Self-adaptive randomized constructive heuristics for the multi-item capacitated lot sizing problem

David Lai[a,*], Yijun Li[c], Emrah Demir[d], Nico Dellaert[b], Tom Van Woensel[b]

[a]*Southampton Business School, University of Southampton, Southampton, United Kingdom*
[b]*School of Industrial Engineering, Eindhoven University of Technology, 5600MB Eindhoven, The Netherlands*
[c]*School of Traffic and Transportation Engineering, Central South University, Changsha 410075, China*
[d]*PARC Institute of Manufacturing, Logistics and Inventory, Cardiff Business School, Cardiff University, Cardiff, United Kingdom*

Abstract

The Capacitated Lot-Sizing Problem (CLSP) and its variants are important and challenging optimization problems. Constructive heuristics are known to be the most intuitive and fastest methods for finding good feasible solutions for the CLSPs and therefore are often used as a subroutine in building more sophisticated exact or metaheuristic approaches. Classical constructive heuristics, such as period-by-period heuristics and lot elimination heuristics, are widely used by researchers. This paper introduces four perturbation strategies to the period-by-period and lot elimination heuristics to further improve the solution quality. We propose a new procedure to automatically adjust the parameters of the randomized period-by-period (RPP) heuristics. The procedure is proved to offer better solutions with reduced computation times by improving time-consuming parameter tuning phase. Combinations of the self-adaptive RPP heuristics with Tabu search and lot elimination heuristics are tested to be effective. Computational experiments provided high-quality solutions with a 0.88% average optimality gap on benchmark instances of 12 periods and 12 items, and an optimality gap within 1.2% for the instances with 24 periods and 24 items.

Keywords: Capacitated lot sizing problem, Constructive heuristics, Self-adaptive, Perturbation strategy, Period-by-period heuristic, Lot elimination heuristic

1. Introduction

The Lot-Sizing Problem answers two key issues in the supply chain under time-varying demands. These include: (*i*) how much inventory should be carried in each period; and (*ii*) when and how many items need to be produced or ordered. The earliest research on this topic starts with the uncapacitated single-item lot-sizing problem (Wagner & Whitin, 1958). These authors proposed a dynamic programming approach based on the assumption of unlimited available resources. Recent studies mainly focus on capacitated lot-sizing problems (CLSP) that incorporate resource capacity. The CLSP determines the production amount in each period within a planning horizon by meeting the known demands and minimizing the total setup and inventory holding costs. There can be additional constraints, such as capacity constraints limiting the total amount of resource usage in each period. The CLSP has a wide variety of applications in supply chain decision-making and logistics optimization de Kok & Graves; Belvaux & Wolsey (2001); Bruno et al. (2014); Brahimi et al. (2017) and is known to be an NP-hard problem. As

*Corresponding author
Email addresses: d.lai@soton.ac.uk (David Lai), yijun_li@csu.edu.cn (Yijun Li), DemirE@cardiff.ac.uk (Emrah Demir), n.p.dellaert@tue.nl (Nico Dellaert), t.v.woensel@tue.nl (Tom Van Woensel)

compared to the single item CLSP, the multi-item CLSP is considerably more difficult to solve and can be time-consuming to determine the optimal solution for large-sized instances. Thus, this paper focuses on heuristics approaches for the multi-item CLSP.

The main objective of this paper is to investigate the effectiveness of introducing perturbation strategies into the existing constructive heuristics. In each iteration of the period-by-period heuristic, lot extension choices are randomized by perturbation strategies. Perturbation strategies are useful for developing heuristics in the context of vehicle routing problems (VRPs), see e.g., Hart & Shogan (1987), Charon & Hudry (2001) and Renaud et al. (2002). The underlying idea is to change fixed sequences determined by the original greedy heuristics through introducing random variants or new functions. Hart & Shogan (1987) introduced several types of perturbation techniques which are then categorized as data perturbation, algorithm perturbation, and solution perturbation by Renaud et al. (2002). Later, Hart & Shogan (1987) extended the savings heuristic for the capacitated VRP using data randomization. In each iteration, a percentage-based rule or a cardinality-based rule was implemented to generate a solution. Charon & Hudry (2001) reviewed the applications of the noising methods on heuristics. Renaud et al. (2002) described and compared seven perturbation heuristics for the pickup and delivery traveling salesman problem (PDTSP). Since there are no existing constructive heuristics developed for the CLSP using perturbation strategies, we explore different ways of introducing perturbation strategies on existing constructive heuristics and compare their effectiveness.

This paper has the following contributions:

1. We develop three randomized period-by-period heuristics and two randomized lot elimination heuristics for the CLSP by introducing four perturbation strategies to the original constructive heuristics.
2. A self-adaptive method is used in the proposed heuristics. This avoids a time-consuming parameter tuning phase without deteriorating the solution quality. The solution quality of several instances has been identified to be better than the ones obtained with extensive parameter tuning.
3. The proposed overall heuristics can find better solutions than the recent results reported in Hein et al. (2018) and outperform the state-of-the-art algorithm in solution quality and time for 24 periods and 24 items benchmark instances.

The remainder of the paper is organized as follows. Section 2 reviews the related heuristics for the CLSP, mainly regarding the constructive heuristics. Section 3 presents the problem description and mathematical formulation. Section 4 proposes three randomized period-by-period heuristics. Then the self-adaptive randomized period-by-period heuristics are described in section 4.3. Section 5 puts forward two randomized lot elimination heuristics by introducing two perturbation strategies. In section 6, our tabu search method is presented. Section 7 presents the computational experiments and results. Lastly, final remarks and future research directions are discussed in section 8.

2. Literature review

This paper focuses on heuristic approaches for the multi-item CLSP. Interested readers can refer to Maes & Van Wassenhove (1988); Karimi et al. (2003); Jans & Degraeve (2007) and Brahimi et al. (2017)

for extensive literature review on lot-sizing problems. For exact methods, we refer to Barany et al. (1984) and Eppen & Martin (1987).

Constructive heuristics for the CLSP and its variants are first proposed in the 1970s and 1980s, see e.g., Dixon & Silver (1981); Dogramaci et al. (1981); Eisenhut (1975); Gu et al. (1987); Karni & Roll (1982); Kirca & Kökten (1994); Lambrecht & Vanderveken (1979); Maes & Van Wassenhove (1986); Selen & Heuts (1989); Van Nunen & Wessels (1978). These heuristics are still relevant today and are widely used as a subroutine in sophisticated methods.

From these early works, researchers developed different methods. These methods for CLSP can be categorized into *i*) period-by-period heuristics, *ii*) improvement heuristics, *iii*) mathematical programming-based heuristics, and finally *iv*) metaheuristics. We will review the four types of heuristics in the following sections.

2.1. Period-by-period heuristics

A period-by-period heuristic iteratively considers one period at a time, starting from the beginning to the end of the planning horizon. At each iteration, the following steps are performed: a ranking step, a lot-sizing step, a feasibility routine, and an improvement step. The ranking step determines the priorities for lot extension for all items in the current period. The lot sizing step revises the lot-sizing matrix according to priority indices. The feasibility routine guarantees that the final solution is feasible, thus future demands may be partially pre-produced in the current lot size particularly when the capacity constraints are tight.

Eisenhut (1975) first proposed a constructive heuristic for the CLSP. A priority index derived from the part-period balancing criterion is used to indicate the potential reduction in cost per period for transferring each future demand to the current lot size. The future demands are moved to the current period in descending order one by one. Lambrecht & Vanderveken (1979) extended Eisenhut's heuristic by adding a feedback mechanism and using a priority index based on the Silver-Meal criterion. Different from the feedback mechanism, a lookahead mechanism is also used to make sure that the current production plan can provide feasibility for future production schedules (Dixon & Silver, 1981; Gu et al., 1987; Maes & Van Wassenhove, 1986).

Selen & Heuts (1989) proposed a modified index for the priority index used in the third step developed by Gu et al. (1987). Maes & Van Wassenhove (1986) presented 72 heuristics (the A/B/C heuristic) by combining six ranking strategies, four priority indices, and three index search directions in the lot-sizing step. The variant performed best out of the 72 heuristics for the problems represented the final solution obtained by the A/B/C heuristic. Recently, Hein et al. (2018) investigated the constructive heuristics for the CLSP and used a Genetic Programming (GP) method to automatically generate some new priority indices based on multiple existing priority indices. The computational results highlighted that the indices obtained by the GP approach always returned lower cost than the Dixon & Silver and A/B/C heuristics. The decision-making of the lot-sizing for the first period is less affected by the demands of the distant future periods. It indicates that the period-by-period heuristics are reliable even though the future demands are updated constantly. Apart from obtaining a lot-sizing schedule period by period, Kirca & Kökten (1994) devised an item-by-item approach. This approach starts from an item-selection strategy, which determines the sequence of items selected in the candidate set.

2.2. Improvement heuristics

Improvement heuristics start with an initial solution disregarding capacity constraints. Then, eliminate infeasibility by shifting lot sizes at minimal increasing costs. The last step usually tries to further get cost savings by shifting lot sizes without violating feasibility (Van Nunen & Wessels, 1978).

As one of the first studies, Dogramaci et al. (1981) developed a four-step method that contains three sub-algorithms. Later, Karni & Roll (1982) disregarded the capacity constraints and applied the Wagner-Within algorithm to each product, obtaining an initial solution. If the initial solution was infeasible, shifted the production amount left to diminish the infeasibility under the requirement of minimal cost increase.

2.3. Mathematical programming-based heuristics

Mathematical programming-based heuristics construct a feasible solution by a linear-programming model iteratively. Many researches employed the Lagrangian relaxation to the capacity constraints to decompose the problem to several single-item uncapacitated problems.

To address the CLSP with setup times, Trigeiro et al. (1989) first applied the Lagrangian relaxations on the capacity constraints, then applied Wagner-Whitin dynamic programs for solving each uncapacitated single-item problem. This work is extended by Hindi et al. (2003), in which the Lagrangian-relaxation heuristic is followed by a variable neighborhood search (VNS) algorithm. Absi & Kedad-Sidhoum (2009); Absi et al. (2013) also applied the same method to CLSP with safety stock and demand shortages.

Relax-and-fix heuristic is one of the MIP-based heuristics. The general steps can be described as: *i*) integer variables are grouped by some strategies; *ii*) in each iteration, one sub-set of the integer variables is maintained integrality while others are relaxed; *iii*) after solving the resulting problem, a part of the integer variables is fixed. However, this heuristic does not guarantee a feasible solution for certain lot sizing problems, such as the multi-item CLSP with setup times (Absi & van den Heuvel (2019)). Interested readers are referred to Absi & van den Heuvel (2019); Toledo et al. (2015); Ferreira et al. (2010); Pedroso & Kubo (2005). Pedroso & Kubo (2005) embedded the relax-and-fix variant heuristic into a Tabu search framework, proposing a hybrid Tabu search heuristic to solve the multi-item multi-machine lot-sizing problem with backlogs.

The fix-and-optimize heuristic is an improvement heuristic, introduced by Sahling et al. (2009). The basic form starts from an initial solution, then uses several strategies to decompose the binary variables. Some criteria will be used to determine the optimized order of subsets and variables. The heuristic is also used to improve the solutions obtained from the relax-and-fix heuristic (Toledo et al. (2015)).

Cattrysse et al. (1990) used several heuristics (including the extended period-by-period A/B/C heuristic based on Maes & Van Wassenhove (1986)) to generate a set of feasible schedules for each item. Then, by solving the LP relaxation of the set partitioning problem formulation, some production schedules for each item were chosen. Subsequently, a column generation was utilized to get a possible fractional solution, followed by some heuristics (e.g., the extended A/B/C heuristic) to convert the fractional solution into an integer one. Cunha et al. (2019) investigated the multi-item CLSP with remanufacturing. **?** provided an extensive survey of different approaches for obtaining lower bounds for a multi-level capacitated lot-sizing problem, including the use of valid inequalities, reformulations, and Lagrangian relaxation. Both theoretical and computational comparisons are provided. **?** derived valid lower bounds on the partial objective function of a single-level multi-item capacitated lot-sizing problem. Effective envelope inequalities are introduced and compared to the (l, S) inequalities in the computational results. Furthermore, the authors perturb the partial objective function coefficients to identify violated inequalities within a cutting-planning algorithm. For exact methods such as the branch-and-price algorithm, readers can refer to Degraeve & Jans (2007).

2.4. Metaheuristics

Since metaheuristics are flexible in solving large and complex problems, they are also developed to solve the variants of the classic CLSP (Hindi, 1996); CLSP with setup carryover (Gopalakrishnan et al., 2001); CLSP with backlogging and setup carryover (Karimi et al., 2006); multi-level CLSP (Özdamar & Barbarosoglu, 2000); multi-resource CLSP with setup times (Hung et al., 2003); multi-machine CLSP with backlogs (Pedroso & Kubo, 2005); CLSP with setup times and without setup costs (Muller et al., 2012); CLSP for multiple plants (Nascimento et al., 2010)); CLSP with setup times, safety stock and demand shortages (Mehdizadeh & Fatehi Kivi, 2014); CLSP with returns and hybrid products (Koken et al., 2018).

Karimi et al. (2006) extended Tabu search heuristic of Hindi (1996) and applied it to the CLSP with backlogging and set-up carryover. A look-ahead mechanism (e.g., Dixon & Silver (1981); Gu et al. (1987); Maes & Van Wassenhove (1986); Dogramaci et al. (1981)) was also applied to ensure feasibility. Mehdizadeh & Fatehi Kivi (2014) proposed three metaheuristic algorithms, namely Simulated Annealing algorithm, Vibration-Damping Optimization algorithm, and Harmony Search algorithm to solve the multi-item CLSP with setup times, safety stock, and demand shortages. Koken et al. (2018) analyzed the CLSP with returns and hybrid products. The authors designed a simulated algorithm with a neighborhood list and compared it with three variants of GA and VNS algorithm.

