Joint optimization of lot-sizing and pricing with backlogging

Ming Liu, Hao Tang, Feng Chu, Feifeng Zheng, Chengbin Chu

Joint optimization of lot-sizing and pricing with backlogging

Ming Liu[a], Hao Tang[a], Feng Chu[b,*], Feifeng Zheng[c], Chengbin Chu[d]

[a]*School of Economics & Management, Tongji University, Shanghai, People's Republic of China*
[b]*IBISC, Univ Évry, University of Paris-Saclay, Évry, France*
[c]*Glorious Sun School of Business & Management, Donghua University, Shanghai, People's Republic of China*
[d]*Univ. Gustave Eiffel, ESIEE Paris and Laboratoire GRETTIA-COSYS, 93162 Noisy-le-Grand CEDEX, FRANCE*

Abstract

Lot-sizing and pricing are two important manufacturing decisions that impact together the profit of a company. Existing works address the joint lot-sizing and pricing problem without backlogging, although it is a usual strategy that permits to satisfy customer demand with delay. In this work, we study a new multi-product joint lot-sizing and pricing problem with backlogging and limited production capacity. The objective is to maximize the total company profit over a finite planning horizon. For the problem, a mixed integer nonlinear programming (MINLP) formulation is given. Then, several optimality properties are provided and a tighter MINLP model is established based on these properties. According to the NP-hard nature and non-linearity of the model, a model based heuristic that focuses on efficiently solving small-sized instances is proposed and a genetic algorithm (GA) with new progressive repair strategy is developed to address large-sized instances. Managerial insights are drawn based an illustrative example. Numerical experiments are conducted on 64 benchmark based instances and 105 randomly generated instances with up to 10 products and 12 periods, which validates the MINLP formulation and shows the efficiency of the proposed solution methods.

Keywords: Lot-sizing; Pricing; Backlogging; Optimality property; Mixed integer nonlinear programming; Genetic algorithm

Acknowledgement

We sincerely thank the editor, area editor and anonymous reviewers for their efforts and contributions on improving this paper.

This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 72021002, 71972146, 71771048, 71432007, 71832001 and 72071144.

*Corresponding author.
Email address: feng.chu@univ-evry.fr (Feng Chu)

Joint optimization of lot-sizing and pricing with backlogging

Abstract

Lot-sizing and pricing are two important manufacturing decisions that impact together the profit of a company. Existing works address the joint lot-sizing and pricing problem without backlogging, although it is a usual strategy that permits to satisfy customer demand with delay. In this work, we study a new multi-product joint lot-sizing and pricing problem with backlogging and limited production capacity. The objective is to maximize the total company profit over a finite planning horizon. For the problem, a mixed integer nonlinear programming (MINLP) formulation is given. Then, several optimality properties are provided and a tighter MINLP model is established based on these properties. According to the NP-hard nature and non-linearity of the model, a model based heuristic that focuses on efficiently solving small-sized instances is proposed and a genetic algorithm (GA) with new progressive repair strategy is developed to address large-sized instances. Managerial insights are drawn based an illustrative example. Numerical experiments are conducted on 64 benchmark based instances and 105 randomly generated instances with up to 10 products and 12 periods, which validates the MINLP formulation and shows the efficiency of the proposed solution methods.

Keywords: Lot-sizing; Pricing; Backlogging; Optimality property; Mixed integer nonlinear programming; Genetic algorithm

1. Introduction

In the manufacturing industry, lot-sizing is one of the most important issues in production planning, which considers the best use of production resources in order to satisfy production requirements and anticipating sales opportunities over the planning horizon (Karimi et al., 2003). Traditionally, the lot-sizing planning is regarded as an operational problem with given tactical and strategic marketing decisions (Díaz-Madroñero et al., 2014). Whereas, the marketing strategy may choose prices such that the induced demand patterns are not well aligned with available production capacity, consequently leading to unsatisfied demand or unutilized capacity (Deng and Yano, 2006).

The lack of coordination and the inconsistent decision-making in manufacturing lead to inefficiency of the overall business performance.

To overcome the above drawback, the joint lot-sizing and pricing problem (JLSP), in which demand is assumed as price-sensitive in a monopolistic market environment, is becoming an increasingly hot topic in recent years. The problem aims to help companies efficiently manage their production considering price-sensitive demand, dynamic pricing strategy, and production capacity. In studies on JLSP, the lot-sizes and prices over the planning horizon are optimized simultaneously, with consideration of production setup costs and limited production capacity. The JLSPs can be classified into single-product and multi-product ones. The single-product JLSP is addressed by Thomas (1970), Gilbert (1999), Deng and Yano (2006) and Askarpoor and Davoudpour (2013). These single-product works consider a linear demand function and develop exact algorithms. The multi-product JLSP is introduced by Gilbert (2000). Since then, some researchers follow this research direction (Haugen et al., 2007; Önal and Romeijn, 2010; González-Ramírez et al., 2011; Lusa et al., 2012; Bajwa et al., 2016a,b; Couzon et al., 2020). In reality, linear demand functions, however, are not applicable to portray the price-demand relationship for all types of products. Therefore, some of the multi-product JLSP related works introduce nonlinear demand functions, such as isoelastic ones. This type of demand function further complicates the problem, for which existing research works (Bajwa et al., 2016a,b; Couzon et al., 2020) only solve instances with 3 products and 6 periods (i.e., 90 decision variables and 60 constraints). In addition, backlogging is a usually used strategy in case where production capacity is insufficient during some periods. This important strategy permits to smooth production and to satisfy customer demand with delay, but results in an additional cost. Although backlogging is widely considered in both uncapacitated and capacitated lot-sizing problems (Cheng et al., 2001; Küçükyavuz and Pochet, 2009; Chu et al., 2013; Wu et al., 2013; Toledo et al., 2013; Wu et al., 2018), it has not been widely included in the studies on JLSP, perhaps due to the complexity of the problem. However, such problem has been a challenge for some real-world practitioners, such as Nvidia Corporation, the leader of Graphics Processing Unit (GPU) industry, who needs to make lot-sizing, pricing and backlogging decisions for a portfolio of GPU products to balance setup cost, inventory holding cost and backlogging cost

and gain more profits under the chip shortage since early 2021 [1,2]. Moreover, the existing works on classic lot-sizing problems with backlogging assume dynamic demand that does not depend on selling price and their developed solution approaches are not appropriate for solving the JLSP with backlogging.

Motivated by the above facts, we investigate a new multi-product joint lot-sizing and pricing problem with backlogging and limited production capacity (JLSPB for short hereinafter). The objective is to maximize the total company profit over a finite planning horizon. In this work, we assume an isoelastic demand function that is nonlinear, continuous, differentiable and strict decreasing in price. For the problem, we give a basic mixed nonlinear programming (MINLP) formulation, and then several optimality properties are provided and a tighter MINLP model is constructed based on these properties. Due to the NP-hard nature and non-linearity of the model, a quick model based heuristic is proposed for small-sized instances and a genetic algorithm (GA) with new progressive repair strategy is developed for large-sized instances. Experiment results on 64 benchmark based instances and 105 randomly generated instances with up to 10 products and 12 periods demonstrate the efficiency of the proposed tighter MINLP model and solution methods. In addition, managerial insights are drawn. The main contributions of this paper can be concluded as follows:

(1) A new multi-product joint lot-sizing and pricing problem with backlogging and limited production capacity is first studied;

(2) Several optimality properties of the considered problem are proved, and based on these properties, a tighter mixed integer nonlinear programming model is established;

(3) A model based heuristic that focuses on efficiently solving small-sized instances is proposed and a genetic algorithm with new progressive repair strategy is developed to solve the instances with up to 10 products 12 periods;

(4) Managerial insights are drawn based on an illustrative example.

The remainder of this paper is organized as follows. In Section 2, a brief literature review is given. Section 3 states the studied problem, and a basic mathematical formulation is proposed. In

[1]Monica J. White, 2021. The GPU Shortage Turns A Year Old While Prices Continue to Rise. https://www.digitaltrends.com/computing/gpu-shortage-turns-year-old-as-prices-continue-to-rise/

[2]Asa Fitch, 2021. Intel CEO Says Chip Shortage Could Stretch Into 2023. The Wall Street Journal. https://www.wsj.com/articles/intel-intc-2q-earnings-report-2021-11626899296?mod=searchresults_pos8&page=1

Section 4, we provide several optimality properties and a tighter formulation. In Section 5, a model based heuristic and a GA are developed to solve the studied problem. Numerical experimentsare conducted in Section 6. The work is concluded and further research directions are indicated in Section 7.

2. Literature review

The earliest research works that address the joint lot-sizing and pricing decisions in the manufacturing industry can date back to 1970s. Among these, Thomas (1970) is one of the first to study a single-product uncapacitated joint lot-sizing and pricing problem (JLSP) with production setup costs, for which the objective is to maximize total profit over a finite planning horizon. The author assumes that demand is a linear function of the decision variable price and proposes a forward planning algorithm to obtain optimal lot-sizing plan and pricing strategy. Since then, many researchers have contribute to the literature by considering various factors in manufacturing process. Below we briefly review these related works in recent decades, which can be classified mainly by the number of product, types of demand function, and solution method.

For the literature considering single product, Gilbert (1999) studies a variant of the classic JLSP with unlimited production capacity and linear demand function, where a constant price need to be determined for all periods. The author proposes an exact algorithm with $O(T^3)$ computational complexity for this problem, where T denotes the number of periods. Deng and Yano (2006) investigate the JLSP with linear demand function and limited production capacity and develop an exact solution approach based on shortest-path algorithm for instances with 6 periods, but the detailed procedure of the approach is not presented in their paper. Askarpoor and Davoudpour (2013) study the same JLSP proposed by Deng and Yano (2006). They propose a 2-index mixed integer nonlinear programming model, which is reduced to a bilinear model using bilinear reduction technique. They prove that solving the primal problem is equivalent to finding a global maximum of the bilinear problem and then propose an efficient heuristic to solve instances with up to 48 periods. In the field of inventory management, Li and Teng (2018) and Chang et al. (2019) study the lot-sizing and pricing decisions for single perishable good and established deterministic model in which the retailer needs to determine the optimal selling prices and inventory levels to maximize total profit. Li and Teng (2018) consider a multivariate demand function of selling price, reference price, product freshness and displayed stocks. Li et al. (2019) extend JLSP study by considering

an advance-cash-credit payment scheme in a two-echelon supplier-retailer chain. In their problem, the optimal selling price, replenishment cycle and shortage interval are simultaneously determined. Feng et al. (2022) also study the JLSP for perishable product but further considered a multivariate demand function of unit price, displayed stocks and product age under a generalized advance-cash-credit payment scheme in a two-echelon supply chain.

For the multi-product JLSP, Gilbert (2000) is the first to investigate a multi-product JLSP with limited production capacity and linear demand function. The author assumes that production setup costs are negligible and a constant price for each product need to be determined for all periods. An exact algorithm is proposed based on the special structure of their problem. Haugen et al. (2007) address a multi-product JLSP with setup costs, limited production capacity and linear demand function. They proposes a Lagrangean relaxation based method to solve the problem. Önal and Romeijn (2010) investigate a multi-product JLSP with setup times, limited production capacity (measured by time) and linear demand function. The authors develop a branch and price algorithm to solve instances with up to 30 products and 20 periods within a time limit of 1200 seconds. González-Ramírez et al. (2011) address the same JLSP proposed by Önal and Romeijn (2010). They propose a Dantzig-Wolfe decomposition based heuristic, which can be used to solve instances with up to 500 products and 50 periods within 2000 seconds.

However, none of the above literature considers a nonlinear demand function and multiple products simultaneously. In Bajwa et al. (2016b), the authors study a multi-product JLSP with nonlinear demand function (i.e., the isoelastic function), setup costs and limited production capacity. Based on the outer approximation methodology, they develop an exact iterative algorithm to solve the instances with 3 periods and 6 products within 135 seconds. Couzon et al. (2020) investigate the same problem proposed by Bajwa et al. (2016b) and develop new lower bounds for prices and upper bounds for sales quantities to cut off non-optimal values of the decision variables. Based on the model reference algorithm (MRA) developed by Bajwa et al. (2016b) for solving their proposed sub-problems, Couzon et al. (2020) devise two efficient model based heuristics to provide near-optimal solutions for the instances proposed by Bajwa et al. (2016b) within less than 1 second. Bajwa et al. (2016a) simplify their former work (Bajwa et al., 2016b) by neglecting the setup costs. Then, a Lagrangian dual approach combined with one-dimensional search algorithm is developed to solve the problem. Lusa et al. (2012) investigate a multi-product JLSP with dynamic production capacity, limited warehouse capacity and convex demand function. In their paper, a mixed integer

nonlinear programming formulation is established to maximize the total profit, which is then linearized to a linear model that can be solved by the commercial solver CPLEX. Most of the works considering multiple products assume linear and isoelastic demand functions and develop heuristics and metaheuristics for solving their problems. In our work, we first extend the multi-product JLSP with isoelastic demand function, setup costs and limited production capacity by introducing backlogging. Due to the complexity of the studied problem, we develop a model based heuristic and a genetic algorithm to find near-optimal solutions.

Table 1: Comparison of researches on the JLSP in recent decades

	The number of products				
Literature	Single	Multiple	Backlogging	Demand function	Solution method
Thomas (1970)	✓			Linear	Exact algorithm
Gilbert (1999)	✓			Linear	Exact algorithm
Deng and Yano (2006)	✓			Linear	Exact algorithm
Askarpoor and Davoudpour (2013)	✓			Linear	Model based heuristic
Gilbert (2000)		✓		Linear	Exact algorithm
Haugen et al. (2007)		✓		Linear	Lagrangian relaxation based heuristic
Önal and Romeijn (2010)		✓		Linear	Branch and price
González-Ramírez et al. (2011)		✓		Linear	Decomposition based heuristic
Lusa et al. (2012)		✓		Convex	CPLEX
Bajwa et al. (2016a)		✓		Isoelastic	Lagrangian dual approach
Bajwa et al. (2016b)		✓		Linear and isoelastic	Outer approximation
Couzon et al. (2020)		✓		Isoelastic	Model based heuristic
This paper		✓	✓	**Isoelastic**	**Model based heuristic and metaheuristic**

To better understand the current research status, the comparison between related works and our work is reported in Table 1.

3. Problem statement and formulation

We consider a joint lot-sizing and pricing problem faced by a manufacturer in a monopolistic market environment, who produces a portfolio of products with unique features and applications using the same equipment under limited production capacity and sells them to price-sensitive customers. The optimization goal is to determine the lot-sizing plan and pricing strategy to maximize total profit over a finite planning horizon. It is assumed that the demand for each product j in each period t is a function of its current price P_{jt} that can change from period to period, and the considered demand function is a widely used isoelastic one in the literature (Huang et al., 2013; Bajwa et al., 2016a,b; Couzon et al., 2020): $D(P_{jt}) = \gamma_{jt}\alpha_j P_{jt}^{-\beta_j}$, where γ_{jt} is the seasonality parameter, α_j is the scaling parameter for price-sensitive demand, and β_j is the price elasticity of

the demand. We adopt the multiplicative demand factors γ_{jt} such that $\sum_{t\in\mathcal{T}}\gamma_{jt}=1$ to portray seasonality, which has been used by Gilbert (2000), Bajwa et al. (2016a), Bajwa et al. (2016b), and Couzon et al. (2020).

Based on the assumption that a same set of equipment is used for production, setup operations are necessary which incurs a setup cost when the equipments are adjusted to produce a new lot of some product (Díaz-Madroñero et al., 2014). We also assume that setup carryover between consecutive periods is negligible and not considered as the same in Haugen et al. (2007), Bajwa et al. (2016b), Bajwa et al. (2016a), Couzon et al. (2020), etc. For each product, a variable production cost per unit and a holding cost per unit of ending inventory in each period are considered. Besides, the cost for backlogging one unit of demand for product j in period t_1 and deliverying it in period t_2 is calculated as $\sum_{t=t_1}^{t_2-1} b_{jt}$, where b_{jt} is the backlogged order maintenance cost per unit. This means that the longer the backlogging time interval, the higher the backlogging cost (Cheng et al., 2001; Karimi et al., 2003; Slama et al., 2020). Without loss of generality, we assume that at the beginning and the ending of the planning horizon, the inventories and backlogs of all products are zero respectively. Overall, the assumptions we made are summarized as follows:

(1) The demand quantity for each product is an isoelastic function with respect to its price only.

(2) The price of each product can be changed in each period and the cost for changing price is negligible.

