Flex Serv Manuf J (2015) 27:534–560
DOI 10.1007/s10696-014-9203-1

Planning of a make-to-order production process in the printing industry

Philipp Baumann · Salome Forrer · Norbert Trautmann

Published online: 11 October 2014

Abstract Offset printing is a common method to produce large amounts of printed matter. We consider a real-world offset printing process that is used to imprint customer-specific designs on napkin pouches. The production equipment used gives rise to various technological constraints. The planning problem consists of allocating designs to printing-plate slots such that the given customer demand for each design is fulfilled, all technological and organizational constraints are met and the total overproduction and setup costs are minimized. We formulate this planning problem as a mixed-binary linear program, and we develop a multi-pass matching-based savings heuristic. We report computational results for a set of problem instances devised from real-world data.

Keywords Real-world production process · Mixed-binary linear program · Symmetry-breaking constraints · Savings heuristic · Matching problem

1 Introduction

In many economic sectors, small and medium enterprises (SMEs) are the primary drivers of innovation and competition. For SMEs in the manufacturing sectors, efficient utilization of the production equipment is particularly important for staying globally competitive. The problem discussed in this paper was reported to us by an SME that operates in the printing industry.

P. Baumann · S. Forrer · N. Trautmann (✉)
University of Bern, Bern, Switzerland
e-mail: norbert.trautmann@pqm.unibe.ch

P. Baumann
e-mail: philipp.baumann@pqm.unibe.ch

S. Forrer
e-mail: salome.forrer@pqm.unibe.ch

The company uses an offset printing process for manufacturing napkin pouches; a significant fraction of the demand is for pouches with customer-specific designs, which are manufactured make-to-order, whereas the remainder is for pouches with standard designs, which are manufactured make-to-stock. Both types of designs are imprinted on the pouches by offsetting the respective inked image from some rotating printing plates to the surface of some paper; afterwards, the pouches are produced by cutting, folding and gluing this paper. The printing plates used here include seven slots; an individual design is allocated to each slot. The planning problem involves determining the number of required printing plates, the allocation of the designs to the slots of these plates, and the number of rotations of each plate such that a given demand for each design is fulfilled, all technological constraints are met, and the total costs are minimized. These costs comprise (a) overproduction costs, which arise when the production quantity of a design exceeds the given demand, and (b) setup costs, which arise for each printing plate that is used. Currently, this problem is solved manually using a time-consuming, spreadsheet-based approach; further drawbacks of this approach are the potential risk of inefficient utilization of the production equipment and of infeasible plans.

To the best of our knowledge, this problem has not yet been discussed in the literature. Related planning problems known from the literature are the job-splitting problem, the layout problem, and the cover-printing problem. Compared to these problems, however, the production process discussed in the present paper comprises various additional technological constraints which increase the complexity of the planning problem.

The contribution of this paper is twofold. First, we formulate the problem as a mixed-binary linear program. To render this optimization problem computationally tractable at least for small-sized instances, we enhance the problem formulation by using constraints that eliminate three different types of symmetries from the search space. Second, we propose a heuristic approach for medium- and large-sized instances. This heuristic is inspired by the well-known savings algorithm of Clarke and Wright (1964) for the vehicle routing problem: in the initial solution of the heuristic, one plate is set up for each design; these plates are then successively merged, where the savings associated with such a merger are devised from the economized setup-cost and the additional overproduction cost. Similar to the heuristics proposed in Altinkemer and Gavish (1991) and Wark and Holt (1994), our heuristic uses repeated matching, i.e., in each iteration of the heuristic, several pairs of plates are merged simultaneously. Eventually, a local-search improvement procedure is applied to the resulting solution. We implement this heuristic as a multi-pass procedure, in which in each pass, the matching problem for identifying the pairs of plates to be merged is varied.

To evaluate the performance of the proposed model formulation and the proposed heuristic, we have generated a set of instances based on the original data provided by the company. Using standard optimization software, the model formulation can be used to determine optimal solutions for small-sized instances within reasonable CPU times. Moreover, the heuristic computes optimal solutions for all small-sized instances and provides good feasible solutions for medium- and large-sized instances.

The remainder of this paper is organized as follows. In Sect. 2, we describe the planning problem in detail. In Sect. 3, we review the related literature. In Sect. 4, we present the formulation as a mixed-binary linear program and the symmetry-breaking constraints. In Sect. 5, we describe the savings heuristic and its implementation as a multi-pass matching-based procedure. In Sect. 6, we report the design and the results of the computational analysis. In Sect. 7, we close the paper with some concluding remarks and directions for future research.

2 Planning situation

In Sect. 2.1, we describe the real-world planning problem to be solved. In Sect. 2.2, we present an illustrative example.

2.1 Planning problem

In offset printing, the image to be printed is indirectly transferred from a metal printing plate to the paper. In general, one printing plate is required for each primary color, and one for black color. The plates are chemically treated such that the ink adheres to specific areas of the plate only. By rotating the plate, the image is first transferred to a rubber cylinder that in turn transfers the image onto the paper as it passes below; the term offset refers to the fact that the image is not printed directly on the paper but on another surface that then makes contact with the paper. In relation to the variable cost, the production and setup costs of the printing plates are relatively high. Therefore, offset printing is best suited for producing large amounts of high-quality printed matter. Typical applications include newspapers, magazines and books. The company involved uses the offset-printing technology to produce napkin pouches.

Each week, a set of customer orders is given, each consisting of a customer-specific design (with or without a white border), a napkin color, and a demand. In typical problem instances of the company mentioned above, between 50 and 90 different customer-specific designs are to be planned. The company concentrates on customer-specific designs; therefore no demand for standard designs is given here, but these designs are produced in separate batches.

In the following, we refer to a set of four printing plates (one for each primary color and one for black) as a single plate. The planning problem discussed in this paper is to determine (1) the number of printing plates to be used; (2) the allocation of the ordered designs to the slots of these plates; and (3) the number of rotations of each plate. The objective is to minimize the total costs, which comprise (a) overproduction costs, which arise when the production quantity of a design exceeds the given demand; and (b) setup costs, which arise for each printing plate that is used. Basically, the constraints of the planning problem are that the given demand for each design must be fulfilled and that the allocation of the designs to the plates must be feasible with respect to the technology and the organization of the shop floor. In detail, the latter can be divided into the following constraints.

