



Production, Manufacturing and Logistics

Mining Pareto-optimal modules for delayed product differentiation

Zhe Song, Andrew Kusiak*

Intelligent Systems Laboratory, MIE, 3131 Seamans Center, The University of Iowa, Iowa City, IA 52242-1527, USA

ARTICLE INFO

Article history:

Received 1 June 2008

Accepted 12 February 2009

Available online 20 February 2009

Keywords:

Data mining

Evolutionary computations

Mass customization

Modularity

Delayed product differentiation

Multi-objective optimization

ABSTRACT

This paper presents a framework for finding optimal modules in a delayed product differentiation scenario. Historical product sales data is utilized to estimate demand probability and customer preferences. Then this information is used by a multiple-objective optimization model to form modules. An evolutionary computation approach is applied to solve the optimization model and find the Pareto-optimal solutions. An industrial case study illustrates the ideas presented in the paper. The mean number of assembly operations and expected pre-assembly costs are the two competing objectives that are optimized in the case study. The mean number of assembly operations can be significantly reduced while incurring relatively small increases in the expected pre-assembly cost.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

In the last decade, manufacturing has been moving from mass production to mass customization. The concept of developing product families and modular architectures are of interest to manufacturing companies in the quest to meet diverse customer requirements while maintaining an economy of scale (Farrell and Simpson, 2003).

In order to implement mass customization successfully, manufacturing companies have to balance their efforts through the life-cycle of their products. Roughly speaking, a product life-cycle can be divided into design and manufacturing. For each stage, different techniques can be adopted to implement the mass customization strategy. For example, during the design stage, modular design and product family concepts are widely used (Fujita and Yoshida, 2004; Huang and Kusiak, 1998; Jiao and Tseng, 1999; Kreng and Lee, 2004; Kusiak and Huang, 1996; Yigit et al., 2002). Basic ideas are to provide diverse products with low technical varieties. Different products can be easily obtained by different combinations of modules.

During manufacturing stage, commonality, postponement (Ma et al., 2002; Swaminathan and Tayur, 1999) are important methods for managing product diversity and maintaining low manufacturing costs. However manufacturing performance can still be affected by product variety (MacDuffie et al., 1996) especially when customer orders are so diverse. Fu et al. (2006) studied the inventory and production decisions for a single product with

uncertain demand and assembly capacity. Lee and Tang (1997) considered the benefits and costs of the delayed differentiation strategy for two products. Hsu and Wang (2004) presented a dynamic programming model for determining the delayed differentiation point in a multi-stage production system. Jewkes and Alfa (2008) used a queueing model to analyze the benefits of delayed differentiation. Gunasekaran and Ngai (2009) reviewed models of make-to-order supply chain. Gupta and Benjaafar (2004) studied the trade-offs between the delayed differentiation, make-to-order, and make-to-stock strategies in the context of a multi-stage assembly system with limited production capacity. These researches provide insights into the manufacturing process under different configurations.

Most research literatures are focused on modular product design or product family design. But in order to improve manufacturing performance, such as delivery time, more researches are needed to develop new techniques to support mass customization in manufacturing stage.

Once a product is designed, it is ready to be manufactured and assembled according to customer orders. Internet-based configuration systems have been gaining popularity in recent years. Customers are able to configure products by selecting desired features, which result in many unique configurations. Maintaining a large number of different product configurations increases production complexity and extends delivery lead time (Da Cunha et al., 2007; Swaminathan and Tayur, 1998). To shorten the lead time, companies may follow a delayed differentiation strategy. The delayed differentiation strategy is a compromise between MTO (make-to-order) and MTS (make-to-stock) strategies (Gupta and Benjaafar, 2004; Swaminathan and Tayur, 1999, 1998). Following

* Corresponding author.

E-mail address: andrew-kusiak@uiowa.edu (A. Kusiak).

the MTO strategy, a company assembles a final product only when a customer order arrives. In the MTS strategy, a company pre-assembles a set of complete products, and once a customer order arrives, the company will search the pre-assembled products. If a match is found, the product is shipped; otherwise, the customer order is assembled from the basic components. In the delayed differentiation strategy, maintaining a set of pre-assembled modules (sub-assemblies or semi-finished products) shortens the final assembly time (Da Cunha et al., 2007; Swaminathan and Tayur, 1998). Once a customer order arrives, the modules needed to assemble a product are retrieved. Then additional components may be assembled with the retrieved modules, if necessary, to make a complete product. Thus customer satisfaction is improved by reducing final assembly time.