(3) Setup carryover between consecutive periods is negligible and not considered.

(4) Multiple products with different consumption rate share and compete for the production capacity that represents the same resources or equipment required for producing them.

(5) The inventories and backlogs of each product are zero at the beginning and the ending of the planning horizon.

A mixed integer nonlinear programming (MINLP) model is formulated for the problem. In the following, we first introduce notations, then the basic MINLP model is given.

3.1. Notations

Indices:

j: index of products;

t: index of periods during the planning horizon.

Problem parameters:

$\mathcal{J}$: set of products, $\mathcal{J} = \{1, \ldots, J\}$, where J is the number of products;

$\mathcal{T}$: set of periods, $\mathcal{T} = \{1, \ldots, T\}$, where T is the number of periods;

c_{jt}: production cost for per unit of product $j \in \mathcal{J}$ in period $t \in \mathcal{T}$;

h_{jt}: inventory cost for per unit of product $j \in \mathcal{J}$ in period $t \in \mathcal{T}$;

b_{jt}: backlogging cost for per unit of product $j \in \mathcal{J}$ in period $t \in \mathcal{T}$;

a_{jt}: setup cost for producing $j \in \mathcal{J}$ in period $t \in \mathcal{T}$;

C_t: production capacity in period $t \in \mathcal{T}$;

v_j: production capacity used for producing one unit $j \in \mathcal{J}$;

α_j: scaling parameter for price-sensitive demand for product $j \in \mathcal{J}$;

β_j: price elasticity of the demand function for product $j \in \mathcal{J}$;

γ_{jt}: demand seasonality of product $j \in \mathcal{J}$ in period $t \in \mathcal{T}$, $\sum_{t \in \mathcal{T}} \gamma_{jt} = 1$.

Decision variables:

P_{jt}: price for product $j \in \mathcal{J}$ in period $t \in \mathcal{T}$;

S_{jt}: quantity of product $j \in \mathcal{J}$ sold in period $t \in \mathcal{T}$, including backlogs that will be fulfilled in the following periods;

X_{jt}: quantity of product $j \in \mathcal{J}$ produced in period $t \in \mathcal{T}$;

I_{jt}: inventory of product $j \in \mathcal{J}$ at the end of period $t \in \mathcal{T}$;

G_{jt}: backlog of product $j \in \mathcal{J}$ that has not been fulfilled at the end of period $t \in \mathcal{T}$;

Y_{jt}: binary, equal to 1 if there is a setup to produce $j \in \mathcal{J}$ in period $t \in \mathcal{T}$; 0, otherwise;

u^I_{jt}: binary, equal to 1 if $I_{jt} > 0$; 0, if $I_{jt} = 0$, where $j \in \mathcal{J}$ and $t \in \mathcal{T}$;

u^G_{jt}: binary, equal to 1 if $G_{jt} > 0$; 0, if $G_{jt} = 0$, where $j \in \mathcal{J}$ and $t \in \mathcal{T}$.

3.2. The basic mixed integer nonlinear programming model

The basic mixed integer nonlinear programming (MINLP) model P_0 is detailed below:

$$[\text{Model} \mid P_0] \ \ max \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}}(P_{jt}S_{jt} - c_{jt}X_{jt} - h_{jt}I_{jt} - b_{jt}G_{jt} - a_{jt}Y_{jt}) \tag{1}$$

subject to:

$$S_{jt} \le \gamma_{jt}\alpha_j P_{jt}^{-\beta_j}, \quad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{2}$$

$$\sum_{j\in\mathcal{J}} v_j X_{jt} \le C_t, \quad \forall t \in \mathcal{T} \tag{3}$$

$$X_{jt} + I_{jt-1} - G_{jt-1} = S_{jt} + I_{jt} - G_{jt}, \quad \forall j \in \mathcal{J}, t \in \mathcal{T}/\{1\} \tag{4}$$

$$X_{j1} = S_{j1} + I_{j1} - G_{j1}, \quad \forall j \in \mathcal{J} \tag{5}$$

$$I_{jT} = 0, \quad \forall j \in \mathcal{J} \tag{6}$$

$$G_{jT} = 0, \quad \forall j \in \mathcal{J} \tag{7}$$

$$v_j X_{jt} \le C_t Y_{jt}, \quad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{8}$$

$$v_j I_{jt} \le u_{jt}^I \sum_{t'=1}^{t} C_{t'}, \quad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{9}$$

$$v_j G_{jt} \le u_{jt}^G \sum_{t'=t+1}^{T} C_{t'}, \quad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{10}$$

$$u_{jt}^I + u_{jt}^G \le 1, \quad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{11}$$

$$X_{jt}, S_{jt}, P_{jt}, I_{jt}, G_{jt} \ge 0, \quad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{12}$$

$$Y_{jt} \in \{0, 1\}, \quad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{13}$$

The objective function (1) maximizes the profit which is the revenue of sales minus total costs including variable production costs, setup costs, inventory holding costs and backlogging costs. Constraints (2) imply that sales quantities for all products cannot exceed corresponding demand during each period. Constraints (3) are production capacity limitation. Constraints (4) and (5) represent the production, inventory, and backlog conservation constraints. The inventory and backlog in the last period are limited to be zero by Constraints (6) and (7) respectively. Constraints (8) imply that there will be a setup cost for each product once it is produced. Constraints (9)-(11) in-

dicate that inventory and backlog cannot exist simultaneously in the same period for each product. Domains of decision variables are given in Constraints (12) and (13).

4. Problem analysis and a tighter formulation

In this section, we analyze the MINLP model P_0 and give the lower bound for decision variable price P_{jt} for all $j \in \mathcal{J}$ and $t \in \mathcal{T}$ and the upper bound for the price P_{jt} in setup periods for all $j \in \mathcal{J}$. Besides, the relationship between prices in two consecutive periods is investigated. Based on these properties of optimal solution, a tighter formulation is established.

4.1. Problem analysis

Optimality property 1. *A lower bound for the optimal price P^*_{jt} of product j in period t is given as follows:*

$$P^*_{jt} \geq \frac{\min_{t' \in \mathcal{T}} A_{jt't}}{1 - \frac{1}{\beta_j}}, \tag{14}$$

where

$$A_{jt't} = \begin{cases} \left(c_{jt'} + \sum_{l=t'}^{t-1} h_{jl} \right)^{\beta_j} & t' \leq t \\ \left(c_{jt'} + \sum_{l=t}^{t'-1} b_{jl} \right)^{\beta_j} & t' > t \end{cases}. \tag{15}$$

Proof. We follow the idea of Couzon et al. (2020) to reformulate the model P_0 with 3-index variables, and the lower bound P^L_{jt} in our problem can be given in a similar way. To reformulate P_0 with only one variable type, some new notations are introduced:

$\mathcal{K}$: set of all possible setup configurations, indexed by k;

$\boldsymbol{Y}^k$: a setup configuration $\boldsymbol{Y}^k \in \mathcal{K}$, where

$$\boldsymbol{Y}^k = \begin{bmatrix} Y^k_{11} & Y^k_{12} & \cdots & Y^k_{1T} \\ Y^k_{21} & Y^k_{22} & \cdots & Y^k_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ Y^k_{J1} & Y^k_{J2} & \cdots & Y^k_{JT} \end{bmatrix};$$

$\mathcal{N}^k$: set of indices associated with the setup configuration $\boldsymbol{Y^k}$, $\mathcal{N}^k = \{(j,t',t) \mid j \in \mathcal{J}, t' \in \mathcal{T}, t \in \mathcal{T} \text{ and } Y^k_{jt'} = 1\}$, where $\boldsymbol{Y^k} \in \mathcal{K}$.

B_{jt}: substitution for $\gamma_{jt} \times \alpha_j$, i.e. $B_{jt} = \gamma_{jt} \times \alpha_j$, where $j \in \mathcal{J}$ and $t \in \mathcal{T}$;

$A_{jt't}$: cost for per unit of product j that is produced in period t' and sold in period t, where $j \in \mathcal{J}$ and $t', t \in \mathcal{T}$;

$X^k_{jt't}$: new decision variable under a given setup configuration k, which denotes quantity of product j produced in period t' and sold in period t, where $(j,t',t) \in \mathcal{N}^k$.

The new model (RP_1) with 3-index variables is formulated as follows:

$$[\text{Model} \mid RP_1] \quad \max_{k \in \mathcal{K}} \; z^k = \sum_{(j,t',t) \in \mathcal{N}^k} \left(X^k_{jt't} \left((B_{jn})^{\frac{1}{\beta_j}} \left(\sum_{l \in \mathcal{T}} X^k_{jlt} \right)^{-\frac{1}{\beta_j}} - A_{jt't} \right) \right) - \sum_{j \in \mathcal{J}} \sum_{t' \in \mathcal{T}} a_{jt'} Y^k_{jt'} \tag{16}$$

subject to:

$$\sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} v_j X^k_{jt't} \leq C_{t'}, \quad \forall t' \in \mathcal{T} \tag{17}$$

$$X^k_{jt't} \geq 0, \quad \forall (j,t',t) \in \mathcal{N}^k \tag{18}$$

The objective function of model RP_1 is transformed from the former one (1) based on the Theorem 1 in Bajwa et al. (2016b) indicating that demand under optimal price is equal to the quantity sold for each product in each period, i.e., the inequality in constraints (2) can be replaced by equality, by which we can derive that $P_{jt} = \left(\frac{S_{jt}}{B_{jt}}\right)^{-\frac{1}{\beta_j}} = \left(\frac{\sum_{l \in \mathcal{T}} X_{jlt}}{B_{jt}}\right)^{-\frac{1}{\beta_j}}$. The objective function (1) of model P_0 can be reformulated as a new one (16) depending only on $X^k_{jt't}$. For a given setup configuration $\boldsymbol{Y}^k \in \mathcal{K}$, the corresponding setup cost in the objective function is a constant $\sum_{j \in \mathcal{J}} \sum_{t' \in \mathcal{T}} a_{jt'} Y^k_{jt'}$, and the profit for per unit of product j produced in t' and sold in t is calculated as $X^k_{jt't}(P^k_{jt} - A_{jt't}) = X^k_{jt't} \left((B_{jn})^{\frac{1}{\beta_j}} \left(\sum_{l \in \mathcal{T}} X^k_{jlt} \right)^{-\frac{1}{\beta_j}} - A_{jt't} \right)$.

We can obtain a restriction model RP^k_1 by fixing the setup variables using one $\boldsymbol{Y}^k \in \mathcal{K}$ as follows:

$$\left[\text{Model} \mid RP^k_1\right] \quad \max \; z^k \tag{19}$$

subject to:

(17) and (18).

The restriction model RP_1^k provides a lower bound for the RP_1. When the capacity constraints (17) is relaxed, the optimal value $X_{jt't}^{k*}$ of $X_{jt't}^k$ in the model RK_1^k can be obtained via setting the first derivative of z^k (with respect to $X_{jt't}^k$) to 0 and solving for $X_{jt't}^k$ as follows:

$$X_{jt't}^{k*} = B_{jt}\left(\frac{1-\frac{1}{\beta_j}}{A_{jt't}}\right)^{\beta_j} - \sum_{l\in\mathcal{T}, l\neq t'} X_{jlt}^k. \tag{20}$$

Without loss of generality, still supposing the capacity constraints (17) is relaxed, we may suppose that each product sold in a period t is manufactured during only one period t' in the planning horizon. Then, it follows that $S_{jt}^k(t') = \sum_{l\in\mathcal{T}} X_{jlt} = X_{jt't}^k = B_{jt}\left(\frac{1-\frac{1}{\beta_j}}{A_{jt't}}\right)^{\beta_j}$, where $S_{jt}^k(t')$ denotes the sold quantity if it is satisfied by production in period t'. The maximum value S_{jt}^{k*} of $S_{jt}^k(t')$ over all possible t' can be given as follows:

$$S_{jt}^{k*} = \max_{t'\in\mathcal{T}} S_{jt}^k(t') = \frac{B_{jt}\left(1-\frac{1}{\beta_j}\right)^{\beta_j}}{\min_{t'\in\mathcal{T}}\left(A_{jt't}\right)^{\beta_j}} \tag{21}$$

Then, we can substitute $B_{jt}\left(P_{jt}^{k*}\right)^{-\beta_j}$ for S_{jt}^{k*} into (21), where P_{jt}^{k*} is the price corresponding to S_{jt}^{k*}, then a lower bound for optimal price can be obtained as follows:

$$P_{jt}^* \geq P_{jt}^{k*} = \frac{\min_{t'\in\mathcal{T}} A_{jt't}}{1-\frac{1}{\beta_j}}. \tag{22}$$

□

Optimality property 2. *For optimal solution, (i) if the demand in period t is completely satisfied by inventory from period $t-1$ (i.e., $Y_{j,t}=0$, $u_{jt-1}^I = 1$ and $u_{jt}^G = 0$), the price P_{jt} in period t must be greater than price P_{jt-1} and $P_{jt} - P_{jt-1} = \frac{h_{jt-1}\beta_j}{\beta_j - 1}$; (ii) if the demand in period t is completely backlogged and satisfied in latter periods (i.e., $Y_{j,t}=0$, $u_{jt-1}^I = 0$ and $u_{jt}^G = 1$), the price P_{jt} in period t must be greater than price P_{jt+1} and $P_{jt} - P_{jt+1} = \frac{b_{jt}\beta_j}{\beta_j - 1}$.*

Proof. Let us suppose that there is an optimal solution denoted by X_{JLSPB}^1 in which $P_{jt}^* \leq P_{jt-1}^*$, $Y_{jt}^* = 0$, $u_{jt-1}^{I*} = 1$ and $u_{jt}^{G*} = 0$ for some product j and period $t > 1$. For simplicity, we omit index j

of parameters and variables in the following. We now consider the following nonlinear programming optimization problem that optimizes prices for period $t-1$ and t while decision variables for other periods remain the same as those in X^1_{JLSPB}:

$$\max \ F(P_{t-1}, P_t) = \gamma_{t-1}\alpha P_{t-1}^{1-\beta} + \gamma_t \alpha P_t^{1-\beta} - h_{t-1}\gamma_t \alpha P_t^{-\beta} \tag{23}$$

subject to:

$$\gamma_{t-1}\alpha P_{t-1}^{-\beta} + \gamma_t \alpha P_t^{-\beta} = K \tag{24}$$

$$P_{t-1}, P_t \geq 0 \tag{25}$$

where $K = \gamma_{t-1}\alpha {P^*_{t-1}}^{-\beta} + \gamma_t \alpha {P^*_t}^{-\beta}$ is a constant representing the overall quantity that can be sold in these two periods. The objective function (23) represents the revenue in period $t-1$ and t minus the inventory holding cost from period $t-1$ to t for satisfying demand in period t. Constraints (24) implies that the total demand in periods $t-1$ and t equals to that of the optimal solution X^1_{JLSPB}. From the KKT condition, we know that there exists λ satisfying (26) for the optimal solution of the nonlinear programming model:

$$\begin{cases} (1-\beta)\gamma_{t-1}\alpha P_{t-1}^{-\beta} + \lambda\beta\gamma_{t-1}\alpha P_{t-1}^{-\beta-1} & = 0 \\ (1-\beta)\gamma_t \alpha P_t^{-\beta} + \beta h_{t-1}\gamma_t \alpha P_t^{-\beta-1} + \lambda\beta\gamma_t\alpha P_t^{-\beta-1} & = 0 \end{cases}. \tag{26}$$

It can be derived from (26) that $P_t - P_{t-1} = \frac{h_{t-1}\beta}{\beta-1} > 0$, which implies that there exists another better solution X^2_{JLSPB} with higher profit, in which $P_{jt} > P_{jt-1}$ and $I_{jt-1} = \gamma_t \alpha P_{jt}^{-\beta}$ while other decision variables remain unchanged as those in X^1_{JLSPB}. Now we have proved (i) of Optimality property 2, and in the same way we can prove (ii) that $P_t - P_{t+1} = \frac{b_t\beta}{\beta-1} > 0$. □

Optimality property 3. *For optimal solution, if product j is produced in period t, the following inequality must be true: $P_{jt} \leq P^{ub}_{jt}$, where P^{ub}_{jt} depends on problem parameters and can be obtained by bisection method.*

Proof. We use $X_{jtt'}$ to denote the quantity of product j produced in period t and sold in period t' and the function F^1_t to denote the profit for producing product j in period t as follows:

$$F^1_t = \gamma_{jt}\alpha P_{jt}^{-\beta_j}(P_{jt} - c_{jt}) - a_{jt} + \sum_{t'>t} X_{jtt'}(P_{jt'} - \sum_{k=t}^{t'-1} h_{jk} - c_{jt}) + \sum_{t'<t} X_{jtt'}(P_{jt'} - \sum_{k=t'}^{t-1} b_{jk} - c_{jt}), \tag{27}$$

where $F_t^2(P_{jt}) = \gamma_{jt}\alpha_j P_{jt}^{-\beta_j}(P_{jt} - c_{jt}) - a_{jt}$ is the revenue for selling product j in period t minus the corresponding variable and setup costs, while other terms represent the profit for satisfying demand for product j in other periods by the production in period t, which must be non-negative for the optimal solution. This is because that if there exist $X_{jtt'} > 0$ and $(P_{jt'} - \sum_{k=t}^{t'-1} h_{jk} - c_{jt}) < 0$ for some $t' > t$, we can increase the price $P_{jt'}$ and set $X_{jtt'} = 0$ such that demand $D_{jt'} = \gamma_{jt'}\alpha_j P_{jt'}^{-\beta_j} = \sum_{t'' \neq t} X_{jt''t'}$, which leads to a higher profit for selling product j in period t'. And it is the same for any $t' < t$ with $X_{jtt'} > 0$ and $(P_{jt'} - \sum_{k=t'}^{t-1} b_{jk} - c_{jt}) < 0$. Therefore, $F_t^1 \geq F_t^2(P_{jt})$ holds true for the optimal solution. Besides, $F_t^1 \geq F_t^2(P_{jt}) > 0$ must be true for the optimal solution, because a setup for producing product j in period t should be profitable.

