International Journal of Knowledge, Innovation and Entrepreneurship
Volume 1 Nos. 1– 2, 2013, pp. 6—23

Leader-followers Joint Optimization of Product Family Configuration and Supply Chain Design

GANG DU, JIANPING YU & LI SUN
Tianjin University, China

RAY Y. WU
University of Westminster, UK

Received 17 June 2013; received in revised form 29 June 2013; approved 02 September 2013

ABSTRACT When designing a new product family, designers not only need to define the product family but also need to consider its supply chains simultaneously. Study of the methods to optimise the product family configuration in conjunction with its supply chains could significantly reduce the overall cost and be beneficial to all partners. In such a design process, it is usually the brand holder who leads the design while the supply chain partners make their own decisions accordingly. The previous research either optimising product family itself or optimising product family and the supply chains at the same level. This paper presents a new method for product family optimisation. In this method, a leader-followers joint decision making model is proposed. The model consists of two optimisation levels, an upper level and a lower level. The upper level determines product family configurations by selecting the components to be assembled for the potential profit gaining. The lower level designs the supply chain of the product family by choosing suppliers and determining the timing for cost minimisation. A genetic algorithm is proposed to solve the model. An example of designing a simple product family is provided to demonstrate the method.

Keywords: Product family, Supply chain design, Leader-followers joint optimisation, Genetic algorithms

Introduction

A product family, as an extension of a product, has the characteristics of both product customisation and mass production. It is not only the core in mass customisation but also an important means for a company take advantages in competition. Comparing to a single product, designing a product family is much more complex. Due to various types of components, which may be supplied from different suppliers, used for the product family, the configuration of components for the product family and its supply chain design may have a significant impact on the operating cost. For instance, when Apple's iPhone was in its late design stage, Steve Jobs went to the Far East himself to choose and secure the assemblers and the component suppliers in order to reduce the operating cost and the production lead time with best possible quality. Therefore, when a product family is being designed it is necessary to have a method to optimise the product family configuration and its corresponding supply chain in order to maximise customers' satisfaction and to minimise the product cost.

A product family refers to a set of similar products that derived from on a common platform, which have different functionalities to meet different requirements of customers (Meyer and Lehnerd, 1997, p. 39). The platform means the shared resources or tools used in the product development process, including technology, design, production process, and components (Erens and Verhulst, 1997). Product family design includes two parts: platform design and product family design based on the platform (Simpson, 2004). Many of the previous study assumed that the platform is known or has been determined (Fujimoto, 1999). For the design of product family based on a platform, methods can be divided into two types: configurational (modular) product family design and scalable (parametric) product family design (Jiao *et al.*, 2007). The advantage of configurational product family design is that each functional element of the product can be configured independently by changing only the corresponding component. It makes standardisation possible for mass production to achieve economy of scale (Ulrich, 1995). Newcomb *et al.* (1998) investigate the methods to design product architectures and to perform configuration for modular products. Jiao *et al.* (1998) propose a product family architecture (PFA) which systematically plans modularity and commonality and their configuration structures across the functional, technical and structural perspectives. Based on the previous product architectures, the recent researches investigate into further extension for optimal solutions. For instance, Kamrani and Gonzalez (2003) apply a genetic algorithm-based method in design of product modularity. Jiao *et al.* (2007) also propose a genetic algorithm for product family modularity and configuration.

The product family design based on modular architectures generates product variety which can meet different customers' needs. Nevertheless, customers' requirements for a product, even in a same market, may vary significantly with economic situations, life styles, and cultural backgrounds. Therefore, how to plan modularity of a product family and to determine its configuration appear to be vital for a company to succeed, while the configuration of the product family is ensured by its supply chain design (Jiao *et al.*, 2007).

A supply chain, as defined by Lee (1993), is a network of manufacturing and distribution facilities (or nodes), of which each performs some functions for the final products, such as raw material procurement, components fabrication, part subassembly, final product assembly, product distribution, and delivery final products to ultimate customers (consumers). Supply chain design is a set of suggested aims or decisions in the supply chain for each department or organisation selection so that the best performance, e.g. the minimum cost, is achieved for the whole supply chain.

For each configuration of a product family there is a bill of material (BOM). On BOM list each item has its corresponding functional node in the supply network (Huang *et al.*, 2005). Therefore, the results of a product family configuration determine the structure of its supply chain. In this sense, the configuration of a product family and the supply chain are interlinked. Therefore, it is worth investigating into the integration of product family configuration and its supply chain configuration. In fact, the previous researches have already studied into the issue. Lamothe, Hadj-Hamou and Aldanondo (2006) assert that when designing a new product family it is necessary to define the product family and its supply chain in two steps. The first step is to configure the product family and its corresponding BOM. The second step is to determine the supply chain structure according to the product family. If necessary, the product family configuration is modified in the first step so that the total cost incurred in the supply chain is minimal. Zhang and Huang (2010) acknowledge that in many cases in practice the manufacturer takes a leading role by making the decisions on platform products configuration and supplier selection. The manufacturer and concerned suppliers then make their ordering and pricing decisions cooperatively with a common objective to maximise the profit. Baud-Lavigne *et al.* (2012) show that solving product standardisation and supply chain design problems separately could result in a suboptimal, or even a bad, decision. They further propose a compound optimisation model to illustrate the impact of standardisation choices on the structure of the supply chain and the gain that can be obtained from solving the problem. Fujita, Amaya and Akai (2012) develop a genetic algorithm based mathematical model for simultaneous design of module commonalisation under the given product architecture and supply chain configuration through selection of manufacturing sites for module production, assembly and final distribution to minimise the total cost.