3. Problem description

The multi-item CLSP aims at determining the production amount for various items during each period, under the constraint of production capacity and demand requirements. The objective is to minimize the sum of overall fixed costs and inventory holding costs.

Let $\mathcal{N} = \{1, 2, ..., N\}$ be the set of items, and $\mathcal{T} = \{1, 2, ..., T\}$ be the set of time periods. For all $i \in \mathcal{N}$ and $t \in \mathcal{T}$, let d_{it} denote the demand for item i in time period t. Shortage and backlog are not allowed. Each unit of item $i \in \mathcal{N}$ requires K_i unit of production time. The total production time for all items in time period t cannot exceed the available production time C_t. When a batch of item i is produced, a fixed setup cost S_i incurs. Let h_i denote the unit inventory holding cost for item i. Furthermore, for all $i \in \mathcal{N}$ and $t \in \mathcal{T}$, let $M_{it} = \min\left\{\frac{C_t}{K_i},\ \sum_{r=t}^{T}\ d_{ir}\right\}$

The decision variables are related to when and how much to produce for each item in each period. For all $i \in \mathcal{N}$ and $t \in \mathcal{T}$, let x_{it} be the lot size (i.e., production quantity) of item i in period t, and I_{it} be the ending inventory of item i in period t. Let y_{it} be a binary decision variable, with $y_{it} = 1$ if and only if item i is produced in period t.

The multi-item CLSP can be formulated as the following mixed-integer linear program (MILP).

$$
\begin{aligned}
(P1):\ \min & \sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}} (S_i y_{it} + h_i I_{it}) && && (1)\\
\text{s.t. } & I_{i,t-1} + x_{it} - I_{it} = d_{it}, && \forall i \in \mathcal{N}, t \in \mathcal{T}, && (2)\\
& \sum_{i\in\mathcal{N}} K_i x_{it} \le C_t, && \forall t \in \mathcal{T}, && (3)\\
& x_{it} \le M_{it} y_{it}, && \forall i \in \mathcal{N}, t \in \mathcal{T}, && (4)\\
& y_{it} \in \{0,1\}, && \forall i \in \mathcal{N}, t \in \mathcal{T}, && (5)\\
& x_{it}, I_{it} \ge 0, && \forall i \in \mathcal{N}, t \in \mathcal{T}. \qquad . && (6)
\end{aligned}
$$

The objective function (1) is to minimize the total setup cost and inventory holding cost. Constraints (2) are the inventory balancing equations. Constraints (3) guarantee that the capacity usage of each period does not exceed the available capacity. Constraints (4) ensure that the corresponding setup cost incurs in the objective function whenever items are produced.

4. Self-adaptive randomized period-by-period heuristics

Period-by-period heuristics for the CLSP are intuitive, easy to implement and require very low computation time (Maes & Van Wassenhove, 1988). Although introduced for more than four decades, some recent applications can be found in e.g., Lee et al. (2005); Tempelmeier & Herpers (2010); Absi et al. (2013) and Hein et al. (2018). A review on existing perturbation strategies can also be found in Hart & Shogan (1987) and Renaud et al. (2002).

4.1. Perturbation strategies

Since priority indices are crucial components in the design of period-by-period heuristics, we focus on introducing randomness into the heuristics by perturbing the priority indices, so that the sequence of lot extensions can be shuffled drastically for diversification purposes. Existing priority indices used in the period-by-period heuristics and their design principles can be found in Eisenhut (1975); Lambrecht & Vanderveken (1979); Dixon & Silver (1981); Gu et al. (1987); Maes & Van Wassenhove (1986); Hein et al. (2018). For completeness, we summarized the priority indices used in our experiments in Appendix A.

It is important to apply the proposed perturbation strategies at the right moment to diversify the search, and to escape from premature convergences without large deterioration on solution quality. To achieve this, a parameter $w \in [0, 1]$ that is referred as *perturbation degree* is introduced as a unified measure for all the proposed perturbation strategies. A larger value of w represents higher extent of perturbation being introduced using the perturbation strategy. Instead of applying perturbation strategies statically as an additional subroutine of the heuristic, we change the perturbation degree dynamically by using the proposed adaptive mechanism presented in section 4.3.

The proposed three perturbation strategies are detailed below.

Perturbation strategy 1 (`PS1`) The first strategy perturbs the heuristic by choosing randomly one of six existing priority indices, instead of using a single fixed one. For all six priority indices $k \in \{1, 2, \dots, 6\}$ and items $i \in \mathcal{N}$, let U_i^k denote the value for item i when priority index k is in use. In which, U_i^1 denotes the value obtained from using the index developed by Gu et al. (1987), and U_i^2 denotes the one from Dixon & Silver (1981). The indices U_i^3, U_i^4, and U_i^5 are obtained respectively from using the SM (Silver & Meal heuristic), LUC (Least-Unit Cost), and AC (Absolute Cost) indices summarized in Hein et al. (2018). Lastly, we denote with U_i^6 the HeinB formula introduced by Hein et al. (2018). The original definitions and design principles of theses indices are shown in the referred articles. For completeness, we summarize all formulae in Appendix A. Furthermore, let $r \in [0, 1]$ denote a number randomly picked between 0 and 1 for all the items, and w denote the perturbation degree. For all items $i \in \mathcal{N}$, the priority index for item i is given by:

$$u_i = \begin{cases} U_i^1, & \text{if } r \leq w/5 \\ U_i^2, & \text{if } w/5 < r \leq 2w/5 \\ U_i^3, & \text{if } 2w/5 < r \leq 3w/5 \\ U_i^4, & \text{if } 3w/5 < r \leq 4w/5 \\ U_i^5, & \text{if } 4w/5 < r \leq w \\ U_i^6, & \text{if } w < r \leq 1 \end{cases} \tag{7}$$

For example, when $w = 0.5$ and $r = 0.6$, the HeinB formula (as shown in Appendix A) is used for setting the priority index. Note that when the value of w is set to a larger number, there is a higher probability of choosing the other five indices.

Perturbation strategy 2 (`PS2`) The second strategy perturbs the heuristic by using only the HeinB formula for setting the priority index.

For all items $i \in \mathcal{N}$, let r_i be a number randomly chosen between $1 - w$ and $1 + w$ for item i where w is the perturbation factor, and let u_i' denote the priority index obtained from using the HeinB formula. The priority index u_i for item i is then given by $r_i u_i'$. When a large value of w is used, the variance for generating the random numbers become larger, which implies that there is a higher chance that the resulting priority index from using the second perturbation strategy deviates from the value obtained from using the original HeinB formula.

Perturbation strategy 3 (`PS3`) The third perturbation strategy perturbs the heuristic by introducing randomness into the instance parameters. This type of strategy is commonly known as *data perturbation* in heuristic design.

Based on our preliminary analysis, it is more effective to introduce randomness into the setup costs than on the inventory holding costs. We will therefore focus on changing slightly the values of the setup costs (which is denoted as S) instead of the other instance parameters. For all items $i \in \mathcal{N}$, let r_i be a number randomly picked between $1 - w$ and $1 + w$ for item i where w is the perturbation degree. The new setup cost S_i of item i is set to $r_i S_i'$ where S_i' is its original value. When a larger value of w is used, there is a higher extent of deviation of the setup costs from its original value, resulting in a higher level of perturbation.

4.2. Randomized period-by-period heuristic

The period-by-period heuristic framework described in Hein et al. (2018), where the priority index is obtained using Genetic Algorithm, can outperform many classical constructive heuristics. Since the results from our preliminary experiments also well align with this, we will evaluate the three proposed perturbation strategies (`PS1`, `PS2`, and `PS3`) on the same period-by-period heuristic framework used by Hein et al. (2018). In Hein et al. (2018), the heuristic is referred to as the variant B of the Dixon & Silver heuristic. We denote the three heuristics as $\texttt{RPP}_1$, $\texttt{RPP}_2$, and $\texttt{RPP}_3$ respectively for the three perturbation strategies. The randomized period-by-period heuristic is presented in Algorithm 2. The heuristic determines the lot size for each item in the current period, and progresses from the beginning period till the last period. Below detail the description on the steps performed in period k.

The surplus capacity in period k is denoted by s_k. As in line 3, the PS-3 strategy can be applied to the input data. For the initialization, set the initial lot size for each item equal to its demand. Then calculate the surplus capacity s_t for all periods $t \in \mathcal{T}$ (line 6). We use u_i and v_i to represent the priority indices of item i during the lot-sizing step and feasibility routine, respectively. Since the priority indices used in the ranking step and the lot-sizing step are the same in the original Dixon & Silver heuristic, the priority index simultaneously determines the sequence of lot extension for items and determines whether it is profitable to pre-produce future demands in the current lot.

In each period k, first the value of α should be calculated. α is the earliest period in which the inequality $\sum_{j=k+1}^{t} -s_j > 0$ is satisfied ($k+1 \leq t < T$). Otherwise, α is equal to $T+1$. The value of α restricts the extended period t_i for item i (lines 8-15). The extended period is the period in which the demand will be considered to be pre-produced in the current lot size. Then in period k, a set M needs to be determined. The set M is a subset of the set $\mathcal{N}$. The items in the set M meet the following conditions, the lot size x_{ik} in the current period k is a positive number, and from the period $k+1$ to the period α, there is a period t in which the lot size x_{it} is also a positive number (line 14). If the set M is empty and α is less than T, to ensure a final feasible solution, the feasibility routine is followed and the lot sizing step is skipped (line 12).

In the lot-sizing step, if the surplus capacity in period k is positive ($s_k > 0$) and the set M is not empty, first determine the extended period t_i for each item $i \in M$ and then calculate the corresponding priority index u_i. Here, either the PS-1 or the PS-2 strategy can be applied, see line 18. Next, the item $\bar{i}$ with the largest u_i as well as its extended period $\bar{t}$ are selected. If $u_{\bar{i}}$ (the priority index for $\bar{i}$) is non-negative and the surplus capacity in period k is no less than the capacity requirement ($K_{\bar{i}} x_{\bar{i}\bar{t}}$) for item $\bar{i}$ in period $\bar{t}$ ($s_k \geq K_{\bar{i}} x_{\bar{i}\bar{t}}$), then the production schedule and the surplus capacity should be updated (lines 22-23). If α is equal to $T+1$, there is no overloaded capacity that must be shifted to period k. Thereby after updating the production schedule, only the extended period $\bar{t}$ and index $u_{\bar{i}}$ for item $\bar{i}$ need to be updated by performing PS-1 or PS-2(line 25). However, if α is less than $T+1$, the overloaded capacity β should be shifted to period k (lines 27-33). But it is noticeable here that in the lot-sizing step, the future demands will not be partially satisfied in period k and only when $u_{\bar{i}}$ is non-negative and $K_{\bar{i}}x_{\bar{i}\bar{t}}$ does not exceed the surplus capacity s_k, the future lot $x_{\bar{i}\bar{t}}$ will be shifted to the current lot in period k. So, after lot-sizing step, the overloaded capacity β can also be positive, requiring a feasibility routine to guarantee the final feasible solution. The lot-sizing step terminates when $s_k \leq 0$ or the set M is empty.

The last step applied in each period k is the feasibility routine. After the lot-sizing step, if the value of α is less than $T+1$ and there exists surplus capacity in period k, then the overloaded capacity Q computed as in line 42 should be shifted to period k with the minimal cost increase. Note that in this step, the future lots can partially be transferred to the lot in period k. After determining the extended period for all items $i \in \mathcal{N}$ and calculate the index v_i by using the PS-1 or PS-2 strategy, the item i' with the largest v_i is selected. Its future lot $x_{i't'}$ is partially (line 46) or fully (line 48) transferred to the lot of period k for eliminating the overloaded capacity Q. This step terminates when the overloaded capacity Q is eliminated which guarantees the final feasible solution.

4.3. Self-adaptive mechanism

We extend the randomized period-by-period heuristics described in section 4.2 ($\texttt{RPP}_1$, $\texttt{RPP}_2$, and $\texttt{RPP}_3$) with a self-adaptive mechanism. The overall heuristics are referred to as the *self-adaptive randomized period-by-period heuristics*, and are respectively denoted by $\texttt{ARPP}_1$, $\texttt{ARPP}_2$, and $\texttt{ARPP}_3$ for the three perturbation strategies. The self-adaptive mechanism attempts to adjust the parameter configurations of the RPP heuristics, so that the heuristics can learn and adapt to features of each instance during the search.

The self-adaptive mechanism controls two parameters of the RPPs: the perturbation degree w, and the number of repetitions m. The perturbation degree w is a parameter between 0 and 1, where a larger value represents more perturbation being introduced to the heuristic. The number of repetitions m controls the number of times the RPP is being invoked. Both parameters aim at controlling the extent of diversification of heuristics, and are apparently interrelated with respected to the performance of the heuristic. For example, heuristics with a higher perturbation degree tend to require more number of repetitions to converge.

The proposed self-adaptive strategy iteratively optimizes these two parameters by using bisection search, and terminate the heuristic when the search converges. Algorithm 1 outlines the procedure we used for adjusting the perturbation degrees during the search.

