. Our implementation of the algorithm ensures that a pattern corresponds to the same set of variables across all the solutions that support it. Secondly, our implementation does not have a maximal phase. The reason for this is purely result-oriented. Knowledge of even non-maximal sequences is essential for a decision maker who wants to understand the effect of individual variables, if any. An additional modification in our implementation of AprioriAll is that the maximum possible support value for any pattern in the dataset is calculated on the basis of the patterns in L_1 and provided to the decision makers to aid them in specifying a minimum support value.

2.1.2. Patterns in Preferred Solutions

Sequential pattern mining is usually applied to the whole dataset. However, the decision maker is often interested in a preferred region of the objective space. Several preference articulation methods have been proposed in the literature (Purshouse et al., 2014), the most common one being the specification of a reference point (Miettinen, 1999; Deb & Sundar, 2006). Decision makers sometimes have vague preferences, which makes it difficult to provide a single reference point. A more natural and interactive approach is to allow the user to ‘brush’ over solutions using a mouse input to select a region of interest in the objective space. This is easily achieved in problems with two objectives. For three and higher-dimensional objective spaces, we display a *scatter matrix* to the user, who can then make a selection incrementally, by brushing over solutions in different two-objective plots. The modified AprioriAll algorithm is applied only to the selected set of solutions, but the patterns discovered in the process can be visualized over all the solutions. We demonstrate this procedure in Section 4 for all the applications. Note that sequential pattern mining is not designed to differentiate between two datasets. In other words, it does not require an “output” feature and the method is therefore unsupervised.

2.2. Classification Trees

A decision tree is an explicit knowledge model that resembles a binary tree with one root node and several leaf nodes. Decision tree learning algorithms, such as the popular ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993) algorithms, work by recursively dividing the dataset using one feature (variable) at a time. At each node, the feature (variable) and its value are chosen, such that the division maximizes the dissimilarity between the two resulting groups in terms of the “output” feature. Each branch of the decision tree therefore represents an ‘if-then’ rule. The branches connecting the root node to a leaf node can be logically conjoined (denoted using $\wedge$) to generate a *decision rule* that is applicable to points (solutions) in that leaf node. The main advantage of using decision trees with MOO datasets is that the decision maker can distinguish between two or more sets of solutions. Decision trees can be of two types, regression trees and classification trees. Regression trees use a continuous output feature and have already been used for data mining MOO datasets (Sugimura et al., 2010; Dudas et al., 2011, 2014, 2015). For example, in Dudas et al. (2015), the distance from a given reference point is used as the output feature. In this work, we use classification trees which require pre-specified class labels as the output feature. Each node uses a *split criterion* to select a variable and its corresponding value. Popular split criteria are the Gini impurity index and information gain (Friedman et al., 2001). We use the Gini impurity index given by $G(t) = 1 - \sum_{k=1}^K [f(k|t)]^2$, where $f(k|t)$ is the fraction of instances belonging to class k at the given node t . The split is based on the condition v (combination of a variable and its value) which maximizes the Gini gain Δ , between the parent node and its child nodes. It is given by

$$\Delta = G(\text{parent}) - \sum_{v \in V} f_{\text{parent},v} G(\text{child}|v), \quad (2)$$

where V is the set of all possible conditions obtained from the features and their sorted values, $f_{\text{parent},v}$ is the fraction of instances in the parent node that satisfies condition v and $G(\text{child}|v)$ is the Gini index of the child node satisfying v .

The preference articulation method described in the previous section is also used here. By brushing over solutions, the decision maker creates two classes, ‘selected’ and ‘unselected’ solutions. The generated rules can sometimes be very lengthy, if the tree is not properly pruned. Moreover, multiple rules may exist that lead to the same class. Our implementation of classification trees uses a pruning method that only results in *dominant rules* that distinguish between the selected and unselected classes. This is achieved by imposing a pruning threshold on the proportion of unselected to selected classes at each leaf node. For example, a pruning threshold of 5% specifies that a rule is dominant only if the percentage of unselected solutions at the leaf node is less than 5%.

2.3. Hybrid Learning

As discussed above, decision trees require a class label to generate rules that distinguish between two or more sets of solutions. However, when no such labels can be generated manually, or when the decision maker does not have a specific preference of solutions, clustering methods can be used in the objective space to find natural clusters occurring in the dataset. Thereafter, the cluster associations of the solutions can be used as labels to perform multi-class classification tree learning. This combination of

unsupervised generation of class labels followed by supervised classification tree learning can be termed *hybrid learning*. The resultant rules can be called *unsupervised rules* to distinguish them from decision rules which are obtained in a supervised manner. The advantage of hybrid learning is that since the classification of the solutions is driven by data itself, the generated dominant rules are expected to be simpler and more insightful to the decision maker. In this paper, we use two clustering methods to generate the class labels.

2.3.1. Agglomerative Hierarchical Clustering

Hierarchical clustering algorithms (Xu & Wunsch, 2005; Friedman et al., 2001) organize the data-points into a dendrogram based on a specified proximity measure. The agglomerative version of hierarchical clustering does this by starting with N singleton clusters (N being the size of the dataset). This represents the finest level of clustering. The two closest clusters are combined to form the next level of clustering. The process continues until all data-points are in the same cluster. Once the hierarchical structure is established among the data-points, the user can quickly vary the number of clusters, by *cutting* the dendrogram at different levels. The advantages of agglomerative clustering are threefold: (i) the number of clusters need not be predefined, (ii) any distance metric and linkage criterion can be used to define proximity, and (iii) the clustering is deterministic. We use agglomerative clustering to generate cluster labels for those problems where a proximity measure could be defined unambiguously. A euclidean distance measure with complete linkage is used to combine clusters at each level. The objective function values are normalized to avoid scaling issues.

2.3.2. Density-based Clustering

MOO datasets that involve discrete variables tend to have feasible solutions in disjoint regions of the decision space, due to variable values ‘jumping’ from one discrete option to another. This often translates to disjointedness in the objective space, where the regions may be arbitrarily shaped, due to non-linear dependencies of the objectives on the variables. Most distance-based clustering algorithms can only find spheroidal clusters, because they use distance as the only criterion. Density-based clustering algorithms (Xu & Wunsch, 2005), on the other hand, define clusters based on a more complex ‘crowdedness’ measure. DBSCAN (Density Based Spatial Clustering of Applications with Noise) (Ester et al., 1996) is a popular density-based technique which uses the notion of *density-reachability* and *density-connectivity* to determine whether two points belong to the same cluster. The basic DBSCAN algorithm is as follows:

```

for each point in dataset do
  if point is not yet classified then
    if point is a core point then
      Collect all points density-reachable from point into new cluster
    else
      Assign point to Noise
    end if
  end if
end for

```

DBSCAN uses two parameters, *MinPts* (minimum points) and *Eps* (neighborhood radius), to define density-reachability. However, some heuristics can be used to set at

least one of the parameters. The added advantage of using DBSCAN on optimization data is that in addition to detecting arbitrarily shaped regions in the objective space, it can also handle noise (isolated solutions).