Because $P_{jt} > 0$, $F_t^2(P_{jt}) = \gamma_{jt}\alpha_j P_{jt}^{-\beta_j}(P_{jt} - c_{jt}) - a_{jt} > 0$ is equivalent to $-a_{jt}P_{jt}^{\beta_j} + \gamma_{jt}\alpha_j P_{jt} - \gamma_{jt}\alpha_j c_{jt} > 0$. Let $F_t^3(P_{jt}) = -a_{jt}P_{jt}^{\beta_j} + \gamma_{jt}\alpha_j P_{jt} - \gamma_{jt}\alpha_j c_{jt}$ and differentiate it with respect to P_{jt}, we obtain

$$F_t^{3'}(P_{jt}) = -a_{jt}\beta_j P_{jt}^{\beta_j - 1} + \gamma_{jt}\alpha_j. \tag{28}$$

Setting this function to zero and solving for P_{jt}, we obtain $P_{jt}^* = \left(\frac{\gamma_{jt}\alpha_j}{a_{jt}\beta_j}\right)^{\frac{1}{\beta_j - 1}}$ and know that $F_t^3(P_{jt})$ increases strictly in the interval $(0, P_{jt}^*)$ and decreases strictly in the interval $(P_{jt}^*, +\infty)$. Obviously, $\left(\frac{\gamma_{jt}\alpha_j}{a_{jt}}\right)^{\frac{1}{\beta_j - 1}} > P_{jt}^*$ and $F_t^3\left(\left(\frac{\gamma_{jt}\alpha_j}{a_{jt}}\right)^{\frac{1}{\beta_j - 1}}\right) < 0$. Without loss of generality, we may suppose $F_t^3(P_{jt}^*) > 0$, and thus P_{jt}^{ub} can be obtained in the interval $(P_{jt}^*, \left(\frac{\gamma_{jt}\alpha_j}{a_{jt}}\right)^{\frac{1}{\beta_j - 1}})$ using bisection method. Because $F_t^1 \geq F_t^2(P_{jt}) > 0$ holds true for the optimal solution, $P_{jt} \leq P_{jt}^{ub}$ is true when product j is produced in period t. □

4.2. A tighter formulation

Based on the above optimality properties, we construct a tighter MINLP model for the studied problem as follows:

$$[\text{Model} \mid P_1] \quad max \sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} (P_{jt}S_{jt} - c_{jt}X_{jt} - h_{jt}I_{jt} - b_{jt}G_{jt} - a_{jt}Y_{jt}) \tag{1}$$

subject to:

$$(2) - (13)$$

$$P_{jt} \geq \frac{\min_{t' \in \mathcal{T}} A_{jt't}}{1 - \frac{1}{\beta_j}}, \quad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{14}$$

$$P_{jt} \geq P_{jt-1} + \frac{h_{jt-1}\beta_j}{\beta_j - 1} - M(1 + Y_{jt} + u^G_{jt} - u^I_{jt-1}), \quad \forall j \in \mathcal{J}, t \in \mathcal{T}/\{1\} \tag{29}$$

$$P_{jt} \leq P_{jt-1} + \frac{h_{jt-1}\beta_j}{\beta_j - 1} + M(1 + Y_{jt} + u^G_{jt} - u^I_{jt-1}), \quad \forall j \in \mathcal{J}, t \in \mathcal{T}/\{1\} \tag{30}$$

$$P_{jt} \geq P_{jt+1} + \frac{b_{jt}\beta_j}{\beta_j - 1} - M(1 + Y_{jt} + u^I_{jt-1} - u^G_{jt}), \quad \forall j \in \mathcal{J}, t \in \mathcal{T}/\{1, T\} \tag{31}$$

$$P_{jt} \leq P_{jt+1} + \frac{b_{jt}\beta_j}{\beta_j - 1} + M(1 + Y_{jt} + u^I_{jt-1} - u^G_{jt}), \quad \forall j \in \mathcal{J}, t \in \mathcal{T}/\{1, T\} \tag{32}$$

$$P_{jt} \leq P^{ub}_{jt} + M(1 - Y_{jt}), \quad \forall j \in \mathcal{J}, t \in \mathcal{T} \tag{33}$$

Constraints (14) mean the price of each product in each period is larger than its lower bound according to Optimality property 1. Constraints (29)-(30) signify that the price of product j in period t is equal to the sum of the price of it in the previous period and $\frac{h_{jt-1}\beta_j}{\beta_j - 1}$, if its demand in period t is satisfied by inventory according to Optimality property 2(i). Constraints (31)-(32) signify that the price of product j in period t is equal to the sum of the price of it in the next period and $\frac{b_{jt}\beta_j}{\beta_j - 1}$, if its demand in period t is backlogged according to Optimality property 2(ii). Constraints (33) mean that the price of product j in setup period t should be less than or equal to P^{ub}_{jt}, according to Optimality property 3. The comparison for the two models P_0 and P_1 is conducted in numerical experiment in Section 6.

5. Solution methods

In this section, we propose a model based heuristic and a genetic algorithm (GA) to solve the studied problem efficiently. The model based heuristic is a kind of fix-and-optimize optimization method, in which the 0-1 variables are fixed using heuristic and the MINLP model with fixed 0-1 variables is solved employing exact algorithm. However, such solution approach cannot deal with large-sized instances efficiently, thus we also develop a GA for the problem.

5.1. Model based heuristic

The model based heuristic is developed to find near-optimal setup configurations for the restriction model RP^k_1, in which the model reference algorithm (MRA) proposed by Bajwa et al. (2016b) is repeatedly called for obtaining the objective values of the $RP1^k$. Our idea is inspired

by Couzon et al. (2020) who first propose two model based heuristics to find near-optimal setup configurations and then feed these fixed setup configurations as the input into MRA to solve the JLSP problem efficiently. However, their proposed heuristic cannot deal with our problem with backlogging decisions. In the following, we first introduce the MRA for solving our RP_1^k, then the model based heuristic capable of handling backlogging decisions is presented.

5.1.1. MRA

In this section, we present main steps of the MRA. Readers may refer to Bajwa et al. (2016b) for detailed illustration and theoretical proof. The MRA can be used to exactly solve resource allocation problems formulated in the form of 3-index model similar to the restriction model RP_1^k. In Bajwa et al. (2016b) and Couzon et al. (2020), the authors use the MRA to allocate resources in each period $t \in \mathcal{T}$ to some periods $t' \geq t$. For our problem considering backlogging, we adapt the MRA to allocate resources in each period $t \in \mathcal{T}$ to some periods $t' \in \mathcal{T}$.

The pseudocode of the MRA is presented in Algorithm 1, in which the notations are the same as those used in the above section. The MRA begins with an initial feasible solution (line 1), iteratively updating the solution if it improves the objective value (line 4 to line 28), and finally stops at the optimal solution for the restriction model. In each iteration of MRA (line 5 to line 27), the algorithm solves T single period resource allocation problems (line 6 to line 26). For each period $t \in \mathcal{T}$, the MRA first checks whether the unconstrained solution violates the capacity constraint (17) or not, then, the single period resource allocation problem is solved based on two different cases that may occur in this period:

Case-1: The capacity constraint (17) is not binding at optimal solution, i.e., the constraint is a strict inequality in period t. The optimal capacity allocation can be found by setting the first derivatives $z^{k\prime}_{X^k_{jtt'}}$ to 0 and solving for $X^{k*}_{jtt'}$ as follows:

$$X^{k*}_{jtt'} = \frac{B_{jt'}\left(1-\frac{1}{\beta_j}\right)^{\beta_j}}{A^{\beta_j}_{jtt'}} - \sum_{l\in\mathcal{T}, l\neq t} X^k_{jlt'}. \tag{34}$$

Case-2: The capacity constraint (17) is binding for the optimal solution. In this case, the Lagrangian operator for RP_1^k is $L = -z^k + \sum_{t\in\mathcal{T}} \lambda^k_t \left(\sum_{j\in\mathcal{J}} \sum_{t'\in\mathcal{T}} v_j X^k_{jtt'} - C_t\right)$. Then, the optimal

Algorithm 1: Pseudocode of the MRA.

Input: A setup configuration $\boldsymbol{Y^k} \in \mathcal{K}$.

1 $X^k_{jtt'} = \epsilon, \forall j \in \mathcal{J}, t \in \mathcal{T}, t' \in \mathcal{T}$ (% ϵ is a sufficient small positive number, $1e-8$ in this paper);

2 $s = 0$ (% Iteration counter);

3 $z^k_0 = 0$, $\Delta z = 1$ (% z^k_s is the objective value in iteration s and Δz is the difference between the objective values at iteration s and $s-1$);

4 **while** $\Delta z > 0$ **do**

5 $\quad s = s+1, t = 1$ (% t is the counter for periods);

6 $\quad$ **while** $t \leq T$ **do**

7 $\quad\quad$ **if** *Case-1 is true* **then**

8 $\quad\quad\quad X^k_{jtt'} = \max\left(\frac{B_{jt'}\left(1-\frac{1}{\beta_j}\right)^{\beta_j}}{A^{\beta_j}_{jtt'}} - \sum_{l\in\mathcal{T}, l\neq t} X^k_{jlt'}, 0\right), \forall j \in \mathcal{J}, t' \in \mathcal{T}$;

9 $\quad\quad$ **else**

10 $\quad\quad\quad$ Sort $\frac{1}{v_j}\frac{\partial z^k}{\partial X^k_{jtt'}}(0)$ in non-increasing order, $\forall j \in \mathcal{J}, t' \in \mathcal{T}$;

11 $\quad\quad\quad$ Denote X_q as the variable associated with the q-th position in the non-increasing order of $\frac{1}{v_j}\frac{\partial z^k}{\partial X^k_{jtt'}}(0)$;

12 $\quad\quad\quad q_t = 0$, $q^*_t = J \times T$ (% q_t and q^*_t are the indices of the non-increasing order of $\frac{1}{v_j}\frac{\partial z^k}{\partial X^k_{jtt'}}(0)$);

13 $\quad\quad\quad$ **while** $q^*_t \neq q_t$ **do**

14 $\quad\quad\quad\quad q_t = q_t + 1$;

15 $\quad\quad\quad\quad$ Determine $\lambda_t(q_t)$ via the bisection method with $\frac{1}{v_{q_t}}\frac{\partial z^k}{\partial X^k_{q_t}}(0)$ as the input;

16 $\quad\quad\quad\quad$ **if** $q^*_t == q_t$ **then**

17 $\quad\quad\quad\quad\quad$ break (% break and turn to line 22);

18 $\quad\quad\quad\quad$ **else if** $\frac{1}{v_{q_t+1}}\frac{\partial z^k}{\partial X^k_{q_t+1}}(0) \leq \lambda_t(q_t)$ **then**

19 $\quad\quad\quad\quad\quad q^*_t = q_t$;

20 $\quad\quad\quad\quad$ **end**

21 $\quad\quad\quad$ **end**

22 $\quad\quad\quad \lambda^*_t = \lambda_t(q^*_t)$;

23 $\quad\quad\quad$ Compute $X^k_{jtt'}$ from equation (24);

24 $\quad\quad$ **end**

25 $\quad\quad t = t+1$;

26 $\quad$ **end**

27 $\quad \Delta z = z^k_s - z^k_{s-1}$;

28 **end**

Output: Solution of the JLSPB with fixed setup decisions.

capacity allocation in period t can be obtained as a function of the optimal λ_t^* as follows:

$$X_{jtt'}^{k*} = \begin{cases} B_{jt'} \frac{\left(1-\frac{1}{\beta_j}\right)^{\beta_j}}{\left(A_{jtt'}+\lambda_t^* v_j\right)^{\beta_j}} - \sum_{l\in\mathcal{T}, l\neq t} X_{jlt'}^k & \text{if } X_{jtt'}^{k*} \in \mathcal{N}_t^{k+} \\ 0 & \text{if } X_{jtt'}^{k*} \in \mathcal{N}_t^{k0} \end{cases}, \tag{35}$$

where N_t^{k+} denotes the set of production variables that take strictly positive values for the optimal capacity allocation in period t, and N_t^{k0} represents the set of production variables equal to 0 for the optimal capacity allocation in period t. The MRA finds the optimal λ_t^* and set N_t^{k+} by employing a problem-specific bisection method in Bajwa et al. (2016b) based on sorting $\frac{1}{v_j}\frac{\partial z^k}{\partial X_{jtt'}^k}(0)$ for all (j,t') in non-increasing order (line 10 to line 23). The optimal λ_t^* is determined by $\lambda_t^* = \lambda_t(q_t^*)$, where $\lambda_t(q_t^*)$ is obtained by the problem-specific bisection method with the q_t^*-th $\frac{1}{v_j}\frac{\partial z^k}{\partial X_{jtt'}^k}(0)$ in the non-increasing order as the input, and variable X_q associated with the q-th position in the non-increasing order of $\frac{1}{v_j}\frac{\partial z^k}{\partial X_{jtt'}^k}(0)$ belongs to $\mathcal{N}_t^{k+}$, if $q \leq q_t^*$; $\mathcal{N}_t^{k0}$, otherwise. Then production variables are updated according to equation (24) (line 23).

The quality of setup configuration directly affects the resulting solution quality for the JLSPB. In the following, a heuristic capable of handling backlogs is developed to obtain near-optimal setup configuration for our JLSPB problem. To improve the configuration obtained by the heuristic, the local search procedure proposed by Couzon et al. (2020) is also adopted.

5.1.2. Heuristic for finding setup configurations

The heuristic consists of (i) a constructive part and (ii) an improving part, whose pseudocodes are presented in Algorithm 2 and Algorithm 3 respectively. In the constructive part, the heuristic first sorts all products in decreasing order using the rule in Couzon et al. (2020) (line 1) according to the values of $\sum_{t\in\mathcal{T}} S_{jt}^{k*} = \alpha_j \left(\frac{1-\frac{1}{\beta_j}}{c_{jt}}\right)^{\beta_j}$, which represents the maximum total production quantity for product j in the entire planning horizon assuming the product is manufactured and sold in the same period and there is sufficient production capacity. For each product $j \in \mathcal{J}$, we use $\mathcal{T}_j^1$ to denote the set of periods during which sales of product j have not been covered and $\mathcal{T}_j^2$ to denote the set of periods with nonzero remaining capacity during which sales of product j have not been covered. The setup is assigned to the period $t \in \mathcal{T}_j^2$ with a maximum value of $\frac{S_{jt}^{k*}}{v_j}$ (line 8 to line 11). Then, setup variables for several periods adjacent to period t in $\mathcal{T}_j^1$ are set to be 0 by prioritizing period t' with a smaller value of $A_{jtt'}S_{jt'}^{k*}$, if the remaining capacity in period t is nonzero (line 12 to

Algorithm 2: Constructive part of the model based heuristic.

Input: Problem parameters, including $\alpha_j, \beta_j, v_j, c_{jt}, C_t$, and $A_{jtt'}$.