1. *Plate constraint* The number of different designs that can be printed using the same plate depends on the size of the plate and the size of the napkin pouches. Here, all pouches have the same size, and a plate comprises seven slots, i.e., at most, seven different designs can be produced using the same plate. It is not possible to leave a slot empty. The number of produced units for a specific design depends on the number of slots allocated to this design and on the number of plate rotations. Whenever a new printing plate is used, setup costs are incurred; these costs include production, cleaning, and installation costs. Note that before and after the respective quantities are imprinted, the plates perform some additional rotations in order to reach the required speed and to stop, respectively; therefore, we assume that the number of plate rotations may be non-integer.
2. *White-border constraint* When the printing plate is wrapped around the cylinder, a small gap between the plate borders arises due to technological reasons. This gap leads to a white stripe on the paper. Therefore, at least two slots on the plate must be occupied by a design that has a white border. Alternatively it is possible to avoid the white stripe by assigning a standard design to one slot. Standard designs use one color only and are printed with an additional fifth cylinder that requires no setup. By rotating the fifth cylinder out of phase, the white stripe is overprinted. It is not feasible to allocate more than one standard design to a plate. A standard design exists for each napkin color.
3. *Color constraint* After printing, the paper is cut, folded and glued. Finally, the napkins are inserted into the resulting pouches using a process-specific machine; this machine can insert napkins of at most two different colors simultaneously. Therefore, a plate must not contain designs with more than two different napkin colors.
4. *Split constraint* The packing of the pouches into the boxes used for delivery imposes an organizational constraint. If a design were allocated to two or more plates (i.e., the order would be split among several plates), then the packing would require significantly more space in the limited packing area and more manpower. Therefore, each design must be allocated to a single plate.

2.2 Illustrative example

We illustrate the planning problem with an example with three customer orders (cf. Table 1). The setup cost per plate is €540 and the overproduction cost per unit of a standard design is €0.001. Due to the color constraint, at least two printing plates are required to fulfill the demand.

Figure 1 shows an optimal solution to the illustrative example.

In this solution, a standard design is allocated to plate B to satisfy the white-border constraint. The number of produced units can be computed by multiplying the number of allocated slots times the corresponding number of rotations of plate p (R^p). In the solution shown in Fig. 1, overproduction costs of €5.83 for the standard design on Plate B occur. The total costs of this solution of €1,085.83 comprise setup costs of €540 for each plate and total overproduction costs of €5.83.

Table 1 Input data of illustrative example

Design	White border	Napkin color code	Demand (units)	Overproduction cost per unit (€)
D1	No	1	15,000	0.0035
D2	Yes	2	20,000	0.0035
D3	No	3	35,000	0.0035

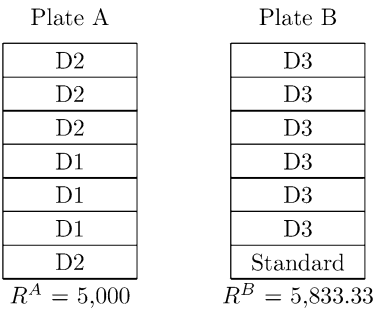

Fig. 1 Optimal solution to the illustrative example

3 Literature review

In this section, we review the literature on problems that are closely related to the planning problem presented in Sect. 2. In particular, we refer to the so-called job-splitting, the layout, and the cover-printing problem, respectively.

The planning problem discussed in this paper can be interpreted as an extension of the job splitting problem (JSP), which we summarize as follows. Given quantities of n products must be produced on a machine with m slots. A particular assignment of a set of jobs to the slots is called a run and requires a setup. The length of a run determines the quantity produced in each slot during that run. The objective is to find a set of runs that minimizes the total setup and overproduction costs. The JSP relates to the planning problem discussed in the present paper as follows. The machine corresponds to a printing plate, the number of slots on the machine corresponds to the number of slots available on the plate, the different types correspond to the different designs, and the run length corresponds to the number of plate rotations. However, the JSP does not include neither the white-border nor the color constraint. Ekici et al. (2010) prove that the JSP is strongly NP-hard, and present a non-linear integer programming formulation, two linear integer programming formulations with some preprocessing steps, and two specific, constructive heuristics.

The so-called layout problem is a variant of the JSP arising in the apparel industry. The layout problem refers to a production process where first several layers of fabric are put on a cutting table, then a combination of several stencils are fixed on top of this stack, and eventually parts of clothing are cut of the fabric. The optimization problem is to find a set of stencil combinations and to determine for each combination the number of layers of fabric that are put together on the cutting

table such that total setup cost and excess is minimized for a given demand. The layout problem compares to the planning problem discussed in the present paper as follows. The combinations of stencils correspond to the printing plates, the number of stencils that fit on the cutting table corresponds to the number of slots available on a printing plate, the different stencils correspond to the different designs, and the number of layers corresponds to the number of plate rotations. However, the layout problem does not include neither the white-border nor the color constraint; moreover, the number of layers per cutting operation is limited from above, and only a part of the cutting table may be used. Degraeve and Vandebroek (1998) formulate this layout problem as a mixed-integer linear program. Degraeve et al. (2002) propose two alternative mixed-integer linear models; the first model includes symmetry-breaking constraints, whereas the second model requires as input a set of all possible stencil combinations and uses binary variables to select a subset of these combinations. Martens (2004) develops two genetic algorithms for the layout problem based on a non-linear and a linear integer programming formulation. Rose and Shier (2007) address a variant of the layout problem in which the entire cutting table must be occupied by stencils except for the last stencil combination, and the demand must be satisfied exactly; the authors present an enumerative approach to minimize the total setup cost.

In the context of book-cover printing, the JSP is also called the cover-printing problem; here, the number of slots is four. Teghem et al. (1995) provide a non-linear and a linear programming formulation and a heuristic of the simulated-annealing type. Elaoud et al. (2007), Tuyttens and Vandaele (2010), and Romero and Alonso-Pecina (2012) propose a genetic algorithm, a greedy random adaptive search procedure and an ad-hoc heuristic, respectively. Peeters and Degraeve (2004) propose a branch-and-price algorithm for the co-printing problem, which is a variant of the cover printing problem. In this problem, the number of rotations is prescribed, not every slot must by occupied by an item, and the number of different item colors per plate is limited from above. Mohan et al. (2007) present a non-linear integer programming formulation and three specific constructive heuristics for an advertisement printing problem, which corresponds to a cover-printing problem with additional bounds on the number of rotations of each plate. Yiu et al. (2007) propose a two-level heuristic for the label-printing problem, which is another variant of the cover-printing problem in which empty slots are allowed, but penalized in the objective function.

4 MBLP model

In this section, we formulate the planning problem presented in Sect. 2 as a mixed-binary linear program. In Sect. 4.1, we formulate the objective function and the technological constraints. In Sect. 4.2, we introduce additional constraints for eliminating symmetric solutions from the search space.

We use the following notation.