2.4. Flexible Pattern Mining

Sequential pattern mining is extremely useful for finding *exact patterns* in the dataset. For example, given the simple one-dimensional dataset $x_1 = \{1, 1, 1, 2, 2, 2, 2, 3, 3, 10, 10, 10\}$, it can detect that the patterns $\langle(x_1 = 1)\rangle$, $\langle(x_1 = 2)\rangle$ and $\langle(x_1 = 10)\rangle$ have supports 3, 4 and 3 respectively. However, it cannot detect the more important rule $\langle(x_1 < 10)\rangle$ which has a support of 9. A decision tree, though capable of generating such rules, requires that the user specify the first nine points as one class to exactly detect the dominant rule. In other words, decision trees are very sensitive to training data and prone to overfitting. This problem is more severe when a large number of features are involved. Exact patterns require that the variable values are exactly the same for all solutions supporting that pattern. This may be too restrictive for pattern mining in MOO datasets, which are usually generated through stochastic optimization algorithms.

To address this inflexibility of sequential pattern mining, we propose an extension called Flexible Pattern Mining (FPM). Like sequential pattern mining, FPM extends the concept of market basket data to MOO datasets, by considering each solution as a customer and each variable as a transaction containing individual items (variable values). However, to enable the generation of rules through the same Apriori algorithm, the optimization data in Table 3 is transformed into a truth table. The columns of this table correspond to each of the n variables being less than ($<$), equal to ($=$) and greater than ($>$) each of its discrete options. If $n_1, n_2, \dots$ and n_n are the number of discrete options for $x_1, x_2, \dots$ and x_n respectively, then the total number of columns in the truth matrix is $3(n_1 + n_2 + \dots + n_n) - 2n$. The second term comes from the fact that no variable x_i can take a value less than d_{i1} or greater than d_{in_i} . The transformed truth vector for one of the solutions from Table 3 is shown in Table 4. Note that due to the length of the vector, it is split across four lines using $\rightarrow$ symbols.

Apriori algorithm can now be applied to the truth table to obtain patterns of 1s. However, before that, the algorithm should be modified to ignore patterns of 0s, which represent false conditions for the corresponding columns and are hence not useful. FPM combines the merits of sequential pattern mining, classification trees and hybrid learning. Firstly, like sequential pattern mining, it is unsupervised and can be applied to the whole dataset in the absence of user preferences. Secondly, like classification trees, it generates rules which are much more flexible than exact patterns. Thirdly, like hybrid learning, it does not require the user to provide class labels. In case the user chooses to express preference by selecting (brushing over) solutions, an additional step can be performed on the rules. By sorting the rules in the decreasing order of the ratio of support in the selected solutions to that in the rest of the solutions, a unique ordering of the rules can be obtained. The topmost rule in such an ordering then represents the *most distinguishing property* of the selected region. Consecutive rules may also be combined to form new rules and their support can be recalculated to reveal their presence within the selected solutions. We illustrate this procedure through the applications in Section 4.

3. Multi-Objective Optimization of Production Systems

Production systems are regarded as complex systems, not only due to their heterogeneous components that usually involve heavy human-machine interactions, but also because of the strict requirements to produce profits. While achieving high profitability is usually the foremost goal, it is composed of many sub-system level objectives, such as high throughput, high utilization, fast customer response with low cycle times, low cost and low inventory, etc. Thus, production system optimization is inherently one with multiple conflicting objectives (Hopp & Spearman, 2011). Although analytical methods, like Petri nets and queuing models, can be used to model production systems, simulation (in particular, discrete-event simulation) is still the most practical choice for industrial practitioners (Tempelmeier, 2003). This is not surprising, since even complex product mixes and their routings can be easily modeled using any modern simulation software. Additionally, simulation is preferable when handling probability distributions that cannot be handled in analytical models. Nevertheless, simulation is not an optimization tool when used standalone. The connection of meta-heuristic search methods to simulation models, referred to as simulation-based optimization, can be used to find the optimal settings for different decision variables in a system design problem (Law & McComas, 2000). With multiple objectives, multiple Pareto-optimal solutions can be expected. A deeper understanding of the behavior of the system can only be gained by analyzing the obtained solutions for hidden knowledge. Therefore, applying data mining methods to MOO datasets from production systems is of prime importance (Pehrsson & Ng, 2011).

In the following sections, we briefly describe three application studies involving multi-objective optimization in production systems. For readers unacquainted with the subject, the problems considered here deal with manufacturing lines which consist of (i) various operations performed at different workstations, and (ii) inter-station buffers. Each workstation has stochastic variables, such as processing time, percentage availability (uptime), time to repair, etc., which directly affect the throughput (production rate) of the line. Buffers are designated storages for each workstation and are meant to compensate for variability in the manufacturing processes, such as variations in processing times, machine breakdowns, tool changes, etc. The optimization of production systems often involves improving throughput by detecting and removing bottlenecks through investments in better machines, more buffers, better processes etc.

3.1. Flexible Machining Cell Design Problem (FMC)

The first application study is based on a flexible machining cell model, initially developed to show the advantages of using multi-objective optimization for production systems' design (Ng et al., 2011). The model is built in FACTS Analyzer (or FACTS hereafter) (Ng et al., 2007), whose interface is shown in Figure 2. FACTS, which stands for Factory Analyses in Conceptual phase using Simulation, is a discrete-event simulation software developed at University of Skövde for analyzing and optimizing production systems in the conceptual phase. The FMC model makes use of the so-called 'selection objects' in FACTS that enable multiple design alternatives to co-exist in a simulation model. By changing the values of the corresponding decision variables, the optimization module in FACTS can switch between different combinations of selection objects. Multi-objective optimization of such a model seeks to find the optimal combination of the design alternatives along with the optimal values of other decision variables.

Figure 2: FACTS simulation model for the FMC design problem.

Figure 2 illustrates how alternative designs can be easily built in FACTS with two nominal variables, S_1 and S_2 . Workstation operations are represented with ‘Op’ and buffers with ‘B’. $S_1 = 2$ represents a choice of having one fast machine, *FastOp1*, instead of two slow machines, *SlowOp1A* and *SlowOp1B*, in parallel (*ParallelSlowOp*) which is represented by $S_1 = 1$. This is considered to be a very common decision-making problem in industry, when considering the replacement of slower old machines with faster new machines. For the three operations, *Op2A*, *Op2B* and *Op2C*, there is a flow selection in which two alternatives are evaluated. The first option ($S_2 = 1$) involves having dedicated flow lines for all the three product variants, *Var A*, *Var B* and *Var C*, as shown in the figure. In the second option ($S_2 = 2$), the three variants are allowed to be flexibly processed in any of the three machines, but with the additional cost of increased setup times during the switching of variants. Therefore, in addition to deciding whether to replace the old machine in the first stage, the decision maker also needs to evaluate whether it is cost-effective to invest in flexible machines in the second stage. Apart from the two nominal variables discussed above, the availabilities of machines *SlowOp1A*, *SlowOp1B*, *Op2A*, *Op2B* and *Op2C* (denoted by A_1 , A_2 , A_3 , A_4 and A_5 , respectively) are allowed to be improved from the default 90% to 95% in steps

of 1%. Moreover, the two buffers B_1 and B_2 are allowed to take integer values in the range $[5, 10]$. FACTS allows the definition of so called ‘multiple choice sets’ (Bernedixen & Ng, 2014) to handle discrete variables in production systems. They take the form $\{\text{lower_bound}; \text{upper_bound} \mid \text{step_size}\}$. Thus, $A_i \in \{90; 95 \mid 1\}$ and $B_i \in \{5; 10 \mid 1\}$.