The most of the researches shown above present the optimization of product family and supply chain at the same level, i.e. the mathematical models are presented as a simple single-level programming problem. In fact, the issue of product family configuration and supply chain design is, as Zhang and Huang (2010) suggest, a leader-followers joint design and cooperative decision making problem. Fujita, Amaya and Akai (2012) also recommend four stages for the product family planning, including a stage of product family design and a stage of supply chain design which is dependent on the product family design. Thus, the optimisation problem of product family and supply chain should be viewed as two sub-problems with different levels. Meeting customers' requirements and profitability would be the main objective of product family design and configuration, while the supply chain design is normally focus on effectiveness and efficiency in operational activities, such as purchasing, manufacturing, and distributing. These two levels are dependent and interlinked. The supply chain design should be compliable to the overall objective for profitability and meeting customers' requirements. Therefore, the whole problem is not a single-level optimization but two-level optimization with leader-follower architecture, i.e. a decision-making model of two decision-making entities – designers and operators.

This paper investigates into product family configuration and its supply chain design as a leader-followers joint optimisation problem. The remainder of the paper is organised as follows. The next section describes the problem of product family configuration and its supply chain design and builds a mathematical model for the leader-followers joint optimisation problem. A genetic algorithm based method is proposed to resolve the model in Section 3. Section 4 shows an example of product family design to illustrate the application of the proposed method. Finally, Section 5 summarises the method proposed and conclusions derived. The directions for further investigation are also provided in this section.

Problem Description

The problem of product family configuration and its supply chain design

This paper studies product family configuration and its supply chain design problem assuming that the platform is known. Product family configuration is to select components from a set of modules to form product family (normally) by a manufacture and then the manufacturer selects a combination of product for production. As shown in Figure 1, there is a set of modules for the product family, denoted as $M = \left\{ M_j \middle| j = 1,\mathrm{L}\ ,J \right\}$.Each module $M_j \in M$ has an option set $M_j^* = \left\{ m_{jk} \middle| k = 1,\mathrm{L}\ ,K_j \right\}$. Platform modules are common modules for the product family without alternatives, i.e. only one option is available. In the figure, M_1 presents a platform module and its option is m_{11}. When configuration constraints are met, a product family $P = \left\{ P_i \middle| i = 1,\mathrm{L}\ ,I \right\}$ can be obtained by selection of different modules. Since each product $\forall P_i \in P$ consists of a set of modules, a product can be presented as $P_i = [m_{jk}]_j$, where m_{jk} is option *k* of module *j*. It is assumed that a product can choose up to one option from each module, i.e. $[m_{jk}]_j \in \left\{ M_1^* \times M_2^* \times \mathrm{L}\ \times M_J^* \right\}$. A product combination *Q* is a set of selected product, $Q = \left\{ P_i \middle| P_i \in P \right\} \subseteq P$, denoted as $card\left(Q\right) = I^+$, $I^+ \leq I$.

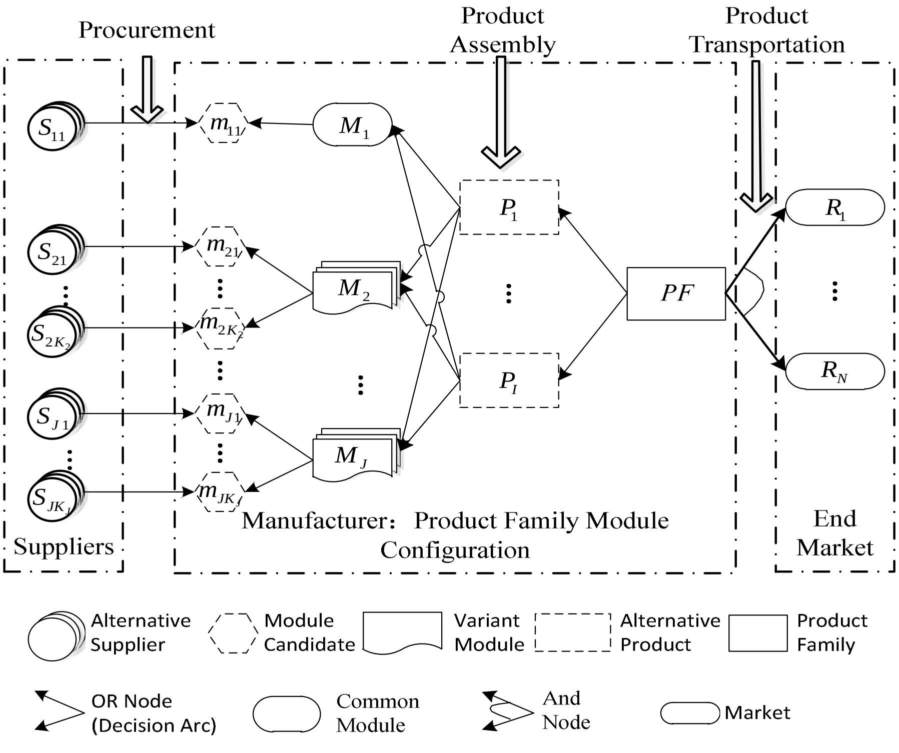

Figure 1 Diagram of product family configuration and supply chain design

After product configuration and module selection, the manufacturer needs to design its supply chain. In this paper, the supply chain is considered as an internal process of the company, which involves purchasing, stocking, production, distribution, and coordination among the functional facilities. In the model, the pricing policy and profit distribution among suppliers are not considered. It is assumed that for every module m_{jk} there is a set of suppliers and for every product there is a set of manufacturers and a set of ways for delivery. Interlink between product family configuration and supply chain design is illustrated in Figure 2.