Sets

$C = \{1, \ldots, \mid C \mid\}$	Color codes
$I = \{1, \ldots, \mid I \mid\}$	Designs
$I^O \subset I$	Customer-specific designs
$I^S \subset I$	Standard designs
$I_c \subseteq I$	Designs with color code c
$I_w \subseteq I$	Designs with white border
$J = \{1, \ldots, \mid J \mid\}$	Slots
$P = \{1, \ldots, \mid P \mid\}$	Plates

Parameters

$\overline{c}$	Maximum number of different color codes per plate
c_i^O	Overproduction cost per unit of customer-specific design i
c^P	Setup costs per plate
c^S	Overproduction cost per unit of standard design
d_i	Demand of design i
M	Sufficiently large number

Continuous variables (all non-negative)

u_{ijp}	Units of design i produced in slot j of plate p
v_i	Overproduced units of design i

Binary variables

W_p	=1, if plate p is used; =0, else
X_{ijp}	=1, if design i is allocated to slot j of plate p; =0, else
Y_{ip}	=1, if customer-specific design i is allocated to plate p; =0, else
Z_{cp}	=1, if any design with color code c is allocated to plate p; =0, else

Due to the split constraint, there is no feasible solution in which the number of plates used exceeds the number of customer-specific designs. Therefore, we define set P such that $|P| = |I^O|$. W.l.o.g., we assume that $d_i > 0$ for all $i \in I^O$. We choose the parameter M to be $M = \max_{i \in I} d_i$.

4.1 Basic model formulation

The objective is to minimize the total cost for (a) overproduction of customer-specific designs, $\sum_{i \in I^o} c_i^O v_i$; (b) overproduction of standard designs, $\sum_{i \in I^s} c^S v_i$; and (c) setups, $\sum_{p \in P} c^P W_p$. Thus, the objective function is

$$\text{Min.} \sum_{\mathrm{i}\in\mathrm{I^O}} \mathrm{c_i^O v_i} + \sum_{\mathrm{i}\in\mathrm{I^S}} \mathrm{c^S v_i} + \sum_{\mathrm{p}\in\mathrm{P}} \mathrm{c^P W_p} \tag{1}$$

The number of overproduced units of a design is the difference between the number of units produced and the demand for this design, i.e.,

$$v_i = \sum_{j\in J;\, p\in P} u_{ijp} - d_i \quad (i \in I) \tag{2}$$

where $d_i = 0$ for standard designs $i \in I^S$. Because of $v_i \geq 0$ $(i \in I)$, constraint (2) implies that the demand will be fulfilled.

If plate p is used, then one design must be allocated to each slot of this plate, i.e.,

$$\sum_{i\in I} X_{ijp} = W_p \quad (j \in J;\, p \in P) \tag{3}$$

If some units of design i are produced in slot j of plate p, then the design is allocated to this slot, i.e.,

$$u_{ijp} \leq M X_{ijp} \quad (i \in I;\, j \in J;\, p \in P) \tag{4}$$

The number of units produced in each slot of a plate must be the same for all slots, i.e., the number of rotations of the plate:

$$\sum_{i\in I} u_{ijp} = \sum_{i\in I} u_{i,j-1,p} \quad (j \in J : j > 1;\, p \in P) \tag{5}$$

To each plate p, designs with at most $\overline{c}$ different color-codes can be allocated, i.e.,

$$\sum_{c\in C} Z_{cp} \leq \overline{c} \quad (p \in P) \tag{6}$$

where $Z_{cp} = 1$ if a design with color code c is allocated to one or several slots of plate p:

$$|J| Z_{cp} \geq \sum_{i\in I_c;\, j\in J} X_{ijp} \quad (c \in C;\, p \in P) \tag{7}$$

Moreover, for each plate, either a white-border design must be assigned to at least two slots or a standard design must be assigned to one slot, i.e.,

$$W_p \leq \sum_{i\in I_w;\, j\in J} \frac{1}{2} X_{ijp} + \sum_{i\in I^S;\, j\in J} X_{ijp} \quad (p \in P) \tag{8}$$

At most one standard design can be allocated to each plate, i.e.,

$$\sum_{i\in I^S;\, j\in J} X_{ijp} \leq 1 \quad (p \in P) \tag{9}$$

A customer-specific design must not be allocated to several plates, i.e.,

$$\sum_{p \in P} Y_{ip} = 1 \quad (i \in I^O) \tag{10}$$

where $Y_{ip} = 1$ if customer-specific design i is allocated to some slots of plate p, i.e.,

$$|J| Y_{ip} \geq \sum_{j \in J} X_{ijp} \quad (i \in I^O; p \in P) \tag{11}$$

In total, the basic optimization problem is

$$(\mathrm{P}) \begin{cases} \text{Min.} & (1) & \\ \text{s.t.} & (2)-(11) & \\ & u_{ijp} \geq 0 & (i \in I; j \in J; p \in P) \\ & v_i \geq 0 & (i \in I) \\ & W_p \in \{0,1\} & (p \in P) \\ & X_{ijp} \in \{0,1\} & (i \in I; j \in J; p \in P) \\ & Y_{ip} \in \{0,1\} & (i \in I^O; p \in P) \\ & Z_{cp} \in \{0,1\} & (c \in C; p \in P) \end{cases}$$

4.2 Symmetry-breaking constraints

The set of feasible solutions to problem (P) contains many symmetric solutions, i.e., solutions that coincide with other solutions except that some slots or plates are interchanged. Jans (2009) demonstrates for the lot-sizing problem on parallel identical machines that the performance of the model improves considerably when symmetric solutions are removed by additional symmetry-breaking constraints. Recently, Jans and Desrosiers (2013) also applied symmetry-breaking constraints successfully to the so-called job grouping problem. In this subsection, we introduce additional constraints to eliminate such symmetries from the solution space w.l.o.g.

The designs allocated to the slots of a plate p can always be sorted according to a non-decreasing design index, i.e.,

$$\sum_{i \in I} i X_{i,j-1,p} \leq \sum_{i \in I} i X_{ijp} \quad (j \in J : j > 1; p \in P) \tag{12}$$

Moreover, the plates with the smallest indices should be used, i.e.,

$$W_{p-1} \geq W_p \quad (p \in P : p > 1) \tag{13}$$

The plates can always be sorted according to a non-decreasing index of the design allocated to the first slot, i.e.,

$$\sum_{i \in I} i X_{i,j,p-1} \geq \sum_{i \in I} i X_{ijp} \quad (j = 1; p \in P : p > 1) \tag{14}$$

In total, the optimization problem with symmetry-breaking constraints is

$$
(\mathrm{PS})\begin{cases}
\text{Min.} & (1) & \\
\text{s.t.} & (2)-(14) & \\
 & u_{ijp} \geq 0 & (i \in I;\, j \in J;\, p \in P) \\
 & v_i \geq 0 & (i \in I) \\
 & W_p \in \{0,1\} & (p \in P) \\
 & X_{ijp} \in \{0,1\} & (i \in I;\, j \in J;\, p \in P) \\
 & Y_{ip} \in \{0,1\} & (i \in I^O;\, p \in P) \\
 & Z_{cp} \in \{0,1\} & (c \in C;\, p \in P)
\end{cases}
$$

5 Heuristic

In this section, we present a savings-based heuristic for the planning problem described in Sect. 2. The heuristic consists of a construction phase, in which an initial solution is computed, and an improvement phase, in which a simple local-search procedure is iteratively applied to the initial solution.