Each additional buffer space requires an investment of \$10,000, while improvements in availability require \$10,000 per step. Switching from $S_1 = 1$ to $S_1 = 2$ involves a one-time investment of \$100,000, whereas that from $S_2 = 1$ to $S_2 = 2$ involves \$90,000. A more complete description of the model and its significance can be found in Ng et al. (2011). A three-objective optimization problem is formulated to maximize the system throughput, minimize the total investment and minimize the total number of buffers ($\text{LeanBuffer} = B_1 + B_2$). In total, there are 9 decision variables: 2 nominal selection variables, 5 practically discrete availability variables and 2 integer buffer variables.

3.2. Improvement Actions for a Balanced Unpaced Flow Line (BFL)

The second application study is a simple unpaced flow line, comprising 13 workstations with 12 inter-station buffers. The workloads of the workstations are perfectly balanced, each with a processing time of 80 minutes per job and each following a log-normal distribution to represent the variability. In a complex flow with unbalanced workloads, the location of the bottleneck and the subsequent improvement of the production rate (throughput) are complicated tasks. The bottleneck depends on many factors including the job flow logic, variability, and the availability of every workstation as well as the buffer capacities. Nevertheless, even for a simple straight flow line with balanced workloads, such as the one considered here, the identification of bottlenecks and improvement actions to be taken is not a trivial task. Multiple improvement actions will improve the throughput, but also require additional investments. The idea of treating this throughput improvement problem as bi-objective optimization that minimizes the number of changes (improvement actions) while maximizing the throughput was first proposed in Pehrsson (2013). For this purpose, a FACTS model is developed with the 13 processing times T_i (practically discrete variables) and the 12 inter-station buffers (integers) with capacities B_i as the decision variables such that $T_i \in \{60; 80 \mid 5\}$ minutes (or $T_i \in \{3600; 4800 \mid 300\}$ seconds) and $B_i \in \{1; 49 \mid 1\}$. An overall buffer constraint $\sum B_i = 60$ is also imposed. During optimization, this linear equality constraint is handled by repairing infeasible solutions by projecting them onto the hyperplane $\sum B_i = 60$. Details can be found in Bernedixen & Ng (2014).

In this application study, the purpose is to identify patterns in processing time reductions and buffer allocations that can improve a balanced unpaced flow line with minimal changes. Such patterns, if found, may provide new insights into the improvement of real-world production systems. The pattern analysis is particularly interesting when applied to the production line models with different levels of variation. Therefore, multi-objective optimization of BFL is performed at two levels of variation, (i) BFL-0.1, classified as ‘low-variance’ corresponds to $c_v = 0.1$, and (ii) BFL-0.5, classified as ‘high-variance’ corresponds to $c_v = 0.5$, where c_v is the coefficient of variation in the processing times of the workstations.

3.3. Automotive Component Manufacturing (ACM)

The third application study is part of an industrial improvement project conducted for an automotive component manufacturer in Sweden. The production line in ques-

tion suffered from high running costs due to capacity loss. At the beginning of the project, it was believed that the system capacity was restricted by several bottlenecks (Pehrsson et al., 2013), which could be relieved if investments were made to improve the availability and/or reduce the processing times of some workstations. The purpose of the optimization study was therefore to choose the best alternative that would, with minimal investments (improvement costs), reduce running costs and achieve the required capacity, in terms of system throughput. A discrete-event simulation model of the production line was developed using FACTS and validated with accurate correspondence to the real average line throughput in the early phase of the project. The model is illustrated in Figure 3 with the workstations and inter-station buffers labeled.

Figure 3: FACTS simulation model of the industrial improvement project.

Complete details of the model were first presented in Pehrsson et al. (2013). Here, we summarize the most important information, including the objectives of the optimization, decision variables, and conclusions from previous data mining experiments which use a distance-based regression tree method (Ng et al., 2012). Three objectives were considered in generating the optimization dataset, (i) minimization of total annual running cost (C_R), (ii) minimization of total (improvement) investment cost (I), and (iii) minimization of total buffer capacity (B). They are formulated as follows,

$$\begin{aligned}
 \text{Minimize } C_R &= C_I + \Delta C_T + \Delta C_A + \Delta C_U \\
 \text{Minimize } I &= \sum_i (Ip_i + Iu_i) \\
 \text{Minimize } B &= \sum_j B_j
 \end{aligned} \tag{3}$$

The total annual running cost is determined by using the incremental cost modeling method developed in Pehrsson et al. (2013). C_R is composed of the initial annual running cost (C_I) and the incremental costs, due to increase in throughput (ΔC_T), attached costs (ΔC_A), such as workstation maintenance, and extra per unit costs (ΔC_U), such as energy use. The total investment cost function (I) is simply the summation of all the investments related to the improvement of processing times (Ip) and the availability or uptime (Iu) of the workstations. The total buffer capacity is simply the summation of the 31 inter-station buffers.

Two groups of decision variables are considered in the case study, (i) the level of improvements (Iu_i or Ip_i) and (ii) the capacities for the inter-station buffers (B_j), where i and j denote the workstations and buffers in the simulation model respectively (see Figure 3). As requested by the production manager, improvements were considered for workstations $Op1E$, $Op1G$, $Op1H$, $Op1J$, $Op1N$ and $Op1O$ at different levels, as shown in Table 5. A level 0 investment meant no improvement, while each of the other levels corresponds to a pre-specified improvement in the processing time or the machine

Table 5: Levels of improvement considered for the first group of variables of the ACM problem and their investment costs in USD (\$). Level 0 corresponds to current state. Hyphens (-) indicate that those levels of improvement are not considered in the optimization problem.

Variables	Levels of improvement (Iu_i in % and Ip_i in seconds)								
	0	1		2		3		4	
	Value	Value	Cost	Value	Cost	Value	Cost	Value	Cost
Iu_{Op1E}	91%	95%	5,443	96%	2,500	97%	4,500	-	-
Ip_{Op1E}	145s	143s	10,000	133s	10,000	-	-	-	-
Iu_{Op1G}	91%	95%	5,000	95.5%	2,500	96%	25,000	97%	2,500
Ip_{Op1G}	145.3s	138.3s	20,000	-	-	-	-	-	-
Ip_{Op1H}	153s	115s	2,500	-	-	-	-	-	-
Ip_{Op1J}	145s	130s	20,000	-	-	-	-	-	-
Iu_{Op1N}	93%	95%	5,000	-	-	-	-	-	-
Ip_{Op1N}	95s	80s	20,000	-	-	-	-	-	-
Ip_{Op1O}	125s	85s	20,000	-	-	-	-	-	-

uptime (availability). These numbers and their associated costs were also provided by the production manager. The bounds for the buffer capacities were $B_j \in \{1; 35 | 1\}$. In terms of our classification of variables, Iu_i and Ip_i are ordinal, while buffers are integer variables.