Jiao and Zhang (2004) applied a genetic algorithm to solve a product portfolio optimization problem for generating complete product configurations. However finding semi-finished products is of interest for the manufacturing company. Determining a set of modules to be pre-assembled and stocked is a complex optimization problem (Da Cunha et al., 2007; Fu et al., 2006; Jiao and Zhang, 2004; Kusiak and Huang, 1996; Swaminathan and Tayur, 1998) in which multiple costs have to be considered and balanced. Da Cunha et al. (2007) developed a linear cost function and heuristic algorithms to find the optimal module combinations that could reduce the mean number of assembly operations. Swaminathan and Tayur (1998) developed an optimization model to determine inventory levels for the modules they selected under variable demand and fixed assembly capacity.

This paper extends and combined the concepts discussed in the literature (e.g. Da Cunha et al., 2007; Swaminathan and Tayur, 1998), by finding optimal modules for a product family described by a set of attributes, where each attribute is associated with a set of components. A multi-objective function is used to find a solution minimizing the mean number of assembly operations and expected pre-assembly cost simultaneously. The optimal solutions considered in this research are semi-finished products. Compared with previous research (Da Cunha et al., 2007), this optimization framework allows users to simultaneously consider multiple cost functions. There is no need to assign a weight for each cost component. In contrast to Swaminathan and Tayur (1998), the proposed solution is more focused on the selection of modules by using historical sales data. Mean assembly time and pre-assembly costs are considered to derive the modules.

This paper is organized in six sections. Section 2 formulates the multi-objective optimization problem. An evolutionary computation algorithm for finding Pareto-optimal solutions is introduced in Section 3. Section 4 discusses the incorporation of historical sales data into the model formulated in Section 2. The three objectives to be achieved are addressed in Section 5. Section 6 discusses an industrial case study based on a truck data set.

2. Problem formulation

A product can be described by a set of attributes (features). Each attribute usually represents a set of components (e.g., a sub-assembly) which the customer selects to create his or her desired configuration. Unlike previous research (Da Cunha et al., 2007; Swaminathan and Tayur, 1998), their configuration (a complete product) is described by a set of components. Each configuration is determined by binary choices, a component is either selected or not selected to be in the configuration.

Definition 1. A product is described by n attributes. Each attribute A_i ($i = 1, \dots, n$) is a set of n_{A_i} components. $A_i(j)$ is the j th component of A_i , $j = 1, \dots, n_{A_i}$.

Based on Definition 1, the following relationships are established:

- In the absence of assembly constraints among the components, there are $\prod_{i=1}^n n_{A_i}$ unique configurations.
- A module could be a single component, a partial configuration, or a complete configuration. There are total $\prod_{i=1}^n (n_{A_i} + 1) - 1$ unique modules in a product.
- The set of modules with only one component is expressed as $\bigcup_{i=1}^n A_i$.
- The set of modules with two components is expressed as $\bigcup_{i=1}^n \bigcup_{j=1, i \neq j}^n A_i \times A_j$.
- The set of modules with three components is expressed as $\bigcup_{i=1}^n \bigcup_{j=1, k=1, i \neq j \neq k}^n A_i \times A_j \times A_k$.
- The set of all complete configurations is expressed as $A_1 \times \dots \times A_n$.

Definition 2. A module M_i ($i = 1, \dots, \prod_{i=1}^n (n_{A_i} + 1) - 1$) is a set of components drawn from different attributes.

There are two questions to be answered here. Which modules should be selected and assembled first? What are the inventory levels of these modules? There are numerous answers to these two questions depending on users' preferences. In this paper, the selected modules should have the ability to form a certain number of unique product configurations. In other words, most customer orders can be assembled from these modules. As we all know, assembling those modules costs a manufacturing company in terms of additional inventory and pre-assembly operations. Thus selecting those modules should also consider these costs.

The ultimate goal of forming these modules is to reduce the final assembly time and deliver products fast to customers. Besides, knowing these modules can help the company utilize its under-used manufacturing resources by avoiding waiting for customer orders.

Definition 3. Let M be a set of modules selected from $\prod_{i=1}^n (n_{A_i} + 1) - 1$ unique modules. $M(i)$ is the i th element of set M , $i = 1, \dots, n_M$. $M(i)$ could be regarded as a set of components from different attributes.

Based on the previous definitions, the optimal set of modules can be determined by solving an optimization problem with multi-objectives. Without loss of generality, it is assumed that there are g objectives to be minimized, i.e., y_1, \dots, y_g . Model (1) states that the optimal set of modules should minimize the g objective functions while satisfying all constraints:

$$\begin{aligned} \min_M \{y_1, y_2, \dots, y_g\} \\ \text{s.t. Constraints.} \end{aligned} \quad (1)$$

The overall solution presented in this paper can be described as following steps:

- Analyze historical sales data to estimate customer demand information and preferences.
- Formulate different objective functions (cost functions) and constraints which the selected modules will optimize.
- Use multi-objective evolutionary algorithms to solve the optimization problem.