Roughly speaking, the construction phase proceeds as follows. First, we assign each design to an individual plate. Then, we iteratively merge plates such that the resulting solution is feasible, and the merger results in a reduction of the total cost. We analyze three alternative strategies to select the plates to be merged. The first strategy is to merge in each iteration one of the pairs of plates that results in the highest reduction of the total cost. The second strategy is similar to the first one, but we use a roulette-wheel procedure for selecting the pair of plates to be merged. The third strategy is to solve a matching problem in each iteration and to determine the pairs of plates to be merged from the solution to this matching problem; we formulate the matching problem such that some edge weights are influenced by random numbers. The solution obtained using the second and third strategies depends on the random numbers used; therefore, we propose implementing these strategies in a multi-pass procedure.

In Sect. 5.1, we describe the construction phase in more detail. In Sect. 5.2, we present the improvement phase.

5.1 Construction phase

In the construction phase, an initial feasible solution is computed as follows.

(C1) Each design is allocated to a separate plate. If necessary, a standard design is allocated to satisfy the white-border constraint.

(C2) For each pair of plates, the savings $S_{pp'}$ that result when merging plate p and plate p' are computed as follows.

$$
S_{pp'}\begin{cases}
= \text{Cost of plate } p + \text{ Cost of plate } p' - \text{Cost of merged plate,} \\
\quad \text{if merger is feasible} \\
= -\infty, \text{ otherwise}
\end{cases}
$$

A higher number of single-color plates tends to increase the number of possible mergers in later iterations. To favor mergers that result in single-color plates, we increase the cost of a plate by a penalty if more than one napkin color is allocated.

The allocation of designs to slots of the merged plate is as follows. First, one slot is allocated to each customer-specific design. If necessary, a standard design is allocated to one slot to satisfy the white-border constraint. Second, for each design i, the minimum number of required rotations $r_i = \frac{\text{Demandd}_i}{\text{Numberofoccupiedslots}}$ is computed. Third, the design with the highest value for r_i receives an additional slot. These three steps are repeated until all slots are occupied. Then, r_i is again updated and the number of rotations of the merged plate is set to the maximum value of r_i. Finally, the total cost of the merged plate, including the penalty cost, is computed.

(C3) If no positive savings exist, then the construction phase terminates.

(C4) Otherwise, one or multiple mergers are selected according to the specific selection strategy.

(C5) The selected mergers are performed, and the savings for the resulting solution are computed (→C2).

As mentioned above, we propose the following three alternative strategies to select the plates to be merged.

1. *Greedy selection* In each iteration, a merger with maximum savings is selected.
2. *Roulette-wheel-based selection* In each iteration, one merger with positive savings is selected using a roulette wheel. The probability $P_{pp'}$ that the merger of plates p and p' is selected is computed as follows:

$$P_{pp'} = \frac{(S_{pp'} - \underline{S})^k}{\sum_{p,p' \in P} (S_{pp'} - \underline{S})^k},$$

where $\underline{S}$ denotes the minimum saving among all positive savings and k is a parameter that controls the relation between the savings and the corresponding selection probabilities. With increasing k, the relative selection probability of large savings increases.

3. *Matching-based selection* In each iteration, a maximum-weighted matching problem is solved to select one or several mergers with positive savings. This problem involves selecting a subset of the edges of a graph such that no two edges are adjacent and the total weight of the selected edges is maximized. We construct the graph for this matching problem as follows. The plates of the current solution represent a subset of the nodes of the underlying graph. If the merger of two plates p and p' results in positive savings, an edge between the corresponding plate nodes p and p' with weight $S_{pp'}$ is introduced. Similar to the matching-based approaches for the vehicle routing problem (cf., e.g., Altinkemer and Gavish 1991), our heuristic performs advantageously when the number of different designs on the plates does not increase uniformly. To this

end, we add a set of n dummy nodes to the graph of the matching problem. We choose n such that it is smaller than both, the number of positive savings and the number of plate nodes (see below). If a plate node is merged with a dummy node, then the corresponding plate will not be merged in the current iteration, i.e., the number of different designs on that plate remains constant.
In order to increase the number of possible mergers in later iterations, we propose rather not to merge plates with few designs and white-border designs in early iterations. Every dummy node d is connected to every plate node p by an edge with weight S_{dp}. We chose this weight S_{dp} as follows. Let $\overline{S}$ denote the maximum saving among all savings, R^D and R^W denote random integers in the interval [0,100), and $|J|$ denote the number of slots on a plate. Moreover, let $\sum_{i \in I^o} Y_{ip}$ denote the current number of customer-specific designs on plate p. We compute S_{dp} by

$$S_{dp} \begin{cases} = \overline{S} + R^D(|J| - \sum_{i \in I^o} Y_{ip}) + R^W, \text{ if the plate associated with } p \\ \quad \text{comprises a white-border design} \\ = \overline{S} + R^D(|J| - \sum_{i \in I^o} Y_{ip}), \text{otherwise} \end{cases}$$

The term $\overline{S}$ ensures that all dummy-nodes are merged in the optimal solution of the matching problem. The term $R^D(|J| - \sum_{i \in I^o} Y_{ip})$ increases with decreasing number of different designs on the plate. The term R^W additionally increases the edge weight if the plate has a white-border design.
Based on a non-negative user-defined parameter $D^R < 1$, which we refer to as the dummy ratio, the number n of dummy nodes is computed as follows:

$$n = \lfloor D^R \min(n_1, n_2) \rfloor$$

where n_1 denotes the number of positive savings and n_2 denotes the number of plate nodes in the current matching problem. The dummy ratio D^R is squared and the matching problem is solved again if no merger is performed despite the existence of positive savings.
Figure 2 depicts the graph of the maximum-weighted matching problem for the illustrative example (cf. Sect. 2.2). The graph refers to the first iteration of the heuristic and was constructed using a dummy ratio of 0.5 and a penalty cost for different napkin colors of €200.

5.2 Improvement phase

After the construction phase, an iterative improvement procedure is applied to the initial schedule. This procedure is as follows.

(I1) For each pair of customer-specific designs that are not allocated to the same plate, it is determined whether the total cost could be decreased by swapping the corresponding designs. The allocation of designs to slots is recomputed as described in step C2. Step I1 is repeated until no further cost reduction can be achieved; then, the improvement procedure continues with step I2.