4. Results and Discussion

The generic methodology adopted in this section is shown as a block diagram in Figure 4. The first block on the left represents the MOO problems from the previous section. The second block represents the MOEA used to solve the MOO problem. Here, we use NSGA-II (Deb et al., 2002) as the MOEA. Multiple randomly-seeded runs with the following parameter settings are used to obtain a representative Pareto-optimal front:

1. Population size, N : 500
2. Simulated Binary Crossover (SBX) probability (Deb & Agrawal, 1995), p_c : 0.9
3. SBX distribution index (Deb & Agrawal, 1995), η_c : 20
4. Polynomial mutation probability (Deb, 2001), p_m : 0.05
5. Polynomial mutation distribution index (Deb, 2001), η_m : 20
6. Number of function evaluations: $\begin{cases} 10,000 \text{ for FMC and BFL-0.1} \\ 20,000 \text{ for BFL-0.5 and ACM} \end{cases}$

The population size and the number of function evaluations were obtained for each problem by gradually increasing them from 50 and 5000 respectively, until a uniform coverage was obtained on the Pareto-optimal front. The other parameters are set to their recommended values.

The third block in Figure 4 represents the MOO dataset obtained from the MOEA. As mentioned in Section 1, it contains the decision variable values and the corresponding objective function values for all solutions evaluated during optimization.

The fourth set of blocks represent the four data mining methods, each of which generates explicit knowledge of a certain form, as shown by the fifth set of blocks.

Figure 4: Block diagram showing the methodology adopted in this paper and how the obtained explicit knowledge can be used with an expert system for controlling the optimization algorithm or modifying the original MOO problem.

Finally, the last block in Figure 4 represents the visualization of the obtained knowledge in the objective space. The most interesting patterns and rules are those whose ‘support solutions’ form well-defined regions in the objective space that match the structure of the objective space. The reason is that such patterns and rules can be stored in a knowledge base and used to improve search in future optimization scenarios through the use of an expert system (Fonseca et al., 1993) as shown in the figure. For example, patterns and rules can be added as constraints to the MOO problem to focus search towards regions of the objective space preferred by the user. The implementation of such an expert system is left for future work.

In order to clearly visualize the knowledge, we adopt the following color scheme. The solutions supporting a pattern or rule are shown in blue. Where applicable, the solutions selected by the user are shown in red. Selected solutions that do not support the pattern or rule under consideration are also shown in red. On the other hand, unselected solutions that support the pattern or rule under consideration are shown in yellow. Clusters obtained by hierarchical clustering and DBSCAN are shown using other colors (green, magenta, cyan, purple and gray). Support is expressed as a percentage of the total number of solutions in the selection. Note that the selections used in the following sections only serve to illustrate the usefulness of the data mining methods and need not correspond to preferences of decision makers in the real-world.

4.1. FMC Optimization Dataset

We first apply sequential pattern mining on the entire dataset. A minimum support condition of 40% yields 14 patterns, shown in Table 6. Pattern 6 has an interesting distribution in the objective space as shown in Figures 5 and 6, the latter of which also illustrates the three-dimensional structure of the objective space. It is applicable to the lower cluster in the two-objective plot (investment vs. throughput) where $S_2 = 1$. Since the only other discrete option for this variable is 2, pattern 10 should be applicable to the upper cluster of solutions. Thus, the two clusters in the objective space, as seen in Figure 5, are the result of S_2 alone. Such knowledge will be vital to a decision maker.

Two other interesting patterns, 11 and 12, are shown in Figures 7 and 8. Note that just by the conjunction of $S_1 = 1$ to $S_2 = 2$, pattern 11 is applicable to near-Pareto-optimal solutions in the upper cluster of solutions. Pattern 12 is even more interesting because it is applicable to near Pareto-optimal solutions in the upper cluster and to dominated solutions in the lower cluster. This means that lower buffer levels ($B_2 = 5$)

Table 6: Patterns obtained using sequential pattern mining on FMC dataset.

ID	Support	Pattern
1	95.15	$S_1 = 1$
2	78.45	$B_1 = 5$
3	74.52	$B_1 = 5 \wedge S_1 = 1$
4	55.62	$A_2 = 90$
5	55.00	$A_1 = 90$
6	53.16	$S_2 = 1$
7	50.76	$S_1 = 1 \wedge A_2 = 90$
8	50.57	$S_1 = 1 \wedge S_2 = 1$
9	50.14	$S_1 = 1 \wedge A_1 = 90$
10	46.84	$S_2 = 2$
11	44.58	$S_1 = 1 \wedge S_2 = 2$
12	42.21	$B_2 = 5$
13	41.11	$B_1 = 5 \wedge A_2 = 90$
14	40.99	$B_1 = 5 \wedge S_2 = 1$

Figure 5: Pattern 6 of FMC data obtained using sequential pattern mining.

Figure 6: Perspective view of Figure 5 showing 'layers' of Lean Buffer.

Figure 7: Pattern 11 of FMC data obtained using sequential pattern mining.

Figure 8: Pattern 12 of FMC data obtained using sequential pattern mining.

Table 7: Dominant rules obtained using classification trees on FMC dataset.

SEL	UNSEL	Rule
8.76	0.00	$S_1 \geq 1.5 \wedge S_2 \geq 1.5 \wedge A_4 \geq 92.5 \wedge A_3 < 91.5 \wedge A_5 < 92.5$
56.18	0.50	$S_1 \geq 1.5 \wedge S_2 \geq 1.5 \wedge A_4 < 92.5 \wedge A_5 < 94.5 \wedge A_5 < 93.5 \wedge A_4 < 91.5$
5.15	0.00	$S_1 \geq 1.5 \wedge S_2 \geq 1.5 \wedge A_4 \geq 92.5 \wedge A_3 \geq 91.5 \wedge A_5 < 91.5 \wedge A_4 < 94.5$

are adequate when using flexible machines ($S_2 = 2$), but lead to sub-optimal operation when dedicated machines ($S_2 = 1$) are used. In other words, the choice of S_2 indirectly affects B_2 .

Sequential pattern mining is also applied to an arbitrary selection shown in Figure 9. One of the obtained patterns, given by

$$S_1 = 2 \wedge A_1 = 90 \wedge A_2 = 90 \wedge S_2 = 2, \quad (4)$$

has a support of 81.52% in the selected region, but an overall support of only 2.26%, which makes it quite unique to the selection, as shown in Figure 10.

Next, we use classification trees on the data with the same selection of solutions as in Figure 9. The unpruned tree has a maximum depth of 9 and is not shown here for brevity. Using a pruning threshold of 5% on the leaf nodes, three dominant rules, as shown in Table 7, are extracted from the decision tree. Here, **SEL** represents the support (in percentage) within the selected solutions and **UNSEL** represents the same within rest of the solutions (unselected). Notice that the obtained rules are longer than those obtained using sequential pattern mining. This is typical of decision trees which have a tendency to overfit data (Friedman et al., 2001). A remedy is to use pruning, but this increases impurity, meaning higher values for **UNSEL** in Table 7. A characteristic of decision tree rules is that variables occurring closer to the root node are considered to be more significant with respect to the split criterion. Thus, out of the three dominant rules, $S_1 = 2$ and $S_2 = 2$ are found to be the distinguishing features of the selected region.

Figure 9: Selected solutions for FMC data.

Figure 10: Perspective view of the pattern shown in Equation (4).

Table 8: Dominant rules obtained using hybrid learning with hierarchical clustering on FMC dataset. The classes are shown in Figure 12.

CLASS	SEL	UNSEL	Rule
1	65.27	≈ 0.00	$S_2 \geq 1.5 \wedge A_3 \geq 91.5 \wedge A_5 \geq 90.5$
2	88.71	≈ 0.00	$S_2 < 1.5 \wedge B_2 < 9.5$

Hybrid learning with hierarchical clustering presents results similar to sequential pattern mining, but in the form of rules. The algorithm first generates a dendrogram, as shown in Figure 11. Based on the linkage distances, the user decides to use two clusters, which generates classes as shown in Figure 12.