Algorithm 1: Self-adaptive procedure

```
1 set $w_1 = 0$ and $w_2 = 100$
2 while $w_2 - w_1 \geq 0.01$ do
3     run Algorithm 2 with perturbation degrees $w_1$ and $w_2$, and let $z_1$ and $z_2$ denote the objective
      values respectively
4     if $z_1 < z_2$ then
5         $w_2 = \left\lfloor (w_1 + w_2)/2 \right\rfloor \big/ 100$
6     else
7         $w_1 = \left\lfloor (w_1 + w_2)/2 \right\rfloor \big/ 100$
8     end
9 end
```

Algorithm 2: The proposed randomized period-by-period (RPP) heuristic

```
1  Data:
2  $d_{it}, K_i, h_i, C_t, TBO_i$;
3  perform PS-3 as described in section 4.1
4  Initialization:
5  set $x_{it} = d_{it}$ for all $i \in \mathcal{N}$ and $t \in \mathcal{T}$;
6  determine the surplus capacity for all periods: $s_t = C_t - \sum_{i \in \mathcal{N}} K_i x_{it}, \;\; \forall t \in \{1, 2, ..., T\}$
7  for $k = 1$ to $T - 1$ do
8      if $\min\{k+1 \le t \le T : \sum_{j=k+1}^{t} -s_j > 0\} \neq \emptyset$ then
9          $\alpha = t, \beta = \sum_{j=k+1}^{t} -s_j$,
           $M = \{i \in \mathcal{N} : x_{ik} > 0$ and there exists $t' \in [k+1, \alpha]$ such that $x_{it'} > 0\}$
10         if $M = \emptyset$ then
11             go to the Feasibility Routine
12         end
13     else
14         $\alpha = T+1, \beta = 0$, $M = \{i \in \mathcal{N} : x_{ik} > 0$ and there exists $t' \in [k+1, T]$ such that $x_{it'} > 0$ $\}$
15     end
16     -Lot Sizing Step:
17     while $s_k > 0$ and $M \neq \emptyset$ do
18         perform PS-1 or PS-2 and calculate the priority index $u_i$ for all $i \in M$ as described in section 4.1
19         select the item $\bar{i}$ in $M$ with the highest rank: $\bar{i} = \arg\max_{i \in M} u_i$ and its extended period $\bar{t}$;
20         if $u_{\bar{i}} \ge 0$ and $s_k \ge K_{\bar{i}} x_{\bar{i}\bar{t}}$ then
21             $P_{\bar{i}} = K_{\bar{i}} x_{\bar{i}\bar{t}}$
22             $x_{\bar{i}k} = x_{\bar{i}k} + x_{\bar{i}\bar{t}}$ and $x_{\bar{i}\bar{t}} = 0$
23             $s_k = s_k - K_{\bar{i}} x_{\bar{i}\bar{t}}; \quad s_{\bar{t}} = s_{\bar{t}} + K_{\bar{i}} x_{\bar{i}\bar{t}}$
24             if $\alpha = T+1$ then
25                 update $\bar{t}$, and then update $u_{\bar{i}}$ only for item $\bar{i}$ by performing PS-1 or PS-2, refer to section 4.1
26             else
27                 if $P_{\bar{i}} \ge \beta$ then
28                     update $\alpha$ and $\beta$ (refer to lines 8 - 12) and update set $M$
29                     perform PS-1 or PS-2 and calculate the index $u_i$ for all items $i \in M$, refer to section 4.1
30                 else
31                     $\beta = \beta - P_{\bar{i}}$
32                     update $\bar{t}$, and then update $u_{\bar{i}}$ only for item $\bar{i}$ by performing PS-1 or PS-2, refer to section 4.1
33                 end
34             end
35         end
36         if $u_{\bar{i}} < 0$ or $s_k < K_{\bar{i}} x_{\bar{i}\bar{t}}$ then
37             remove $\bar{i}$ from $M$
38         end
39     end
40     -Feasibility Routine:
41     if $\alpha \le T$ and $s_k > 0$ then
42         $Q = \max_{t=\alpha,\alpha+1,...,T} \left\{ \sum_{j=k+1}^{t} -s_j \right\}$
43         while $s_k \ge Q > 0$ do
44             perform PS-1 or PS-2 and calculate the index $v_i$ for all items $i \in \mathcal{N}$, refer to section 4.1
45             select item $i' = \arg\max_{i \in \mathcal{N}} v_i$ and its extended period $t'$ if $x_{i't'} > Q/K_{i'}$ then
46                 $x_{i'k} = x_{i'k} + Q/K_{i'}; \quad x_{i't'} = x_{i't'} - Q/K_{i'}; \quad Q = 0$
47             else
48                 $x_{i'k} = x_{i'k} + x_{i't'}; \quad x_{i't'} = x_{i't'} - x_{i't'}; \quad Q = Q - K_i x_{i't'}$
49                 update $t'$, and then update $v'$ for item $i'$ by performing PS-1 or PS-2, refer to section 4.1
                   $s_k = s_k - x_{i't'}; \quad s_{t'} = s_{t'} + x_{i't'}$
50             end
51         end
52     end
53 end
```

5. Randomized lot-elimination heuristic

We now introduce perturbation strategies into the lot elimination heuristic of CLSPs. The lot elimination procedure starts with an initial solution and then attempts to improve the solution by eliminating production lots (i.e., fixing the values of some of the setup decision variables y to zero in P1).

Lot elimination is a basic greedy procedure that is commonly used as an additional step for improving the initial solution obtained from using a constructive heuristic, see e.g. Dixon & Silver (1981); Günther (1988); Cattrysse et al. (1990), Fragkos et al. (2016) and Degraeve & Jans (2007). In addition, lot elimination heuristics are also used in the exact approaches developed for the CLSP, e.g., Fragkos et al. (2016) and Degraeve & Jans (2007).

5.1. Evaluation function

A solution $(\bar{x}, \bar{y})$ of (P1) is represented by the active production lots, denoted as $Y = \{(i,t) \in \mathcal{N} \times \mathcal{T} : \bar{y}_{it} = 1\}$, and its complement set $\bar{Y} = \mathcal{N} \times \mathcal{T} \setminus Y$.

Since the setup decision variables are fixed, the setup cost is a constant and given by $\sum_{(i,t)\in Y} S_{it}$. To determine the production quantity $\bar{x}$ and the corresponding production cost, we solve the linear-programming model (8) – (11). We note that whenever the solution is updated in the heuristic, only the objective cost coefficients need to be modified. We can, therefore, speed up the computation of the evaluation function by using the network simplex algorithm with warm-start. This implementation matters since the evaluation function is invoked frequently in the heuristic.

$$c(Y) = \sum_{(i,t)\in Y} S_{it} + \min \sum_{(i,t)\in \bar{Y}} \eta x_{it} + \sum_{(i,t)\in Y} h_i I_{it}, \tag{8}$$

$$\text{s.t. } I_{i,t-1} + x_{it} - I_{it} = d_{it}, \qquad \forall i \in \mathcal{N}, t \in \mathcal{T}, \tag{9}$$

$$\sum_{i\in\mathcal{N}} K_i x_{it} \leq C_t, \qquad \forall t \in \mathcal{T}, \tag{10}$$

$$x_{it}, I_{it} \geq 0, \qquad \forall i \in \mathcal{N}, t \in \mathcal{T}. \tag{11}$$

The objective function (8) minimizes the total inventory holding cost, and the penalty for using an inactive production lot where η is a sufficiently large positive number. Constraints (9) are the inventory balancing equations. Constraints (3) guarantee that the usage of each period does not exceed the available capacity.

5.2. Standard lot-elimination procedure

The heuristic begins by initializing all the production lots to be active i.e. $Y = \mathcal{N} \times \mathcal{T}$ and $\bar{Y} = \emptyset$. Then it progressively eliminates an active production lot $(i,t) \in Y$ that can lead to any cost savings, one by one following the descending order of setup cost S_{it}. Although the algorithm runs in a basic fashion, there are different variants used in the literature. We follow the procedure described more precisely as follows.

Step 1: set $Y = \mathcal{N} \times \mathcal{T}$

Step 2: sort all the production lots in an order list $\mathcal{L}$ with descending setup cost, where for any distinct items $(i,t), (i't') \in \mathcal{L}$, we have $(i,t) \prec (i't')$ iff $S_{it} \geq S_{i't'}$.

Step 3: select the next unprocessed item $(\bar{i}, \bar{t}) \in \mathcal{L}$ that has the largest setup cost.

Step 4: if $c(Y \setminus \{(\bar{i}, \bar{t})\}) < c(Y)$ then set $Y = Y \setminus \{(\bar{i}, \bar{t})\}$

Step 5: mark $(\bar{i}, \bar{t})$ as processed and go to step 3, until all the items in $\mathcal{L}$ are processed.

5.3. Randomized lot elimination procedure

We extend the lot elimination procedure described in section 5.2 by introducing two perturbation strategies. The resulting heuristics are referred as the randomized lot elimination (RLE) heuristics.

In the standard lot elimination heuristic, the production lots are sorted by the descending order of the items' setup costs. To develop a randomized version, we perturb the heuristic by using the following two strategies:

PS3 using the PS3 strategy to introduce randomness on the setup costs;

PS4 introduce randomness on the lot-elimination decisions by revising Step 4 in section 4.1 as: if $c(Y \setminus \{(\bar{i}, \bar{t})\}) < c(Y)$ and $r_i > w$ then set $Y = Y \setminus \{(\bar{i}, \bar{t})\}$, where r_i is a number randomly picked in [0,1], and w is the perturbation factor.

6. LP-based Tabu search heuristic

After obtaining an initial solution, we further improve the solution quality with a Tabu search metaheuristic. Given a current solution, the Tabu search metaheuristic aims to guide a local search process to explore the neighborhood space and escape from a local optimum. The development of Tabu search can date back to the 1960s, and it is proposed as a general heuristic by Glover (1989, 1990). Since then it has been widely used for solving a large variety of optimization problems. We will present Tabu search metaheuristic developed for the CLSP based on the approach of Hindi (1996). Algorithm 3 outlines Tabu search metaheuristic used in our experiments. We denote this as TS. The major components used in the algorithm are described below.

Neighborhood structure We represent a solution $(\bar{x}, \bar{y})$ of $(P1)$ by its corresponding active production lots $Y = \{(i,t) \in \mathcal{N} \times \mathcal{T} : \bar{x}_{it} > 0\}$ and its complement set $\bar{Y} = \mathcal{N} \times \mathcal{T} \setminus Y$. A move operation is to relocate an item from Y to $\bar{Y}$, or from $\bar{Y}$ to Y. This represents opening a currently closed production lot, or closing a currently active production lot. Let $\mathcal{X}$ denote the search space. For any solution $x \in \mathcal{X}$, let $N(x)$ denote the solutions in the neighborhood of x and are defined as the solutions after applying a single move operation on the current solution x. There are at most $|\mathcal{N}| \times |\mathcal{T}|$ solutions in the neighborhood.

Move evaluation Each move operation is evaluated by solving the LP model (8) – (11) as described in section 5.1. At each iteration, we evaluate all possible moves that are applicable to the current solution excluding the ones that are in the Tabu list. The best non-Tabu neighborhood solution that satisfies the aspiration criterion is then selected.

Aspiration criterion Aspiration by objective is applied. When a neighborhood solution is feasible and has a lower total cost than the best solution found by the heuristic so far, this neighborhood solution will be accepted and used to update the current solution regardless of whether its status is Tabu or not.

Tabu list to prevent cycling, the move operation that is accepted at each iteration is declared as Tabu in the next θ iteration. If a move operation related to the production lot, $(\bar{i}, \bar{t})$ is applied at iteration k. Then, the same operation and it's reverse operation can only be applied again after the $k + \theta$ iteration. The parameter θ is known as the length of the Tabu list, and we set $\theta = 3T/5$ where T is the number of time periods.

Stopping criterion The TS metaheuristic terminates when the best feasible solution obtained has not been improved after a given number of iterations.

Main procedure Algorithm 3 outlines the TS metaheuristic used in our experiments.

Algorithm 3: Tabu search metaheuristic

```
 Data: initial solution $x$, initial total cost $c(x)$, the length of tabu list $L_{min}$ and stopping criteria;
 1  Input: the current solution $x$ and its total cost $c(x)$;
 2  Output: $x^*$, $c^*$;
 3  Set $x^* = x, c^* = c(x), x_{now} = x, c_{now} = c(x)$;
 4  while neither of the stopping criteria are satisfied do
 5      get all possible move operations for the current solution $x$;
 6      for each move operation do
 7          solve the model P2 and obtain the solution $x'$, forming the neighborhood $N(x)$;
 8          calculate its total cost $c(x')$;
 9      end
10      select $x'' \in N(x)$ that leads to the minimal total cost $c(x'')$;
11      if $c(x'') < c^*$ then
12          set $x^* = x''$,$x_{now} = x''$, $c^* = c(x'')$,$c_{now} = c(x'')$;
13      end
14      if $c(x'') \geq c^*$ then
15          if the corresponding move operation for $x''$ is not in the tabu list then
16              set $x_{now} = x''$, $c_{now} = c(x'')$;
17          else
18              accept a $x' \in N(x)$ that leads to the minimal total cost among all the other solutions
                  for which the corresponding move operations are not in the tabu list ;
19              set $x_{now} = x'$, $c_{now} = c(x')$;
20          end
21      end
22      set the accepted move operation as tabu
23  end
```

7. Computational results

We describe the proposed solution approaches, the datasets and computer environment in Section 7.1, and the parameter tuning experiments in Sections 7.2 and 7.3. In section 7.4, we evaluate the performance of the `RPP` and `RLE` heuristics, and benchmark the heuristics with nine existing construction heuristics developed for the CLSPs. The effectiveness of the proposed perturbation strategies is also examined. Section 7.5 explores the effectiveness of the `ARPP` and `RPP` heuristics when solving instances with different available capacities and setup costs. To verify the effectiveness of the heuristics, the results obtained by the combined heuristics will be compared with the results from Hein et al. (2018). In Section 7.6, we further analyse the best-performing heuristic $\texttt{ARPP}_3$ when it is used in combination with the tabu search and lot elimination heuristics, respectively. In Section 7.7, we compare the performance of $\texttt{ARPP}_3$, $\texttt{ARPP}_3\texttt{-LE}$ to CPLEX on very large-sized instances with up to 96 periods and 192 items.