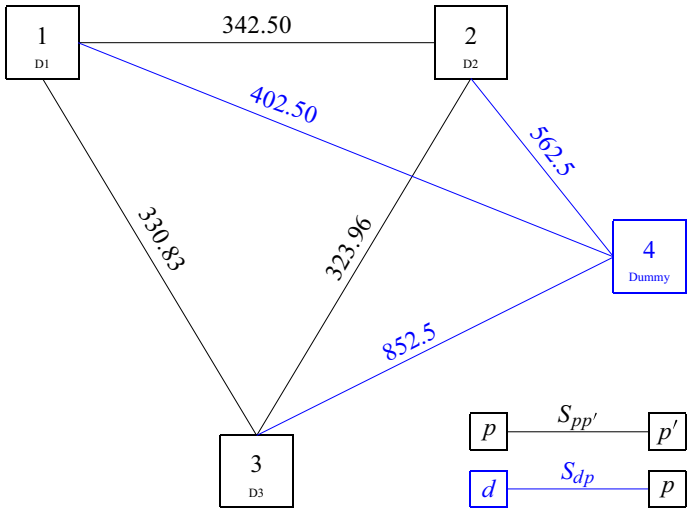

Fig. 2 Graph of the weighted-matching problem for the first iteration of the illustrative example

(I2) For each design, every feasible move to another plate is evaluated. The allocation of designs to slots is recomputed as described in step C2. If some cost reduction can be achieved, the improvement procedure continues with step I1. Otherwise, the procedure stops.

6 Results

In this section, we report the results of an experimental analysis of the mixed-binary linear model presented in Sect. 4 and the multi-pass matching-based heuristic presented in Sect. 5. In Sect. 6.1, we describe the set of problem instances used for the analysis. In Sect. 6.2, we report the results. First, we compare the results of the MBLP formulations with and without the symmetry-breaking constraints. Second, we compare the results of the heuristic under the three alternative selection strategies against the best result obtained using the MBLP formulations.

We implemented the MBLP models in AMLP and used the Gurobi Mixed-Integer Optimizer 5.6.2 as the solver. For the implementation of the multi-pass heuristic, we used ANSI-C; thereby, we used the Gurobi solver to obtain a solution of the matching problems. All computations were performed on a standard workstation with two Intel Xeon CPUs (model E5-2687W) with 3.10 GHz clock speed and 128 GB of RAM.

6.1 Test set and experimental design

We have generated the set of benchmark instances by varying the following problem characteristics.

- *Number of designs N* This parameter mainly drives the size of the MBLP model in terms of the number of decision variables and constraints. We have generated small-sized instances with 5, 10, and 15 different customer-specific designs, medium-sized instances with 20 and 25 different customer-specific designs, and large-sized instances with 30, 50, 70, and 90 different customer-specific designs.
- *White-border ratio WR* This parameter controls the number of designs with a white border relative to the total number of designs. We have generated instances with a low white-border ratio of 0.33 and with a high white-border ratio of 0.66. For each design, a random number between 0 and 1 is drawn. If this number is equal to or smaller than the white-border ratio, then the design belongs to the set of white-border designs.
- *Color-code ratio CR* This parameter is used to compute the maximum number of different napkin colors relative to the total number of designs. We have generated instances with a low color-code ratio of 0.15 and with a high color-code ratio of 0.3. For each instance, we first computed the maximum number of different napkin colors by rounding up the product of the corresponding values for *CR* and *N*. Then, an integer random number between 1 and the computed maximum number of different napkin colors is drawn for each customer-specific design of this instance; this random number is the color code of the respective design.
- *Demand ratio DR* Similar to the color-code ratio, the demand ratio computes the maximum number of different demand values relative to the total number of designs. We have generated instances with a low demand ratio of 0.2 and a high demand ratio of 0.4. For each instance, we computed the maximum number of different demand values by rounding up the product of the corresponding values of *DR* and *N*. Then, we divide the interval [5,000, 80,000] (which has been derived from the original data provided by the company) evenly into several intervals; the number of these intervals is defined by the computed maximum number of different demand values. Then, one of these intervals is chosen randomly; the demand value is then computed by rounding the center of this interval to the nearest multiple of 500.

We have generated one instance for each combination of the above-described parameters. In total, the test set comprises 40 small- and medium-sized instances and 32 large-sized instances. In all instances, the overproduction cost are €0.0035 per unit of customer-specific design and €0.001 per unit of standard design. The setup cost per plate are €540.

We applied the MBLP model with and without symmetry-breaking constraints to all of the 72 instances. Thereby, we set a CPU time limit of 1,800 seconds per instance for the Gurobi Optimizer. For the heuristic approaches, we have used a penalty cost for the allocation of different napkin colors of €200. We applied both the heuristic with the roulette-wheel-based and the heuristic with the matching-based selection strategy 50 times to each instance.

For the roulette-wheel-based selection strategy, we used the value $k = 3$ for the computation of the relative selection probabilities. We tested the values $k = 0, 1, \ldots, 15, 20, \ldots, 95, 100$; Table 2 indicates for each value of k, the number

Table 2 Number of best solutions (best) obtained for different values of parameter k within the roulette-wheel-based heuristic

k	0	1	2	3	4	5	6	7	8	9	10	11	12	15	20	25
Best	22	21	27	28	27	26	28	27	27	25	24	27	24	25	24	21
k	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100	
Best	20	22	21	21	22	21	24	21	20	21	19	19	20	20	20	

of instances for which the best objective function value among these values for k was obtained. The best results were obtained for $k = 3$ and for $k = 6$.

For the matching-based selection strategy, we set the dummy ratio to $D^R = 0.5$ and prescribed a CPU time limit of 5 s per iteration for the Gurobi Optimizer.

6.2 Numerical results

The numerical results of our analysis for the MBLP formulations and the heuristic with the different selection strategies are reported in Tables 3, 4, 5 and 6, respectively. In the first part of this subsection, we are concerned about the results obtained with the MBLP formulations; in the second part of this subsection, we discuss the results obtained with the heuristic approach.

The second to the fifth column of Table 3 refer to the characteristics that were varied to generate each small- and medium-sized instance. In columns 6–11 of Table 3, we present for each instance and the MBLP model without symmetry-breaking constraints, i.e., model (P), and the MBLP model with symmetry-breaking constraints, i.e., model (PS), the objective function value (OF), the MIP gap (G) and the required CPU time (T). For each instance, the best objective function value found is marked in bold face. The entry lim means that the solver was stopped because the prescribed time limit had been reached; the entry na indicates that no feasible solution was found within the prescribed time limit.