Historical sales data contains important information about customer demand and preferences. For example, knowing which configurations (finished complete product) are frequently purchased by customers will help identify those semi-finished products. As a result, the final assembly time of complete products ordered by customers will be decreased.

Once historical sales data is analyzed, different cost functions and constraints can be formulated based on domain knowledge and application requirements. Then the optimization model is solved by multi-objective evolutionary algorithms to find those modules satisfying Pareto-optimal definitions.

3. Evolutionary algorithms and Pareto-optimal solutions

Optimization model (1) is a multi-objective optimization problem and it may have a set of optimal solutions. Before solving model (1), several definitions of Pareto-optimal solutions are restated for consistency with the definitions introduced in this paper. Based on the definition of Pareto-optimality, heuristic algorithms can be designed to derive the solutions.

Definition 4. Let M_1 and M_2 be two solutions of model (1); solution M_1 dominates solution M_2 , if and only if: $\forall i \in \{1, \dots, g\}$, $y_i(M_1) \leq y_i(M_2)$; $\exists i \in \{1, \dots, g\}$, $y_i(M_1) < y_i(M_2)$. Solutions which are not dominated by any other solutions in the entire search space are called non-dominated solutions.

Non-dominated solutions of model (1) are called Pareto-optimal and form a *Pareto-optimal set*. The corresponding objective values of the *Pareto-optimal solutions* form a *Pareto-optimal front*.

An evolutionary computation approach has proven to be an effective technique to solve multi-objective optimization problems (Deb, 2001). To solve model (1) by an evolutionary algorithm, solution M is encoded into a binary vector with $\prod_{i=1}^n (n_{A_i} + 1) - 1$ entries (genes) (Da Cunha et al., 2007). Each entry “1” denotes the presence of a module; “0” indicates the absence of a module. Recall that all modules are numbered from 1 to $\prod_{i=1}^n (n_{A_i} + 1) - 1$.

If $M(i) = 1$, module M_i exists in the solution. If $M(i) = 0$, module M_i does not exist in the solution. The number of modules in solution M is expressed as $\sum_{i=1}^{\prod_{i=1}^n (n_{A_i} + 1) - 1} M(i)$.

Once the solution is encoded by a binary vector, an evolutionary algorithm can be used to solve model (1) (Eiben and Smith, 2003). The basic operators of an evolutionary computation algorithm are:

- mutation;
- recombination;
- selection.

Mutation

One common mutation scheme for binary encoding is bitwise mutation, which allows each entry to flip (i.e., from 1 to 0 or 0 to 1) with some small probability p_{mu} .

Recombination

Two parent solutions are selected to generate a child. In this paper the uniform crossover operator is used to recombine two parents. Uniform crossover is implemented by generating $\prod_{i=1}^n (n_{A_i} + 1) - 1$ random numbers from the uniform distribution $[0, 1]$. For each gene, if the random value is smaller than a parameter p_{cross} (usually 0.5), the gene is inherited from the first parent, otherwise from the second.

Selection

Tournament selection (Eiben and Smith, 2003) with replacement is used in this paper to select promising individuals going into the next generation. The tournament size k is a predefined parameter to control the selection pressure.

3.1. The strength Pareto evolutionary algorithm (SPEA) (Zitzler and Thiele, 1999)

In this paper, a multi-objective evolutionary algorithm called SPEA is used as a basic algorithm to solve the optimization problem. The basic steps of SPEA are shown next.

The SPEA algorithm (Zitzler and Thiele, 1999)

- 1: Initialize three empty sets P , O and E . Randomly generate λ feasible individuals (solutions) to form the initial children population and place them in O .
- 2: Repeat until the stopping criterion is satisfied.
 - 2.1: Find non-dominated solutions in O and copy them into E . Remove dominated solutions in E . Reduce the size of E by clustering if necessary.
 - 2.2: Fitness assignment: assign fitness to individuals in O and E .
 - 2.3: Selection: use tournament selection to select individuals from $O \cup E$ and store them in P .
 - 2.4: Recombination: generate a new population O by selecting two parents in P .
 - 2.5: Mutation: mutate the individuals in O .
 - 2.6: Assign fitness to the individuals in O .

4. Sales data to estimate demand information

Analysis of the historical sales data may allow future configurations and customer preferences to be predicted. Assume the information considered in the three definitions presented next is available at the beginning of a selling period. Based on this information, the optimization knows which modules are important for reducing final assembly time, which modules will cause more pre-assembly costs.

Definition 5. Let x_D be a random variable of product demand following probability density function f .

Knowing the product demand information is important for planning inventory levels of the modules. Demand information is also needed for finding optimal modules. For example, modules with high demand should be considered and assembled first as they are important to reduce the expected final assembly time over all customer orders.