7.1. Experimental design

Datasets

The heuristics are tested on four datasets: i) 360 instances with 12 products and 12 periods (the 12*12 instances), ii) 360 instances with 24 products and 24 periods (the 24*24 instances); iii) three instances with 96 products and 48 periods (the 96*48 instances) ; and iv) six instances with 192 items and 96 periods (the 192*96 instances). All the instances are obtained using the generation procedure described in Hein et al. (2018) and Maes & Van Wassenhove (1986), including the same control parameter values. Each dataset consists of 72 classes of instances with various characteristics which are detailed below.

Table 1 shows the control parameters of the five factors: standard deviation of demand, capacity absorption, the average time between orders, tightness of capacity (i.e., total production capacity divided by total production resource requirement), demand type (normal or lumpy). For the normal demand pattern, average demand is fixed at 100 units per period, three types of demand standard deviation (low, medium, high) are applied to generate demands.

For the lumpy demand pattern, 25% of the periods have zero demand (i.e., 12 periods have three periods with no demand), other periods have demands with normal distribution, but the average demand is 125 units. All demands are assumed to be positive. There are two types of capacity absorption for items, one is to set the production capacity usage for all items equal to one, another is to generate capacity absorption for each item from a uniform distribution. Tightness of capacity is 1.11, 1.25, and 2 times of total capacity absorption. Then, total available capacity is spread evenly over all periods.

The time between orders (TBO) for each item is also generated from high and low uniform distributions. Holding cost is always 1 for each item in every period. As $\mathrm{TBO}_i = \sqrt{\frac{2S_i}{h_i d_i}}$, the set-up cost for each item can be computed. We round up the available capacity in each period, the resource usage for each item, the set-up cost, and the demand for each item in each period to an integer value.

We limit the experiments to nine very large instances (the 96*48 and 192*96 instances), since both the heuristics and CPLEX take too much computation time to find a satisfying solution. Table 7 (in the Appendix) summarizes the characteristics of these instances.

As we notice, for tight capacity problems, the instances are easily infeasible initially. For instance, one of the situations is that the available capacity in the first period C_1 may be less than the total capacity requirement in period 1 ($C_1 < \sum_{i \in \mathcal{N}} K_i x_{i1}$). Since we assume that there is no initial inventory for each item, this situation mentioned above does not have any feasible solutions. In the numerical evaluation of CLSP constructive heuristics, Günther (1988) also guaranteed that all the instances used for testing must have at least one feasible solution. Thus, we add a feasibility ensuring procedure during the instance generation. Once an instance is proven infeasible, it is disregarded and a new instance is generated.

Table 1: Control parameters used for instance generation

	Parameter	Value
1	Std. deviation demand	(a)High: uniform [0,50]; (b)medium: uniform [0,25];(c)low: uniform [0,10]
2	Capacity absorption	(d)Constant: 1; (e)random: uniform [1,5]
3	Average TBO	(f)High: uniform [1,6]; (g)low: uniform [1,2]
4	Tightness of capacity	(h)High: 111%; (i)medium: 125%; (j)low: 200%
5	Demand lumpiness	(k)Normal; (l)lumpy

Solution approaches

We summarize the proposed solution approaches evaluated in our experiments. We also implemented an existing constructive heuristics as a benchmark which are used in the experimental settings.

$\texttt{RLE}_x$: construct a solution using a randomized lot elimination (RLE) heuristic. Starts with an initial solution obtained by solving $(P2)$ where all production lots are opened, and then improve the solution by using the randomized lot elimination heuristics with a perturbation strategy. Heuristics $\texttt{RLE}_1$ and $\texttt{RLE}_2$ refer to the RLE with strategies PS3 and PS4 respectively. As a baseline approach, heuristic $\texttt{RLE}_r$ refers to RLE heuristic with a randomly generated priority index.

$\texttt{ARPP}_x$**:** construct a solution using the adaptive randomized period-by-period (ARPP) heuristic. Heuristics $\texttt{ARPP}_1$, $\texttt{ARPP}_2$ and $\texttt{ARPP}_3$ refer to the ARPP with strategies PS1, PS2 and PS3 respectively.

$\texttt{ARPP}_3$**-LE:** construct an initial solution obtained by $\texttt{ARPP}_3$ first, and then improve the solution using the standard lot elimination heuristic.

$\texttt{ARPP}_3$**-TS:** construct an initial solution obtained by $\texttt{ARPP}_3$ first, and then improve the solution using the tabu search algorithm described in section 6.

Computational environment

All tests are done on a computer with an Intel Core i5-4200M processor with 2.5GHz and 4GB of main memory. All algorithms are coded in Python and implemented in Spyder 3.3.6. The MILP models are solved using CPLEX Optimization Studio 12.9 with the default settings. The average MIP gap of all instances for each problem size solved in CPLEX is also reported.

We measure the solution quality of the heuristics based on the optimality gaps given by:

$$\left(\frac{C_{heuristic} - C_{cplex}}{C_{cplex}}\right) \times 100\% \tag{12}$$

where C_{cplex} is the objective function value obtained by `CPLEX`, and $C_{heuristic}$ is the objective function value obtained using the heuristic. If `CPLEX` cannot find the proven optimal solution within two hours, the best lower bound is used.

7.2. Parameter tuning of the `RPP` heuristics

The `RPP` heuristics are controlled by two parameters: the number of repetitions m and the perturbation factor w. In this experiment, we examine the impact of these two parameters on the solution quality, and accordingly set the initial parameter values of the `RPP` heuristics. We vary parameter m to values 5, 20, 100, 200 and 500, and parameter w to values from 5% to 90% with steps of 5%. We report the average gaps (measured by 12) from using the three `RPP` heuristics. The 360 instances with 12 periods and 12 items are used in this experiment. The results are shown in figure 1. The results are detailed in table 10 in Appendix B.

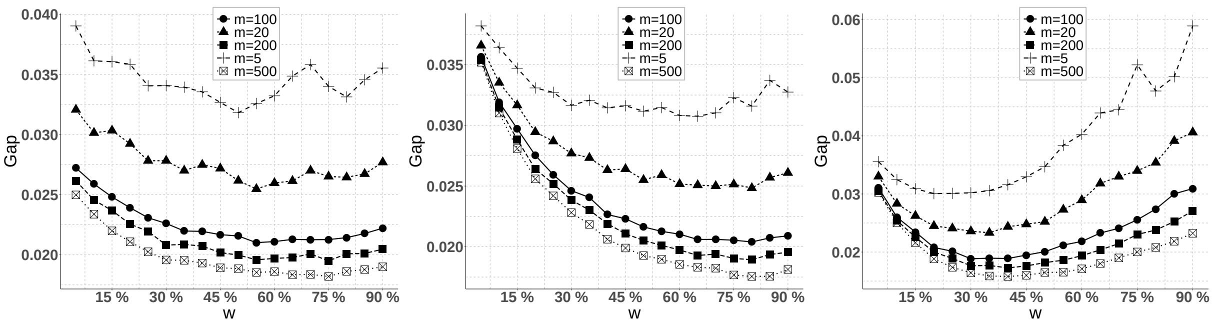


Figure 1: Average gaps from using $\texttt{RPP}_1$ (left), $\texttt{RPP}_2$ (middle) and $\texttt{RPP}_3$ (right) with various parameter values

As shown in figure 1, the average gaps decreases with more repetitions (i.e., larger value of m). Furthermore, with a fixed value of m, the average gaps of all three `RPP` heuristics exhibit an approximately continuous and convex trend. This trend becomes more obvious when m has a larger value. This observation reveals that it is possible to identify the best perturbation factor w for each specific value of m in order to achieve the best performance of the `RPP` heuristic. This result also inspired the use of a bisection search procedure as a learning mechanism for the `RPP` heuristics. The perturbation factor w can be viewed as being optimized by a unconstrained continuous optimization procedure.

7.3. Parameter tuning of the `RLE` heuristics

Similar to the `RPP` heuristics, we conduct a parameter tuning experiment for the `RLE` heuristics. We vary m to values 5 or 20, and vary w from 5% to 90% in steps of 5%. A less extensive experiment is performed since the `RLE` heuristics require more computation time than the `RPP` heuristics.

Fig. 2 shows the average gaps of $\texttt{RLE}_1$ (left) and $\texttt{RLE}_2$ (right). Concerning $\texttt{RLE}_1$, the average gaps fluctuate when w increases from 5% to 90%. For $\texttt{RLE}_2$, the average gaps initially exhibit a slight fluctuation when w changes from 5% to 55%; and when w exceeds 60%, the gap grows exponentially. This reveals that: *i*) it is not practical to select a common perturbation factor w for $\texttt{RLE}_1$ to reach the near-optimal average solution quality for all instances; and *ii*) it is ineffective to introduce more perturbations in $\texttt{RLE}_2$. Thus, we do not embed the similar self-adaptive procedure into the `RLE` heuristics.

7.4. Effectiveness of the `RPP` and `RLE` heuristics

In this section, we evaluate the effectiveness of the `RPP` and `RLE` heuristics. The heuristics are compared with `CPLEX`, and the following nine other existing constructive heuristics:

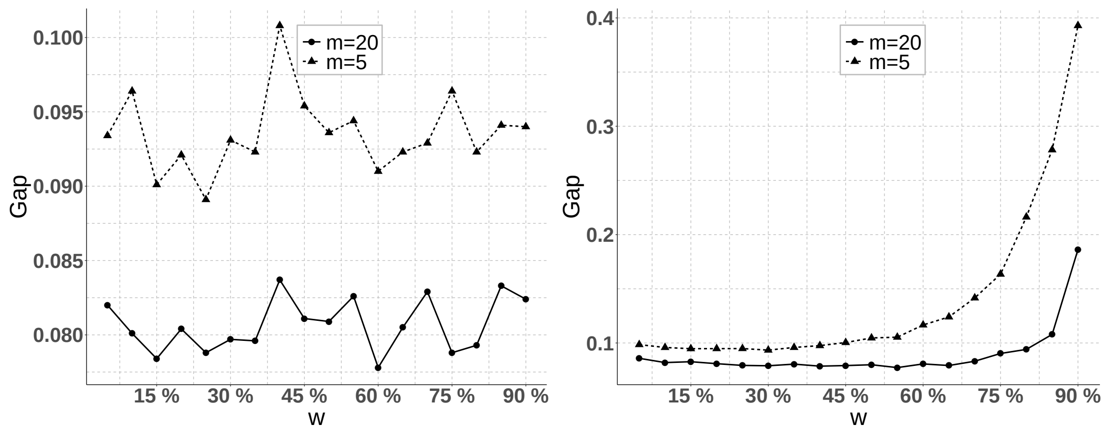


Figure 2: Average gaps of 12*12 instances of $\texttt{RLE}_1$ and $\texttt{RLE}_2$ when fixing m and varying w

- `Gunther`: The period-by-period heuristic for CLSP proposed in Gu et al. (1987).
- `DS`: The period-by-period heuristic for CLSP proposed in Dixon & Silver (1981).
- `HeinA1, HeinA2_LUC, HeinA2_SM, HeinA3_LTC, HeinA3_AC`: According to the description of Hein et al. (2018), we replace the priority indices used in the ranking step and feasibility routine by the best rules found in experiment A and keep the original lot-sizing index used in the Dixon & Silver (1981) heuristic unchanged to get the heuristic HeinA1. The only difference among HeinA1 and HeinA2_LUC, HeinA2_SM, HeinA3_LTC, HeinA3_AC is that the priority index used in the lot-sizing step in HeinA1 is the index used in Dixon & Silver heuristic. Whereas, the index used in the lot-sizing step is replaced by rule LUC, rule SM, rule LTC and rule AC in HeinA2_LUC, HeinA2_SM, HeinA3_LTC, HeinA3_AC respectively.
- `HeinB`: Based on the Dixon & Silver's heuristic and replace the priority indices used in the three steps by the best indices found in experiment B in Hein et al. (2018), we get the heuristic HeinB.
- `SLE`: The standard lot elimination heuristic for CLSP described in section 5, which can refer to Fragkos et al. (2016); Degraeve & Jans (2007).

All heuristics start with the same lot-for-lot initial solution. To restrict the computation time, we set the repetitions m to two for all `RPP` heuristics, and the value of w set according to the parameter tuning experiment result (the best value when $m = 5$ in table 10 in Appendix B). For the two `RLE` heuristics, m is set to 20. Table 2 shows the average gaps and computation time for all algorithms.

In terms of the existing constructive heuristics, results show that based on the same algorithm structure, only using different priority indices will have a significant impact on the algorithm performance. Among the eight period-by-period heuristics, `HeinB` performs best. When perturbation strategies are embedded, all the three `RPP` heuristics can find better solutions on the instances with 12 periods and 12 items within similar CPU time. For the instances with 24 periods and 24 items, only $\texttt{RPP}_2$ and $\texttt{RPP}_3$ generate lower average gaps then `HeinB`. Results of `RPP-random` indicate that if the lot extension order for items is randomly determined instead of using priority indices to guide the search direction in `RPP` heuristics, the solution quality will become worse than that of `HeinB`, and it is difficult to approach the optimal solution (more detailed results are presented in table 8 in Appendix). This result also demonstrates that the perturbation strategies are effective when they are embedded into the period-by-period heuristics.