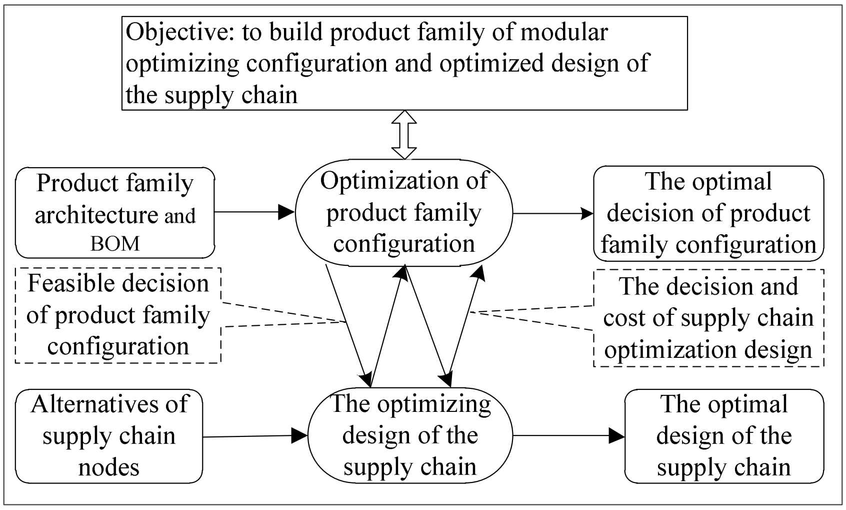
Objective: to build product family of modular optimizing configuration and optimized design of the supply chain
Product family architecture and BOM
Optimization of product family configuration
The optimal decision of product family configuration
Feasible decision of product family configuration
The decision and cost of supply chain optimization design
Alternatives of supply chain nodes
The optimizing design of the supply chain
The optimal design of the supply chain

Figure 2 Interlink between product family configuration and supply chain design

In order to simplify the module, the following assumptions are also made:

1. The set of modules have been previously determined.
2. The supply network is configured for initial product production for product release and in growth stage, which is usually in a time frame within six months. Further supply network dynamics is not considered.
3. The price of products and components and cost of delivery are fixed within the time frame indicated above, as pricing is a complicated issue, which are not only considered by the market and operations issues, e.g., operating cost and quality, of an organisation in the supply chain but also the competitors' performance (and their pricing policy) outside the supply chain.
4. There is no limitation to the capacity of manufacturers and suppliers, i.e. each supplier can fulfil the demand on request.
5. Quality variation of each supplier is not included in the model and the product quality or brand does not affect the demand.
6. One set of market segments $D_n(n=1,2,\mathrm{L}\ ,N)$ is considered. The demand of each market segment is previously given and selling price varies from segments.

The decision mechanism of leader-follower hierarchy

This research takes the whole problem as a cooperative leader-follower decision-making relationship for the product family configuration and its supply chain design. The theoretical framework of the problem can be expressed in the mathematical model. The main decision maker (leader) is product family designer and the follower is the supply chain designer, where both designers can be a real designer, a group of designers, or a virtual decision making body. The leader, *L*, determines the decision vector *X* for his objective $F(X,v(X))$ and the follower, *F*, determines the decision vector *Y* for the objective $f(X,Y)$, where $v(X)$ is the optimum obtained by the follower after a decision *X* has been given by the leader. Therefore, for the follower *F*, the decision set will be constrained by *X*, to be a reasonable responding set (RRS) as follows.

$$RRS(X)=\left\{Y(X)\middle|f(X)=\max_{Y\in D(X)}f(X,Y),X\in D\right\} \tag{1}$$

For the decision leader, the objective can be expressed as:

$$\begin{aligned}&\max F(X,v(X))\\&\text{s. t. } Y(X)\in RRS(X)\end{aligned} \tag{2}$$

Equation (1) and (2) expresses the leader-follower decision-making relationship. A general mathematical module hence can be written as follows.

$$\begin{aligned}&\max_X F(X,v(X))\\&\text{s. t. }\begin{cases}G(X)\geq 0\\H(X)=0\end{cases}\\&\quad v(X)=\min_Y f(X,Y)\\&\quad\text{s. t. }\begin{cases}g(X,Y)\geq 0\\h(X,Y)=0\end{cases}\end{aligned} \tag{3}$$

This is a value-type two-level programming model, i.e. the follower sends the value of his objective to the decision leader. The constraint set for higher level is $D=\{X\mid G(X)\geq 0,\ H(X)=0\}$ and for lower level $D_L(X)=\{Y|g(X,Y)\geq 0,\ h(X,Y)=0\}$. If $X^*\in D$ and $Y^*\in RRS(X^*)$ exist for any $X\in D$, $Y\in RRS(X)$ and $F(X,v(X))\leq F(X^*,v(X^*))$, (X^*,Y^*) is then a solution of Equation (3).

Mixed Integer Programming Model

In this section, the genetic model shown in the previous section will be converted into an integer programming model for genetic algorithms.

Model for the higher level

For product family configuration, $X^p = \left\{ x_i^p \middle| i = 1, \ldots, I \right\}$ is denoted as product selection vector and $X^m = \left\{ x_{ijk}^m \middle| j = 1, \ldots, J; k = 1, \ldots, K_j \right\}$ as module selection vector, i.e.

$$x_i^p = \begin{cases} 0, \text{ product } i \text{ is not selected} \\ 1, \text{ product } i \text{ is selected} \end{cases}, \quad i = 1, \ldots, I.$$

$$x_{ijk}^m = \begin{cases} 0, \text{ module } m_{jk} \text{ is not selected in product } i \\ 1, \text{ module } m_{jk} \text{ is selected in product } i \end{cases}, \quad j = 1, \ldots, J; k = 1, \ldots, K_j.$$

Therefore, the decision vector *X* for the leader is thus $X = (X^p, X^m)$.