Definition 6. Let C be a subset of $A_1 \times \dots \times A_n$, i.e., $C \subset A_1 \times \dots \times A_n$. C is a set of predicted complete configurations the customers will purchase. $C(i)$ is the i th element of set C , $i = 1, \dots, n_C$. $C(i)$ could be regarded as a set composed of components from different attributes.

Although the manufacturer could produce $\prod_{i=1}^n n_{A_i}$ complete configurations, only a small portion of the total portfolio is sold during each period. Finding optimal modules for the total portfolio makes little sense, as many configurations are not going to be sold. Thus optimization guided by C may generate more practical results.

Definition 7. Let x_{A_i} be a discrete random variable whose value assumes any component of A_i . Let $p(x_{A_1}, \dots, x_{A_n})$ be the joint probability mass function of the random variables $(x_{A_1}, \dots, x_{A_n})$.

The joint distribution can be estimated from historical sales data. For a complete configuration $C(i)$, the demand probability is $p(C(i))$, and it is assumed that $\sum_{C(i) \in C} p(C(i)) = 1$. In other words, configurations not defined in C have zero demand probability.

5. Objectives for mining Pareto-optimal modules

Three different objectives of determining optimal modules are considered to illustrate the ideas presented in this paper. Intuitively the optimal modules should allow to construct certain number of complete product configurations to satisfy the customer preferences. Generally speaking, mean number of assembly operations and expected pre-assembly costs are two major competing objectives and thus it is interesting to show how they compete with each other during the optimization process. However, there are other objectives that could be considered for various industrial applications. For example, penalty costs due to customer waiting can be considered. Transportation and inventory costs could be used as potential objectives.

Objective 1: A certain percentage of unique complete configurations in C could be assembled from solution M .

A configuration $C(i)$ can be obtained from solution M , if and only if, there exists a subset m of M . The intersection of any two elements in the subset is an empty set, and the union of all elements in the subset is equal to $C(i)$, i.e., $\exists m \subseteq M$, with $m(k) \cap m(j) = \emptyset$, $\forall m(k), m(j) \in m$, $k \neq j$, and $\bigcup_{j=1}^{|m|} m(j) = C(i)$. The number of assembly operations needed to form $C(i)$ from m is $|m| - 1$. Although there are many potential subsets of M , in this paper m is selected to form $C(i)$ with the minimum assembly operation. Minimum assembly operations equal the minimum $|m|$, where $|\cdot|$ is the cardinality operator of a set.

Algorithm 1. Determine subset m with minimum assembly operations for configuration $C(i)$.

- 1: Arrange elements of M in descending order based on the cardinality.
- 2: Set $start = 1$;
- 3: **DO**
 - 3.1: Let $C(i)$ be the i^{th} complete configuration in C , empty m .
 - 3.2: **FOR** $j = start$ to n_M

IF $M(j) \subseteq C(i)$, **THEN** add $M(j)$ into m in descending cardinality order, and let $C(i) = C(i) - M(j)$.
 - 3.3: **END FOR**
 - 3.4: **IF** $C(i) = \emptyset$, **THEN** stop, output m ;
 - 3.5: **ELSE IF** m is not empty, **THEN** let $start$ be $m(1)$'s index in M plus 1, **ELSE** let $start = start + 1$.
 - 3.6: **WHILE** $start \leq n_M$.
- 4: **IF** $C(i) \neq \emptyset$, **THEN** let $m = \emptyset$, which implies that no solution exists.

Algorithm 1 is a greedy heuristic finding subset m , which is similar to the concept presented in Chvatal (1979). If **Algorithm 1** cannot output m , $C(i)$ cannot be assembled from solution M . The computational complexity of **Algorithm 1** is primarily determined by n_M and n . In the worst scenario, **Algorithm's 1** computational complexity is $O(n_M n^2)$. Although configurations in C may cover most customer orders, it is possible that a new customer configuration may emerge during the selling period. In order to make sure that all possible configurations can be assembled from the solution, all components of the attributes are incorporated into the solution. In other words, $A_i(j)$ must exist in solution M , for $j = 1, \dots, n_{A_i}$, $i = 1, \dots, n$. Another benefit of adding $A_i(j)$ into the solution is that the computational complexity of **Algorithm 1** will be significantly reduced.

Objective 2: Mean number of assembly operations of a complete configuration from solution M .

Definition 8. For module set M , $m_{C(i)}$ is a subset of M by which $C(i)$ can be assembled, $C(i) \in C$. If $m_{C(i)}$ is an empty set, $C(i)$ cannot be

obtained from M , and the number of assembly operations is defined as infinity ∞ . If all configurations in C are obtainable from M , the mean number of assembly operations is defined as $\sum_{C(i) \in C} (|m_{C(i)}| - 1)p(C(i))$.

If a configuration is directly assembled from the components of attributes, the number of assembly operations is $n - 1$. Thus the mean number of assembly operations can be scaled as $\sum_{C(i) \in C} \frac{|m_{C(i)}| - 1}{n - 1} p(C(i))$.