Out of the 10 dominant rules extracted from the classification tree, one for each class is shown in Table 8. Here, **SEL** represents the support (in percentage) for the current class and **UNSEL** represents the same for the other class.

Finally, we use flexible pattern mining for the selected solutions in Figure 9. The obtained rules are shown in Table 9. Note that the rules are sorted according to the ratio of support in the selected solutions, **SEL**, to the support in the unselected solutions, **UNSEL**. The topmost rule is therefore the most distinguishing rule of the selected region. This ordering also allows us to combine rules progressively from the top as follows:

1. top two rules: $S_1 > 1 \wedge S_2 > 1$,
2. top three rules: $S_1 > 1 \wedge S_2 > 1 \wedge A_2 = 90$,
3. top four rules: $S_1 > 1 \wedge S_2 > 1 \wedge A_2 = 90 \wedge A_1 = 90$,
4. ...

The support values for the combined rules can also be calculated. For example, the first four rules have a combined support of 79.58% in the selected region and 0.52% in the unselected part, which increases the highest ratio of **SEL** to **UNSEL**, from $79.36/3.17 = 25.03$ in Table 9 to $79.58/0.52 = 153.04$. Thus, the combined rule is a much better distinguishing feature of the selection in Figure 9. This same rule was obtained earlier in Equation (4).

Figure 11: Dendrogram for hierarchical clustering on FMC data.

Figure 12: Two clusters found using hierarchical clustering on FMC data.

Table 9: Unsupervised rules obtained using flexible pattern mining on FMC dataset for the selection shown in Figure 9.

SEL	UNSEL	Rule
79.36	3.17	$S_1 > 1$
94.95	45.75	$S_2 > 1$
83.49	54.99	$A_2 = 90$
82.11	54.38	$A_1 = 90$
55.05	41.92	$B_2 = 5$
51.83	40.05	$A_5 < 92$
57.80	46.06	$A_4 < 92$
49.08	41.43	$A_3 < 92$
76.15	78.50	$B_1 = 5$

Figure 13: Pattern 13 of BFL-0.1 data obtained using sequential pattern mining.

Figure 14: Selected solutions for BFL-0.1 data.

This application study shows that flexible pattern mining is capable of generating simple rules, like those from sequential pattern mining, while ordering them by the importance of the variables, in a manner similar to classification trees.

4.2. BFL-0.1 Optimization Dataset

For this dataset, sequential pattern mining with a minimum support condition of 40% presents 18 patterns. However, none of them have an interesting structure in the objective space. For example, one of the obtained patterns, $B_2 = 1 \wedge B_{11} = 1$, with 44.06% support has solutions supporting it as shown in Figure 13. Such patterns, though valid, are of little use to decision makers, who often make decisions by looking at the objective space.

For applying classification trees, we use the selection shown in Figure 14. Three dominant rules are obtained from the unpruned tree, which all have the same first five terms, as depicted below:

$$T_1 < 4050 \wedge T_7 < 4050 \wedge T_4 < 4050 \wedge T_{10} < 4050 \wedge T_{12} < 4050 \wedge \dots \quad (5)$$

The terms indicate that the distinguishing features of the selected region (82.19% of the selection) are that processing times T_1 , T_4 , T_7 , T_{10} and T_{12} are either 3600 or 3900 seconds.

Hybrid learning with hierarchical clustering generates a dendrogram, as shown in Figure 15, based on which three clusters are used to obtain the partitioning shown in Figure 16. A total of 28 dominant rules are extracted from the unpruned tree corresponding to the classes. Due to the length of the obtained rules, we only show the first few terms of the most dominant rule for each class in Table 10. As before, **SEL** represents the support for the current class and **UNSEL** represents the support for all other classes combined.

Note that the clusters seen in Figure 16 do not agree with the structure of the objective space. Clearly, there are five distinct clusters. Since these clusters are elongated, certain points in one cluster can be closer to points from other clusters than to their

Figure 15: Dendrogram for hierarchical clustering on BFL-0.1 data.

Figure 16: Three clusters found using hierarchical clustering on BFL-0.1 data.

Table 10: Dominant rules obtained using hybrid learning with hierarchical clustering on BFL-0.1 dataset. The classes are shown in Figure 16.

CLASS	SEL	UNSEL	Rule
1	76.92	0.00	$T_8 < 4350 \wedge T_{10} < 3750 \wedge T_4 < 4050 \wedge \dots$
2	34.99	≈ 0.00	$T_8 < 4350 \wedge T_{10} \geq 3750 \wedge T_{13} < 4350 \wedge \dots$
3	71.72	≈ 0.00	$T_8 \geq 4350 \wedge T_7 \geq 4050 \wedge T_{12} \geq 4050 \wedge \dots$

own cluster center. This misleads distance-based clustering algorithms like hierarchical clustering, especially when using a euclidean distance measure. A decision maker would be more interested in knowledge that conforms to the visually obvious structure of the objective space. Therefore, we now perform hybrid learning with the DBSCAN clustering algorithm. With $MinPts = 20$, five clusters are obtained, as shown in Figure 17, that exactly match the structure of solutions. The first few terms of the most dominant rules for the five classes are shown in Table 11. Note that even with such well-defined clusters, the support for the dominant rules in the first four classes is relatively low. Moreover, the dominant rule for Class 4 does not exactly match with either that for Class 1 in Table 10 or with Equation (5), even though they represent the same region in the objective space. This illustrates a fundamental drawback of classification tree based techniques; the rules

Table 11: Dominant rules obtained using hybrid learning with DBSCAN on BFL-0.1 dataset. The classes are shown in Figure 17.

CLASS	SEL	UNSEL	Rule
1	49.89	0.00	$T_8 \geq 4350 \wedge T_{12} \geq 4350 \wedge T_6 \geq 4650$
2	30.16	0.00	$T_8 < 4350 \wedge T_1 \geq 4050 \wedge T_1 \geq 4350 \wedge \dots$
3	42.07	0.00	$T_8 < 4350 \wedge T_1 \geq 4050 \wedge T_1 < 4350 \wedge \dots$
4	40.71	0.00	$T_8 < 4350 \wedge T_1 < 4050 \wedge T_{10} < 3750 \wedge \dots$
5	100.0	0.00	$T_8 < 4350 \wedge T_1 < 4050 \wedge T_{10} < 3750 \wedge \dots$

Figure 17: Five clusters found using DBSCAN on BFL-0.1 data.

Figure 18: Rule $T_i < 4200 \forall i$ obtained using flexible pattern mining on BFL-0.1 data for the selection in Figure 14.

depend not only on the selection (or current class), but also on the unselected solutions (or other classes).

Flexible pattern mining circumvents this problem by generating multiple rules based solely on user-selection, but sorting them based on the ratio of their support in the selected region to that in the unselected region. For the selection shown in Figure 14, we obtain 25 simple rules (not shown here) in the decreasing order of their support ratios. The first 13 rules indicate that the processing times of all the 13 workstations in the BFL-0.1 line should be less than 4200, i.e. $T_i < 4200 \forall i$ (or $T_i \in \{3600, 3900\} \forall i$). Moreover, all 13 rules have 100% support in the selected region and very low support among the unselected solutions, which indicates that combining them could yield additional knowledge about the problem. Indeed, as Figure 18 shows, the combined rule is not only applicable to the selected region with 100% support, but also to all solutions above the chosen region.