Lot elimination heuristic typically has the worst performance. Even if embedding the perturbation strategies and consuming more computation times, the average quality of the solutions found by RLE is not as good as that of RPP heuristics. The results also reveal that although CPLEX can solve most instances to optimality, it requires significantly more computation time on the large-sized instances. This highlights the advantages of using period-by-period heuristics for solving the CLSPs.

7.5. Effectiveness of the ARPP and RPP heuristics

This section explores the effectiveness of the ARPP and RPP heuristics when solving instances with different available capacities and setup costs.

In order to control the computation time within a comparable level, on 12-period-12-item (12*12) instances, for RPP heuristics, set m to 20, and for ARPP heuristics, set m to 6; on 24-period-24-item (24*24) instances, for RPP heuristics, set m to 20, and for ARPP heuristics, set m to 3.

Specific computation time and the average gap of each heuristic can refer to table 9 in Appendix. On 12*12 instances, the average gaps obtained from RPP heuristics are the results. In which, the parameter values are set to the best value in the parameter tuning experiment. Nevertheless, on 24*24 instances, it is a time-consuming task to perform an extensive parameter tuning experiment. Thus, we keep the same values used in on the 12*12 instances.

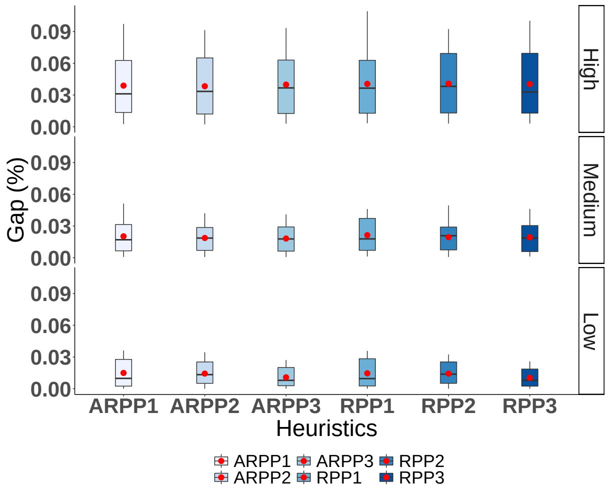
Fig.3 (a) and (b) show the performance of RPP and ARPP heuristics for solving 12*12 and 24*24 instances, respectively on instances with low, medium, and high tightness of capacity. High, medium, and low represent that the available capacities are 1.11 times, 1.25 times, and 2 times the original total capacity requirement, respectively. Each category is an average gap of 120 instances for each heuristic. As shown in Fig.3, when the capacity constraint is not so tight (that is, tightness of capacity varies from high to low), the solution quality obtained by RPP and ARPP heuristics will be better. Even if the best perturbation value of w is found through the parameter tuning experiment, the average gap of the solutions found by the ARPP heuristics is still the same or slightly better than that of the RPP heuristics (Fig.3 (a)). Furthermore, when solving instances with different tightness of capacity, the performance of the ARPP heuristics is better than that of the RPP heuristics from Fig.3 (b).

In Fig.4, results distinguish between low, and high TBO, which represents low and high setup costs, respectively. Each category is an average gap of 180 instances for each heuristic. Fig.4 (a) and (b) respectively demonstrate the performance of RPP and ARPP heuristics for solving 12*12 and 24*24 instances. When the setup cost for each item is generally lower, both RPP and ARPP heuristics tend to find better solutions. We note that the solution quality obtained from the ARPP heuristic is equivalent to or slightly better than that of the RPP heuristic after the parameter tuning experiment. Therefore, after adding the self-adaptive procedure, a time-consuming parameter tuning experiment can be avoided, and the solution quality can be ensured simultaneously.

Among the three ARPP heuristics, $\mathtt{ARPP}_3$ generally performs best, followed by the $\mathtt{ARPP}_2$ heuristic. Thus, we further compare $\mathtt{ARPP}_3$ to the tabu search and lot elimination heuristics based on the same

Table 2: Results for constructive heuristics and randomized constructive heuristics

Algorithm	**Problem Size**		Gunther	DS	HeinA1	HeinA2 **LUC**	HeinA2 **SM**	HeinA3 **LTC**	HeinA3 **AC**	HeinB
Gap	12*12	**Average**	**6.59%**	**4.55%**	**4.44%**	**4.89%**	**4.44%**	**7.26%**	**5.48%**	**4.14%**
		Worst	24.25%	26.26%	29.11%	29.11%	29.11%	29.21%	29.11%	26.26%
		Best	0.36%	0.00%	0.00%	0.00%	0.00%	0.70%	0.00%	0.00%
	24*24	**Average**	**4.61%**	**3.23%**	**2.96%**	**3.52%**	**2.96%**	**6.49%**	**7.32%**	**2.69%**
		Worst	13.45%	16.60%	16.40%	14.89%	16.40%	16.66%	28.53%	11.91%
		Best	0.45%	0.00%	0.00%	0.00%	0.00%	1.94%	0.11%	0.00%
Runtime (s)	12*12	Average	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
	24*24	Average	0.05	0.04	0.05	0.04	0.04	0.04	0.04	0.05
Algorithm			$\mathtt{RPP}_1$	$\mathtt{RPP}_2$	$\mathtt{RPP}_3$	$\mathtt{RPP}_r$	SLE	$\mathtt{RLE}_1$	$\mathtt{RLE}_2$	CPLEX
Gap	12*12	**Average**	**3.77%**	**3.76%**	**3.44%**	**64.81%**	**15.60%**	**10.34%**	**9.48%**	**0.00%**
		Worst	19.14%	21.72%	20.38%	281.16%	88.71%	74.84%	39.94%	0.00%
		Best	0.00%	0.00%	0.00%	8.84%	0.00%	0.00%	0.05%	0.00%
	24*24	**Average**	**2.98%**	**2.59%**	**2.65%**	**147.46%**	**26.70%**	**16.05%**	**16.30%**	**0.01%**
		Worst	11.13%	10.89%	10.41%	759.13%	116.00%	99.92%	102.04%	0.33%
		Best	0.00%	0.00%	0.00%	23.67%	0.12%	0.12%	0.55%	0.00%
Runtime (s)	12*12	Average	0.01	0.01	0.01	0.01	0.9	14.98	6.25	1.46
	24*24	Average	0.08	0.08	0.07	0.09	5.7	25.67	16.71	445.24


(a) 12 periods and 12 items

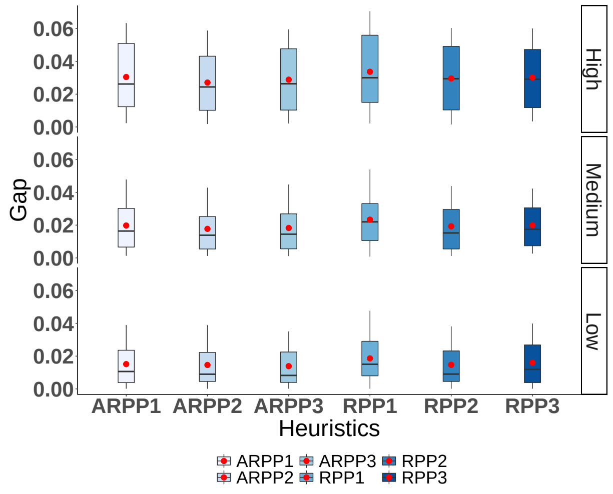
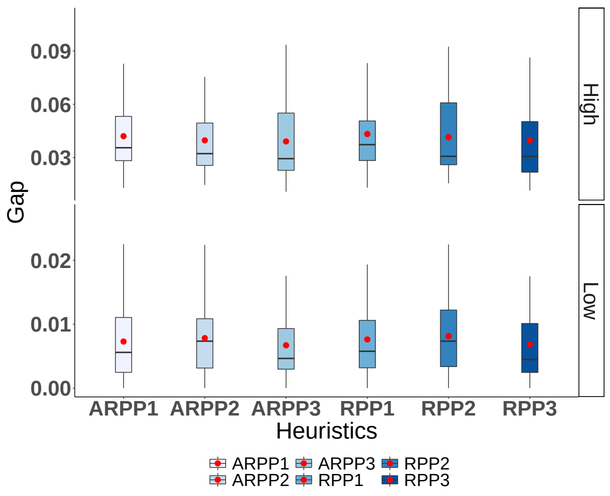

(b) 24 periods and 24 items

Figure 3: Results for different tightness of capacity


(a) 12 periods and 12 items

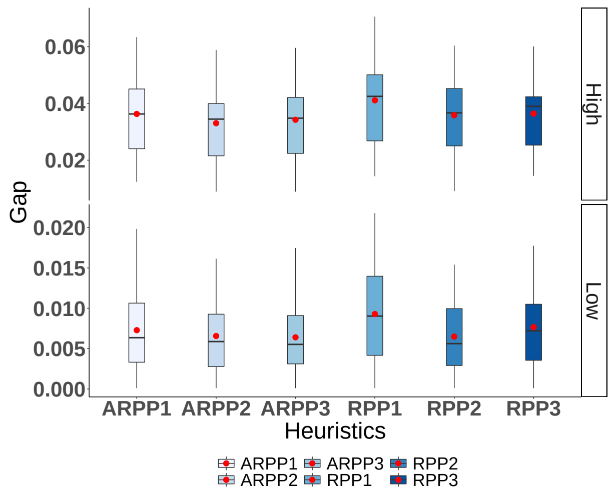

(b) 24 periods and 24 items

Figure 4: Results for different setup cost

initial solutions obtained by HeinB. From Table 3, the third column presents the average gaps for $\mathtt{ARPP}_3$ and the computation time when fixing repetitions m (the first part in brackets shows the computation time).

Although the three algorithms start from the same initial solutions, $\mathtt{ARPP}_3$ performs best under the same time-limit. Within one second, $\mathtt{ARPP}_3$ can decrease the average gap to 1.98% on the 360 12-period-12-item instances, while the lot elimination heuristic can only decrease the average gap to 2.59%. The results obtained from $\mathtt{ARPP}_3$ within one second can beat the TS and the lot elimination heuristics no matter for which problem sizes. It indicates that it is more worthwhile to adopt $\mathtt{ARPP}_3$ than using the TS method or the lot elimination heuristic.

Compared to the fourth and the fifth columns, the TS can always return lower average gaps while consuming more computation time than the lot elimination heuristic. It is reasonable given that the TS method explores more neighborhood solutions. The results also reveal the limitation of the lot elimination heuristic, which only tries to eliminate production schedules.

Table 3: The results for ARPP$_3$, TS and lot elimination

Problem Size	ARPP$_3$	Tabu search	Lot elimination
12*12	2.37% (0.21, $m = 5$) 1.98% (0.84, $m = 20$) 1.61% (3.99, $m = 100$) 1.49% (8.12, $m = 200$) 1.38% (18.89, $m = 500$)	2.11% (4.45)	2.59% (0.70)
24*24	1.92% (1.04, $m = 5$) 1.70% (3.79, $m = 20$) 1.47% (18.21, $m = 100$)	1.95% (32.35)	2.12% (4.11)

7.6. Effectiveness of the combined heuristics

Since the TS method and the lot elimination heuristic can also be applied to the improvement stage, ARPP$_3$ is adopted to generate initial solutions and then using the two improvement algorithms to further improve the solution quality.

Table 4 summarizes the results for the proposed ARPP$_3$ heuristic, and the two combined heuristics in terms of average gap and computation time (the number in the brackets). Each cell is an average result of 360 instances. In the first row, column three presents the result for ARPP$_3$ when setting m to 5; column four presents the result for applying ARPP$_3$ (m is set to 5) to generate an initial solution for each instance and then using TS to get a final result; column five shows the result for applying ARPP$_3$ (m is set to 5) to generate an initial solution for each instance and then using lot elimination to improve the solution.

The ARPP$_3$-LE heuristic eliminates production schedules based on the descending order of the setup costs of all items, which means that the search order is guided by the simple priority index of setup cost. We note that TS method applies restricted neighborhood search on 24-period-24-item problem size instances. When applying ARPP$_3$-TS, the lowest average gaps both on 12*12 and 24*24 instances are achieved. This combined method can get a 0.88% gap for 12*12 size instances with 18.86 seconds and a 1.15% gap for 24*24 size instances within 53 seconds.

Subsequently, to measure the efficiency of our two combined methods and compare them with the recent research results, we show the best results found by our combined methods under the same time limit reported by Hein et al. (2018). In the study of Hein et al. (2018), they applied two genetic algorithms to further improve the solution quality, namely biased random key genetic algorithm (BRKGA) with random seeds and biased random key genetic algorithm (BRKGA) with good seeds.

- BRKGA with random seeds: Biased random key genetic algorithm (BRKGA) with random seeds used in Hein et al. (2018).

- BRKGA with good seeds: Biased random key genetic algorithm (BRKGA) with good seeds used in Hein et al. (2018).

For the BRKGA with good seeds and with random seeds, we only compare the best solutions found by our combined heuristics under the same time limit with the results reported by Hein et al. (2018).

Table 4: Results for ARPP$_3$, ARPP$_3$-TS and ARPP$_3$-LE

Problem Size	**m**	**Algorithm**		
		ARPP$_3$	ARPP$_3$-TS	ARPP$_3$-LE
12*12	5	2.37% (0.21)	1.42% (4.18)	1.59% (0.92)
	20	1.98% (0.84)	1.15% (4.38)	1.40% (1.57)
	100	1.61% (3.99)	1.02% (6.17)	1.19% (5.06)
	200	1.49% (8.12)	0.94% (10.26)	1.10% (9.03)
	500	1.38% (18.89)	0.88% (18.86)	1.04% (19.38)
24*24	5	1.92% (1.04)	1.48% (19.03)	1.60% (6.60)
	20	1.70% (3.79)	1.32% (21.88)	1.43% (7.91)
	100	1.47% (18.21)	1.19% (32.75)	1.26% (22.50)
	200	1.44% (35.76)	1.15% (52.38)	1.22% (39.17)

As reported in Hein et al. (2018), the computation times for solving 12*12 instances are both one second for BRKGA with or without good seeds. The computation times for solving 24*24 instances are 17 seconds and 16 seconds for BRKGA with random seeds and with good seeds, respectively. Therefore, we present the best results found by our combined methods within the same time limit for different problem-size instances. We reset the stopping criteria of TS and apply restricted neighborhood search both on 12*12 and 24*24 instances.