Objective 3: Expected pre-assembly cost.

Once the solution M is fixed, the inventory level of each module in M can be determined based on the demand x_D . Thus there is a pre-assembly cost related with solution M .

Definition 9. Let $h_{M(j)}$ be the pre-assembly cost of module $M(j)$, $q_{M(j)}$ be the inventory level of $M(j)$.

Here $h_{M(j)}$ is assumed to be proportional to some functional form of the number of components in $M(j)$. Different functions can be used, for example, $h_{M(j)}$ could be proportional to $e^{(|M(j)|)}$, or $|M(j)|^2$.

It is assumed that $C(i)$ can be obtained from M . The demand of $C(i)$ is $x_D p(C(i))$. Thus the demand of $M(j)$ can be calculated from $\sum_{i=1}^{n_C} \{x_D p(C(i)) : M(j) \in m_{C(i)}\}$. To satisfy all customer demands, $q_{M(j)}$ has to be equal to $\sum_{i=1}^{n_C} \{x_D p(C(i)) : M(j) \in m_{C(i)}\}$. Thus the pre-assembly cost is calculated as $\sum_{j=1}^{n_M} \{h_{M(j)} q_{M(j)}\}$. Since x_D is a random variable, the expected pre-assembly cost is $E\left(\sum_{j=1}^{n_M} \{h_{M(j)} q_{M(j)}\}\right)$.

To illustrate the concepts, two extreme cases are considered. In the first case, M equals $\bigcup_{i=1}^n A_i$. In this case all configurations are to be assembled from the components, which is the make-to-order (MTO) scenario. The mean number of assembly operations of a configuration is $\sum_{C(i) \in C} (n - 1)p(C(i)) = n - 1$. Since all modules in M are components, $m_{C(i)}$ is essentially equivalent to $C(i)$. Thus $q_{M(j)}$ is calculated from $x_D p(M(j))$. In this paper $h_{M(j)}$ is assumed to be proportional to an exponential function of the number of components in $M(j)$, i.e., $h_{M(j)} \propto e^{(|M(j)|)}$. Module $M(j)$ with more components calls for a greater assembly effort that translates into more space and storage space, and implies a higher risk once customer needs fluctuate. Note that in real applications the pre-assembly cost function need to be defined accordingly.

The expected pre-assembly cost is $E\left(\sum_{j=1}^{n_M} e^{x_D} p(M(j))\right) = E\left(e^{x_D} \sum_{j=1}^{n_M} \sum_{i=1}^{n_{A_i}} p(A_i(j))\right) = n e E(x_D)$, where $\sum_{j=1}^{n_{A_i}} p(A_i(j))$ should be equal to 1.

The second scenario is the make-to-stock (MTS), where all configurations are assembled first. M in this case is equal to C . The mean number of assembly operations of a configuration is 0 since all configurations have already been assembled. The expected pre-assembly cost is $E\left(\sum_{i=1}^{n_C} \{e^{x_D} p(C(i))\}\right) = e^{x_D} E(x_D)$. MTS usually incurs more risks compared to MTO.

Observation 1. If $h_{M(j)}$ is proportional to the number of components in $M(j)$, i.e., $h_{M(j)} \propto |M(j)|$, the expected pre-assembly costs of MTO and MTS are equal.

It is easy to see that Objectives 2 and 3 are competing with each other. In a delayed differentiation system, the mean number of assembly operations is reduced by stocking a certain number of modules, which increases the pre-assembly cost. Based on these three objectives, model (1) is instantiated as model (2).

$$\begin{aligned} \min_M \{y_2, y_3\} \\ \text{s.t. } y_1 = 1. \end{aligned} \tag{2}$$

In model (2), y_2 denotes Objective 2, the mean number of assembly operations of a configuration. y_3 represents Objective 3, the expected pre-assembly cost, and y_1 represents Objective 1. To

facilitate the discussion of the optimization results, Objective 1 is purposely considered as the constraint. $y_1 = 1$ requires that solutions be able to form all configurations in **C**.

6. Industrial case study

To illustrate the ideas introduced in this paper, a data set for trucks is used. The truck manufacturer is interested in reducing product complexity due to diverse customer orders. The manufacturer has focused on identifying complete product configurations to cover different customer segmentations. A set of predefined configurations, marketing strategies are used to steer customers to purchase the predefined configurations. Thus the number of unique customer orders decreases, and production complexity is reduced. The next level of mass customization is to use the assemble-to-order concept. The Pareto-optimal modules allow the company to analyze the trade-offs between the pre-assembly and the final assembly time and cost.