This application problem illustrates that flexible pattern mining can reveal patterns even when the user partially selects the region in which they are present. To further support this claim, we use two other user-selections shown in Figures 19 and 21. The (combined) rules corresponding to these selections are $T_i < 4500 \forall i$ and $T_i < 4800 \forall i$ respectively, shown in Figures 20 and 22 respectively. From these rules, it can be seen that in order to improve the level of throughput significantly, say from Throughput > 0.75 to Throughput > 0.8 , the processing times of all the workstations have to be reduced simultaneously, from $T_i < 4800 \forall i$ to $T_i < 4500 \forall i$. Thus, the performance of BFL-0.1 production line is significantly affected by the processing times of all the workstations.

4.3. BFL-0.5 Optimization Dataset

As in the case of BFL-0.1, sequential pattern mining again generates valid but uninteresting patterns. With a minimum support condition of 30%, we obtain 11 patterns, all of which only have one variable (one-itemset sequence). Two of the patterns are shown in Figures 23 and 24.

Figure 19: Selected solutions for BFL-0.1 data.

Figure 20: Rule $T_i < 4500 \forall i$ obtained using flexible pattern mining on BFL-0.1 data for the selection in Figure 19.

Figure 21: Selected solutions for BFL-0.1 data.

Figure 22: Rule $T_i < 4800 \forall i$ obtained using flexible pattern mining on BFL-0.1 data for the selection in Figure 21.

Figure 23: Pattern $B_{12} = 5$ obtained using sequential pattern mining on BFL-0.5 dataset.

Figure 24: Pattern $T_6 = 3600$ obtained using sequential pattern mining on BFL-0.5 dataset.

Figure 25: Selected solutions for BFL-0.5 data.

Figure 26: Pattern shown in Equation (6).

Selecting the solutions as shown in Figure 25 results in 19 patterns for a minimum support condition of 60%, each containing either one or two variables (one or two-itemset sequences). Overall, the patterns, with varying degrees of support, indicate that the machine processing times are equal to 4800 (upper bound of T_i) in the selected region. For example, one of the patterns,

$$T_3 = 4800 \wedge T_{12} = 4800, \quad (6)$$

having a support of 63.64%, is shown in Figure 26. This rule corresponds to no investment in the processing times of workstations 3 and 12.

For the same selection of solutions, classification tree generates two dominant rules of support 28.36% and 9.17%. The first two terms of the former rule,

$$T_3 \geq 4650 \wedge T_{12} \geq 4650 \wedge \dots,$$

Figure 27: Dendrogram for hierarchical clustering on BFL-0.5 data.

Figure 28: Three clusters found using hierarchical clustering on BFL-0.5 data.

Table 12: Dominant rules obtained using hybrid learning with hierarchical clustering on BFL-0.5 dataset. The classes are shown in Figure 28.

CLASS	SEL	UNSEL	Rule
1	54.75	≈ 0.00	$T_1 \geq 4050 \wedge T_{13} \geq 4350 \wedge T_9 \geq 4350 \wedge \dots$
2	25.37	0.00	$T_1 \geq 4050 \wedge T_{13} < 4350 \wedge T_8 \geq 3750 \wedge \dots$
3	55.81	≈ 0.00	$T_1 < 4050 \wedge T_{13} < 4050 \wedge T_2 < 4050 \wedge \dots$

confirm the pattern in Equation (6), since all processing times have an upper bound of 4800. However, note the huge difference in support values. The reason for the lower support of the above rule is that classification trees are built to *distinguish* between two or more classes, whereas sequential pattern mining (even with user-preference) only finds patterns in the selected region. This is also the reason why the former method is supervised, while the latter is unsupervised. From the third term onwards, the above rule tries to distinguish the selected region from the unselected part. Because of the tendency of decision trees to overfit, the rule becomes lengthy and the end node, though purely belonging to one class, has low support (very few solutions) for it.

Hybrid learning is also based on classification trees, but the classes here are determined in an unsupervised manner (i.e., through clustering). If the clustering method used succeeds in finding the *natural* clusters in the data, then meaningful and possibly compact rules can be obtained. For this dataset, hierarchical clustering generates the dendrogram shown in Figure 27. With three clusters, the grouping of solutions is as presented in Figure 28. The classification tree obtained for the classes shown in the figure contains 29 dominant rules, of which the most dominant for each class is shown in Table 12. The fundamental problem with classification trees is again observed here. In order to distinguish between the three classes, the algorithm generates lengthy rules that are too specific and overfit, thereby leading to low support values in each class.

DBSCAN fails to identify any clusters in this dataset for the recommended values of *MinPts*. This can be explained by the lack of visually distinct clusters in the objective space. DBSCAN sees a uniform density of points and hence fails to partition the dataset.

Table 13: Unsupervised rules obtained using flexible pattern mining on BFL-0.5 dataset.

Selection: Figure 29			Selection: Figure 31		
SEL	UNSEL	Rule	SEL	UNSEL	Rule
100.00	34.89	$T_1 < 4200$	100.00	81.72	$T_1 < 4800$
100.00	38.10	$T_{12} < 4200$	100.00	84.62	$T_{11} < 4800$
100.00	39.69	$T_8 < 4200$	100.00	85.19	$T_{12} < 4800$
100.00	42.23	$T_9 < 4200$	100.00	86.20	$T_7 < 4800$
100.00	43.03	$T_{10} < 4200$	100.00	88.16	$T_3 < 4800$
100.00	43.71	$T_6 < 4200$	100.00	88.29	$T_9 < 4800$
100.00	59.17	$T_{11} < 4500$	100.00	88.97	$T_8 < 4800$
100.00	63.16	$T_7 < 4500$	100.00	89.04	$T_{10} < 4800$
100.00	64.66	$T_3 < 4500$	100.00	90.84	$B_{11} > 1$
100.00	66.81	$T_5 < 4500$	100.00	91.01	$T_4 < 4800$
100.00	69.30	$T_2 < 4500$	100.00	92.21	$T_5 < 4800$
100.00	69.55	$T_4 < 4500$	100.00	92.41	$T_2 < 4800$
100.00	97.62	$B_{11} < 15$	100.00	92.46	$B_3 > 1$
100.00	97.69	$B_7 < 14$	100.00	92.77	$B_7 > 1$
100.00	97.84	$B_9 < 17$	100.00	93.49	$T_6 < 4800$
100.00	97.91	$B_{12} < 15$	100.00	95.87	$B_9 < 13$
100.00	97.99	$B_{10} < 13$	100.00	96.87	$B_6 < 13$
100.00	98.16	$B_8 < 13$	100.00	97.49	$B_4 < 12$
100.00	98.45	$B_3 < 14$	100.00	97.84	$B_{12} < 14$
100.00	98.45	$B_4 < 14$	100.00	98.04	$B_1 < 13$
100.00	98.85	$B_6 < 17$	100.00	98.11	$B_2 < 11$
100.00	98.87	$B_5 < 15$	100.00	98.12	$B_5 < 12$
100.00	99.20	$B_1 < 18$	100.00	98.34	$B_8 < 13$
100.00	99.25	$B_2 < 16$	100.00	98.43	$B_{10} < 14$

As discussed earlier, BFL-0.5 uses the same simulation model as BFL-0.1. Therefore, some similarities between their optimization datasets are expected. In order to investigate this further, we use two user-selections resembling those in Figures 19 and 21. These are shown in Figures 29 and 31, respectively. All the rules obtained in each case for a minimum support condition of 50% are presented in Table 13, again in decreasing order of the ratio of **SEL** to **UNSEL**. The **SEL** values are 100% for all the rules, which in itself is interesting. The first 12 rules in each case are not exactly the same, but closely resemble those shown in Figures 20 and 22 (i.e., $T_i < 4500$ and $T_i < 4800$). The regions where the combination of the first 12 rules is applicable in the two selections are illustrated in Figures 30 and 32. These also resemble the coverage of rules as seen earlier in Figures 20 and 22.