For small-sized instances, model (PS) considerably outperforms model (P). Model (PS) solves all of the 24 small-sized instances to optimality, whereas model (P) solves only 12 instances to optimality within the prescribed time limit. For all medium-sized instances, however, the solutions found by model (P) have the same or a better objective function value than the solutions found by model (PS). However, for instances with up to $N = 20$ customer-specific designs, the MIP gaps obtained with model (PS) are smaller than the gaps obtained with model (P); in particular for $N = 20$, the lower bounds provided by model (PS) are considerably higher than for model (P).

In Table 4, we report analogously the results of both MBLP models for the 32 large-sized instances. Model (P) finds feasible solutions for 23 of all large-sized instances, whereas model (PS) can only devise feasible solutions for two large-sized instances. Hence, the symmetry-breaking constraints seem to exclude large areas of the search space, such that finding a feasible solution gets more difficult. From the results for model (P), we conclude that a high color-code ratio seems to render the problem instance more difficult.

Table 3 Numerical results of the MBLP-models (P) and (PS) for small- and medium-sized instances

Inst	*N*	*WR*	*CR*	*DR*	Model (P)			Model (PS)			Model (PS) + Hbest		
					OF (€)	G (%)	T (s)	OF (€)	G (%)	T (s)	OF (€)	G (%)	T (s)
1	5	0.33	0.15	0.2	**731.3**	0.0	20	**731.3**	0.0	18	731.3	0.0	19
2	5	0.33	0.15	0.4	**665.3**	0.0	19	**665.3**	0.0	15	665.3	0.0	14
3	5	0.33	0.30	0.2	**731.3**	0.0	20	**731.3**	0.0	17	731.3	0.0	18
4	5	0.33	0.30	0.4	**1,079.3**	0.0	19	**1,079.3**	0.0	16	1,079.3	0.0	18
5	5	0.66	0.15	0.2	**731.3**	0.0	17	**731.3**	0.0	16	731.3	0.0	18
6	5	0.66	0.15	0.4	**610.9**	0.0	18	**610.9**	0.0	16	610.9	0.0	17
7	5	0.66	0.30	0.2	**731.3**	0.0	18	**731.3**	0.0	14	731.3	0.0	17
8	5	0.66	0.30	0.4	**816.8**	0.0	23	**816.8**	0.0	16	816.8	0.0	17
9	10	0.33	0.15	0.2	**1,354.5**	18.5	Lim	**1,354.5**	0.0	40	1,354.5	0.0	42
10	10	0.33	0.15	0.4	**1,310.5**	17.6	Lim	**1,310.5**	0.0	48	1,310.5	0.0	45
11	10	0.33	0.30	0.2	**1,202.3**	0.0	747	**1,202.3**	0.0	36	1,202.3	0.0	36
12	10	0.33	0.30	0.4	**1,375.8**	21.5	Lim	**1,375.8**	0.0	37	1,375.8	0.0	37
13	10	0.66	0.15	0.2	**1,223.3**	5.6	Lim	**1,223.3**	0.0	39	1,223.3	0.0	37
14	10	0.66	0.15	0.4	**1,242.6**	8.8	Lim	**1,242.6**	0.0	46	1,242.6	0.0	40
15	10	0.66	0.30	0.2	**1,333.5**	0.0	1,699	**1,333.5**	0.0	37	1,333.5*	0.0	42
16	10	0.66	0.30	0.4	**1,363.1**	10.2	Lim	**1,363.1**	0.0	46	1,363.1*	0.0	36
17	15	0.33	0.15	0.2	**1,904.4**	14.9	Lim	**1,904.4**	0.0	355	1,904.4*	0.0	808
18	15	0.33	0.15	0.4	**2,051.0**	21.0	Lim	**2,051.0**	0.0	688	2,051.0*	0.0	770
19	15	0.33	0.30	0.2	**2,018.1**	19.7	Lim	**2,018.1**	0.0	124	2,018.1*	0.0	140
20	15	0.33	0.30	0.4	**2,028.0**	20.1	Lim	**2,028.0**	0.0	271	2,028.0*	0.0	157
21	15	0.66	0.15	0.2	**1,676.3**	3.4	Lim	**1,676.3**	0.0	173	1,676.3	0.0	355
22	15	0.66	0.15	0.4	**1,802.6**	10.1	Lim	**1,802.6**	0.0	654	1,802.6*	0.0	457
23	15	0.66	0.30	0.2	**1,818.8**	10.9	Lim	**1,818.8**	0.0	176	1,818.8*	0.0	681

Table 3 continued

Inst	*N*	*WR*	*CR*	*DR*	Model (P)			Model (PS)			Model (PS) + Hbest		
					OF (€)	G (%)	T (s)	OF (€)	G (%)	T (s)	OF (€)	G (%)	T (s)
24	15	0.66	0.30	0.4	**2,023.4**	19.9	Lim	**2,023.4**	0.0	213	2,023.4*	0.0	136
25	20	0.33	0.15	0.2	**2,053.1**	23.3	Lim	**2,053.1**	21.1	Lim	2,053.1*	0.0	1,662
26	20	0.33	0.15	0.4	2,616.0	38.1	Lim	**2,563.8**	36.8	Lim	2,616.0*	35.1	Lim
27	20	0.33	0.30	0.2	3,233.4	49.9	Lim	**3,027.4**	28.7	Lim	2,832.9*	23.8	Lim
28	20	0.33	0.30	0.4	3,016.4	46.3	Lim	**2,988.6**	27.7	Lim	2,988.6*	45.8	Lim
29	20	0.66	0.15	0.2	**2,172.8**	25.4	Lim	**2,172.8**	25.4	Lim	2,172.8*	10.4	Lim
30	20	0.66	0.15	0.4	**2,594.0**	40.5	Lim	2,740.0	40.9	Lim	2,491.8*	34.2	Lim
31	20	0.66	0.30	0.2	**2,416.4**	33.0	Lim	3,162.4	31.7	Lim	2,936.2	44.8	Lim
32	20	0.66	0.30	0.4	**2,785.6**	41.8	Lim	**2,785.6**	22.5	Lim	3,014.3*	28.3	Lim
33	25	0.33	0.15	0.2	**3,055.6**	35.5	Lim	3,418.1	36.8	Lim	3,330.4*	35.1	Lim
34	25	0.33	0.15	0.4	**3,364.9**	44.1	Lim	3,813.0	43.4	Lim	3,353.6	35.6	Lim
35	25	0.33	0.30	0.2	**3,708.1**	48.1	Lim	4,783.8	54.8	Lim	3,390.4*	36.3	Lim
36	25	0.33	0.30	0.4	**3,850.3**	50.1	Lim	na	na	Lim	3,762.8	42.6	Lim
37	25	0.66	0.15	0.2	**2,973.8**	27.4	Lim	3,004.4	28.1	Lim	3,341.3	35.4	Lim
38	25	0.66	0.15	0.4	**2,898.0**	25.5	Lim	3,192.0	32.3	Lim	3,273.0*	34.0	Lim
39	25	0.66	0.30	0.2	**3,622.5**	40.4	Lim	na	na	Lim	3,694.0	41.5	Lim
40	25	0.66	0.30	0.4	**4,061.5**	46.8	Lim	4,835.6	57.2	Lim	4,124.1	47.6	Lim