Sort the products in decreasing order according the value of $\alpha_j \left(\frac{1-\frac{1}{\beta_j}}{c_{jt}}\right)^{\beta_j}$;

$Y_{jt}^k = 0, \forall j \in \mathcal{J}, t \in \mathcal{T}$ (% Initialization for setup variables);

$C_t^{remaining} = C_t, \forall t \in \mathcal{T}$ (% Remaining capacity in period t);

$\mathcal{T}_j^1 = \mathcal{T}, \forall j \in \mathcal{J}$ (% $\mathcal{T}_j^1$ is the set of periods during which the sales of product j have not been covered);

$\mathcal{T}_j^2 = \mathcal{T}, \forall j \in \mathcal{J}$ (% $\mathcal{T}_j^2$ is the set of periods with nonzero capacity during which the sales of product j have not been covered);

while $\sum_{j \in \mathcal{J}} |\mathcal{T}_j^2| > 0$ **do**

  for $j \in \mathcal{J}$ **do**

    if $|\mathcal{T}_j^2| > 0$ **then**

      $t = \arg\max_{t \in \mathcal{T}_j^2} \frac{S_{jt}^{k*}}{v_j}$ (% S_{jt}^{k*} is the maximum sales quantity calculated by euqation (21));

      $Y_{jt}^k = 1$;

      $C_t^{remaining} = \max\{C_t^{remaining} - S_{jt}^{k*} v_j, 0\}$;

      $\mathcal{T}_j^+ = \{t' \mid t' > t \text{ and } t' \in \mathcal{T}_j^1\}$ (% $\mathcal{T}_j^+$ is arranged in increasing order);

      $\mathcal{T}_j^- = \{t' \mid t' < t \text{ and } t' \in \mathcal{T}_j^1\}$ (% $\mathcal{T}_j^-$ is arranged in decreasing order);

      $i^+ = 1, i^- = 1$ (% Indices for sets $\mathcal{T}_j^+$ and $\mathcal{T}_j^-$);

      while $C_t^{remaining} > 0$ *and* $\left(i^+ \leq len(\mathcal{T}_j^+) \text{ or } i^- \leq len(\mathcal{T}_j^-)\right)$ **do**

        $t^+ = \mathcal{T}_j^+(i^+)$, $t^- = \mathcal{T}_j^-(i^-)$;

        if $A_{jtt^+} S_{jt^+}^{k*} \leq A_{jtt^-} S_{jt^-}^{k*}$ *or* $i^- > len(\mathcal{T}_j^-)$ **then**

          $Y_{jt^+}^k = 0$;

          $C_t^{remaining} = \max\{C_t^{remaining} - S_{jt^+}^{k*} v_j, 0\}$;

          $i^+ = i^+ + 1$;

        else

          $Y_{jt^-}^k = 0$;

          $C_t^{remaining} = \max\{C_t^{remaining} - S_{jt^-}^{k*} v_j, 0\}$;

          $i^- = i^- + 1$;

        end

      end

      Update $\mathcal{T}_j^1$ and $\mathcal{T}_j^2$;

    end

  end

end

Output: Initial setup configuration $\boldsymbol{Y^k}$.

line 27). For example in Figure 1, suppose $\mathcal{T}_j^1$ and $\mathcal{T}_j^2$ for a product j with $v_j = 1$ are $\{2, 3, 4, 5, 6\}$

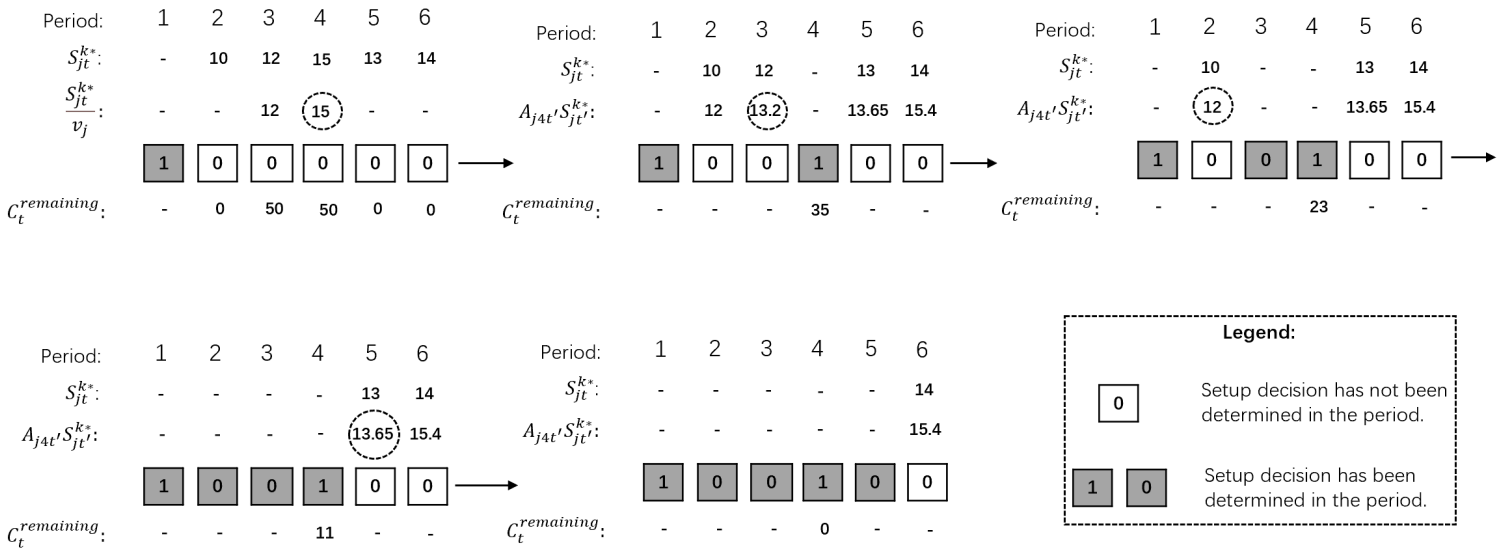


Figure 1: An illustrative example for the constructive part of the model based heuristic

and $\{3, 4\}$ respectively, and the capacity C_t is 50, where $t \in \mathcal{T}_j^2$. Suppose $\frac{S_{jt}^*}{v_j}$ for period $t \in \mathcal{T}_j^2$ is $\{12, 15\}$, then setup is assigned to period 4 with a larger value of $\frac{S_{jt}^*}{v_j}$. Suppose $S_{jt'}^*$ and $A_{jtt'}S_{jt'}^*$ for period $t' \in \mathcal{T}_j^1/\{4\}$ are $\{10, 12, 13, 14\}$ and $\{12, 13.2, 13.65, 15.4\}$ respectively, then setup variables for periods $3, 2$ and 5 are set to be 0 sequentially according line 15 to line 26. The process is repeated until capacities in all periods have been fully utilized or sales of all products in all periods have been covered (line 6 and line 32).

In the improving part of the heuristic, the initial setup configuration obtained by the Algorithm 2 for each product is repeatedly evaluated and improved if it is more profitable to combine two setups into one with a larger value of $\frac{S_{jt}^{k*}}{c_{jt}}$ (line 4 to line 23). For example in Figure 2, suppose the initial setup configuration for product 1 in 6 periods is $\{1, 1, 0, 0, 1, 0\}$, and $\frac{S_{1t}^*}{c_{1t}}$ for periods $\{1, 2, 5\}$ are $\{10, 12, 13\}$. According line 6 to line 16 in the Algorithm 3, there are two temporary configurations: (i) the first one is obtained by combining setups in periods 1 and 2 into one setup in period 2 which has a larger value of $\frac{S_{jt}^*}{c_{jt}}$; (ii) Similarly, the second one is obtained by combining setups in periods 2 and 5. Then, the temporary configurations are evaluated by the MRA and thus the configuration for product 1 is updated to be the first temporary configuration as it leads to a larger objective value. Next, the algorithm attempts to combine setups for product 1 in periods 2

Algorithm 3: Improving part of the model based heuristic.

Input: Initial setup configuration $\boldsymbol{Y}^k$ obtained by Algorithm 2;
Problem parameters including c_{jt}.

1 best_obj = MRA($\boldsymbol{Y}^k$);
2 $\boldsymbol{Y}^{k\prime} = \boldsymbol{Y}^k$;
3 $\mathcal{T}_j^Y = \{t \mid Y_{jt}^{k\prime} = 1\}, \forall j \in \mathcal{J}$;
4 **for** $j \in \mathcal{J}$ **do**
5 **while** $|\mathcal{T}_j^Y| \geq 2$ **do**
6 temp_best_obj = best_obj;
7 **for** $i = 1 : |\mathcal{T}_j^Y| - 1$ **do**
8 $t_1 = \mathcal{T}_j^Y(i), t_2 = \mathcal{T}_j^Y(i+1)$ ($\%\mathcal{T}_j^Y(i)$ denotes the i-th element in $\mathcal{T}_j$);
9 $\boldsymbol{Y}^{temp} = \boldsymbol{Y}^{k\prime}$;
10 Combine $Y_{jt_1}^{temp}$ and $Y_{jt_2}^{temp}$ into one with a larger value of $\frac{S_{jt}^*}{c_{jt}}$;
11 temp_obj = MRA($\boldsymbol{Y}^{temp}$);
12 **if** *temp_obj* > *temp_best_obj* **then**
13 $\boldsymbol{Y}_{best}^{temp} = \boldsymbol{Y}^{temp}$;
14 temp_best_obj = temp_obj;
15 **end**
16 **end**
17 **if** *temp_best_obj* > *best_obj* **then**
18 $\boldsymbol{Y}^{k\prime} = \boldsymbol{Y}_{best}^{temp}$;
19 best_obj = temp_best_obj;
20 Update $\mathcal{T}_j^Y$;
21 **else**
22 break (% break and turn to line 4);
23 **end**
24 **end**
25 **end**

Output: The final setup configuration $\boldsymbol{Y}^{k\prime}$.

and 5 into one setup in period 5, however, such combination leads to a lower objective value and the configuration is not updated. The algorithm continues to try combinations of setups for products 2 and 3 till the stop criterion is met.

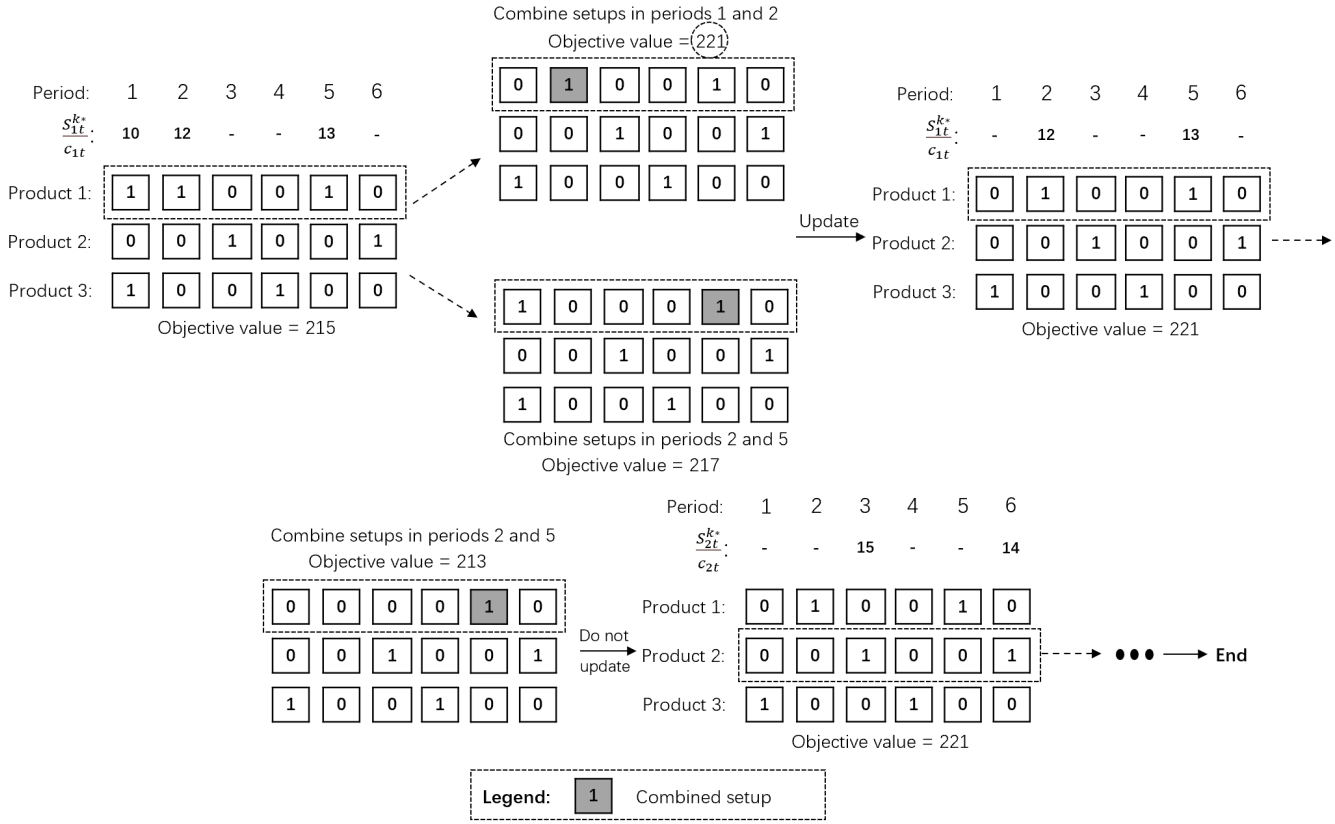


Figure 2: An illustrative example for the improving part of the model based heuristic

5.1.3. Local search procedure for the model based heuristic

Three local search moves proposed in Couzon et al. (2020) are applied to find better solutions by exploring the immediate neighborhood of the incumbent solution. The neighborhood of incumbent solution is represented by the moves used to revise setup configuration obtained by the Algorithm 3. All the three local search moves are used with a "first-improvement" policy to limit the computational load of the local search, which is consistent with Couzon et al. (2020). Note that feasibility of the solution is guaranteed in this local search procedure, as the MRA can always find a set of 3-index variables satisfying the capacity constraints either in *Case-1* or in *Case-2* for any given

setup configuration. Three moves are executed in sequence after the improving part as follows:

- Move 1:
  - Find two periods during which two products have a setup assigned to both periods;
  - Unassign a setup randomly for each product.
- Move 2:
  - Find two periods during which two products have a setup assigned to one of the two periods respectively;
  - Swap the two setups, i.e. reassign the setup to the other one period for each product.
- Move 3: Shift one setup of one product to another period.

5.2. Genetic algorithm

Since the problem formulated above is NP-hard and nonlinear which cannot be solved by the state-of-art commercial solvers efficiently, we also propose a GA to solve it. Two crossover operators and four mutation operators are proposed based on the features of the problem. The probabilities of them are adaptively adjusted according to the fitness values of individuals generated by them. As the crossover and mutation operators can generate an awful of infeasible individuals for the studied problem, we propose three repair operators to progressively deal with infeasibilities caused by unmet backlogs, violations of the capacity constraints and excess inventories. In addition, a local search procedure is also incorporated at the end of each iteration to exploit more solution space and avoid premature. The general scheme of the proposed GA is displayed in Algorithm 4 as follows:

In Algorithm 4, the procedure Initialization() is used to generate initial populations. In each iteration, child individuals are generated by Crossover() from a set of parent individuals selected by function Parent_Selection(). Then the population is perturbed by Mutation(). In addition, function Repair() is called after Crossover() and Mutation(). After these procedures, Loca_Search() function is called. The details of these procedures are presented in the following subsections.

5.2.1. Solution representation

Based on the mathematical model P_0 and Theorem 1 in Bajwa et al. (2016a) mentioned before, we know that if production quantities and prices of all products are determined, the other decision

Algorithm 4: Framework of the proposed GA.

Input: Problem parametersincluding $\alpha_j, \beta_j, v_j, \gamma_{jt}, c_{jt}, h_{jt}, b_{jt}$, and C_t;
MaxIt (% Maximum iteration number);
MaxIt_{no} (%Maximum number of iterations without improving the best solution).