Table 1 shows that there are nine major attributes considered in this paper (i.e., $n = 9$). There are a total of $\prod_{i=1}^9 n_{A_i} = 34,020$ unique configurations with no assembly constraints. There are a total of $\prod_{i=1}^9 (n_{A_i} + 1) - 1 = 442,367$ unique modules. Based on the historical sales data, the expected annual demand $E(x_D)$ for the trucks is 1317. There are 292 unique configurations in **C**, and each configuration is associated with a demand probability calculated from the historical sales data.

The mean number of assembly operations and the expected pre-assembly cost for the MTO strategy are 8 and 32,219.79, respectively. For the MTS strategy, the mean number of assembly operations and the expected pre-assembly cost are 0 and 10,671,761.53.

Before performing computational experiments, some parameters of the evolutionary computation need to be fixed. Default values of the SPEA parameters are as follows: mutation probability is 0.0001, $\mu = 20$, $\lambda = 120$, tournament size $k = 4$, $p_{cross} = 0.5$. Some of these parameters will be changed in other experiments to observe their impact on the final solutions. In each figure to follow, the horizontal axis is the mean number of assembly operations, and the vertical axis is the expected pre-assembly cost.

The ratio between the parent size and the offspring size (i.e., μ/λ) is an important parameter in SPEA. In the experiments leading to Fig. 1, all parameters were fixed, while the ratio μ/λ changed. Based on Fig. 1, it is easy to see that the mean number of assembly operations and expected pre-assembly cost are competing objectives. The values of $\mu = 20$, $\lambda = 80$ resulted in a limited number of Pareto-optimal solutions. For $\mu = 20$, $\lambda = 120$ more Pareto-optimal solutions were found and with a lower mean number of assembly operations and expected pre-assembly cost compared to the solutions generated from $\mu = 20$, $\lambda = 80$. Based on Fig. 1, the best ratio found was $\mu = 20$, $\lambda = 120$, i.e., $\mu/\lambda = 1/6$.

The initial population size is another interesting parameter in SPEA. In this paper, different population sizes were evaluated while

Table 1
Description of product attributes.

Attribute	Description	Number of components
A ₁	Operator station	3
A ₂	AMS	3
A ₃	Comfort package	2
A ₄	Remote cylinder control	3
A ₅	Power take-off	3
A ₆	Hitch quick coupler and drawbar	5
A ₇	Hydraulic pump	2
A ₈	Rear axles	7
A ₉	Front axles	3

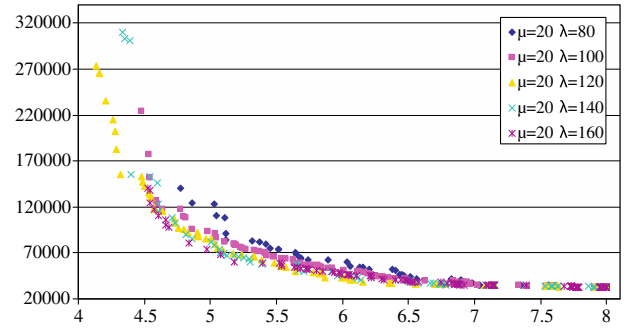


Fig. 1. Solutions in elite set *E* at 50th generation with fixed μ and different λ .

establishing $\mu/\lambda = 1/6$ and other parameters. Fig. 2 shows that increasing the population size enlarges the solution space and produces better quality solutions. However, increasing the population size significantly increases the computational time.

Mutation probability is interesting to investigate. After establishing $\mu/\lambda = 20/120$ and other parameters, different values of the mutation probability are used. Based on Fig. 3, when p_{mu} is increased, the algorithm finds more diverse solutions. However, when $p_{mu} = 0.00025$ or $p_{mu} = 0.0003$, in areas where the mean number of assembly operations is between 4 and 5 (as shown in Fig. 3), the algorithm tends to find worse solutions than for $p_{mu} = 0.0002$ or $p_{mu} = 0.00015$.

Computational experiments have been performed for different values of the tournament size $k = 2, 4, 6, 8, 10$ with other parameters at default values. Based on Fig. 4, $k = 4$ or $k = 6$ appear to be good choices. The solutions for $k = 8$ or $k = 10$ are of lower quality around areas where the mean number of assembly operations is between 4 and 5.5.

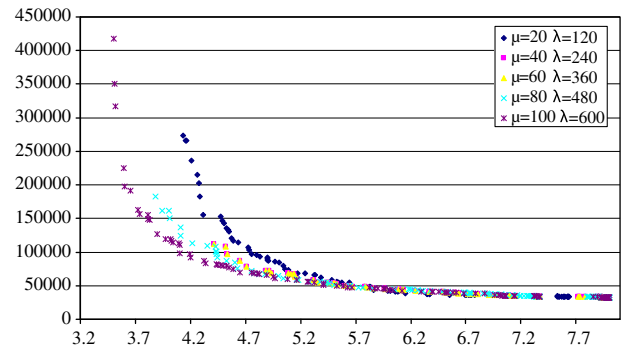


Fig. 2. Solutions in elite set *E* at 50th generation with fixed $\mu/\lambda = 1/6$, and different values of μ and λ .