Table 4 Numerical results of the MBLP-models (P) and (PS) for large-sized instances

Inst	*N*	*WR*	*CR*	*DR*	Model (P)			Model (PS)			Model (PS) + Hbest		
					OF (€)	G (%)	T (s)	OF (€)	G (%)	T (s)	OF (€)	G (%)	T (s)
41	30	0.33	0.15	0.2	**3,832.3**	41.4	Lim	na	na	Lim	4,088.0	34.0	Lim
42	30	0.33	0.15	0.4	**4,226.0**	36.1	Lim	na	na	Lim	4,434.5	50.9	Lim
43	30	0.33	0.30	0.2	**4,812.9**	43.9	Lim	na	na	Lim	4,847.9	44.3	Lim
44	30	0.33	0.30	0.4	**4,927.8**	55.1	Lim	na	na	Lim	4,622.1	41.6	Lim
45	30	0.66	0.15	0.2	**3,569.2**	35.6	Lim	4,572.3	49.4	Lim	3,933.2	31.4	Lim
46	30	0.66	0.15	0.4	**3,957.4**	43.9	Lim	na	na	Lim	4,361.6	47.0	Lim
47	30	0.66	0.30	0.2	**4,245.6**	46.6	Lim	na	na	Lim	4,148.4	34.9	Lim
48	30	0.66	0.30	0.4	**4,421.3**	50.8	Lim	na	na	Lim	4,494.8	39.9	Lim
49	50	0.33	0.15	0.2	**7,954.8**	48.4	Lim	na	na	Lim	7,641.8	48.8	Lim
50	50	0.33	0.15	0.4	**7,253.5**	49.5	Lim	na	na	Lim	6,879.8	55.1	Lim
51	50	0.33	0.30	0.2	**8,063.6**	59.8	Lim	na	na	Lim	7,253.1	52.8	Lim
52	50	0.33	0.30	0.4	na	na	Lim	na	na	Lim	7,992.3	56.7	Lim
53	50	0.66	0.15	0.2	**7,372.9**	48.6	Lim	na	na	Lim	6,528.0	48.3	Lim
54	50	0.66	0.15	0.4	**6,977.9**	45.8	Lim	na	na	Lim	6,525.7	48.2	Lim
55	50	0.66	0.30	0.2	**8,434.7**	55.2	Lim	na	na	Lim	7,043.1	53.8	Lim
56	50	0.66	0.30	0.4	**8,699.0**	50.3	Lim	na	na	Lim	6,968.8	57.0	Lim
57	70	0.33	0.15	0.2	**14,311.7**	62.3	Lim	na	na	Lim	9,087.1	61.5	Lim
58	70	0.33	0.15	0.4	**12,034.2**	57.2	Lim	na	na	Lim	9,285.7	65.9	Lim
59	70	0.33	0.30	0.2	na	na	Lim	na	na	Lim	10,161.7*	66.7	Lim
60	70	0.33	0.30	0.4	**13,809.2**	63.5	Lim	na	na	Lim	10,777.4	62.3	Lim
61	70	0.66	0.15	0.2	**11,667.4**	56.9	Lim	na	na	Lim	9,096.8	50.7	Lim
62	70	0.66	0.15	0.4	**10,244.2**	49.9	Lim	na	na	Lim	9,415.5	46.0	Lim
63	70	0.66	0.30	0.2	na	na	Lim	na	na	Lim	10,624.4	71.4	Lim
64	70	0.66	0.30	0.4	na	na	Lim	na	na	Lim	9,743.9	61.5	Lim
65	90	0.33	0.15	0.2	na	na	Lim	na	na	Lim	12,263.8	68.1	Lim
66	90	0.33	0.15	0.4	na	na	Lim	na	na	Lim	12,297.3	69.8	Lim
67	90	0.33	0.30	0.2	na	na	Lim	na	na	Lim	12,375.2	69.0	Lim
68	90	0.33	0.30	0.4	na	na	Lim	na	na	Lim	13,204.1	72.1	Lim
69	90	0.66	0.15	0.2	**18,005.0**	68.1	Lim	na	na	Lim	11,952.4	70.9	Lim
70	90	0.66	0.15	0.4	na	na	Lim	na	na	Lim	12,259.6	66.7	Lim
71	90	0.66	0.30	0.2	**15,983.8**	67.9	Lim	na	na	Lim	12,946.0	74.0	Lim
72	90	0.66	0.30	0.4	na	na	Lim	na	na	Lim	12,943.9	70.3	Lim

As model (PS) often has difficulties to find a feasible solution within the prescribed time limit, we analyzed the usefulness of providing an initial feasible solution to model (PS), which we generated using the heuristic under the greedy selection strategy (cf. also Tables 5, 6). In the last three columns of Tables 3 and 4, we list the respective objective function values, the gaps, and the required CPU time. Instances for which the MILP solver found a better solution than the provided initial solution are marked with

Table 5 Numerical results of the heuristic approaches for small- and medium-sized instances