To objective of product family configuration is to maximise the profit, which is the total sales minus the total cost of the supply chain. Based on the model built by Jiao *et al*. (2007), the higher level programming model can be written as follows.

$$\text{Profit}(X) = \max_X \left(\sum_{n=1}^{N} \sum_{i=1}^{I} p_{ni} D_{ni} x_i^p - TC \right) \qquad (4)$$

Subject to,

$$p_{ni} = p(X, D_{ni}), n = 1, 2, \ldots, N \qquad (5)$$

$$\sum_{k=1}^{K_j} x_{ijk}^m = 1,\ i = 1, \ldots, I; j = 1, \ldots, J; \qquad (6)$$

$$\sum_{j=1}^{J} \sum_{k=1}^{K_j} \left| x_{i_1 jk}^m - x_{i_2 jk}^m \right| > 0, i_1 \neq i_2; \qquad (7)$$

$$x_{ij_1k_1}^m + x_{ij_2k_2}^m \leq 1, j_1 \neq j_2, \text{ for any } k_1 \in \{1, \ldots, K_{k_1}\}, k_2 \in \{1, \ldots, K_{k_2}\}; \qquad (8)$$

$$x_{ij_3k_3}^m \leq x_{ij_4k_4}^m, j_3 \neq j_4, \text{ for any } k_3 \in \{1, \ldots, K_{k_3}\}, k_4 \in \{1, \ldots, K_{k_4}\}; \qquad (9)$$

$$\sum_{i=1}^{I} x_i^p \leq I^+; \qquad (10)$$

$$x_i^p, x_{ijk}^m \in \{0, 1\} \qquad (11)$$

Where total cost *TC* is the objective of the lower level model. The price function $p(X, D_{ni})$ in equation set (5) presents the product price regarding to the corresponding markets and D_{ni} is demand in market *n* for product *i*. Constraint set (6) ensures up to one option can be selected in a product. Constraint set (7) provides difference among products in a product family. Constraint set (8) means modules $m_{j_1k_1}$ and $m_{j_2k_2}$ ($j_1 \neq j_2$, j_1 and j_2 are given) are not compatible in a product. Constraint set (9) means if module $m_{j_3k_3}$ is used module $m_{j_4k_4}$ must be also selected, where j_1 and j_2 are given.

Model for the lower level

For supply chain design, Y^s, Y^a and Y^d are denoted as supplier selection vector, assembly (and production) selection vector and delivery selection vector, i.e.

$$y_{jkr}^s = \begin{cases} 0, & \text{supplier } r \text{ for module } j \text{ option } k \text{ is not chosen} \\ 1, & \text{supplier } r \text{ for module } j \text{ option } k \text{ is selected} \end{cases}$$

$$y_{iu}^a = \begin{cases} 0, & \text{assemblier } u \text{ for product } i \text{ is not chosen} \\ 1, & \text{assemblier } u \text{ for product } i \text{ is selected} \end{cases}$$

$$y_{nv}^d = \begin{cases} 0, & \text{haulier } v \text{ to market } n \text{ is not chosen} \\ 1, & \text{haulier } v \text{ to market } n \text{ is selected} \end{cases}$$

In supply chain management, stocking of raw materials, components and products cannot be avoided. The level of inventory is mainly determined by demand and the frequency of replenishment (or delivery in terms of distribution of products). Moreover, the decision vectors T^r and T^d are replenishment interval of module m_{jk} and dispatch interval to market *n*, denoted as t_{jk}^r and t_n^d respectively. The decision vector *Y* for the follower is hence $Y = (Y^s, Y^a, Y^d, T^r, T^d)$.

As mentioned above, the objective for the supply chain design is to minimise the total cost, which is the sums of the costs of purchase, the costs of stocking, the costs of production/ assembly, and the cost of distribution. Therefore,

$$TC = \min_{Y \in RRS(X)} \left\{ \begin{array}{l} \sum_{j=1}^{J} \sum_{k=1}^{K_j} \left(C_{jk}^s + c_{jk}^s d_{jk} + \frac{OR_{jk}}{t_{jk}^r} + \frac{h_{jk} d_{jk} t_{jk}^r}{2} \right) z_{jk} + \\ + \sum_{i=1}^{I} \left(C_i^a + c_i^a d_i \right) x_i^p \\ + \sum_{n=1}^{N} \left(\frac{OR_n}{t_n^d} + \sum_{i=1}^{I} \frac{h_i D_{ni} x_i^p t_n^d}{2} \right) \\ + \sum_{n=1}^{N} \left(C_n^d + c_n^d \sum_{i=1}^{I} D_{ni} x_i^p \right) \end{array} \right\} \qquad (12)$$

where h_i and h_{jk} are the holding cost of product *i* and module m_{jk} respectively; OR_n and OR_{jk} are fixed ordering cost of each purchase made for product *i* and module m_{jk} respectively; d_i and d_{jk} are demand of product *i* and module m_{jk} respectively. They can be calculated the demand of products in each market as follows.

$$d_i = \sum_{n=1}^{N} D_{ni}, i = 1,\ldots, I; \tag{13}$$

$$d_{jk} = \sum_{i=1}^{I} d_i x_{ijk}^m x_i^p, \; j = 1,\ldots, J, k = 1,\ldots, K_j; \tag{14}$$

$z_{jk} \in \{0,1\}$ is determined bythe decision made by the leader in the higher decision level. If a product module $m_{jk} \quad (j = 1,\ldots, J, k = 1,\ldots, K_j)$ is selected in the higher decision level, the module has to be purchased in the lower level, i.e. if $x_{ijk}^m \neq 0$ for any $i \in \{1,\ldots, I\}$, then $z_{jk} = 1$, otherwise $z_{jk} = 0$ in Equation (12). Or it is expressed as constraints:

$$z_{jk} \le \sum_{i=1}^{I} x_i^p x_{ijk}^m \le MM \cdot z_{jk}, j = 1,\ldots, J, k = 1,\ldots, K_j; \tag{15}$$