POP = Initialization() (% POP denotes the population);
It = 0 (% Iteration counter);
$\text{It}_{no} = 0$ (% Iteration counter for counting the number of iterations without improvement);
best_obj_It = 0 (% Objective value of the best individual in this iteration);
best_obj = 0 (% Objective value of the best individual found so far);
while $It < MaxIt$ *and* $It_{no} < MaxIt_{no}$ **do**
    It = It + 1;
    Evaluation(POP);
    if $best\_fitness\_It - best\_obj \leq 1e-4$ **then**
        $\text{It}_{no} = \text{It}_{no} + 1$;
    else
        best_obj = best_obj_It;
        $\text{It}_{no} = 0$;
    end
    Parent_POP = Parent_Selection(POP);
    POP = Parent_POP ∪ Crossover(Parent_POP);
    Repair(POP);
    Mutation(POP);
    Repair(POP);
    Local_Search(POP);
end

Output: Solution of the JLSPB.

variables can be calculated uniquely. Therefore, a real number encoding scheme is employed to represent the solution with two gene chains, where the decision variables X_{jt} and P_{jt} are encoded as chromosomes respectively. Thus, each gene chain contain $J \times T$ genes. Figure 3 illustrates this representation scheme for a problem with two products and four periods, thus the individual has $2J \times T = 16$ genes in all.

Period 1		Period 2		Period 3		Period 4	
X_{11}	X_{21}	X_{12}	X_{22}	X_{13}	X_{23}	X_{14}	X_{24}
P_{11}	P_{21}	P_{12}	P_{22}	P_{13}	P_{23}	P_{14}	P_{24}

Figure 3: Representation for a problem with two products and four periods.

5.2.2. Population initialization

The initial population is generated randomly. For each individual, the production quantities of all products in period t are initialized as follows:

step 1 generate an increasing sequence with $J+1$ numbers $K_0, K_1, \ldots, K_J$, where $K_0 = 0, K_J = C_t$ and $K_j, j \in \{1, 2, \ldots, J-1\}$, is a real number randomly generated in the interval $[0, C_t]$;

step 2 calculate the initial production quantity of product j in period t by the equation $X_{jt} = K_j - K_{j-1}$.

Note that the capacity constraints (3) are guaranteed in the above production quantities initializing process. After generating initial X_{jt}, next the initial price of each product is randomly generated from a feasible price interval. The lower bound for optimal price given by Optimality property 1 1 is adopted as the lower bound for the feasible price interval for initialization. It is much easier to get an upper bound for linear demand functions, but not vice versa for an isoelastic one. The reason is that the price of product j during period t can be arbitrary large when there has been a setup for this product in another period t_0, i.e., the profit of selling product j in period t is always nonzero as long as the price P_{jt} is larger than the unit inventory or backlogging cost from period t_0 to t. To tackle this problem, the upper bound for the feasible price interval is set to be $2P_{jt}^{LB}$, i.e., the initial price of product j during period t will be randomly generated from $[P_{jt}^{LB}, 2P_{jt}^{LB}]$. Note that this interval only represents the range of prices in initial solutions. Considering that the optimal prices may be larger than $2P_{jt}^{LB}$, so in the following iterative search procedure of proposed GA, the prices may be adjusted to be larger than $2P_{jt}^{L}$ in crossover and repair procedures.

5.2.3. Fitness evaluation

In the evaluation process, the fitness value of an individual i is calculated as follows:

$$fitness_i = \frac{\frac{1}{rank_i}}{\sum_{k=1}^{POP_{size}} \frac{1}{rank_k}}, \tag{36}$$

where $rank_i$ is the rank of this individual in the decreasing order of objective values for the population. Note that each individual may be infeasible due to violation of the capacity constraints (3) or having excess inventories or unmet backlogs in the end period. Therefore, penalty costs are adopted for capacity violation and unmet backlogs, while excess inventories are not penalized because we have paid for them when calculating the objective function. The penalty cost for per unit of unmet backlog of a product is set to be its maximum price of all periods, and penalty cost for per unit

of production quantity exceeding capacity during one period is set to be the maximum price of all products in that period.

5.2.4. Parent selection

Individuals are selected from the current population to generate offsprings for the new generation by some methods, such as roulette wheel selection, tournament selection, rank selection, elitism, etc (Michalewicz, 2013). Here, roulette wheel selection and elitism method are adopted, because the elitism method ensures the best individual to remain in the new population (Davis, 1991), and roulette wheel selection tends to promote the diversity of the population and thus avoids premature (Liu et al., 2008). Specifically, the best individual is first copied to the new population, while the rest are selected by roulette wheel selection. The selection of the first parent is performed by using the probability distribution $\frac{2(\mathrm{POP}_{size}+1-i)}{\mathrm{POP}_{size}(\mathrm{POP}_{size}+1)}$, where POP_{size} is the population size and i represents the i-th chromosome in the increasing order of the fitness values. A random number between 0 and 1 is generated and the first individual with a cumulative probability that is greater than the random number is chosen as the first parent. The second parent is randomly selected with equal probability among others.

5.2.5. Crossover

Two different crossover operators are designed to combine information from both parents. One is single-point linear crossover that combines genes from two parent individuals on the basis of one cross point cp , which is faster for the real-coded representation compared to the binary crossover (Rezaei and Davoodi, 2012). Superscripts pa and ch of decision variables are used to denote parent individuals and child individuals respectively. With the two selected individuals pa_1 and pa_2, the child individual ch_1 and ch_2 are generated as follows:

$$\begin{cases} X_{jt}^{ch_1} = rX_{jt}^{pa_1} + (1-r)X_{jt}^{pa_2} & j \times t \leq cp \\ X_{jt}^{ch_2} = (1-r)X_{jt}^{pa_1} + rX_{jt}^{pa_2} & j \times t > cp \end{cases} \quad \begin{cases} P_{jt}^{ch_1} = rP_{jt}^{pa_1} + (1-r)P_{jt}^{pa_2} & j \times t \leq cp \\ P_{jt}^{ch_2} = (1-r)P_{jt}^{pa_1} + rP_{jt}^{pa_2} & j \times t > cp \end{cases}, \tag{37}$$

where r is a random real number, and cp is an integer randomly generated between 0 and $J \times T$. The range for random number r is determined empirically depending on a particular problem, for instance $[-0.5, 1.5]$ in Michalewicz (2013) and $[1, 1.25]$ in Liu et al. (2008). In our case, r is randomly generated in $[0, 1.5]$, which allows the variables, especially the prices, to vary in relatively large ranges. Note that decision variables of child individuals generated by this crossover operator

may be negative, thus we update these negative variables to be a sufficient small positive number. For the example, two child individuals are generated from parent solutions by crossover operator

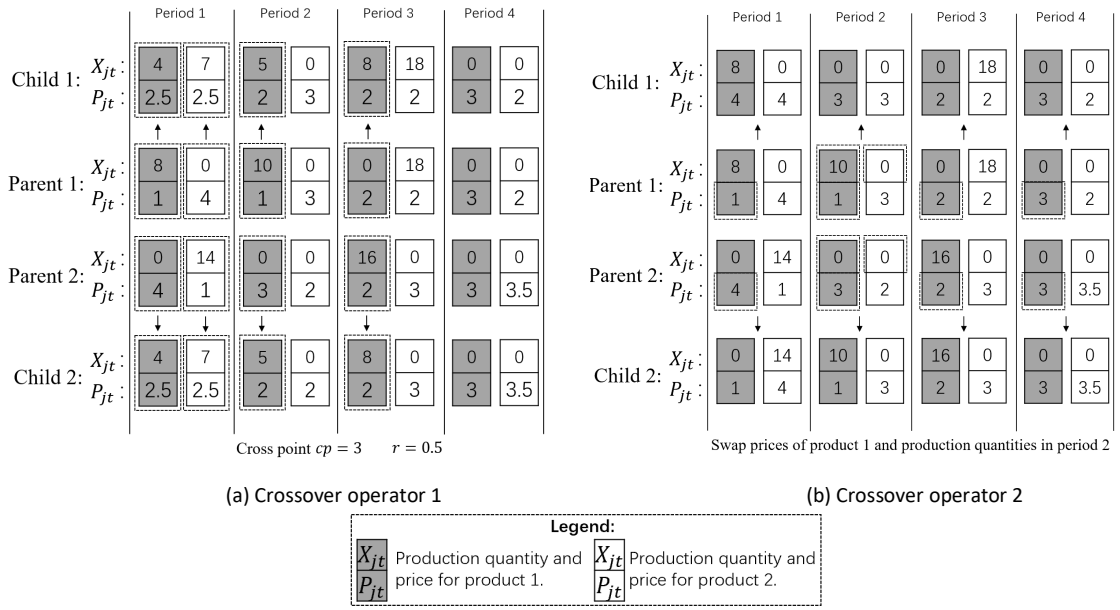


Figure 4: The illustrative examples for the crossover operators.

1 in Figure 4(a), with a random selected crossover point $cp = 3$. Then for two child individuals, the genes (variables) with index combinations $(1, 1)$, $(1, 2)$, $(2, 1)$ and $(1, 3)$ are generated by the equation (26), while the rest genes are copied from the parents individuals. The other crossover operator is a problem-specific one that swaps production quantities and prices of the selected parents respectively. The production quantities in a randomly selected period t are swapped between the parents, while the prices of a randomly selected product j are swapped between them. For example, two child individuals are generated by swapping production quantities in period 2 and swapping prices of product 1 as depicted in Figure 4(b).

In the crossover procedure, two parent individuals are selected to generate child individuals with a specified crossover probability P_c, during which one of the two crossover operators will be applied according their weights by roulette wheel selection. The initial weight ω_i^c of crossover operator i is set to be 0.5. To choose a better crossover operator as the GA goes on, the weights of them are adjusted dynamically, every time point when 100 child individuals have been generated, based

on the objective values of these 100 child individuals. We use n_i to indicate the number of child individuals generated by operator i among the 100 child individuals and π_i to indicate the number of child individuals generated by operator i which are better than their parents in terms of objective value. The score sc_i of operator i can be calculated as $\frac{\pi_i}{n_i}$. Then, the new weight $\bar{\omega}_i$ of operator i will be updated as follows:

$$\bar{\omega}_i^c = \frac{sc_i}{\sum_{k=1}^{2} sc_k}. \tag{38}$$

5.2.6. Mutation

To avoid premature and allow for a wide exploration of the solution space, four mutation operators are designed based on the features of the problem. The first two mutation operators modify production quantities while the latter two change prices of individuals.

The first mutation operator randomly selects a period t and reallocates production capacity for all products as the gene initialization procedure. Production quantities of two randomly selected products j_1 and j_2 are swapped during two randomly selected periods t_1 and t_2 by the second mutation operator as follows:

$$\begin{cases} X'_{j_1t_1} = \frac{v_{j_2} X_{j_2t2}}{v_{j_1}} \\ X'_{j_1t_2} = \frac{v_{j_2} X_{j_2t1}}{v_{j_1}} \end{cases} \qquad \begin{cases} X'_{j_2t_1} = \frac{v_{j_1} X_{j_1t2}}{v_{j_2}} \\ X'_{j_2t_2} = \frac{v_{j_1} X_{j_1t1}}{v_{j_2}} \end{cases} . \tag{39}$$

For each product, the price of it during a randomly selected period is regenerated between its initial price interval in the same way as the initialization procedure when the third mutation operator is performed. The last mutation operator is dedicated to swap the prices of each product during two randomly selected periods. The illustrative examples for these four mutation operators on an instances with two products and four periods are illustrated in Figure 6. Specifically, in Figure 5(a), production quantities in period 2 are re-initialized to be 13 and 7 by mutation operator 1. As depicted in 5(b), production quantities for both products in periods 3 and 4 are swapped in by mutation operator 2. Prices for both products in period 1 are re-initialized by mutation operator 3 in 5(c), and prices for each product in periods 3 and 4 are swapped by mutation operator 4 in 5(d).

Each individual in the population, except the best one, will mutate with mutation probability P_m in the mutation procedure. Similar to crossover procedure, one of these four mutation operators is applied according their weights by roulette wheel selection. The initial weight ω_i^m of mutation

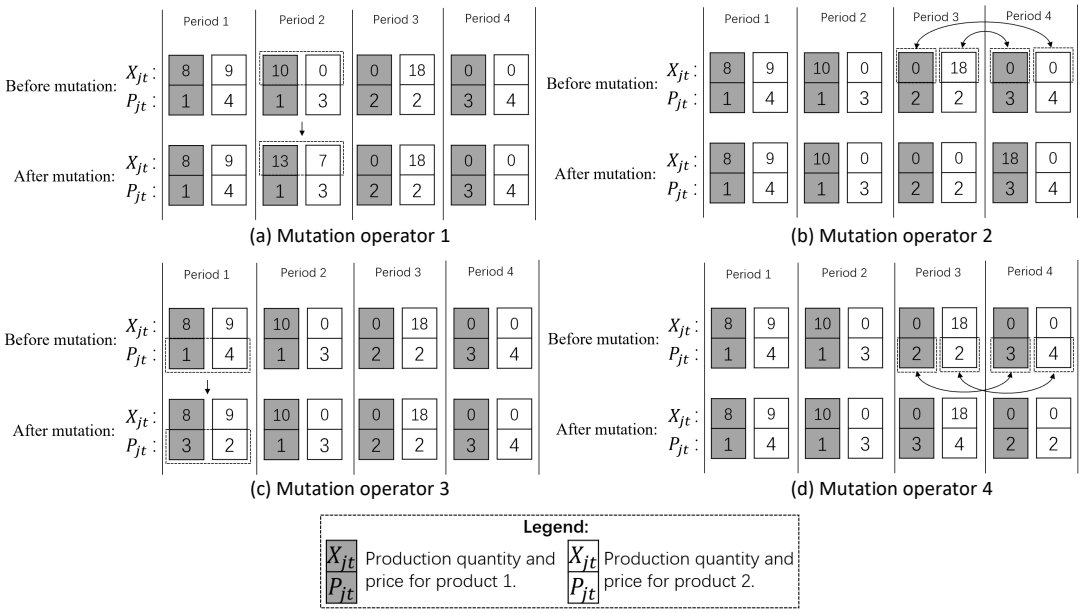


Figure 5: The illustrative examples of the mutation operators.

operator i is set to be 0.25. During the iterative procedure, the weights are adjusted dynamically, every time point when 100 new individuals have been generated, based on the objective values of the 100 new individuals. The weights are adjusted in the same way as mentioned in crossover section, thus we do not repeat describing the process again here.

5.2.7. Repair

New individuals generated by crossover and mutation operators may be infeasible for capacity violations and excess inventories or unmet backlogs at the end of the last period in the planning horizon. Therefore, repair operators are designed to reallocate production capacity and adjust prices so as to reduce infeasibilities. After a new individual is generated by crossover and mutation operators, a checking procedure first examines the used production capacity in each period and excess inventories or unmet backlogs of each product at the end of the last period (excess inventories or unmet backlogs for short in the following statement). If infeasibilities exist, three repair operators are performed in sequence.

- Repair operator 1: Repair operator 1 is called if and only if the infeasibilities are caused only by

excess inventories. In other words, if there exist excess inventories but no unmet backlogs, and production capacity constraints are satisfied, then for each product j with excess inventories, the repair operator will lower prices or reduce production quantities during some randomly selected periods, until there are no excess inventories.

- Repair operator 2: Repair operator 2 is designed for eliminating infeasibilities caused by unmet backlogs. If there exist unmet backlogs, the following steps are conducted: (i) The repair operator will first examine whether there also exist excess inventories and then reallocate production capacity consumed by excess inventories of some products to produce more products with unmet backlogs; (ii) When excess inventories do not exist, idle production capacity will be assigned to fulfill unmet backlogs; (iii) In case that production capacity of all periods has been fully utilized, prices of some randomly selected products will be increased to balance demand and supply such that unmet backlogs can be reduced to be zero.

- Repair operator 3: Repair operator 3 is designed for eliminating infeasibilities caused by violations of the capacity constraints. If the production capacity constraints are violated, the following steps are conducted: (i) The repair operator first examines whether there also exist excess inventories. Then production quantities that lead to excess inventories are reduced in the periods with violation of the capacity constraints. (ii) If there do not exist excess inventories, the repair operator searches the periods with idle production capacity, and then the production quantities exceeding the capacity will be shifted to these periods with idle capacity. (iii) In case that there exist neither excess inventories nor idle capacity, the prices of products with nonzero production quantities in the periods with capacity violations, will be increased to suppress the corresponding demand. Then, the production quantities exceeding the capacity will be reduced to eliminate infeasibilities.