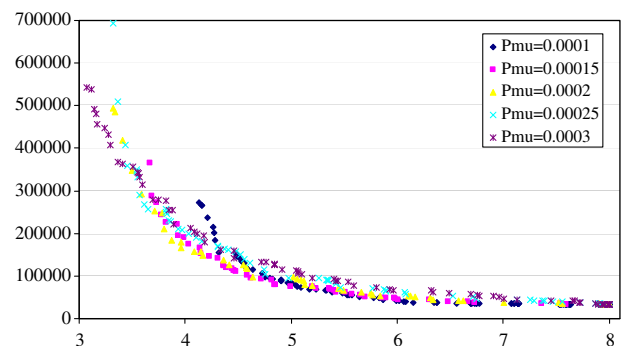


Fig. 3. Solutions in elite set *E* at 50th generation with different mutation probability p_{mu} .

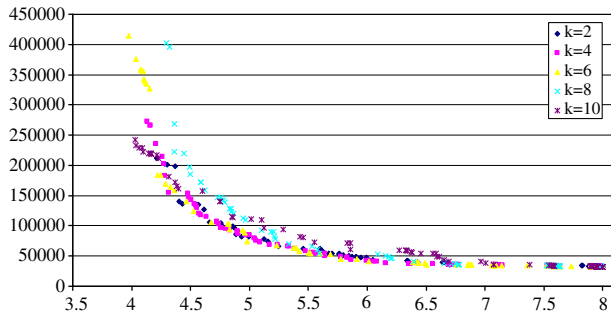


Fig. 4. Solutions in elite set E at 50th generation with tournament size k .

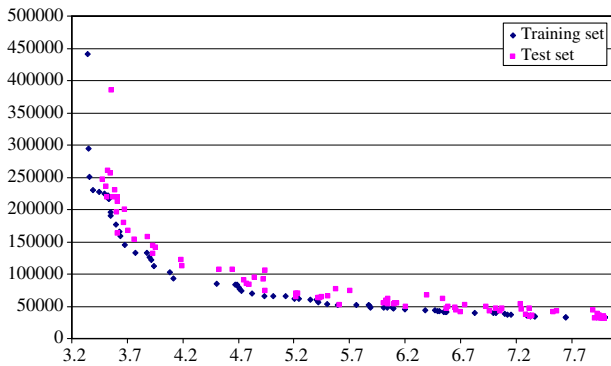


Fig. 5. Solutions in elite set E at 50th generation evaluated at training and test data sets.

Based on previous experiments, the algorithm parameters are heuristically fixed at $p_{mu} = 0.00015$, $\mu = 100$, $\lambda = 600$, $k = 4$, $p_{cross} = 0.5$. Then, the next-period sales data is used to evaluate the Pareto-optimal solutions obtained by solving the optimization model. This sales data reflects the actual demand information and configurations ordered by customers. The objective is to evaluate whether the Pareto-optimal solutions mined from previous sales data are still valid in the next-period. Let \mathbf{M}^* be the optimal solution generated by solving model (2) based on \mathbf{C} and the demand probabilities of each configuration in \mathbf{C} . Suppose the next-period sales data is available and the unique configurations sold are in \mathbf{C}_{test} . There are a total of 1271 configurations sold in this test data. The demand probability of each unique configuration $p(\mathbf{C}(i))$ is the ratio of the actual demand $D_{C(i)}$ of this configuration over the total demand of 1217.

For each Pareto-optimal solution \mathbf{M}^* , the mean number of assembly operations for the configurations in \mathbf{C}_{test} is computed from $\sum_{\mathbf{C}(i) \in \mathbf{C}_{test}} (|\mathbf{m}_{\mathbf{C}(i)}^*| - 1)p(\mathbf{C}(i))$, where $\mathbf{m}_{\mathbf{C}(i)}^*$ is a subset of \mathbf{M}^* from which $\mathbf{C}(i)$ can be assembled. $\mathbf{m}_{\mathbf{C}(i)}^*$ is determined by Algorithm 1. The expected pre-assembly cost of the test data set is $E\left(\sum_{j=1}^{|\mathbf{M}^*|} \{h_{\mathbf{M}^*(j)} q_{\mathbf{M}^*(j)}\}\right)$, where $h_{\mathbf{M}^*(j)} = e^{|\mathbf{M}^*(j)|}$, $E(q_{\mathbf{M}^*(j)}) = \sum_{i=1}^{|\mathbf{C}_{test}|} \{D_{C(i)} : \mathbf{M}^*(j) \in \mathbf{m}_{\mathbf{C}(i)}^*\}$. Fig. 5 shows that Pareto-optimal solutions are still valid for the test data set with some small degradation. In other words, it is feasible to mine the Pareto-optimal modules from the previous sales data and use them in the next selling period.