We introduce a parameter F, when the number of iterations exceeds the value of F or the gap is less than 0.00, the TS stops. Besides, we modify the repetitions m of the ARPP$_3$ heuristic. For ARPP$_3$- TS, we set $m = 10$ for ARPP$_3$ and set F to 2 on 12*12 instances; on 24*24 instances, we set F to 2, set $m = 50$ and $m = 55$ for ARPP$_3$ respectively for obtaining best solutions within 16 and 17 seconds. For ARPP$_3$-LE, we set the parameter m of ARPP$_3$ to control the overall computation time. In detail, set $m = 20$ for ARPP$_3$ on 12*12 instances and set $m = 75$ and $m = 80$ for controlling the CPU time within 16 and 17 seconds on 24*24 instances, respectively.

Table 5 demonstrates the average gap and the computation time (the number in the brackets) for each algorithm on 360 12*12 and 24*24 instances, respectively. Our two combined methods can beat the BRKGA with random seeds no matter for which problem size.

When compared to BRKGA with good seeds, the ARPP$_3$-LE is slightly better on 24*24 instances under the same time limit; while on 12*12 instances, neither the two proposed combined methods can find better solutions within one second. But it is noticed that from table 4, our two combined methods can achieve better solutions on both two problem sizes than 'BRKGA with good seeds', but require more CPU time. This can be explained by the following reason. The lot elimination heuristic is a very simple constructive heuristic and the TS method used in this study is an easy-implemented version, their capabilities of optimization are limited. It also shed light on two future directions, one is to apply more effective improvement heuristics to further improve the solution quality; another is to use some methods to reduce the computation time for the proposed heuristics.

Table 5: Performance of our combined method and the combined methods used in Hein et al. (2018)

Problem Size	**BRKGA with random seeds**	**BRKGA with good seeds**	ARPP$_3$-TS	ARPP$_3$-LE
12*12	1.56% (1)	1.19% (1)	1.56% (1.46)	1.40% (1.57)
24*24	2.40% (17)	1.32% (16)	1.35% (16.20)	1.29% (16.04)
			1.34% (17.45)	1.28% (17.32)

7.7. Results on large-size instances

The ARPP$_3$-LE heuristic can obtain better results than the combined heuristics of Hein et al. (2018) on 24*24 size instances within similar computation time and can achieve the same solution quality on 12*12 instances with five seconds. Thus, in this section, we compare the performance of ARPP$_3$, ARPP$_3$-LE with CPLEX on large-size instances, namely 48-period-96-item size, and 96-period-192-item size. More details of the characteristics for each instance can refer to table 7 in Appendix.

Table 6 summarizes the results of the proposed two heuristics and CPLEX on larger-size instances, with different tightness of capacity and different setup costs. The run time of CPLEX is limited to respectively 60, 300, 600, and 3600 seconds for each instance. The run times of ARPP$_3$ and ARPP$_3$-LE are mainly controlled by the parameter m. For 48-period-96-item instances, the value of m is 200. For 96-period-192-item instances, m equals 5 for the two heuristics. The value of each cell represents the average result of running five replications using the respective approaches for one instance. The optimality gaps of the two proposed heuristics, as well as the results from using CPLEX with different runtimes, are reported based on the lower bounds obtained by CPLEX with a timelimit of 3600 seconds.

When the problem size is smaller (48*96), although CPLEX cannot find optimal solutions (the gaps are not 0) within 3600 s, the obtained gaps for the three instances are within 0.25%. When instance problem size gradually increases, CPLEX could not find satisfactory solutions within a reasonable time. For example, on instances 7 and 8, the heuristics can find solutions with lower optimality gaps and less CPU time than CPLEX. It can be seen that when the tightness of capacity becomes higher and the problem size becomes larger, the solution quality of CPLEX gradually decreases. This result is also consistent with the results of Hein et al. (2018). For the smaller sized problems (instances 1 – 3), CPLEX can find better solutions (with lower optimality gaps) in longer computation times (e.g. with timelimits of 600 seconds and 3600 seconds) than the heuristics; and the heuristics can usually outperform CPLEX with runtimes up to 300 seconds. For the instances with low TBO (instances 4 – 6), CPLEX performs especially well and the heuristics can only outperform CPLEX with runtimes up to 60 seconds. For the most challenging instances (instances 7 – 9 with high TBO, 96 items and 192 periods), the heuristics can significantly outperform CPLEX with runtimes up to 600 seconds, and can sometimes obtain better solutions (with lower optimality gaps) than CPLEX after a runtime of 3600 seconds.

8. Conclusions

Classical constructive heuristics, such as the period-by-period heuristics and lot elimination heuristics, are known to be the most intuitive and fastest method for finding good feasible solutions for the CLSPs,

Table 6: Results for ARPP$_3$, ARPP$_3$-LE, and CPLEX on large-sized instances

Ins	Size	Capacity Tightness	Ave. TBO		CPLEX with different time limits				Heuristics	
					60 s	300 s	600 s	3600 s	ARPP$_3$	ARPP$_3$-LE
1	48-96	High: 111%	High	Opt. gap	51.9181%	18.0779%	0.6226%	0.2398%	1.6596%	1.3760%
				Run time	60.156	300.64	600.172	3603.70	423.62	510.19
2	48-96	Medium: 125%	High	Opt. gap	60.4551%	10.9017%	0.2520%	0.1942%	1.1719%	0.9303%
				Run time	60.938	300.17	600.141	3600.20	427.08	534.62
3	48-96	Low: 200%	High	Opt. gap	29.1967%	0.0430%	0.0399%	0.0314%	0.4010%	0.3709%
				Run time	60.141	300.11	600.156	3602.59	446.56	567.04
4	96-192	High: 111%	Low	Opt. gap	12.2608%	0.0059%	0.0059%	0.0047%	0.1123%	0.1029%
				Run time	60.094	300.48	600.375	3601.61	58.53	2359.69
5	96-192	Medium: 125%	Low	Opt. gap	12.6326%	0.0027%	0.0023%	0.0016%	0.1304%	0.1178%
				Run time	63.735	302.55	600.312	3600.44	57.06	2310.56
6	96-192	Low: 200%	Low	Opt. gap	3.4114%	0.0000%	0.0000%	0.0000%	0.0477%	0.0477%
				Run time	60.14	125.75	101.328	131.44	61.10	2327.67
7	96-192	High: 111%	High	Opt. gap	80.7311%	76.5474%	39.7258%	4.1829%	1.6135%	1.4075%
				Run time	60.188	302.39	600.422	3600.41	56.93	1649.68
8	96-192	Medium: 125%	High	Opt. gap	85.7756%	81.9826%	33.1162%	0.9484%	1.0892%	0.9182%
				Run time	61.75	301.77	601.312	3600.52	59.47	1413.47
9	96-192	Low: 200%	High	Opt. gap	91.1543%	77.0041%	37.7422%	0.1905%	0.6538%	0.5442%
				Run time	60.265	301.16	600.359	3601.20	60.38	1389.12

and therefore are often used as a subroutine in building more sophisticated exact and metaheuristic approaches. Extending from these constructive heuristics, we have developed three randomized period-by-period heuristics and two randomized lot elimination heuristics by introducing four perturbation strategies.

Experimental results highlighted that the proposed perturbation strategies can significantly improve the solution quality of the original constructive heuristics and that the improvement is more effective on period-by-period heuristics than on lot elimination heuristics. For the proposed randomized constructive heuristics, two parameters, namely the number of repetitions (m) and the perturbation factor (w), are used to control the heuristics. Concerning the three RPP heuristics, when fixing m and gradually increased the value of w, the average gaps first declined and then started to rise, exhibiting an approximate continuous and convex trend. With this observation from our experimental results, a bisection search method was embedded into the three RPP heuristics for automatically adjusting the parameters.

The resulting ARPP heuristics can automatically choose suitable values of the parameters without the need of performing time-consuming parameter-turning experiments. As a result, the ARPP heuristics can find slightly better solutions than the RPP heuristics, even when the best parameter values were fixed for all instances with extensive parameter tuning for the RPP heuristics. Furthermore, the ARPP heuristics were effective and could find better solutions with less computation time when compared to tabu search and lot elimination heuristics.

When the ARPP$_3$ was used in the Tabu search framework, high-quality solutions with 0.88% average gap can be obtained on benchmark instances of 12 periods and 12 items, and average gap within 1.2% for the instances with 24 periods and 24 items. When compared the results of the two combined methods to those reported in Hein et al. (2018), the proposed ARPP$_3$-LE heuristic could achieve lower average gap for the 24-period-24-item instances with similar run times. Finally, compared ARPP$_3$, ARPP$_3$-LE with CPLEX on large-size instances. Results showed that it would be more efficient to adopt the two heuristics when problem size becomes larger, the tightness of capacity becomes higher, and the setup cost becomes larger. They can achieve better solution quality within reasonable computation time than `CPLEX`.

As for future research, it is worthwhile to further improve the period-by-period heuristics by developing new priority indices that are critical to the performance of the constructive heuristics. With the success of using the self-adaptive procedure, the proposed self-adaptive randomized heuristics can be adapted for other CLSP variants for addressing stochastic demand and multiple-stage decisions.

Acknowledgement

We would like to thank the three anonymous reviewers' constructive and valuable comments to the improvement of this manuscript.

References

Absi, N., Detienne, B., & Dauzère-Pérès, S. (2013). Heuristics for the multi-item capacitated lot-sizing problem with lost sales. *Computers & Operations Research*, *40*, 264–272.

Absi, N., & van den Heuvel, W. (2019). Worst case analysis of relax and fix heuristics for lotsizing problems. *European Journal of Operational Research*, *279*, 449–458.

Absi, N., & Kedad-Sidhoum, S. (2009). The multi-item capacitated lot-sizing problem with safety stocks and demand shortage costs. *Computers & Operations Research*, *36*, 2926–2936.

Barany, I., Van Roy, T. J., & Wolsey, L. A. (1984). Strong formulations for multi-item capacitated lot sizing. *Management Science*, *30*, 1255–1261.

Belvaux, G., & Wolsey, L. A. (2001). Modelling practical lot-sizing problems as mixed-integer programs. *Management Science*, *47*, 993–1007.

Brahimi, N., Absi, N., Dauzère-Pérès, S., & Nordli, A. (2017). Single-item dynamic lot-sizing problems: An updated survey. *European Journal of Operational Research*, *263*, 838–863.

Bruno, G., Genovese, A., & Piccolo, C. (2014). The capacitated lot sizing model: A powerful tool for logistics decision making. *International Journal of Production Economics*, *155*, 380–390.

Cattrysse, D., Maes, J., & Van Wassenhove, L. N. (1990). Set partitioning and column generation heuristics for capacitated dynamic lotsizing. *European Journal of Operational Research*, *46*, 38–47.

Charon, I., & Hudry, O. (2001). The noising methods: A generalization of some metaheuristics. *European Journal of Operational Research*, *135*, 86–101.

Cunha, J. O., Kramer, H. H., & Melo, R. A. (2019). Effective matheuristics for the multi-item capacitated lot-sizing problem with remanufacturing. *Computers & Operations Research*, *104*, 149–158.

Degraeve, Z., & Jans, R. (2007). A new dantzig-wolfe reformulation and branch-and-price algorithm for the capacitated lot-sizing problem with setup times. *Operations research*, *55*, 909–920.

Dixon, P. S., & Silver, E. A. (1981). A heuristic solution procedure for the multi-item, single-level, limited capacity, lot-sizing problem. *Journal of opérations management*, *2*, 23–39.

Dogramaci, A., Panayiotopoulos, J. C., & Adam, N. R. (1981). The dynamic lot-sizing problem for multiple items under limited capacity. *AIIE transactions*, *13*, 294–303.

Eisenhut, P. (1975). A dynamic lot sizing algorithm with capacity constraints. *AIIE transactions*, *7*, 170–176.

Eppen, G. D., & Martin, R. K. (1987). Solving multi-item capacitated lot-sizing problems using variable redefinition. *Operations Research*, *35*, 832–848.

Ferreira, D., Morabito, R., & Rangel, S. (2010). Relax and fix heuristics to solve onestage one-machine lot-scheduling models for small-scale soft drink plants. *Computers & Operations Research*, *37*, 684–691.

Fragkos, I., Degraeve, Z., & De Reyck, B. (2016). A horizon decomposition approach for the capacitated lot-sizing problem with setup times. *INFORMS Journal on Computing*, *28*, 465–482.

Glover, F. (1989). Tabu search—part i. *ORSA Journal on computing*, *1*, 190–206.

Glover, F. (1990). Tabu searc—part ii. *ORSA Journal on computing*, *2*, 4–32.

Gopalakrishnan, M., Ding, K., Bourjolly, J.-M., & Mohan, S. (2001). A tabu-search heuristic for the capacitated lot-sizing problem with set-up carryover. *Management science*, *47*, 851–863.

Gu, H. et al. (1987). Planning lot sizes and capacity requirements in a single stage production system. *European Journal of Operational Research*, *31*, 223–231.

Günther, H.-O. (1988). Numerical evaluation of heuristics for the multi-item singlelevel capacitated lotsize problem. *Engineering costs and production economics*, *14*, 233–243.