An illustrative examples for the repair operators are presented in Figure 6. In Figure 6(a), there are 16 units of excess inventories for product 1. Repair operator 1 first reduces production quantity X_{13} to zero, and then decreases price P_{14} to improve the corresponding demand such that the excess inventories for product 1 are reduced to be zero. As depicted in Figure 6(b), there are 13 unit of unmet backlogs for product 1 in the beginning. Repair operator 2 first shifts 10 units of capacity consumed by excess inventories of product 2 in period 2 to satisfy unmet backlogs of product 1. Then, 2 units of idle capacity in period 1 are used to satisfy unmet backlogs of product 1 and the

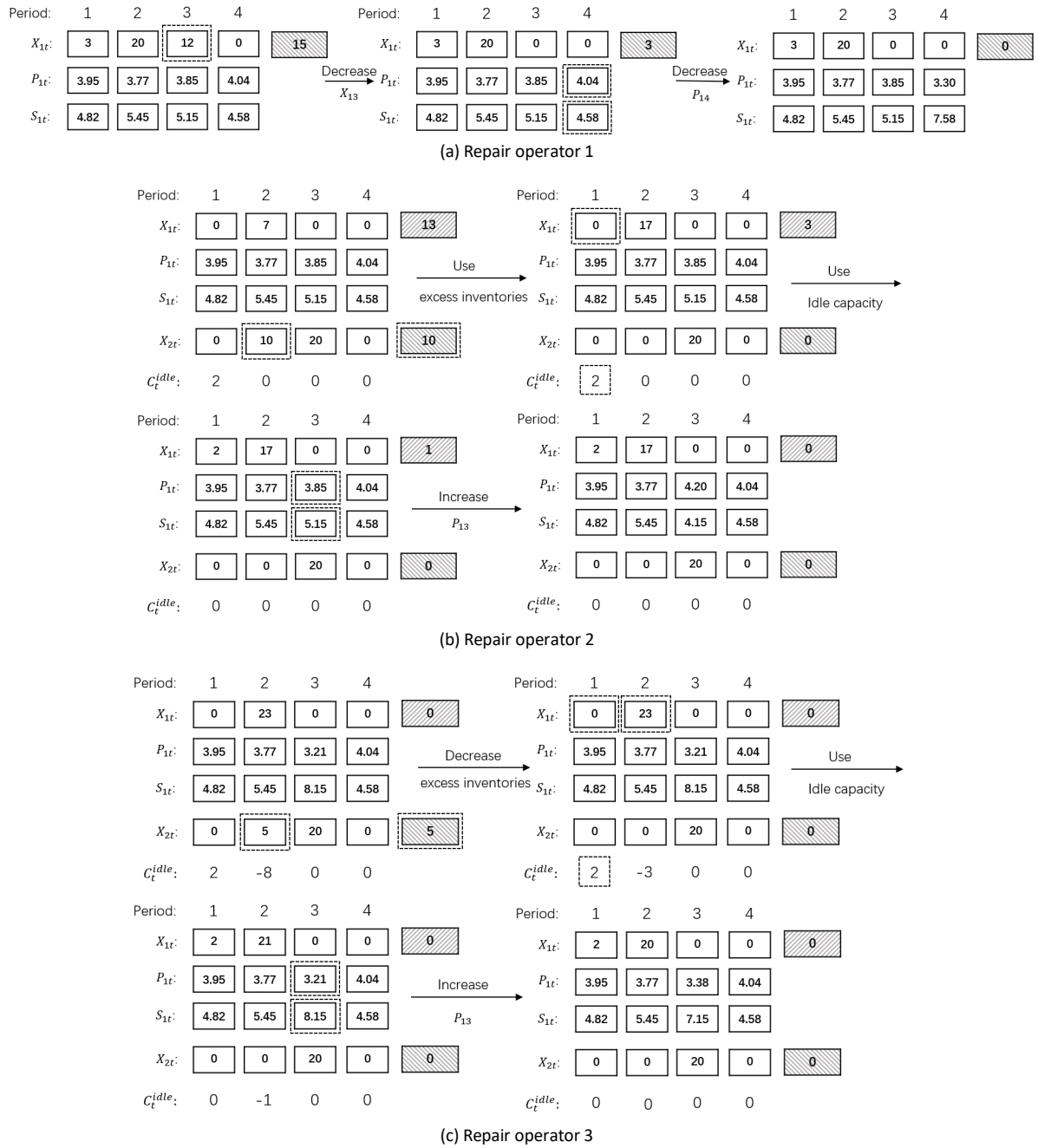


Figure 6: The illustrative examples for the repair operators.

price P_{13} is increased to suppress demand such that the unmet backlogs are finally eliminated. The example for eliminating infeasibilities caused by violations of capacity constraints is presented in Figure 6(c), in which negative idle capacity in period 2 denotes that the capacity constraints is violated. Repair operator 3 first reduces 5 units of production quantities for product 2 in period 2 that leads to excess inventories. Then, 2 units of production for product 1 in period 2 are shifted to be produced in period 1 using the idle capacity. Finally, the violation is eliminated by increasing price P_{13} and reducing production quantity X_{12}.

Note that individuals adjusted by repair operators 2 and 3 may still have excess inventories, as the repair operator 1 is called only if the infeasibilities are caused by excess inventories. The algorithm does not need to repair these individuals immediately, because they will either be weeded out due to low fitness or be repaired in the next iteration.

5.2.8. Local search procedure in the GA

Local search has been widely used in the literature to enhance the performance of meta-heuristic algorithms. Thus, to help the proposed GA escape from local optima and revive the search further, a problem-specific local search is designed to exploit more information from incumbent solutions. The local search procedure will be performed on each individual with a specified probability P_l, except the best one. Pseudocode of the proposed local search is presented in Algorithm 5. The main idea of this local search is to combine production quantities in different periods, which is beneficial for improving objective value of the solution when production capacity is relatively sufficient.

6. Numerical experiments

In this section, we first present an illustrative example to help readers understand better the MINLP model and discuss managerial insights. Then, the efficiency of LINGO solver on MINLP models P_0 and P_1 is compared, and finally the performance of the model based heuristic and the GA is evaluated based on a series of instances. Both the solution methods are coded in Python and computational results have been conducted on a personal computer with 2.4 GHz Intel Core i5 and 16 GB RAM, except that parameter tuning procedure is conducted on a cloud server with 3.8 GHz Intel Xeon Platinum (Cooper Lake) and 24 GB RAM. In addition, the nonlinear commercial solver LINGO 18.0 is called to solve the MINLP models P_0 and P_1 with a CPU time limit of 3600s.

Algorithm 5: Pseudocode of the local search procedure in the GA.

Input: A selected individual in the population;
C_t^{idle} (% Idle capacity in period t);
$\mathcal{T}_j = \{t \mid X_{jt} > 0 \text{ and } t \in \mathcal{T}\}, \forall j \in \mathcal{J}$(% A set of periods during which $X_{jt} > 0$).

1 Shuffle $\mathcal{J}$ randomly (% Improve randomness of the procedure);
2 **for** $j \in \mathcal{J}$ **do**
3 Shuffle $\mathcal{T}_j$ randomly (% Improve randomness of the procedure);
4 **while** $|\mathcal{T}_j| \geq 2$ **do**
5 $t = \mathcal{T}_j(1)$ (% $\mathcal{T}_j(1)$ denotes the first element in $\mathcal{T}_j$);
6 **for** $t' \in \mathcal{T}_j / \{t\}$ **do**
7 **if** $C_t^{idle} \geq v_j X_{jt'}$ **then**
8 Shift $X_{jt'}$ to period t ;
9 $C_t^{idle} = C_t^{idle} - v_j X_{jt'}$;
10 $\mathcal{T}_j = \mathcal{T}_j / \{t'\}$;
11 break (% break and turn to line 4);
12 **else if** $C_{t'}^{idle} \geq v_j X_{jt}$ **then**
13 Shift X_{jt} to period t' ;
14 $C_{t'}^{idle} = C_{t'}^{idle} - v_j X_{jt}$;
15 $\mathcal{T}_j = \mathcal{T}_j / \{t\}$;
16 break (% break and turn to line 4);
17 **else**
18 $t'' = \arg\max_{t \in \mathcal{T}_j} C_t^{idle}$;
19 **if** $C_{t''}^{idle} \geq v_j \left(X_{jt} + X_{jt'}\right)$ **then**
20 Shift X_{jt} and $X_{jt'}$ to period t'' ;
21 $C_{t''}^{idle} = C_{t''}^{idle} - v_j \left(X_{jt} + X_{jt'}\right)$;
22 $\mathcal{T}_j = \mathcal{T}_j / \{t, t'\}$;
23 **if** $t'' \notin \mathcal{T}_j$ **then**
24 $\mathcal{T}_j = \mathcal{T}_j \cup \{t''\}$
25 **end**
26 break (% break and turn to line 4);
27 **end**
28 **end**
29 **end**
30 **end**
31 **end**

Output: A new individual.

Table 2: Numerical parameters list

Name	Type	Range	Description
POP_{size}	Integer	$[20, 100]$	Population size
P_c	Real	$[0.5, 1]$	Crossover probability for the parent individuals
P_m	Real	$[0.01, 0.4]$	Mutation probability for each individual
P_l	Real	$[0.01, 1]$	Local search probability for each individual
MaxIt_{no}	Integer	$[20, 100]$	Maximum iterations without improving the best solution

The objective values of each instance obtained by both solution methods and the corresponding solution times are reported with an average of ten runs.

6.1. Parameter tuning

The parameters to be tuned of the proposed GA are listed in Table 2. These parameters has been tuned by employing an automatic algorithm configuration tool, the IRACE package (López-Ibánez et al., 2016). This automatic approach has been widely used for tuning configurable algorithms with a number of parameters that can be preset and affect the computational efficiency (Dell et al., 2016; Pinto et al., 2018; Franzin and Stützle, 2019). The IRACE package tunes parameters in parallel mode based on a method named iterated racing that consists of three main steps: (1) sampling parameter combinations according to a particular distribution; (2) selecting the best combinations from newly sampled ones via racing method; (3) updating the sampling distribution to bias the procedure towards better combinations. The IRACE package needs the target algorithm, the ranges of the parameters to be tuned, a set of training instances and a set of initial parameters combinations (optional) as the input. A training budget is preset by users to control the maximum number of experiments to be executed, after which a set of elite parameters combinations will be returned.

Table 3: Initial and elite parameter combination

	Initial combination				Elite combination			
Parameter	1	2	3	4	1	2	3	4
POP_{size}	30	60	70	80	68	84	74	84
P_c	0.90	0.65	0.80	0.80	0.89	0.88	0.85	0.82
P_m	0.05	0.10	0.15	0.30	0.12	0.08	0.06	0.07
P_l	0.50	0.75	0.95	1.00	0.95	0.98	0.90	0.95
MaxIt_{no}	50	70	50	100	100	92	91	94

The training instances are generated based on benchmark instances with production capacity 40 and 80 from Bajwa et al. (2016b) and Couzon et al. (2020). As backlogging is not allowed in Bajwa et al. (2016b) and Couzon et al. (2020), in our training instances, backlogging cost b_{jt} is

set to be $2h_{jt}$ in line with Cheng et al. (2001), Toledo et al. (2013), and Goren and Tunali (2016). The training budget of IRACE is set to be 3000 in our experiments, and a set of initial parameters combinations, obtained by preliminary experiments, is fed to enhance the tuning procedure (see Table 3). Finally, 4 elite combinations returned by the IRACE package are also presented in Table 3. In the following experiments, we adopt the first elite combination 1 (i.e., $\text{POP}_{size} = 68, P_c = 0.89, P_m = 0.12, P_l = 0.95$, and $\text{MaxIt}_{no} = 100$) as the actual parameter values.

6.2. An illustrative example

Table 4: Parameters of the illustrative example

Product	α_j	β_j	v_j	c_{jt}	h_{jt}	b_{jt}	a_{jt}
1	500	1.9	1	1.6	0.02	0.04	8.5
2	400	1.6	1	1.3	0.05	0.1	4.5
3	600	2.5	1	1.5	0.04	0.08	7.5

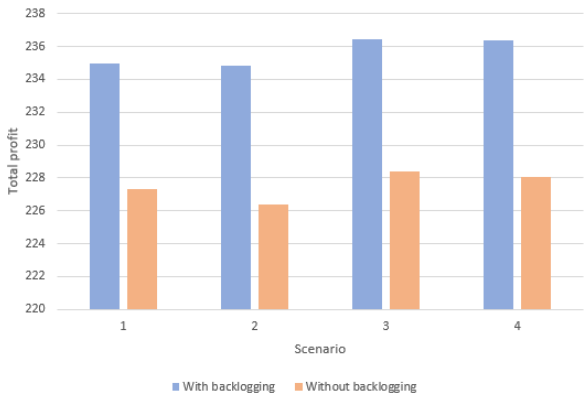


Figure 7: The comparison of total profits for the illustrative example.

In this section, an illustrative example with 3 products and 6 time periods is provided to help readers understand better the MINLP model P_1 and Optimality property 2 and discuss managerial insights. The example is generated based on an industrial glove manufacturer in Bajwa et al. (2016b) in which backlogging is not permitted. The product parameters in Table 4 and production capacity of 80 are identical to those in Bajwa et al. (2016b). In line with Cheng et al. (2001), backlogging cost is set as $2h_{jt}$. Demand scenarios are set as the same in Bajwa et al. (2016b), which is presented in Table 6 in Section 6.3. The example is formulated by the model P_1 and exactly solved by calling the well-known LINGO solver. The comparison of total profits with or

without backlogging for the example is shown in Figure 7. It can be observed from Figure 7 that the total profit with backlogging under all scenarios is higher than that without backlogging. Furthermore, the optimal prices of scenarios 1-4 with backlogging are illustrated in Figure 8(a)-(d) respectively, in which dotted circles represent the setup periods (as illustrated in Figure 8(e)). The details of the optimal solution including production quantity, inventory quantity, backlogging

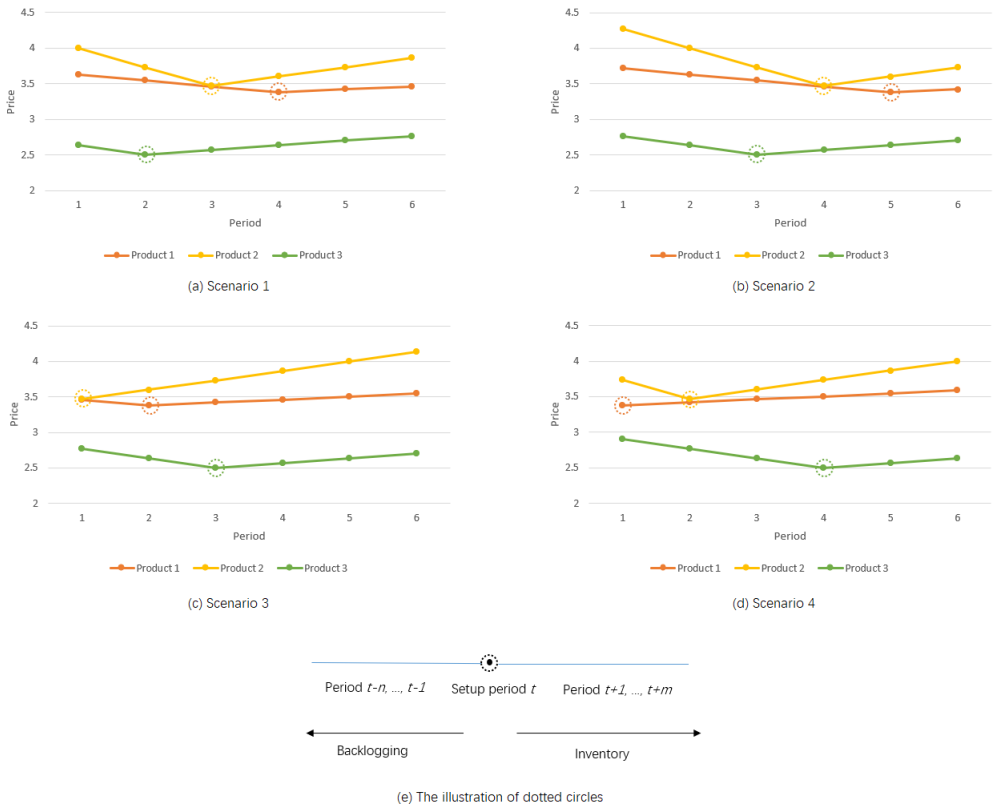


Figure 8: The optimal prices for the illustrative example.

quantity and price are provided in the supplementary material. We can observe from Figure 8(a) that products 1-3 are produced in periods 4, 3 and 2, respectively. For each product, the demands before and after setup period are satisfied by backlogging and inventory, respectively. And the optimal prices of a product decrease before setup period and increase after setup period. The trend of prices in Figure 8(a) is in line with that described in Optimality property 2, i.e., (i) if the demand in period t is completely satisfied by inventory, its optimal price is greater than that in the next period; (ii) if the demand in period t is completely satisfied by backlogging, its optimal price is

greater than that in the previous period. And the price difference can be calculated by the formulas provided in Optimality property 2. For example, $P_{11} - P_{12} = 3.4622 - 3.3778 = \frac{b_{11}\beta_1}{\beta_1 - 1} = 0.0844$, and $P_{16} - P_{15} = 3.5466 - 3.5044 = \frac{h_{15}\beta_1}{\beta_1 - 1} = 0.0422$. The prices in other demand scenarios (Figure 8(b)-(d)) show the same trend. For the example and Optimality property 2, we can give the following managerial insights:

(1) The formulated MINLP model can help manufacturers make the decision on whether applying backlogging strategy or not and when to backlog the demand;

(2) If demand in a period is satisfied by backlogging strategy, it is more profitable to set its price higher than that in the next period, and the larger the backlogging cost, the higher its price should be;

(3) If demand in a period is satisfied by inventory, it is more profitable to set its price higher than that in the previous period, and the larger the inventory holding cost, the higher its price should be.