It is assumed that we choose up to one supplier for a module m_{jk}, i.e.

$$\sum_{r=1}^{R_{jk}} y_{jkr}^s = z_{jk}, \; j = 1,\ldots, J, k = 1,\ldots, K_j; \tag{16}$$

It is also assumed that the purchase cost is equal to the purchase cost from the supplier selected, including fixed purchase cost C_{jk}^s and variable purchase cost c_{jk}^s respectively.

$$C_{jk}^s = \sum_{r=1}^{R_{jk}} C_{jkr}^s y_{jkr}^s, \; j = 1,\ldots, J, k = 1,\ldots, K_j; \tag{17}$$

$$c_{jk}^s = \sum_{r=1}^{R_{jk}} c_{jkr}^s y_{jkr}^s, \; j = 1,\ldots, J, k = 1,\ldots, K_j; \tag{18}$$

where C_{jkr}^s and c_{jkr}^s are the supplier fixed cost, including development costs, and variable cost for module m_{jk} from supplier *r*.

Similar to Equations (16), (17) and (18), the assumptions for assembly (production) and delivery respectively are made as follows.

$$\sum_{u=1}^{U_i} y_{iu}^{a} = x_i^{p}, i = 1,\mathrm{L}\ \ , I; \qquad (19)$$

$$C_i^{a} = \sum_{u=1}^{U_i} C_{iu}^{a} y_{iu}^{a}, i = 1,\mathrm{L}\ \ , I; \qquad (20)$$

$$c_i^{a} = \sum_{u=1}^{U_i} c_{iu}^{a} y_{iu}^{a}, i = 1,\mathrm{L}\ \ , I; \qquad (21)$$

$$\sum_{v=1}^{V_n} y_{nv}^{d} = 1, n = 1,\mathrm{L}\ \ , N; \qquad (22)$$

$$C_n^{d} = \sum_{v=1}^{V_n} C_{nv}^{d} y_{nv}^{d}, n = 1,\mathrm{L}\ \ , N; \qquad (23)$$

$$c_n^{d} = \sum_{v=1}^{V_n} c_{nv}^{d} y_{nv}^{d}, n = 1,\mathrm{L}\ \ , N. \qquad (24)$$

Joint model

Integrating the higher level and the lower level, the joint module can be written as follows.

$$\mathrm{Pr\ ofit}(X) = \max_X \left(\sum_{n=1}^{N} \sum_{i=1}^{I} p_{ni} D_{ni} x_i - TC \right)$$

Subject to: Equation (5) to (11).

where TC is the minimum of the problem below:

$$TC = \min_{Y \in RRS(X)} \left\{ \begin{array}{l} \displaystyle\sum_{j=1}^{J} \sum_{k=1}^{K_j} \left(C_{jk}^{s} + c_{jk}^{s} d_{jk} + \frac{OR_{jk}}{t_{jk}^{r}} + \frac{h_{jk} d_{jk} t_{jk}^{r}}{2} \right) z_{jk} + \\ \displaystyle + \sum_{i=1}^{I} \left(C_i^{a} + c_i^{a} d_i \right) x_i^{p} \\ \displaystyle + \sum_{n=1}^{N} \left(\frac{OR_n}{t_n^{d}} + \sum_{i=1}^{I} \frac{h_i D_{ni} x_i^{p} t_n^{d}}{2} \right) \\ \displaystyle + \sum_{n=1}^{N} \left(C_n^{d} + c_n^{d} \sum_{i=1}^{I} D_{ni} x_i^{p} \right) \end{array} \right\}$$

Subject to: Equation (13) to (24) and

$$x_i^{p}, x_{ijk}^{m}, y_{jkr}^{s}, y_{iu}^{a}, y_{nv}^{d}, z_{jk} \in \{0,1\}; t_{jk}^{r} > 0, t_n^{d} > 0.$$

Proposed method

Optimisation methods can be divided into two categories, direct methods and indirect methods. Direct methods find a solution to the model directly. Examples of direct methods include implicit enumeration (Li and Sun, 2006), satisfactory solution procedure (Lai, 1996; Emam, 2006). Indirect methods convert a two-level problem to one (or two) equivalent single-level problem and apply ordinary methods to find a solution. An example (Fortuny-Amat and McCarl, 1981) is to transform the sub-problem by its Kuhn-Tucker conditions. In practise, there are many options in the model (26). As the problem becomes large, the methods above are shown to be inefficient (Jiao *et al.*, 2007).

Genetic algorithms have the advantages for large and complex combinatorial optimisation problems. Some researchers have applied genetic algorithms in bi-level programming problems. Liu (2000) uses Stackelberg-Nash equilibrium with genetic algorithms for multilevel optimization. Niwa *et al.* (1998) apply double strings in genetic algorithms for two-level 0-1 programming. Oduguwa and Roy (2002) also propose a bi-level genetic algorithm to encourage limited asymmetric cooperation between the two players. Li and Wang (2008) propose a method based a genetic-algorithm incorporating with Lemke algorithm. In their method, the follower's problem, a convex quadratic programming problem, is transformed by using Karush-Kuhn-Tucher conditions.

In this paper, a genetic-algorithm-based solution finding strategy is proposed for bi-level optimisation problems. In the strategy, any newly generated solution is firstly checked for the feasibility in the higher level. If it is feasible, it will be used for iterations in genetic algorithms in the lower level to obtain a solution $Y^*(X)$ and use the solution of lower level to calculate the fitness of the higher level for the genetic algorithm to select, crossover, and mutate until the optimal (or near optimal) solution $Y^*(X^*)$ is found. The programme diagram of the propose method is shown in Figure 3.