Hart, J. P., & Shogan, A. W. (1987). Semi-greedy heuristics: An empirical study. *Operations Research Letters*, *6*, 107–114.

Hein, F., Almeder, C., Figueira, G., & Almada-Lobo, B. (2018). Designing new heuristics for the capacitated lot sizing problem by genetic programming. *Computers & Operations Research*, *96*, 1–14.

Hindi, K. S. (1996). Solving the clsp by a tabu search heuristic. *Journal of the Operational Research Society*, *47*, 151–161.

Hindi, K. S., Fleszar, K., & Charalambous, C. (2003). An effective heuristic for the clsp with set-up times. *Journal of the Operational Research Society*, *54*, 490–498.

Hung, Y.-F., Chen, C.-P., Shih, C.-C., & Hung, M.-H. (2003). Using tabu search with ranking candidate list to solve production planning problems with setups. *Computers & Industrial Engineering*, *45*, 615–634.

Jans, R., & Degraeve, Z. (2007). Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches. *European journal of operational research*, *177*, 1855–1875.

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

Karimi, B., Ghomi, S. F., & Wilson, J. M. (2006). A tabu search heuristic for solving the clsp with backlogging and set-up carry-over. *Journal of the Operational Research Society*, *57*, 140–147.

Karni, R., & Roll, Y. (1982). A heuristic algorithm for the multi-item lot-sizing problem with capacity constraints. *IIE Transactions*, *14*, 249–256.

Kirca, Ö., & Kökten, M. (1994). A new heuristic approach for the multi-item dynamic lot sizing problem. *European Journal of Operational Research*, *75*, 332–341.

de Kok, A., & Graves, S. (). Dynamic models of transportation operations, .

Koken, P., Seok, H., & Yoon, S. W. (2018). A simulated annealing algorithm with neighbourhood list for capacitated dynamic lot-sizing problem with returns and hybrid products. *International Journal of Computer Integrated Manufacturing*, *31*, 739–747.

Lambrecht, M., & Vanderveken, H. (1979). Heuristic procedures for the single operation, multi-item loading problem. *AIIE Transactions*, *11*, 319–326.

Lee, W.-S., Han, J.-H., & Cho, S.-J. (2005). A heuristic algorithm for a multi-product dynamic lot-sizing and shipping problem. *International Journal of Production Economics*, *98*, 204–214.

Maes, J., & Van Wassenhove, L. (1988). Multi-item single-level capacitated dynamic lot-sizing heuristics: A general review. *Journal of the Operational Research Society*, *39*, 991–1004.

Maes, J., & Van Wassenhove, L. N. (1986). A simple heuristic for the multi item single level capacitated lotsizing problem. *Operations research letters*, *4*, 265–273.

Mehdizadeh, E., & Fatehi Kivi, A. (2014). Three metaheuristic algorithms for solving the multi-item capacitated lot-sizing problem with product returns and remanufacturing. *Journal of optimization in industrial engineering*, *7*, 41–53.

Muller, L. F., Spoorendonk, S., & Pisinger, D. (2012). A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research*, *218*, 614–623.

Nascimento, M. C., Resende, M. G., & Toledo, F. M. (2010). Grasp heuristic with path-relinking for the multi-plant capacitated lot sizing problem. *European Journal of Operational Research*, *200*, 747–754.

Özdamar, L., & Barbarosoglu, G. (2000). An integrated lagrangean relaxationsimulated annealing approach to the multilevel multiitem capacitated lot sizing problem. *International Journal of production economics*, *68*, 319–331.

Pedroso, J. P., & Kubo, M. (2005). Hybrid tabu search for lot sizing problems. In *International Workshop on Hybrid Metaheuristics* (pp. 66–77). Springer.

Renaud, J., Boctor, F. F., & Laporte, G. (2002). Perturbation heuristics for the pickup and delivery traveling salesman problem. *Computers & Operations Research*, *29*, 1129–1141.

Sahling, F., Buschkühl, L., Tempelmeier, H., & Helber, S. (2009). Solving a multi-level capacitated lot sizing problem with multi-period setup carry-over via a fix-and-optimize heuristic. *Computers & Operations Research*, *36*, 2546–2553.

Selen, W. J., & Heuts, R. M. (1989). A modified priority index for günther's lot-sizing heuristic under capacitated single stage production. *European journal of operational research*, *41*, 181–185.

Tempelmeier, H., & Herpers, S. (2010). Abc β–a heuristic for dynamic capacitated lot sizing with random demand under a fill rate constraint. *International Journal of Production Research*, *48*, 5181–5193.

Toledo, C. F. M., da Silva Arantes, M., Hossomi, M. Y. B., França, P. M., & Akartunalı, K. (2015). A relax-and-fix with fixandoptimize heuristic applied to multilevel lotsizing problems. *Journal of heuristics*, *21*, 687–717.

Trigeiro, W. W., Thomas, L. J., & McClain, J. O. (1989). Capacitated lot sizing with setup times. *Management science*, *35*, 353–366.

Van Nunen, J., & Wessels, J. (1978). Multi-item lot size determination and scheduling under capacity constraints. *European Journal of Operational Research*, *2*, 36–41.

Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management science*, *5*, 89–96.

Appendix A

Priority index

Besides the notations mentioned in section 3, the following notations are still required to compute the priority indices used in the period-by-period heuristics. The formulas used to calculate the priority indices of period-by-period heristics are concluded in Fig. 5 and 6.

Notations:

k: the current period, in which the lot size for each item is determined;

s_k: the surplus capacity in period k;

T: the number of total periods;

S_i: the fixed set-up cost for item i;

h_i: the unit inventory holding cost for item i;

K_i: production time for item i;

α: the earliest period in which the inequality $\sum_{j=k+1}^{t} -s_j > 0$ is satisfied, where $k+1 \le t < T$. Otherwise, α is equal to $T+1$;

T_{i1}:the number of periods whose demands are satisfied by the current lot for item i in period k (e.g., if $x_{ik} = d_{ik}$, then $T_{i1} = 1$);

t_i: the extended period of product i. If the current lot is extended, the demands in period t_i will be satisfied by the pre-production in period k;

x_{ik}: the current lot size for item i in period k.

T_{i2}: after the lot extension of item i, the number of periods whose demands are satisfied by the updated lot in period k, $T_{i2} = t_i - k + 1$ (e.g., if $x_{ik} = d_{ik} + d_{i,k+1}$, then $t_i = k+1$, $T_{i2} = 2$);

y_{ik}: binary variable, if the lot size for product i in period k is positive ($x_{ik} > 0$), $y_{ik} = 1$; otherwise, $y_{ik} = 0$;

$H_i^{T_{i1}}$: the inventory holding cost incurred with the current lot in period k of item i;

$H_i^{T_{i2}}$: the additional inventory holding cost incurred with the pre-production for future demands d_{it_i} of item i in period k, $H_i^{T_{i2}} = h_i \sum_{j=k}^{t_i}(j-k)d_{ij}$;

$\overline{d_i}$: average demand for product i during the entire planning periods, $\overline{d_i} = \frac{\sum_{j=1}^{T}(d_{ij})}{T}$;

$\widetilde{d_{ik}}$: average demand for product i during the periods from $k+1$ to T after the lot extention of item i, $\widetilde{d_{ik}} = \frac{\sum_{j=k+1}^{T}(x_{ij})}{T-k}$. Note that the value of $\widetilde{d_{ik}}$ may change after a lot-extention because of the pre-production of future demands;

TBO_i: time between orders for item i, $TBO_i = \sqrt{\frac{2S_i}{h_i \overline{d_i}}}$;

E_i: the expected cost savings for item i obtained through combining demands after TBO periods with the lot size in the current period. $E_i = S_i(TBO_i - 1) - (TBO_i(TBO_i - 1)\overline{d_i}h_i)/2$;

CO_t: the capacity overload in period t, $CO_t = \max\{0; \max\limits_{\alpha=k+1,\ldots,t}\{-\sum_{j=k+1}^{\alpha} s_j\}\}$ and $CO_k = 0$;

CH_t: the capacity that can be shifted from period t to the current period k by pre-production, $CH_t = s_k - CO_{t-1}$;

q_{it_i}: the maximum amount of product i that can be shifted from period t_i to period k in Gunther's feasibility routine. $q_{it_i} = \min\{d_{it_i}, \frac{CH_{t_i}}{K_i}\}$;

z_i: the priority index used in the ranking step;

u_i: the priority index used in the lot-sizing step;

v_i: the priority index used in the feasibility routine;

Priority indices	z_i	u_i	v_i
Silver & Meal (SM)[1]		$\frac{(S_i + H_i^{T_{i1}})}{T_{i1}} - \frac{(S_i + H_i^{T_{i2}})}{T_{i2}}$	
Least-Unit Cost (LUC)[1]		$\frac{(S_i + H_i^{T_{i1}})}{x_{ik}} - \frac{(S_i + H_i^{T_{i2}})}{x_{ik} + d_{it_i}}$	
Least Total Cost (LTC)[1]		$(S_i - H_i^{T_{i1}}) - (S_i - H_i^{T_{i2}})$	
Absolute Cost (AC)[1]		$S_i - (H_i^{T_{i2}} - H_i^{T_{i1}})$	
Eisenhut (1975)[2]		$\frac{(S_i - H_i^{T_{i1}})}{T_{i1}^2 d_{it_i}}$	
Lambrecht & Vanderveken (1979)[3]		$\frac{(S_i + H_i^{T_{i1}} - h_i T_{i1}^2 d_{it_i})}{T_{i1}(T_{i1} + 1) d_{it_i}}$	
Gunther (1987)[4]	$\frac{\left(\frac{2S_i}{h_i} - T_{i1} T_{i2} d_{it_i}\right)}{K_i d_{it_i}}$	$\frac{\left(\frac{2S_i}{h_i} - T_{i1} T_{i2} d_{it_i}\right)}{K_i d_{it_i}}$	$\frac{[q_{it_i} h_i (t_i - k) + S_i (1 - y_{ik})]}{q_{it_i} K_i}$
Dixon & Silver (1981)[5]	$\frac{\left[\frac{(S_i + H_i^{T_{i1}})}{T_{i1}} - \frac{(S_i + H_i^{T_{i2}})}{T_{i1} + 1}\right]}{K_i d_{it_i}}$	$\frac{\left[\frac{(S_i + H_i^{T_{i1}})}{T_{i1}} - \frac{(S_i + H_i^{T_{i2}})}{T_{i1} + 1}\right]}{K_i d_{it_i}}$	$\frac{\left[\frac{(S_i + H_i^{T_{i1}})}{T_{i1}} - \frac{(S_i + H_i^{T_{i2}})}{T_{i1} + 1}\right]}{K_i d_{it_i}}$

[1]The priority index formula refers to Hein et al., 2018
[2]The priority index formula refers to Eisenhut, 1975
[3]The priority index formula refers to Lambrecht & Vanderveken, 1979
[4]The priority index formula refers to Gu et al., 1987
[5]The priority index formula refers to Dixon & Silver, 1981