This application problem again illustrates how flexible pattern mining can work with user-selections that only partially cover an actual hidden rule cluster. The unique order of the rules and the ability to combine them progressively helps uncover such hidden clusters, as seen here.

Figure 29: Selected solutions for BFL-0.5 data.

Figure 30: Combination of the top 12 rules on the left in Table 13.

Figure 31: Selected solutions for BFL-0.5 data.

Figure 32: Combination of the top 12 rules on the right in Table 13.

Table 14: Patterns obtained using sequential pattern mining on ACM dataset.

ID	Support	Pattern
1	76.53	$Ip_{Op1O} = 1$
2	57.95	$Ip_{Op1N} = 1$
3	57.13	$Ip_{Op1H} = 1$
4	53.90	$Ip_{Op1N} = 1 \wedge Ip_{Op1O} = 1$
5	50.73	$Ip_{Op1O} = 1 \wedge Ip_{Op1H} = 1$
6	41.96	$Ip_{Op1N} = 1 \wedge Ip_{Op1H} = 1$
7	40.35	$Ip_{Op1N} = 1 \wedge Ip_{Op1O} = 1 \wedge Ip_{Op1H} = 1$

Figure 33: Pattern 4 of ACM data obtained using sequential pattern mining (cost vs. buffer capacity).

Figure 34: Pattern 4 of ACM data obtained using sequential pattern mining (cost vs. investment).

4.4. ACM Optimization Dataset

Sequential pattern mining with a minimum support condition of 40% generates the seven patterns shown in Table 14. All the patterns lack a clear structure in the objective space. Pattern 4, for example, is spread out across visibly different regions of the objective space as presented in Figures 33 and 34. In a previous study that used a distance-based regression tree learning method (Dudas et al., 2014), it was shown that the improvements of $Op1O$ and $Op1N$ and buffer capacity conditions $B_2 > 1$ and $B_8 > 1$ are essential to guarantee solutions that lie in the preferable low cost regions of the objective space. The patterns generated by sequential pattern mining agree in terms of improvement variables, but do not accurately describe the lower cost solutions, since the method is incapable of capturing rules.

Classification trees provide better insights in this problem, because the clusters of solutions in the objective space correspond to clusters of rules. We used three different preferred regions, shown in Figures 35a, 35b and 35c, and in each case obtained very simple trees. The dominant rules are shown together in Table 15. These rules closely resemble those obtained through a regression tree-based approach in Ng et al. (2012).

Due to the elongated shape of clusters in the objective space, hierarchical clustering

Figure 35: Three different user-selections for ACM data.

Table 15: Dominant rules obtained using classification trees on ACM dataset.

Selection	SEL	UNSEL	Rule
Figure 35a	64.56	0.00	$Ip_{Op1O} < 0.5 \wedge B_2 \geq 1.5 \wedge B_8 \geq 1.5 \wedge \dots$
	17.99	≈ 0.00	$Ip_{Op1O} < 0.5 \wedge B_2 \geq 1.5 \wedge B_8 \geq 1.5 \wedge \dots$
Figure 35b	100.0	0.00	$B_8 < 1.5 \wedge B_2 \geq 1.5$
Figure 35c	99.42	0.00	$B_2 < 1.5$

faces the same issues as with the BFL-0.1 dataset. The dendrogram in Figure 36 suggests the use of three clusters, yet the clustering obtained in Figure 37 does not correspond to the accepted notion of clusters. As a result, even the most dominant rules are found to have very low support.

On the other hand, DBSCAN performs well on this dataset. The clusters obtained with $MinPts = 20$ are shown in Figure 38. Table 16 presents the most dominant rules for the three classes. Note the similarity of rules corresponding to classes 2 and 3 with those in Table 15.

Next, we use flexible pattern mining on the selections shown in Figures 35a, 35b and 35c. For a minimum support condition of 50%, we obtain 40 unsupervised rules in each case, sorted in the decreasing order of ratio of support in the selected set to that in the unselected set. However, in all three cases, the topmost rule alone has a very high support ratio, so that its combination with other rules down the list makes little sense. Hence, the topmost rule for each selection is shown in Table 17 and the solutions supporting them are shown in Figures 39, 40 and 41. Here, the distinction between the

Table 16: Dominant rules obtained using hybrid learning with DBSCAN on ACM dataset. The classes are shown in Figure 38.

CLASS	SEL	UNSEL	Rule
1	78.84	0.00	$B_2 \geq 1.5 \wedge B_8 \geq 1.5 \wedge B_{16} < 23.5 \wedge \dots$
2	95.93	0.00	$B_2 < 1.5 \wedge B_{15} < 18.5 \wedge B_{16} < 16 \wedge \dots$
3	90.06	0.00	$B_2 \geq 1.5 \wedge B_8 < 1.5 \wedge B_{27} < 31.5 \wedge \dots$

Figure 36: Dendrogram for hierarchical clustering on ACM data.

Figure 37: Three clusters found using hierarchical clustering on ACM data.

Figure 38: Three clusters found using DBSCAN on ACM data.

Figure 39: Most distinguishing rule $Ip_{Op1O} = 0$ obtained using flexible pattern mining on ACM data for the selection in Figure 35a.

Table 17: Unsupervised rules obtained using flexible pattern mining on ACM dataset.

Selection	SEL	UNSEL	Rule
Figure 35a	99.63	3.98	$Ip_{Op1O} = 0$
Figure 35b	100.0	0.17	$B_8 = 1$
Figure 35c	99.42	0.00	$B_2 = 1$

Figure 40: Most distinguishing rule $B_8 = 1$ obtained using flexible pattern mining on ACM data for the selection in Figure 35b.

Figure 41: Most distinguishing rule $B_2 = 1$ obtained using flexible pattern mining on ACM data for the selection in Figure 35c.

three regions becomes clear. The obtained rules also confirm that low cost solutions must at least satisfy $B_2 > 1$, $B_8 > 1$ and $Ip_{Op10} \neq 0$. Both sequential pattern mining and clustering-based classification trees could not clearly identify these conditions.

5. Conclusions

The main research goal of this paper was to develop data mining methods for knowledge discovery from multi-objective optimization datasets. As opposed to most existing techniques, the developed methods can:

1. use preferences in the objective space to discover knowledge in the decision space,
2. handle discrete variables (including ordinal and nominal variables),
3. represent knowledge in an explicit form.