Computational Example

Case description

In order to illustrate the use of the proposed method, the design of a product family of laptop computers from a case study is taken as an example. The product family includes a platform and four modules, in which the number of options varies between 2 and 4. Specifically, the optional modules are m_{11}, m_{21}, m_{22}, m_{31}, m_{32}, m_{33}, m_{41}, m_{42}, m_{43}, m_{44}, m_{45}. Among them, m_{21} and m_{33}, m_{21} and m_{45}, m_{22} and m_{31} do not compatible. From the feasible combination of these modules, 16 types of product, P_1, P_2, …, P_{15}, P_{16} ,can be produced. The configuration of each type of products is shown in Table 1. The demand and price of the products in each market is shown in Table 2 and 3 respectively.

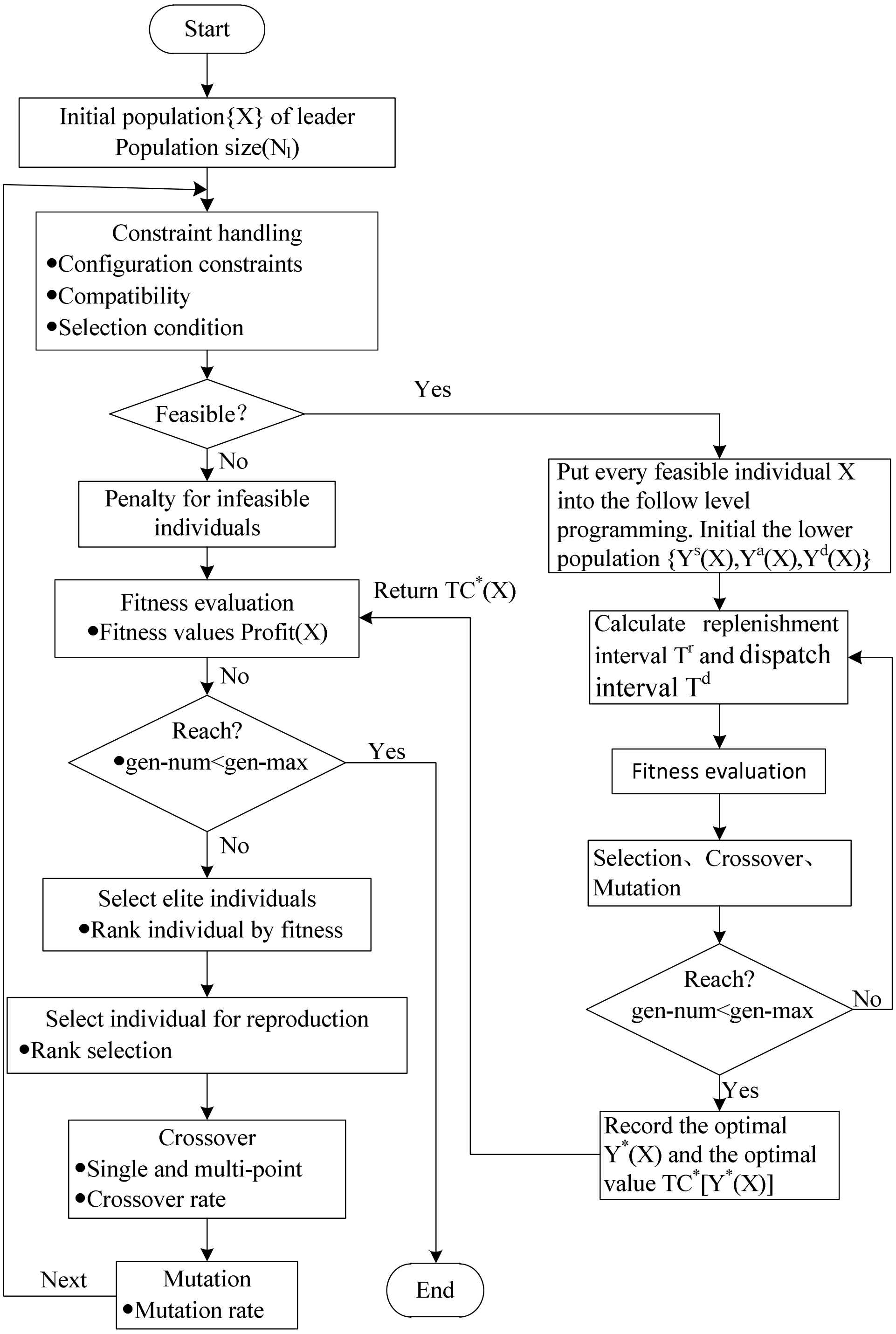
Start
Initial population{X} of leader
Population size(Nl)
Constraint handling
•Configuration constraints
•Compatibility
•Selection condition
Feasible?
Yes
No
Penalty for infeasible individuals
Fitness evaluation
•Fitness values Profit(X)
Return TC*(X)
No
Reach?
•gen-num<gen-max
Yes
No
Select elite individuals
•Rank individual by fitness
Select individual for reproduction
•Rank selection
Crossover
•Single and multi-point
•Crossover rate
Next
Mutation
•Mutation rate
End
Put every feasible individual X into the follow level programming. Initial the lower population {Ys(X),Ya(X),Yd(X)}
Calculate replenishment interval Tr and dispatch interval Td
Fitness evaluation
Selection、Crossover、Mutation
Reach?
gen-num<gen-max
No
Yes
Record the optimal Y*(X) and the optimal value TC*[Y*(X)]