Figure 5: Priority indices used in the period-by-period heuristics-Part I

Priority indices	z_i	u_i	v_i
HeinA1	$\frac{\frac{S_i^2E_i}{\left(K_i\widetilde{d_{ik}}(T-k)\right)^2}+K_i\widetilde{d_{ik}}(T-k)+S_i(1+\lvert y_{ik}S_i-H_i^{T_{i1}}\rvert)}{T_{i1}K_id_{it_i}(H_i^{T_{i2}}-H_i^{T_{i1}})}$	$\frac{\left[\frac{(S_i+H_i^{T_{i1}})}{T_{i1}}-\frac{(S_i+H_i^{T_{i2}})}{T_{i1}+1}\right]}{K_id_{it_i}}$	$\frac{\frac{S_i^2E_i}{\left(K_i\widetilde{d_{ik}}(T-k)\right)^2}+K_i\widetilde{d_{ik}}(T-k)+S_i(1+\lvert y_{ik}S_i-H_i^{T_{i1}}\rvert)}{T_{i1}K_id_{it_i}(H_i^{T_{i2}}-H_i^{T_{i1}})}$
HeinA2_LUC	$\frac{\frac{S_i^2E_i}{\left(K_i\widetilde{d_{ik}}(T-k)\right)^2}+K_i\widetilde{d_{ik}}(T-k)+S_i(1+\lvert y_{ik}S_i-H_i^{T_{i1}}\rvert)}{T_{i1}K_id_{it_i}(H_i^{T_{i2}}-H_i^{T_{i1}})}$	$\frac{(S_i+H_i^{T_{i1}})}{\sum_{i\in N}x_{ik}}-\frac{(S_i+H_i^{T_{i2}})}{(\sum_{i\in N}x_{ik}+d_{it_i})}$	$\frac{\frac{S_i^2E_i}{\left(K_i\widetilde{d_{ik}}(T-k)\right)^2}+K_i\widetilde{d_{ik}}(T-k)+S_i(1+\lvert y_{ik}S_i-H_i^{T_{i1}}\rvert)}{T_{i1}K_id_{it_i}(H_i^{T_{i2}}-H_i^{T_{i1}})}$
HeinA2_SM	$\frac{\frac{S_i^2E_i}{\left(K_i\widetilde{d_{ik}}(T-k)\right)^2}+K_i\widetilde{d_{ik}}(T-k)+S_i(1+\lvert y_{ik}S_i-H_i^{T_{i1}}\rvert)}{T_{i1}K_id_{it_i}(H_i^{T_{i2}}-H_i^{T_{i1}})}$	$\frac{(S_i+H_i^{T_{i1}})}{T_{i1}}-\frac{(S_i+H_i^{T_{i2}})}{T_{i1}+1}$	$\frac{\frac{S_i^2E_i}{\left(K_i\widetilde{d_{ik}}(T-k)\right)^2}+K_i\widetilde{d_{ik}}(T-k)+S_i(1+\lvert y_{ik}S_i-H_i^{T_{i1}}\rvert)}{T_{i1}K_id_{it_i}(H_i^{T_{i2}}-H_i^{T_{i1}})}$
HeinA3_LTC	$\frac{\frac{S_i^2E_i}{\left(K_i\widetilde{d_{ik}}(T-k)\right)^2}+K_i\widetilde{d_{ik}}(T-k)+S_i(1+\lvert y_{ik}S_i-H_i^{T_{i1}}\rvert)}{T_{i1}K_id_{it_i}(H_i^{T_{i2}}-H_i^{T_{i1}})}$	$(S_i-H_i^{T_{i1}})-(S_i-H_i^{T_{i2}})$	$\frac{\frac{S_i^2E_i}{\left(K_i\widetilde{d_{ik}}(T-k)\right)^2}+K_i\widetilde{d_{ik}}(T-k)+S_i(1+\lvert y_{ik}S_i-H_i^{T_{i1}}\rvert)}{T_{i1}K_id_{it_i}(H_i^{T_{i2}}-H_i^{T_{i1}})}$
HeinA3_AC	$\frac{\frac{S_i^2E_i}{\left(K_i\widetilde{d_{ik}}(T-k)\right)^2}+K_i\widetilde{d_{ik}}(T-k)+S_i(1+\lvert y_{ik}S_i-H_i^{T_{i1}}\rvert)}{T_{i1}K_id_{it_i}(H_i^{T_{i2}}-H_i^{T_{i1}})}$	$S_i-(H_i^{T_{i2}}-H_i^{T_{i1}})$	$\frac{\frac{S_i^2E_i}{\left(K_i\widetilde{d_{ik}}(T-k)\right)^2}+K_i\widetilde{d_{ik}}(T-k)+S_i(1+\lvert y_{ik}S_i-H_i^{T_{i1}}\rvert)}{T_{i1}K_id_{it_i}(H_i^{T_{i2}}-H_i^{T_{i1}})}$
HeinB	$z_i=u_i=v_i=\frac{y_{ik}S_i-\frac{K_i(\widetilde{d_{ik}})^2H_i^{T_{i1}}(H_i^{T_{i2}}-H_i^{T_{i1}})}{K_i(\widetilde{d_{ik}})^2\left(K_i(\widetilde{d_{ik}})^2+(H_i^{T_{i2}}-H_i^{T_{i1}})\right)+y_{ik}d_{it_i}\left(K_ix_{ik}+K_id_{it_i}+(H_i^{T_{i2}}-H_i^{T_{i1}})\right)}-(H_i^{T_{i2}}-H_i^{T_{i1}})}{(T_{i1}+1)K_id_{it_i}}$		

All the priority index formulas listed here refer to Hein et al., 2018

Figure 6: Priority indices used in the period-by-period heuristics-Part II

Appendix B

Table 7: Characteristics of the very large sized instances

Instances	**Periods**	**Items**	**Tightness of capacity**	**Average TBO**	**Capacity absorbation**	**Std. deviation demand**	**Demand type**
1	48	96	High: 111%	High	1	uniform[0,10]	Normal
2	48	96	Medium: 125%	High	1	uniform[0,10]	Normal
3	48	96	Low: 200%	High	1	uniform[0,10]	Normal
4	96	192	High: 111%	Low	1	uniform[0,10]	Normal
5	96	192	Medium: 125%	Low	1	uniform[0,10]	Normal
6	96	192	Low: 200%	Low	1	uniform[0,10]	Normal
7	96	192	High: 111%	High	1	uniform[0,10]	Normal
8	96	192	Medium: 125%	High	1	uniform[0,10]	Normal
9	96	192	Low: 200%	High	1	uniform[0,10]	Normal

Table 8: Comparison between $\texttt{RPP}_3$ and $\texttt{RPP}_r$

Problem	m	Time		Gap	
Size		$\texttt{RPP}_r$	$\texttt{RPP}_3$	$\texttt{RPP}_r$	$\texttt{RPP}_3$
12*12	5	0.02	0.04	60.60%	3.01%
	20	0.07	0.18	55.46%	2.33%
	100	0.36	0.86	50.55%	1.88%
	200	0.73	1.68	48.43%	1.73%
	500	1.79	3.72	45.93%	1.58%

Table 9: Results for RPP and ARPP heuristics under similar CPU time

Problem		Solution Approaches					
Size		$\texttt{RPP}_1$	$\texttt{RPP}_2$	$\texttt{RPP}_3$	$\texttt{ARPP}_1$	$\texttt{ARPP}_2$	$\texttt{ARPP}_3$
12*12	m	20	20	20	6	6	6
	w	0.55	0.8	0.35	-	-	-
	Ave. Gap	2.55%	2.48%	2.33%	2.47%	2.38%	2.29%
	Time	0.16	0.19	0.18	0.25	0.24	0.23
24*24	m	20	20	20	3	3	3
	w	0.55	0.8	0.35	-	-	-
	Ave. Gap	2.52%	2.12%	2.20%	2.18%	1.98%	2.03%
	Time	0.67	0.72	0.69	0.73	0.71	0.79

Table 10: Parameter tuning results for three randomized period-by-period heuristics on 12*12 instances

			Perturbed percentage w																	
RPP	**m**		**5%**	**10%**	**15%**	**20%**	**25%**	**30%**	**35%**	**40%**	**45%**	**50%**	**55%**	**60%**	**65%**	**70%**	**75%**	**80%**	**85%**	**90%**
$\texttt{RPP}_1$	5	Gap	3.903%	3.613%	3.608%	3.595%	3.406%	3.407%	3.394%	3.355%	3.268%	3.192%	3.258%	3.322%	3.486%	3.582%	3.401%	3.313%	3.465%	3.553%
		Time	0.06	0.06	0.06	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.04	0.04	0.04	0.04
	20	Gap	3.207%	3.014%	3.034%	2.924%	2.782%	2.793%	2.701%	2.749%	2.718%	2.617%	2.548%	2.597%	2.625%	2.701%	2.652%	2.644%	2.673%	2.77%
		Time	0.21	0.2	0.2	0.2	0.19	0.18	0.18	0.21	0.19	0.2	0.22	0.21	0.23	0.18	0.2	0.2	0.18	0.19
	100	Gap	2.722%	2.589%	2.481%	2.392%	2.318%	2.261%	2.199%	2.195%	2.168%	2.157%	2.101%	2.110%	2.128%	2.125%	2.129%	2.143%	2.177%	2.22%
		Time	1.24	1.17	1.13	1.18	1.14	1.17	1.22	1.18	1.14	1.15	1.14	1.2	1.12	1.15	1.17	1.15	1.15	1.25
	200	Gap	2.615%	2.456%	2.368%	2.255%	2.194%	2.081%	2.087%	2.072%	2.019%	1.998%	1.959%	1.970%	1.978%	2.007%	1.950%	2.008%	2.012%	2.047%
		Time	2.29	2.33	2.22	2.36	2.38	2.2	2.28	2.18	2.34	2.26	2.3	2.19	2.2	2.37	2.23	2.37	2.28	2.23
	500	Gap	2.499%	2.338%	2.200%	2.109%	2.025%	1.967%	1.954%	1.931%	1.89%	1.885%	1.854%	1.860%	1.834%	1.837%	1.810%	1.862%	1.887%	1.901%
		Time	4.95	4.88	4.82	4.92	4.98	4.72	4.78	4.88	4.73	4.93	4.83	4.98	4.94	4.73	4.82	4.93	4.72	4.95
$\texttt{RPP}_2$	5	Gap	3.820%	3.650%	3.471%	3.309%	3.272%	3.175%	3.207%	3.145%	3.162%	3.117%	3.127%	3.082%	3.076%	3.103%	3.238%	3.159%	3.381%	3.275%
		Time	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
	20	Gap	3.668%	3.352%	3.166%	2.946%	2.869%	2.770%	2.733%	2.632%	2.641%	2.551%	2.592%	2.517%	2.508%	2.505%	2.515%	2.484%	2.570%	2.609%
		Time	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.18	0.19	0.19	0.19	0.19	0.19	0.19	0.19	0.18	0.19	0.19
	100	Gap	3.568%	3.192%	2.972%	2.754%	2.596%	2.462%	2.406%	2.266%	2.230%	2.163%	2.127%	2.103%	2.060%	2.060%	2.054%	2.039%	2.073%	2.090%
		Time	0.95	0.94	0.94	0.94	0.94	0.93	0.94	0.94	0.95	0.95	0.94	0.94	0.95	0.94	0.94	0.94	0.94	0.94
	200	Gap	3.537%	3.146%	2.883%	2.640%	2.519%	2.384%	2.302%	2.199%	2.107%	2.052%	2.011%	1.974%	1.928%	1.939%	1.914%	1.893%	1.936%	1.955%
		Time	1.88	1.88	1.88	1.88	1.87	1.86	1.87	1.88	1.89	1.86	1.89	1.86	1.89	1.88	1.87	1.89	1.86	1.86
	500	Gap	3.520%	3.102%	2.809%	2.560%	2.419%	2.282%	2.184%	2.062%	1.990%	1.927%	1.897%	1.855%	1.831%	1.823%	1.767%	1.755%	1.756%	1.812%
		Time	3.78	3.79	3.79	3.79	3.79	3.79	3.79	3.79	3.78	3.78	3.78	3.78	3.78	3.78	3.78	3.78	3.78	3.78
$\texttt{RPP}_3$	5	Gap	3.556%	3.257%	3.090%	3.007%	3.010%	3.020%	3.057%	3.159%	3.295%	3.468%	3.838%	4.028%	4.395%	4.450%	5.223%	4.773%	5.019%	5.894%
		Time	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
	20	Gap	3.301%	2.844%	2.624%	2.450%	2.409%	2.358%	2.332%	2.436%	2.479%	2.523%	2.731%	2.894%	3.182%	3.300%	3.399%	3.541%	3.914%	4.06%
		Time	0.18	0.18	0.18	0.18	0.18	0.18	0.19	0.18	0.18	0.18	0.19	0.18	0.18	0.19	0.19	0.19	0.18	0.18
	100	Gap	3.104%	2.599%	2.344%	2.079%	2.015%	1.884%	1.894%	1.888%	1.948%	2.006%	2.120%	2.185%	2.327%	2.409%	2.556%	2.740%	3.002%	3.092%
		Time	0.86	0.86	0.86	0.87	0.86	0.86	0.85	0.85	0.86	0.86	0.85	0.87	0.85	0.85	0.87	0.85	0.85	0.86
	200	Gap	3.053%	2.542%	2.256%	1.999%	1.890%	1.763%	1.768%	1.727%	1.761%	1.818%	1.860%	1.938%	2.038%	2.144%	2.302%	2.379%	2.524%	2.708%
		Time	1.68	1.68	1.68	1.68	1.68	1.68	1.68	1.68	1.68	1.68	1.69	1.69	1.68	1.68	1.68	1.68	1.69	1.68
	500	Gap	3.026%	2.504%	2.159%	1.883%	1.737%	1.643%	1.587%	1.589%	1.600%	1.648%	1.652%	1.714%	1.802%	1.900%	2.002%	2.079%	2.186%	2.322%
		Time	3.41	3.42	3.42	3.42	3.42	3.42	3.42	3.42	3.42	3.41	3.42	3.42	3.42	3.42	3.42	3.42	3.42	3.42

Table 11: Parameter tuning results for two randomized lot elimination heuristics on 12*12 instances

			Perturbed percentage w																	
RLE	**m**		**5%**	**10%**	**15%**	**20%**	**25%**	**30%**	**35%**	**40%**	**45%**	**50%**	**55%**	**60%**	**65%**	**70%**	**75%**	**80%**	**85%**	**90%**
$\texttt{RLE}_1$	5	Gap	9.34%	9.64%	9.01%	9.22%	8.91%	9.31%	9.23%	10.08%	9.54%	9.36%	9.44%	9.10%	9.23%	9.29%	9.64%	9.23%	9.41%	9.40%
		Time	4.16	3.83	3.76	3.74	3.76	3.61	3.72	3.7	3.86	3.65	3.57	3.45	3.52	3.71	3.59	3.62	3.71	3.75
	20	Gap	8.20%	8.01%	7.84%	8.04%	7.88%	7.97%	7.97%	8.37%	8.11%	8.09%	8.26%	7.78%	8.05%	8.29%	7.88%	7.93%	8.33%	8.24%
		Time	15.12	15.12	15.06	15.02	15.07	15.08	14.97	15.11	15.06	15.02	15.06	14.98	14.96	15.03	14.98	14.91	14.87	14.87
$\texttt{RLE}_2$	5	Gap	39.29%	27.82%	21.62%	16.35%	14.16%	12.40%	11.66%	10.54%	10.46%	10.04%	9.76%	9.59%	9.34%	9.49%	9.48%	9.47%	9.58%	9.85%
		Time	0.39	0.51	0.67	0.91	1.83	1.12	1.33	1.51	1.72	3.41	1.97	2.27	2.52	2.68	5.23	3.1	3.22	3.32
	20	Gap	18.62%	10.79%	9.41%	9.03%	8.32%	7.92%	8.07%	7.73%	7.99%	7.93%	7.86%	8.05%	7.90%	7.93%	8.09%	8.27%	8.19%	8.59%
		Time	1.15	1.94	2.59	3.38	4.09	4.79	5.52	6.25	7.03	7.69	8.48	9.19	9.84	10.44	11.16	11.82	12.51	13.37