The explicit knowledge generated by the data mining methods can be used to form the knowledge base for an expert system. Such a system can in turn enhance the performance of a multi-objective optimization algorithm in preferred regions by, for example, confining search to the patterns and rules that are unique to those regions. We intend to address this topic in our future work.

The four methods, (i) sequential pattern mining, (ii) clustering + classification trees, (iii) hybrid learning, and (iv) flexible pattern mining, are all extended from existing data mining techniques. These extensions are not limited to the case studies in this paper and should be applicable to all kinds of discrete numerical datasets. It is well understood in the data mining community that most frequent does not necessarily mean most interesting. Our experiments with sequential pattern mining illustrate this in at least two of the case studies (BFL-0.1 and BFL-0.5). Nevertheless, we recommend that this method should be one of the first data mining techniques to be applied on a given dataset, due to its efficiency in handling large numbers of solutions and variables. As suspected, classification trees generated lengthy overfit rules in all cases. However, being the only

supervised algorithm capable of generating explicit rules, they can be useful when trying to understand how certain solutions differ from others. We also introduced the concept of hybrid learning, which combines unsupervised clustering with classification trees to find natural clusters in the data. The approach was found to be reasonably effective, subject to the performance of the clustering method. However, since the unsupervised component is independent, the method is not restricted to hierarchical clustering or DB-SCAN. Other clustering algorithms may be more suited for certain problems. In our experiments, we found flexible pattern mining to be the most consistent. The discovery of rules is unsupervised, but the ordering of these rules is based on how well they distinguish the preferred solutions from the rest of the solutions. This allows the user to combine rules progressively, starting from the most distinguishing rule.

With regard to the applications themselves, it can be seen that, in general, all the methods associate specified regions of the objective space with similar rules and patterns. In fact, for most user-selections, the obtained rules are found to be intuitive. Overall, flexible pattern mining is the most effective of the four methods. As observed in the BFL-0.5 case study, flexible pattern mining is able to find important rules, even when the user selects only a part of the true rule cluster. The added advantage of combining rules that are ordered by their distinguishing ability, gives flexible pattern mining an edge over the other methods.

Acknowledgments

The first author acknowledges the financial support received from KK-stiftelsen (Knowledge Foundation, Stockholm, Sweden) for the ProSpekt 2013 project KDISCO. All the authors would like to thank Dr. Leif Pehrsson (Volvo Car Corporation, Skövde, Sweden) for sharing the simulation models used in this study.

References

- Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on Very Large DataBases, VLDB* (pp. 487–499).
- Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering* (pp. 3–14). IEEE.
- Bernedixen, J., & Ng, A. H. C. (2014). Practical production systems optimization using multiple-choice sets and manhattan distance based constraints handling. In *Proceedings of the 12th Industrial Simulation Conference* (pp. 97–103).
- Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. Wiley.
- Deb, K., Agarwal, S., Pratap, A., & Meyarivan, T. (2002). A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6, 182–197.
- Deb, K., & Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, 9, 115–148.
- Deb, K., Bandaru, S., Greiner, D., Gaspar-Cunha, A., & Tutum, C. C. (2014). An integrated approach to automated innovization for discovering useful design principles: Case studies from engineering. *Applied Soft Computing*, 15, 42–56.
- Deb, K., & Srinivasan, A. (2006). Innovization: Innovating design principles through optimization. In *Proceedings of the 8th annual conference on genetic and evolutionary computation, GECCO '06* (pp. 1629–1636). ACM.
- Deb, K., & Sundar, J. (2006). Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th annual conference on genetic and evolutionary computation, GECCO '06* (pp. 635–642). New York, NY, USA: ACM.

- Dudas, C., Frantzén, M., & Ng, A. H. C. (2011). A synergy of multi-objective optimization and data mining for the analysis of a flexible flow shop. *Robotics and Computer-Integrated Manufacturing*, 27, 687–695.
- Dudas, C., Ng, A. H. C., & Boström, H. (2015). Post-analysis of multi-objective optimization solutions using decision trees. *Intelligent Data Analysis*, 19, 259–278.
- Dudas, C., Ng, A. H. C., Pehrsson, L., & Boström, H. (2014). Integration of data mining and multi-objective optimization for decision support in production system development. *International Journal of Computer Integrated Manufacturing*, 27, 824–839.
- Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of KDD-96* (pp. 226–231).
- Fonseca, C. M., Fleming, P. J. et al. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms* (pp. 416–423). Morgan Kaufman Publishers.
- Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The Elements of Statistical Learning*. Springer.
- Hopp, W. J., & Spearman, M. L. (2011). *Factory physics*. Waveland Press.
- Law, A. M., & McComas, M. G. (2000). Simulation-based optimization. In *Proceedings of the 32nd conference on Winter simulation* (pp. 46–49).
- Liao, S.-h. (2003). Knowledge management technologies and applications - literature review from 1995 to 2002. *Expert systems with applications*, 25, 155–164.
- Miettinen, K. (1999). *Nonlinear multiobjective optimization*. Kluwer Academic Publishers.
- Ng, A. H. C., Bernedixen, J., Moris, M. U., & Jägstam, M. (2011). Factory flow design and analysis using internet-enabled simulation-based optimization and automatic model generation. In *Proceedings of the 2011 Winter Simulation Conference* (pp. 2181–2193). IEEE.
- Ng, A. H. C., Dudas, C., Pehrsson, L., & Deb, K. (2012). Knowledge discovery in production simulation by interleaving multi-objective optimization and data mining. In *Proceedings of the 5th Swedish Production Symposium (SPS'12), Linköping, Sweden* (pp. 6–8).
- Ng, A. H. C., Urenda Moris, M., Svensson, J., Skoogh, A., & Johansson, B. (2007). FACTS Analyser: An innovative tool for factory conceptual design using simulation. In *The Swedish Production Symposium in Göteborg 28-30 August 2007* (pp. 1–8).
- Pehrsson, L. (2013). *Manufacturing Management and Decision Support Using Simulation-Based Multi-Objective Optimisation*. Ph.D. thesis De Montfort University, Leicester, UK.
- Pehrsson, L., & Ng, A. H. C. (2011). An applied framework for simulation-based multi-objective production system optimization within production system development. In *Proceedings of the Industrial Simulation Conference, ISC'2011* (pp. 1–8).
- Pehrsson, L., Ng, A. H. C., & Stockton, D. (2013). Industrial cost modelling for multi-objective optimisation of production systems. *International Journal of Computer and Industrial Engineering*, 66, 1036–1048.
- Purshouse, R. C., Deb, K., Mansor, M. M., Mostaghim, S., & Wang, R. (2014). A review of hybrid evolutionary multiple criteria decision making methods. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1147–1154).
- Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1, 81–106.
- Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.
- Srikant, R., & Agrawal, R. (1996). Mining sequential patterns: Generalizations and performance improvements. In G. Apers, Peter and Bouzeghoub, Mokrane and Gardarin (Ed.), *Advances in Database Technology EDBT '96* (pp. 1–17). Springer.
- Sugimura, K., Obayashi, S., & Jeong, S. (2010). Multi-objective optimization and design rule mining for an aerodynamically efficient and stable centrifugal impeller with a vaned diffuser. *Engineering Optimization*, 42, 271–293.
- Tempelmeier, H. (2003). Practical considerations in the optimization of flow production systems. *International Journal of Production Research*, 41, 149–170.
- Xu, R., & Wunsch, D. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16, 645–678.