6.3. Numerical experiment on small-sized instance

The small-sized instances are generated based on 2 benchmark instance sets proposed by Bajwa et al. (2016a,b) with additional backlogging information. These instance sets, which are combined with 4 demand scenarios, are collected from a real-world company manufacturing selling industrial gloves to institutional buyers. The detailed data of the instance sets and demand scenarios are given in Table 5 and Table 6. The demand scenarios mainly differ in demand seasonality γ_{jt} for each product. The first scenario indicates the case without seasonality. The second scenario has an increasing demand for each product in the planning horizon, while the demand for each product decreases over the horizon under the third scenario. Finally, the last scenario is a combination of the second and third scenarios. Production capacity is the same and constant in each period over the planning horizon. In addition, 7 different production capacities varying from 40 to 110 are tested for the instance sets. In total, there are 64 small-sized instances that are solved by calling LINGO solver. Computation time limit of LINGO is set as 3600s. LINGO can propose a feasible solution or an optimal solution with the time limit.

The comparison of the efficiency of MINLP models P_0 and P_1 is reported in Table 7. The column "Average obj" represents average objective value obtained by LINGO solver, and the column "Best" reports the number of best solutions obtained for the models P_0 and P_1. From Table 7, we can

Table 5: Data for small-sized instances

	α_j	β_j	v_j	c_{jt}	h_{jt}	b_{jt}	a_{jt}
	500	1.9	1.0	1.6	0.02	0.04	8.5
Instance set 1	400	1.6	1.0	1.3	0.05	0.10	4.5
	600	2.5	1.0	1.5	0.04	0.08	7.5
	20000	3.5	1.0	3.0	0.035	0.07	10.5
Instance set 2	18000	4.0	1.0	3.0	0.035	0.07	4.5
	800	5.5	1.0	3.0	0.013	0.026	3.5

[1] Backlogging cost b_{jt} is set to be $2h_{jt}$ in the same way as in Cheng et al. (2001), Toledo et al. (2013) and Goren and Tunali (2016).

Table 6: Demand scenarios

	Period					
Scenario	1	2	3	4	5	6
	0.1667	0.1667	0.1667	0.1667	0.1667	0.1667
Scenario 1	0.1667	0.1667	0.1667	0.1667	0.1667	0.1667
	0.1667	0.1667	0.1667	0.1667	0.1667	0.1667
	0.1	0.1	0.1	0.2	0.2	0.3
Scenario 2	0.1	0.1	0.1	0.2	0.2	0.3
	0.1	0.1	0.1	0.2	0.2	0.3
	0.3	0.2	0.2	0.1	0.1	0.1
Scenario 3	0.3	0.2	0.2	0.1	0.1	0.1
	0.3	0.2	0.2	0.1	0.1	0.1
	0.3	0.2	0.2	0.1	0.1	0.1
Scenario 4	0.3	0.2	0.2	0.1	0.1	0.1
	0.1	0.1	0.1	0.2	0.2	0.3

observe for the Instance set 1 that the average profits of models P_0 and P_1 are comparable, and the number of best solutions obtained for P_0 and P_1 are 29 and 32 respectively. For the Instance set 2, the average profit of P_0 and P_1 are 185,77 and 221,59, respectively. And 13 and 32 best solutions are obtained on P_0 and P_1. The experiment results show that the tighter model P_1 is more efficient than model P_0.

Table 7: Comparison of the efficiency of LINGO solver on the two MINLP models

	Model P_0		Model P_1	
	Average obj	Best	Average obj	Best
32 instances of set 1	235.39	29	235.43	32
32 instances of set 2	185.77	13	221.59	32

The numerical results of model based heuristic and the GA on small-sized instances are presented in Table 8. As a comparison, we also report the best objective value out of two MINLP models obtained by LINGO solver for each instance in this table. The objective values obtained by LINGO, the model based heuristic and the proposed GA are presented in the "Objective" columns in the table. The corresponding computational times are presented in the "Time" columns

Table 8: Numerical results on small-sized instances

		Instance 1								Instance2							
		LINGO		Model based heuristic			The proposed GA			LINGO		Model based heuristic			The proposed GA		
Scenario	Capacity	Objective	Time (s)	Objective	Time (s)	Gap (%)	Objective	Time (s)	Gap (%)	Objective	Time (s)	Objective	Time (s)	Gap (%)	Objective	Time (s)	Gap (%)
Scenario 1	40	232.48	3600	**232.47**	3.08	0.00	232.13	10.26	0.15	207.29	3600	**207.12**	10.26	0.08	206.22	13.61	0.52
	50	234.86	3600	**234.86**	0.41	0.00	234.84	12.23	0.01	217.48	3600	**217.47**	10.02	0.00	217.43	13.31	0.02
	60	234.94	3600	**234.94**	0.12	0.00	234.91	13.58	0.01	222.44	3600	**222.44**	9.39	0.00	218.71	11.99	1.68
	70	234.94	3600	**234.94**	0.13	0.00	234.92	12.85	0.01	222.43	3600	**223.23**	9.61	-0.36	221.49	12.23	0.43
	80	234.94	3600	233.67	2.14	0.54	**234.92**	13.72	0.01	223.27	3600	**223.26**	7.27	0.00	222.14	10.38	0.51
	90	235.22	3600	233.93	1.75	0.55	**234.92**	13.86	0.13	225.41	3600	**224.99**	3.32	0.19	224.24	12.92	0.52
	100	235.39	3600	233.99	0.17	0.59	**235.35**	13.09	0.02	227.56	3600	**227.25**	1.95	0.14	227.11	11.24	0.20
	110	235.39	3600	234.53	0.02	0.36	**235.34**	12.99	0.02	229.41	3600	229.29	1.87	0.05	**229.33**	13.12	0.04
Scenario 2	40	232.38	3600	**231.08**	8.54	0.56	230.98	11.54	0.60	207.21	3600	**206.93**	3.63	0.14	206.91	13.24	0.15
	50	234.69	3600	233.91	0.87	0.33	**234.70**	11.08	0.00	217.28	3600	215.96	5.92	0.61	**216.77**	14.02	0.23
	60	234.81	3600	233.79	0.08	0.43	**234.76**	13.20	0.02	220.65	3600	220.51	6.20	0.07	**222.17**	15.89	-0.69
	70	234.81	3600	233.94	0.03	0.37	**234.76**	12.64	0.02	221.92	3600	222.28	3.37	-0.16	**222.96**	14.47	-0.47
	80	234.81	3600	232.70	5.85	0.90	**234.78**	13.58	0.01	222.99	3600	220.60	3.78	1.07	**222.23**	12.09	0.34
	90	235.06	3600	232.53	1.32	1.08	**234.75**	12.71	0.13	225.13	3600	221.81	3.47	1.48	**224.26**	14.67	0.39
	100	235.43	3600	232.60	0.55	1.20	**235.20**	12.55	0.10	227.35	3600	225.49	1.81	0.82	**226.96**	12.33	0.17
	110	235.45	3600	233.40	0.06	0.87	**235.42**	14.91	0.01	229.16	3600	224.86	2.11	1.87	**229.12**	11.57	0.01
Scenario 3	40	233.66	3600	**233.32**	2.72	0.14	232.01	14.45	0.70	205.68	3600	**205.57**	10.70	0.05	205.22	14.53	0.22
	50	236.31	3600	**236.19**	1.98	0.05	236.08	14.55	0.10	216.65	3600	**216.65**	8.78	0.00	216.57	13.51	0.04
	60	236.43	3600	236.21	1.34	0.09	**236.40**	15.49	0.01	221.46	3600	**221.46**	9.56	0.00	219.12	12.48	1.05
	70	236.43	3600	236.11	1.14	0.13	**236.40**	13.92	0.01	222.20	3600	**222.20**	8.38	0.00	221.20	13.94	0.45
	80	236.43	3600	235.41	4.86	0.43	**236.40**	14.44	0.01	220.33	3600	**222.59**	6.19	-1.03	222.30	12.24	-0.90
	90	236.65	3600	**236.57**	3.28	0.03	236.40	16.03	0.10	223.56	3600	**226.06**	2.09	-1.12	225.88	11.71	-1.04
	100	236.89	3600	**236.88**	0.97	0.00	236.81	13.57	0.03	227.30	3600	**229.20**	2.05	-0.83	229.16	11.96	-0.82
	110	236.94	3600	**236.91**	0.07	0.01	236.89	12.23	0.02	228.63	3600	231.20	6.85	-1.12	**231.25**	13.19	-1.15
Scenario 4	40	233.67	3600	231.45	2.24	0.95	**232.20**	11.90	0.63	206.40	3600	**207.67**	15.40	-0.62	206.67	12.69	-0.13
	50	236.30	3600	235.98	0.48	0.14	**236.24**	11.66	0.03	218.21	3600	**218.09**	7.47	0.05	217.79	11.79	0.19
	60	236.39	3600	236.05	0.14	0.14	**236.35**	13.34	0.01	223.32	3600	223.18	9.99	0.07	**223.27**	13.09	0.02
	70	236.39	3600	235.88	2.39	0.22	**236.35**	13.87	0.02	224.18	3600	**224.14**	9.04	0.01	220.17	13.40	1.79
	80	236.39	3600	233.79	1.95	1.10	**236.35**	13.23	0.02	224.22	3600	**224.07**	10.37	0.07	223.36	13.23	0.38
	90	236.39	3600	234.01	0.64	1.01	**236.36**	12.94	0.01	226.37	3600	226.20	8.31	0.07	**226.37**	11.45	0.00
	100	236.47	3600	233.53	0.02	1.24	**236.43**	12.53	0.02	226.98	3600	228.68	4.66	-0.75	**229.30**	13.35	-1.02
	110	236.47	3600	233.85	0.01	1.11	**236.44**	12.92	0.01	228.30	3600	230.28	4.54	-0.87	**231.42**	12.42	-1.37
Average		235.43	3600	234.36	1.54	0.46	235.21	13.18	0.09	221.59	3600	221.59	6.51	0.00	221.47	12.88	0.06

in the table. The "gap" columns in Table 7 are calculated as: gap $= \frac{\text{Obj\_L}-\text{Obj\_M}}{\text{Obj\_L}} \times 100\%$ and gap $= \frac{\text{Obj\_L}-\text{Obj\_G}}{\text{Obj\_L}} \times 100\%$ respectively, where Obj_L, Obj_M and Obj_G represent the objective values obtained by LINGO, the model based heuristic and the GA respecitvely. From Table 7, we can observe that the average gaps between the GA and LINGO are 0.09% on Instance set 1 and 0.06% on Instance set 2 respectively, which are very close to those between the model based heuristic and LINGO, with an average of 0.46% on Instance set 1 and 0.00% on Instance set 2 respectively. Specifically, the proposed GA outperforms the model based heuristic on 22 out of 32 scenario-capacity combinations for Instance set 1, while the model based heuristic yields better objective values on 19 out of 32 scenario-capacity combinations for Instance set 2, compared to the objective values obtained by the GA. Besides, the computational time of the model based heuristic ranges from 0.01s to 8.54s on Instance set 1 and from 1.81s to 15.40s on Instance set 2, with an average of 1.54s and 6.51s respectively. Whereas, the proposed GA spends more computational time to obtain near-optimal solutions on small-sized instances, which varies from 10.26s to 16.03s on Instance set 1 and from 10.38s to 15.89s on Instance set 2. Therefore, the model based heuristic are more time-efficient on relative small-sized instances.

6.4. Numerical experiment on large-sized instance

With the growth of instance sizes, the formulated MINLP models cannot be solved by LINGO efficiently, due to its NP-hard nature. Therefore, in this part, only the model based heuristic and the GA are evaluated by large-sized instances that are randomly generated based on the small-sized instance sets used above. We generate large-sized instances with three combinations of products and periods, i.e. (5 products, 12 periods), (10 products, 6 periods), and (10 products, 12 time periods). And (5P, 6T), (10P, 6T), (10P,12T) are used to denote these combinations for simplicity hereinafter. For each combination, 5 instance sets are randomly generated. Problem parameters for large-sized instance sets are randomly generated in the ranges listed in Table 9. In addition, each large-sized instance set is solved under demand scenario without seasonality and 7 different production capacities varying from to 50 to 200. In total, there are 105 large-sized instances. The data of all these large-sized instances is included in the supplementary material and available on the internet [1].

[1] https://drive.google.com/drive/folders/1QgERGcZLf0Q28PT4msCZiv9OvL6PctEm?usp=sharing

Table 9: Parameter range for large-sized instances

Parameter	Range	Parameter	Range
α_j	[400,20000]	c_{jt}	[1,3]
β_j	[1.5,5.5]	h_{jt}	[0.01,0.05]
v_j	[0.75,1.25]	a_{jt}	[3.5,10.5]

Table 10: Numerical results on (10 products, 6 time periods) instances

	Capacity	model based heuristic		The proposed GA		
Instance set		Objective	Time (s)	Objective	Time (s)	Imp (%)
(10P, 6T)_1	50	2720.49	4.32	3039.59	102.69	11.73
	75	3235.34	8.86	3632.74	135.00	12.28
	100	3626.84	3.83	4097.40	111.81	12.97
	125	3964.63	12.81	4506.36	118.58	13.66
	150	4240.12	11.40	4855.18	117.24	14.51
	175	4450.17	3.84	5148.52	125.72	15.69
	200	4845.32	11.16	5421.93	116.59	11.90
(10P, 6T)_2	50	2324.84	7.81	2594.99	110.49	11.62
	75	2753.60	7.69	3095.23	122.13	12.41
	100	3077.65	5.52	3445.86	123.01	11.96
	125	3333.74	8.00	3697.82	140.17	10.92
	150	3540.96	8.05	3916.46	135.35	10.60
	175	3710.76	9.28	4115.21	106.18	10.90
	200	3850.67	13.48	4267.76	108.22	10.83
(10P, 6T)_3	50	1760.59	8.43	1826.12	143.55	3.72
	75	2159.70	8.43	2216.02	144.49	2.61
	100	2481.52	5.43	2588.98	176.08	4.33
	125	2752.27	5.19	2871.71	157.06	4.34
	150	2986.05	7.30	3132.87	163.01	4.92
	175	3190.96	6.66	3398.39	264.50	6.50
	200	3385.06	9.03	3596.90	156.48	6.26
(10P, 6T)_4	50	2701.08	2.99	3114.50	123.05	15.31
	75	3166.19	2.78	3658.95	103.56	15.56
	100	3520.32	2.34	4059.37	125.94	15.31
	125	3807.49	2.77	4400.58	100.95	15.58
	150	4076.64	5.97	4685.10	137.48	14.93
	175	4344.73	9.12	4920.88	122.57	13.26
	200	4460.23	5.18	5105.53	89.63	14.47
(10P, 6T)_5	50	1694.03	6.63	1926.74	114.22	13.74
	75	2059.68	3.24	2390.28	129.40	16.05
	100	2358.57	6.76	2723.40	124.54	15.47
	125	2623.56	17.27	2975.63	147.49	13.42
	150	2886.86	16.59	3201.43	136.64	10.90
	175	3024.40	10.69	3468.38	127.31	14.68
	200	3208.23	14.18	3660.32	110.83	14.09
Average		3209.24	7.80	3593.06	130.63	11.64

Numerical results on instances with 10 products and 6 time periods are reported in Table 10. The first two columns represent instance set index and capacity, and the objective values and the corresponding computational times are reported from columns 3 to 6. The last column represents the improvement of the objective value obtained by the GA compared to that of the model based heuristic, which is calculated as Imp = $\frac{\text{Obj\_G - Obj\_M}}{\text{Obj\_M}} \times 100\%$, where Obj_G and Obj_M are the objective values obtained by the proposed GA and the model based heuristic respectively. Comparison of the third and fifth columns demonstrates that the GA provides better solutions than those of model based heuristic on all instances, with an average improvement of 11.64%. Comparing solution times recorded in Table 7 and Table 9, it is seen that the average computational time of the GA increases from less than 14s to 130.63s, with the growth of the number of products. On the contrary, the model based heuristic can solve these instances with 6 periods much faster with an average computational time of 7.80s, which is not impacted by the increase of the number of products greatly.