Figure 3 The diagram of proposed bi-level genetic algorithm

Table 2 Demands in different market

	R_1	R_2	R_3	Total
P_1	6,000	5,000	3,000	14,000
P_2	8,000	6,000	4,000	18,000
P_3	9,000	7,000	4,500	20,500
P_4	10,000	7,500	5,000	22,500
P_5	7,000	6,000	4,000	17,000
P_6	9,500	7,000	4,500	21,000
P_7	9,000	7,500	4,000	20,500
P_8	10,000	8,000	5,000	23,000
P_9	8,000	7,000	4,000	19,000
P_{10}	8,500	6,500	5,000	20,000
P_{11}	8,500	6,000	4,000	18,500
P_{12}	6,000	4,500	2,000	12,500
P_{13}	7,000	6,000	3,000	16,000
P_{14}	8,500	6,500	5,000	20,000
P_{15}	8,000	6,000	3,000	17,000
P_{16}	4,000	2,500	1,000	7,500
Total	127,500	99,000	61,000	

Table 1 Product model configuration

Table 3 Prices at different market

	R_1	R_2	R_3
P_1	3,700	3,750	3,800
P_2	3,900	3,950	4,000
P_3	4,100	4,150	4,200
P_4	4,400	4,450	4,500
P_5	4,100	4,150	4,200
P_6	4,200	4,250	4,300
P_7	4,350	4,400	4,450
P_8	4,600	4,650	4,700
P_9	4,300	4,350	4,400
P_{10}	4,700	4,750	4,800
P_{11}	5,200	5,250	5,300
P_{12}	6,500	6,550	6,600
P_{13}	4,600	4,650	4,700
P_{14}	5,000	5,050	5,100
P_{15}	5,600	5,650	5,700
P_{16}	6,900	6,950	7,000

Furthermore, according to the case study,

$S_{11}^{p}=S_{21}^{p}=S_{22}^{p}=S_{31}^{p}=S_{32}^{p}=S_{33}^{p}=S_{41}^{p}=S_{42}^{p}=S_{43}^{p}={}_{44}^{p}=S_{45}^{p}=2$, $S_{1}^{a}=S_{2}^{a}=\mathrm{L}\ =S_{15}^{a}=S_{16}^{a}=3$,

$S_{1}^{t}=S_{2}^{t}=S_{3}^{t}=2$, and $I^{+}=6$.

Computational results

In the genetic algorithm used, the parameters are set as follows. In the higher level, the size of generations and the number of iterations were 50 and 100 respectively; the probabilities of crossover and mutation were 0.8 and 0.01 respectively. In the lower level, the size of generations and the number of iterations were 50 and 200 respectively; the probabilities of crossover and mutation were 0.8 and 0.01 respectively. The iteration results and the final results obtained by the proposed genetic algorithm are respectively shown in Figure 4 and Table 4.

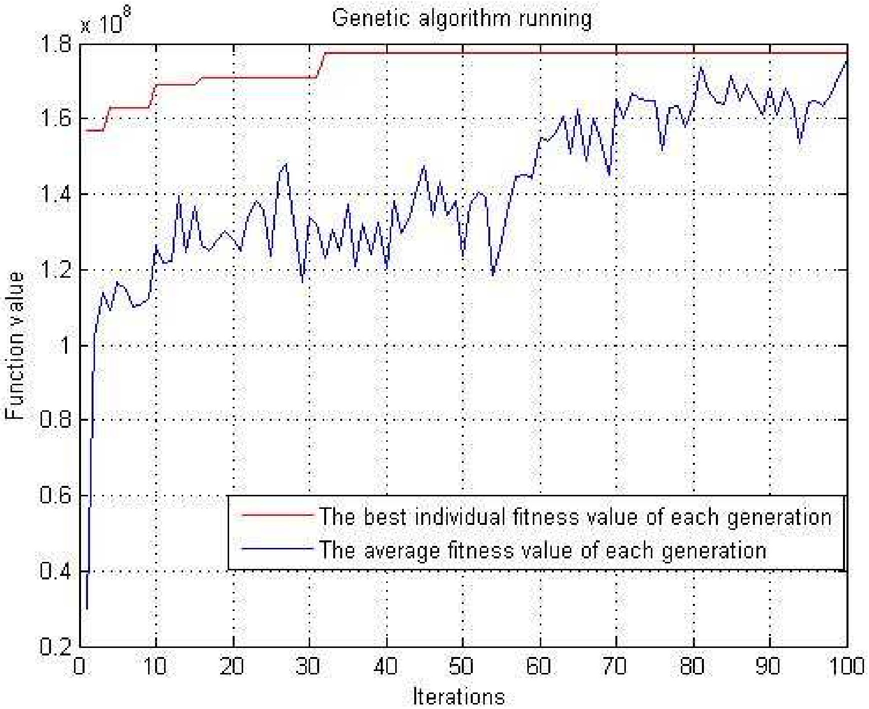

Figure 4 The results of each generation obtained by the genetic algorithm

The result shows that production of products P_6 , P_{10} , P_{11} , P_{12} , P_{14} , P_{15} , with the corresponding selection of suppliers, manufacturers and haulers collaborating in the supply chain, to fulfil the demands in the markets maximises the total profit.