7. Conclusion

A general framework of mining Pareto-optimal modules from historical sales data was presented in this paper. Using an evolutionary computation algorithm allows modules to be selected based on multi-objective criteria. The Pareto-optimal solutions of-

fer opportunities to select the desired solutions based on domain knowledge. An industrial data set was used to illustrate the ideas presented in this paper, with the mean number of assembly operations and expected pre-assembly cost optimized simultaneously. Since the proposed method is data-driven, the efficacy of the Pareto-optimal solutions depends on the predicted demand information and customer preferences assigned to the configurations. A reliable estimation of the demand and preference information is critical in applying the proposed approach. The optimization model presented in this paper can be further extended by considering the assembly capacity or the optimal initial inventory level for each module.

Future research could focus on other representations of the individuals in the evolutionary algorithm to improve the efficiency of the algorithm.

Acknowledgement

This research has been partially funded by the Iowa Energy Center, Grant No. 06-04.

References

- Chvatal, V., 1979. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research* 4, 233–235.
- Da Cunha, C., Agard, B., Kusiak, A., 2007. Design for cost: Module-based mass customization. *IEEE Transactions on Automation Science and Engineering* 4, 350–359.
- Deb, K., 2001. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons, New York.
- Eiben, A.E., Smith, J.E., 2003. *Introduction to Evolutionary Computation*. Springer-Verlag, New York.
- Farrell, R.S., Simpson, T.W., 2003. Product platform design to improve commonality in customer products. *Journal of Intelligent Manufacturing* 14, 541–556.
- Fu, K., Hsu, V.N., Lee, C.Y., 2006. Inventory and production decisions for an assemble-to-order system with uncertain demand and limited assembly capacity. *Operations Research* 54, 1137–1150.
- Fujita, K., Yoshida, H., 2004. Product variety optimization simultaneously designing module combination and module attributes. *Concurrent Engineering: Research and Applications* 12, 105–118.
- Gunasekaran, A., Ngai, W.T., 2009. Modeling and analysis of build-to-order supply chains. *European Journal of Operational Research* 195, 319–334.
- Gupta, D., Benjaafar, S., 2004. Make-to-order, make-to-stock, or delay product differentiation? A common framework for modeling and analysis. *IIE Transactions* 36, 529–546.
- Hsu, S., Wang, W., 2004. Dynamic programming for delayed product differentiation. *European Journal of Operational Research* 156, 183–193.
- Huang, C.C., Kusiak, A., 1998. Modularity in design of products and systems. *IEEE Transactions on Systems, Man, and Cybernetics – Part A* 28, 66–77.
- Jewkes, E.M., Alfa, A.S., 2008. A queueing model of delayed product differentiation. *European Journal of Operational Research*. doi:10.1016/j.ejor.2008.08.001.
- Jiao, J., Tseng, M., 1999. A methodology of developing product family architecture for mass customization. *Journal of Intelligent Manufacturing* 10, 3–20.
- Jiao, J., Zhang, Y., 2004. Product portfolio planning with customer–engineering interaction. *IIE Transactions* 37, 801–814.
- Kreng, V., Lee, T.P., 2004. Modular product design with grouping genetic algorithm – A case study. *Computers and Industrial Engineering* 46, 443–460.
- Kusiak, A., Huang, C.C., 1996. Development of modular products. *IEEE Transactions on Components, Packaging, and Manufacturing Technology, Part A* 19, 523–538.
- Lee, H.L., Tang, C.S., 1997. Modeling the costs and benefits of delayed product differentiation. *Management Science* 40, 40–53.
- Ma, S., Wang, W., Liu, L., 2002. Commonality and postponement in multistage assembly systems. *European Journal of Operational Research* 142, 523–538.
- MacDuffie, J.P., Sethuraman, K., Fisher, M.L., 1996. Product variety and manufacturing performance: Evidence from the international automotive assembly plant study. *Management Science* 42, 350–369.
- Swaminathan, J.M., Tayur, S.R., 1999. Managing design of assembly sequence for product lines that delay product differentiation. *IIE Transactions* 31, 1015–1026.
- Swaminathan, J.M., Tayur, S.R., 1998. Managing broader product lines through delayed differentiation using vanilla boxes. *Management Science* 44, S161–S172.
- Yigit, A.S., Galip-Ulsoy, A.G., Allahverdi, A., 2002. Optimize modular product design for reconfigurable manufacturing. *Journal of Intelligent Manufacturing* 13, 309–316.
- Zitzler, E., Thiele, L., 1999. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation* 3, 257–271.