Table 11 reports numerical results on instances with 5 products and 12 periods. In comparison with the third and fifth columns, the objective values obtained by the GA are larger than those of the model based heuristic on most instances, with the exception of instances (5P, 12T)_4 under capacity 175 and 200. The average improvement 9.58% on the objective values also shows the superiority of the proposed GA in terms of solution quality. Comparing the fourth and sixth columns, the computation times of GA vary from 93.93s to 171.23s, with an average of 125.21s, which is less than that of the model based heuristic ranging from 121.58s to 652.07s, with an average of 276.26s. The prosed GA is more time-efficient on 34 out of 35 instances. Therefore, the proposed GA outperforms the model based heuristic in terms of both solution quality and time efficiency on these instances with 5 products and 12 periods.

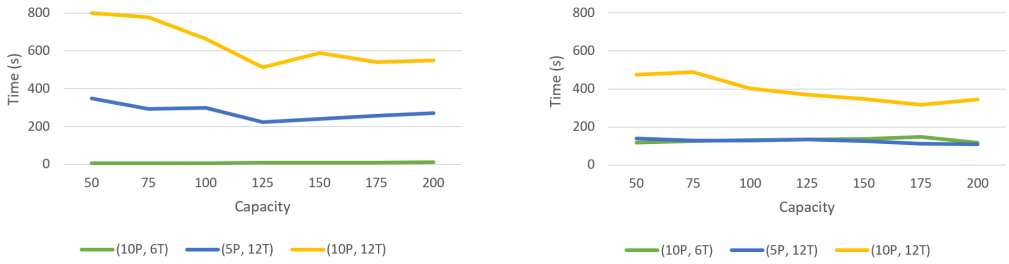
Table 12 reports numerical results on instances with 10 products and 12 periods. In comparison of the objective values in the third and fifth columns, it demonstrates that the GA yields better solutions compared with those obtained by the model based heuristic, with an average improvement of 11.45%. It is observed from the fourth and sixth that the proposed GA is also more time-efficient on 30 out of 35 instances, with an average computational time of 392.04s, compared to the model based heuristic with an average computational time of 633.29s. It is obvious that the proposed GA outperforms the model based heuristic on instances with 10 products and 12 periods in terms of both solution quality and time efficiency.

Table 11: Numerical results on (5 products, 12 time periods) instances

		model based heuristic		The proposed GA		
Instance set	Capacity	Objective	Time (s)	Objective	Time (s)	Imp (%)
(5P, 12T)_1	50	1880.67	471.51	2345.28	137.33	24.70
	75	2073.99	269.27	2789.25	144.04	34.49
	100	2326.58	230.75	3115.00	129.11	33.89
	125	2538.97	216.33	3366.11	137.59	32.58
	150	2790.08	173.65	3577.02	122.57	28.20
	175	2905.22	186.67	3754.01	112.43	29.22
	200	3290.04	208.43	3902.29	126.52	18.61
(5P, 12T)_2	50	1926.83	283.14	2081.86	148.82	8.05
	75	2261.09	226.71	2389.68	125.86	5.69
	100	2460.49	246.28	2611.56	146.98	6.14
	125	2678.97	214.74	2765.87	122.23	3.24
	150	2779.31	175.41	2878.77	113.78	3.58
	175	2898.14	160.39	2957.71	114.47	2.06
	200	2946.93	155.90	3016.57	97.00	2.36
(5P, 12T)_3	50	2028.98	221.70	2317.98	163.98	14.24
	75	2441.66	271.80	2644.78	143.38	8.32
	100	2636.86	217.76	2864.17	121.49	8.62
	125	2831.37	190.22	3029.73	114.69	7.01
	150	3001.18	190.40	3161.34	120.34	5.34
	175	3109.61	160.69	3259.33	105.30	4.81
	200	3191.56	160.44	3334.50	129.98	4.48
(5P, 12T)_4	50	499.07	121.58	524.97	107.26	5.19
	75	579.47	158.64	608.12	127.02	4.94
	100	640.55	175.47	655.04	149.27	2.26
	125	675.52	166.77	683.65	171.23	1.20
	150	697.84	225.68	698.39	142.23	0.08
	175	712.61	314.00	711.35	125.56	-0.18
	200	720.36	338.20	714.49	102.86	-0.82
(5P, 12T)_5	50	3157.96	652.07	3462.14	134.53	9.63
	75	3513.83	542.82	3805.65	96.84	8.30
	100	3702.71	621.28	3989.42	94.47	7.74
	125	3862.24	323.44	4075.98	126.69	5.53
	150	3984.29	439.76	4094.86	126.79	2.78
	175	4033.12	468.07	4104.69	105.78	1.77
	200	4056.56	489.11	4100.91	93.93	1.09
Average		2452.42	276.26	2696.93	125.21	9.58

Table 12: Numerical results on (10 products, 12 time periods) instances

	Capacity	The model based heuristic		The proposed GA		
Instance set		Objective	Time (s)	Objective	Time (s)	Imp (%)
(10P, 12T)_1	50	1913.22	622.26	2247.57	350.66	17.48
	75	2253.15	590.56	2665.57	375.90	18.30
	100	2510.54	586.74	2932.72	253.22	16.82
	125	2712.21	621.30	3152.96	215.73	16.25
	150	2875.29	819.36	3301.74	228.44	14.83
	175	3146.28	780.20	3437.43	229.50	9.25
	200	3258.73	552.69	3527.24	317.93	8.24
(10P, 12T)_2	50	2778.14	675.70	2956.04	551.71	6.40
	75	3227.34	564.64	3420.21	538.82	5.98
	100	3546.92	492.67	3795.92	519.09	7.02
	125	3785.34	418.60	4037.02	354.99	6.65
	150	3971.07	473.15	4233.36	340.06	6.61
	175	4109.12	428.98	4387.67	314.59	6.78
	200	4214.47	672.32	4505.49	377.44	6.91
(10P, 12T)_3	50	2394.04	786.97	2614.52	571.96	9.21
	75	2857.99	858.98	3224.33	481.97	12.82
	100	3209.13	620.56	3644.42	331.29	13.56
	125	3482.73	497.07	3983.43	284.80	14.38
	150	3705.84	562.86	4260.19	297.94	14.96
	175	3899.77	476.18	4451.17	206.06	14.14
	200	4051.26	559.62	4629.25	284.98	14.27
(10P, 12T)_4	50	3541.54	507.00	4021.52	485.73	13.55
	75	4138.89	566.45	4773.22	508.67	15.33
	100	4585.12	450.42	5270.32	392.31	14.94
	125	4941.61	243.07	5621.60	553.67	13.76
	150	5230.15	256.23	5923.35	479.74	13.25
	175	5474.69	306.72	6155.94	455.22	12.44
	200	5675.30	242.73	6342.69	396.72	11.76
(10P, 12T)_5	50	3571.46	1404.41	4075.14	411.06	14.10
	75	4136.11	1305.64	4602.33	538.74	11.27
	100	4566.63	1171.21	5017.99	520.42	9.88
	125	4863.58	786.09	5294.08	437.34	8.85
	150	5113.55	826.63	5513.15	393.01	7.81
	175	5339.29	720.36	5694.74	375.12	6.66
	200	5500.45	716.64	5839.58	346.46	6.17
Average		3845.17	633.29	4272.97	392.04	11.45



(a) Model based heuristic

(b) The GA

Figure 9: Average computational time under different capacity.

Figure 9 depicts the average computational time of the model based heuristic and the GA under different capacity for large-sized instances. As can be seen from Figure 9(a), the impact of the capacity on the model based heuristic can be neglected for instances with 10 products and 6 periods, whereas it takes more time for the model based heuristic to solve instances with 12 periods under smaller production capacity. We observe in Figure 9(b) that the GA spends more time on solving instances with 10 products and 12 periods under smaller capacity, while the capacity cannot greatly impact the computational efficiency of the GA on instances (5P, 12T) and (10P, 6T).

We can conclude from the numerical results that: (i) for small-sized instances, both the model based heuristic and the GA can provide high-quality solutions, while the model based heuristic is more time-efficient; (ii) as problem-size increases, we recommend the proposed GA as the solution method because of its efficiency.

7. Conclusion and future research

In this paper, we consider the joint optimization of the lot-sizing and pricing decisions in a manufacturing company selling multiple products over a finite planning horizon while considering isoelastic demand function, backlogging and limited production capacity. For the problem, a mixed integer nonlinear programming (MINLP) model is firstly formulated. Then, several optimality properties are provided and a tighter MINLP model is constructed. Due to the NP-hardness and non-linearity of the problem, a model based heuristic for small-sized instances and a genetic algorithm with new progressive repair strategy for large-sized instances are proposed. Numerical results on a variety of instances demonstrate efficiency of the tighter MINLP formulation and the

solution methods. Besides, the following managerial insights are drawn: (i) the formulated MINLP model can help manufacturer make the decision on whether applying backlogging strategy or not and when to backlog the demand; (ii) if demand in a period is satisfied by backlogging strategy, it is more profitable to set its price higher than that in the next period, and the larger the backlogging cost, the higher its price should be; (iii) if demand in a period is satisfied by inventory, it is more profitable to set its price higher than that in the previous period, and the larger the inventory holding cost, the higher its price should be.

Future research directions may include: (i) Some other factors that affect demand can be integrated into the demand function. For example, cross-price elasticity can be included for manufacturers selling substitute or complement products. Besides, the customer behaviors are also impacted by reference price, product age (freshness) and displayed stock volume, as given in Li and Teng (2018) and Feng et al. (2022), which can be incorporated in the multi-product JLSPB for certain categories; (ii) Stochastic factors can be considered in our future work by taking uncertainty of the demand and yield into account; (iii) Other real-world objectives can be investigated to optimize the studied JLSPB from different aspects, such as environmental goals and service levels, thus our research can be extended to a multi-objective problem.

References

Askarpoor, H.R., Davoudpour, H., 2013. An effective approximation algorithm for joint lot-sizing and pricing problem. The International Journal of Advanced Manufacturing Technology 65, 1429–1437.

Bajwa, N., Fontem, B., Sox, C.R., 2016a. Optimal product pricing and lot sizing decisions for multiple products with nonlinear demands. Journal of Management Analytics 3, 43–58.

Bajwa, N., Sox, C.R., Ishfaq, R., 2016b. Coordinating pricing and production decisions for multiple products. Omega 64, 86–101.

Chang, C.T., Ouyang, L.Y., Teng, J.T., Lai, K.K., Cárdenas-Barrón, L.E., 2019. Manufacturer's pricing and lot-sizing decisions for perishable goods under various payment terms by a discounted cash flow analysis. International Journal of Production Economics 218, 83–95.

Cheng, C.H., Madan, M.S., Gupta, Y., So, S., 2001. Solving the capacitated lot-sizing problem with backorder consideration. Journal of the Operational Research Society 52, 952–959.

Chu, C., Chu, F., Zhong, J., Yang, S., 2013. A polynomial algorithm for a lot-sizing problem with backlogging, outsourcing and limited inventory. Computers & Industrial Engineering 64, 200–210.

Couzon, P., Ouazene, Y., Yalaoui, F., 2020. Joint optimization of dynamic pricing and lot-sizing decisions with nonlinear demands: Theoretical and computational analysis. Computers & Operations Research 115, 104862.

Davis, L., 1991. Handbook of genetic algorithms. Van Nostrand Reinhold.

Dell, M., Iori, M., Novellani, S., Stützle, T., et al., 2016. A destroy and repair algorithm for the bike sharing rebalancing problem. Computers & Operations Research 71, 149–162.

Deng, S., Yano, C.A., 2006. Joint production and pricing decisions with setup costs and capacity constraints. Management Science 52, 741–756.

Díaz-Madroñero, M., Mula, J., Peidro, D., 2014. A review of discrete-time optimization models for tactical production planning. International Journal of Production Research 52, 5171–5205.

Feng, L., Wang, W.C., Teng, J.T., Cárdenas-Barrón, L.E., 2022. Pricing and lot-sizing decision for fresh goods when demand depends on unit price, displaying stocks and product age under generalized payments. European Journal of Operational Research 296, 940–952.

Franzin, A., Stützle, T., 2019. Revisiting simulated annealing: A component-based analysis. Computers & 0perations Research 104, 191–206.

Gilbert, S.M., 1999. Coordination of pricing and multi-period production for constant priced goods. European Journal of Operational Research 114, 330–337.

Gilbert, S.M., 2000. Coordination of pricing and multiple-period production across multiple constant priced goods. Management Science 46, 1602–1616.

González-Ramírez, R.G., Smith, N.R., Askin, R.G., 2011. A heuristic approach for a multi-product capacitated lot-sizing problem with pricing. International Journal of Production Research 49, 1173–1196.

Goren, H.G., Tunali, S., 2016. A comparative study of hybrid approaches for solving capacitated lot sizing problem with setup carryover and backordering. European Journal of Industrial Engineering 10, 683–702.

Haugen, K.K., Olstad, A., Pettersen, B.I., 2007. The profit maximizing capacitated lot-size (pclsp) problem. European Journal of Operational Research 176, 165–176.

Huang, J., Leng, M., Parlar, M., 2013. Demand functions in decision modeling: A comprehensive survey and research directions. Decision Sciences 44, 557–609.

Karimi, B., Ghomi, S.F., Wilson, J., 2003. The capacitated lot sizing problem: a review of models and algorithms. Omega 31, 365–378.

Küçükyavuz, S., Pochet, Y., 2009. Uncapacitated lot sizing with backlogging: the convex hull. Mathematical Programming 118, 151–175.

Li, R., Liu, Y., Teng, J.T., Tsao, Y.C., 2019. Optimal pricing, lot-sizing and backordering decisions when a seller demands an advance-cash-credit payment scheme. European Journal of Operational Research 278, 283–295.

Li, R., Teng, J.T., 2018. Pricing and lot-sizing decisions for perishable goods when demand depends on selling price, reference price, product freshness, and displayed stocks. European Journal of Operational Research 270, 1099–1108.

Liu, X., Tu, Y., Zhang, J., Watson, L., 2008. A genetic algorithm heuristic approach to general outsourcing capacitated production planning problems. International Journal of Production Research 46, 5059–5074.

López-Ibánez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43–58.

Lusa, A., Martinez-Costa, C., Mas-Machuca, M., 2012. An integral planning model that includes production, selling price, cash flow management and flexible capacity. International Journal of Production Research 50, 1568–1581.

Michalewicz, Z., 2013. Genetic algorithms+ data structures= evolution programs. Springer Science & Business Media.

Önal, M., Romeijn, H.E., 2010. Multi-item capacitated lot-sizing problems with setup times and pricing decisions. Naval Research Logistics 57, 172–187.

Pinto, B.Q., Ribeiro, C.C., Rosseti, I., Plastino, A., 2018. A biased random-key genetic algorithm for the maximum quasi-clique problem. European Journal of Operational Research 271, 849–865.

Rezaei, J., Davoodi, M., 2012. A joint pricing, lot-sizing, and supplier selection model. International Journal of Production Research 50, 4524–4542.

Slama, I., Ben-Ammar, O., Dolgui, A., Masmoudi, F., 2020. New mixed integer approach to solve a multi-level capacitated disassembly lot-sizing problem with defective items and backlogging. Journal of Manufacturing Systems 56, 50–57.

Thomas, J., 1970. Price-production decisions with deterministic demand. Management Science 16, 747–750.

Toledo, C.F.M., De Oliveira, R.R.R., França, P.M., 2013. A hybrid multi-population genetic algorithm applied to solve the multi-level capacitated lot sizing problem with backlogging. Computers & Operations Research 40, 910–919.

Wu, J., Teng, J.T., Chan, Y.L., 2018. Inventory policies for perishable products with expiration dates and advance-cash-credit payment schemes. International Journal of Systems Science: Operations & Logistics 5, 310–326.

Wu, T., Zhang, C., Liang, Z., Leung, S.C., 2013. A lagrangian relaxation-based method and models evaluation for multi-level lot sizing problems with backorders. Computers & Operations Research 40, 1852–1863.