Concluding Remarks

This paper has clarified the leader-follower relationship in design of product families and the corresponding supply network, in which product designers leads product family configura-

Table 4 Decision results by the proposed genetic algorithm

	Purchase/production decision	Supplier/hauler selected	Replenishment/delivery interval (year)
P_{1}	0		
P_{2}	0		
P_{3}	0		
P_{4}	0		
P_{5}	0		
P_{6}	1	1	
P_{7}	0		
P_{8}	0		
P_{9}	0		
P_{10}	1	1	
P_{11}	1	2	
P_{12}	1	3	
P_{13}	0		
P_{14}	1	2	
P_{15}	1	3	
P_{16}	0		
m_{11}	1	2	0.0137
m_{21}	1	2	0.0690
m_{22}	1	1	0.0350
m_{31}	0		
m_{32}	1	1	0.0445
m_{33}	1	1	0.0621
m_{41}	0		
m_{42}	1	2	0.0713
m_{43}	1	1	0.0500
m_{44}	1	1	0.0562
m_{45}	1	2	0.0894
R_{1}		2	0.0658
R_{2}		1	0.0763
R_{3}		1	0.0956

tion and operation managers interactively determines the supply chain structure. Based on this perception, an optimisation model with leader-follower hierarchy for achieving overall profit of the product families and supply network has been built and a bi-level genetic algorithm has been proposed, while achieving overall optimisation by the previous methods is difficult. A computational example of a laptop family design case study shows goodness of the proposed and feasibility of extension of the model and the method in application to similar problems in other area. The integration of product family design and supply network design reduces the total cost for the products and product development lead times. Therefore, the research proposed may help practitioners make better decisions to improve competitiveness.

Due to timing and scope of the research, many assumptions have been made in the model, which presents a major limitation of the study. Since nowadays supply chain changes rapidly, another limitation is that the model is static in both product structure and the supply network. There are many areas which can be further explored. The near future research directions include to enclose the platform design in the model, to refine the leader-follower hierarchic structure, to make the model more adaptable to rapidly changing situations to reflect the supply network dynamics, and to put more detailed considerations, such as pricing policy among suppliers, capacity limitations, refined market segments with demand correlations, lead times, inventory availability and supply chain responsiveness, in the model in order to make the model and the method more applicable.

Acknowledgement

This research is supported by National Science Foundation of China under project number 71071104.

Correspondence

Professor Gang Du
Tianjin University
No. 92 WeiJin Road
Nankai District, Tianjin
China
Email: tddg@tju.edu.cn

References

Baud-Lavigne B, Agard B, Penz B (2012) Mutual impacts of product standardization and supply chain design. *International Journal of Production Economics*, **135**(1), 50-60.

Emam OE (2006) A fuzzy approach for bi-level integer non-linear programming problem. *Applied Mathematics and Computation*, **172**(1), 62-71.

Fortuny-Amat J, McCarl B (1981) A representation and economic interpretation of a two-level programming problem. *Journal of the Operational Research Society*, **32**(9), 783-792.

Erens F, Verhulst K (1997) Architectures for product families. *Computers in Industry*, **33**(2-3), 165-178.

Fujimoto T (1999) The evolution of a manufacturing system at Toyota. New York: Oxford University Press.

Fujita K, Amaya H, Akai R (2012) Mathematical model for simultaneous design of module commonalization and supply chain configuration toward global product family. *Journal of Intelligent Manufacturing*, **24**(134), 1990-2013.

Huang GQ, Zhang XY, Liang L (2005) Towards integrated optimal configuration of platform products, manufacturing processes, and supply chains. *Journal of Operations Management*, **23**(3-4), 267-290.

Jiao J, Tseng MM, Duffy VG, Lin F (1998) Product family modeling for mass customization. *Computers and Industrial Engineering*, **35**(3-4), 495-498.

Jiao JR, Simpson TW, Siddique Z (2007) Product family design and platform-based product development, a state-of-the-art review. *Journal of Intelligent Manufacturing*, 18(1), 5-29.

Jiao JR, Zhang Y, Wang Y (2007) A heuristic genetic algorithm for product portfolio planning. *Computers and Operations Research*, **34**(6), 1777-1799.

Kamrani AK, Gonzalez R (2003) A genetic algorithm-based solution methodology for modular design. *Journal of Intelligent Manufacturing*, **14**(6), 599-616.

Lai YJ (1996) Hierarchical optimization – A satisfactory solution. *Fuzzy Sets and Systems*, **77**(3), 321-335.

Lamothe J, Hadj-Hamou K, Aldanondo M (2006) An optimization model for selecting a product family and designing its supply chain. *European Journal of Operational Research*, **169**(3), 1030-1047.

Lee HL (1993) Design for supply chain management, concept and examples, in Rakesh K. (ed.) Perspectives in Operations Management. Boston: Kluwer (Springer LINK), 45-65.

Li H, Wang Y (2008) Hybrid genetic algorithm for several classes of nonlinear bi-level programming problems. *Systems Engineering and Electronics*, **30**(6), 1168-1172.

Li D, Sun XL (2006) Towards strong duality in integer programming. *Journal of Global Optimization*, **35** (2), 255-282.

Liu B (1998) Stackelberg-Nash equilibrium for multilevel programming with multiple follows using genetic algorithms. *Computers and Mathematics with Applications*, **36**(7), 79-89.

Meyer MH, Lehnerd AP (1997) The power of product platform, building value and cost leadership. New York: Free Press.

Newcomb PJ, Bras B, Rosen DW (1998) Implications of modularity on product design for the life cycle. *Journal of Mechanical Design*, **120**(3), 483-490.

Niwa K, Nishizaki I, Sakawa M (1998) Decentralized two-level 0-1 programming through genetic algorithms with double strings. 2nd International Conference on Knowledge-Base Intelligent Electronic Systems, 278-284.

Oduguwa V, Roy R (2002) Bi-level Optimisation using Genetic Algorithm. Proceedings of the 2002 IEEE International Conference on Artificial Intelligence Systems, 322-327.

Simpson TW (2004) Product platform design and customization – Status and promise. *Artificial Intelligence for Engineering Design*, **18**(1), 3-20.

Ulrich K (1995) The role of product architecture in the manufacturing firm. *Research Policy*, **24**(3), 419 -440.

Zhang X, Huang GQ (2010) Game-theoretic approach to simultaneous configuration of platform products and supply chains with one manufacturing firm and multiple cooperative suppliers. *International Journal of Production Economics*, **124**(1), 121-136.