Inst	Heuristic best				Heuristic roulette				Heuristic matching			
	OF^C (€)	OF^I (€)	T (s)	D (%)	OF^C (€)	OF^I (€)	T (s)	D (%)	OF^C (€)	OF^I (€)	T (s)	D (%)
	731.3	**731.3**	<1	0.0	**731.3**	**731.3**	<1	0.0	**731.3**	**731.3**	1	0.0
2	**665.3**	**665.3**	<1	0.0	**665.3**	**665.3**	<1	0.0	**665.3**	**665.3**	1	0.0
3	**731.3**	**731.3**	<1	0.0	**731.3**	**731.3**	<1	0.0	**731.3**	**731.3**	1	0.0
4	**1,079.3**	**1,079.3**	<1	0.0	**1,079.3**	**1,079.3**	<1	0.0	**1,079.3**	**1,079.3**	1	0.0
5	**731.3**	**731.3**	<1	0.0	**731.3**	**731.3**	<1	0.0	**731.3**	**731.3**	1	0.0
6	**610.9**	**610.9**	<1	0.0	**610.9**	**610.9**	<1	0.0	**610.9**	**610.9**	1	0.0
7	**731.3**	**731.3**	<1	0.0	**731.3**	**731.3**	<1	0.0	**731.3**	**731.3**	1	0.0
8	**816.8**	**816.8**	<1	0.0	**816.8**	**816.8**	<1	0.0	**816.8**	**816.8**	1	0.0
9	1,690.1	**1,354.5**	<1	0.0	**1,354.5**	**1,354.5**	<1	0.0	**1,354.5**	**1,354.5**	3	0.0
10	1,723.0	**1,310.5**	<1	0.0	1,519.5	**1,310.5**	<1	0.0	1,519.5	**1,310.5**	2	0.0
11	1,350.8	**1,202.3**	<1	0.0	1,330.3	**1,202.3**	<1	0.0	1,330.3	**1,202.3**	2	0.0
12	1,814.0	**1,375.8**	<1	0.0	**1,375.8**	**1,375.8**	<1	0.0	**1,375.8**	**1,375.8**	3	0.0
13	1,547.3	**1,223.3**	<1	0.0	**1,223.3**	**1,223.3**	<1	0.0	1,333.5	**1,223.3**	3	0.0
14	1,437.0	**1,242.6**	<1	0.0	1,285.9	**1,242.6**	<1	0.0	1,433.3	**1,242.6**	3	0.0
15	1,461.5	1,461.5	<1	9.6	**1,333.5**	**1,333.5**	<1	0.0	**1,333.5**	**1,333.5**	3	0.0
16	1,739.8	1,739.8	<1	27.6	**1,363.1**	**1,363.1**	<1	0.0	1,507.0	**1,363.1**	2	0.0
17	2,165.0	1,970.6	<1	3.5	2,104.4	**1,904.4**	<1	0.0	2,057.5	**1,904.4**	3	0.0
18	2,602.9	2,178.4	<1	6.2	2,093.1	2,059.2	<1	0.4	2,178.4	**2,051.0**	3	0.0
19	2,040.0	2,040.0	<1	1.1	2,085.6	**2,018.1**	<1	0.0	2,040.0	**2,018.1**	2	0.0
20	2,401.0	2,401.0	<1	18.4	2,236.3	**2,028.0**	<1	0.0	**2,028.0**	**2,028.0**	2	0.0
21	2,158.1	**1,676.3**	<1	0.0	1,989.4	**1,676.3**	<1	0.0	1,703.8	**1,676.3**	2	0.0
22	2,249.1	1,977.6	<1	9.7	1,943.5	**1,802.6**	<1	0.0	1,985.8	1,832.6	3	1.7
23	2,216.9	2,173.1	<1	19.5	2,255.0	**1,818.8**	<1	0.0	1,866.9	**1,818.8**	3	0.0

Table 5 continued

Inst	Heuristic best				Heuristic roulette				Heuristic matching			
	OF^C (€)	OF^I (€)	T (s)	D (%)	OF^C (€)	OF^I (€)	T (s)	D (%)	OF^C (€)	OF^I (€)	T (s)	D (%)
24	2,265.9	2,265.9	<1	12.0	2,025.4	**2,023.4**	<1	0.0	**2,023.4**	**2,023.4**	2	0.0
25	2,640.8	2,640.8	<1	28.6	2,682.1	**2,394.3**	<1	16.6	2,513.9	**2,394.3**	4	16.6
26	2,886.8	2,776.8	<1	8.3	2,709.8	**2,563.8**	<1	0.0	2,576.0	**2,563.8**	3	0.0
27	3,186.0	3,186.0	<1	19.1	3,223.5	2,895.0	<1	8.2	2,837.3	**2,675.7**	3	0.0
28	3,558.0	3,160.1	<1	5.7	3,016.4	2,986.8	<1	−0.1	3,098.9	**2,940.8**	4	−1.6
29	2,646.6	2,544.5	<1	17.1	2,623.8	**2,172.8**	<1	0.0	2,310.4	**2,172.8**	4	0.0
30	2,979.8	2,586.8	<1	2.7	2,656.8	**2,361.3**	<1	−6.3	2,826.8	2,405.4	4	−4.5
31	2,986.7	2,936.2	<1	21.5	**2,587.9**	**2,587.9**	<1	7.1	2,594.0	**2,587.9**	3	7.1
32	3,256.2	3,251.1	<1	15.9	3,186.0	**2,675.4**	<1	−4.6	2,902.0	**2,675.4**	3	−4.6
33	3,840.0	3,452.5	<1	14.1	3,487.5	3,100.4	<1	2.5	3,363.1	**3,056.0**	43	1.0
34	3,914.6	3,353.6	<1	−0.7	3,640.5	3,416.8	<1	1.2	3,640.5	**3,304.3**	45	−2.1
35	3,752.8	3,752.8	<1	1.8	3,894.8	3,435.3	<1	−6.8	3,603.8	**3,319.0**	18	−10.0
36	3,831.8	3,762.8	<1	−11.3	3,987.8	**3,713.3**	<1	−12.5	3,828.0	3,713.6	26	−12.5
37	3,746.5	3,341.3	<1	11.2	3,544.4	3,089.0	<1	2.8	3,347.5	**2,973.8**	44	−1.0
38	4,032.1	3,559.5	<1	8.8	3,199.5	**2,898.0**	<1	−11.5	3,683.3	3,163.9	38	−3.3
39	4,061.5	3,694.0	<1	−0.1	3,944.4	3,638.9	<1	−1.6	3,617.4	**3,449.1**	18	−6.7
40	4,280.8	4,124.1	<1	8.6	4,097.3	3,771.8	<1	−0.7	**3,613.9**	**3,613.9**	17	−4.8
∅				6.5				−0.1				−0.6

Table 6 Numerical results of the heuristic approaches for large-sized instances

Inst	Heuristic best				Heuristic roulette				Heuristic matching			
	OF^C (€)	OF^I (€)	T (s)	D (%)	OF^C (€)	OF^I (€)	T (s)	D (%)	OF^C (€)	OF^I (€)	T (s)	D (%)
41	4,505.8	4,088.0	<1	−4.1	4,266.1	3,951.0	1	−7.3	4,120.8	**3,844.3**	79	−9.8
42	4,993.3	4,434.5	<1	11.1	4,819.8	4,048.0	1	1.4	4,428.8	**3,693.1**	62	−7.5
43	5,022.1	4,847.9	<1	2.4	5,070.4	4,854.4	<1	2.5	4,644.1	**4,450.0**	46	−6.0
44	4,622.1	4,622.1	<1	−3.8	4,585.3	4,246.1	<1	−11.6	4,252.4	**4,097.2**	42	−14.7
45	4,499.8	3,933.2	<1	−0.8	4,209.4	3,757.0	1	−5.3	4,204.1	**3,616.1**	81	−8.8
46	4,836.0	4,361.6	<1	5.1	4,441.5	3,893.0	1	−6.2	4,230.1	**3,868.0**	73	−6.8
47	4,160.7	4,148.4	<1	−3.0	4,661.5	4,312.5	<1	0.8	3,969.2	**3,903.9**	39	−8.7
48	4,739.8	4,494.8	<1	−11.3	4,926.9	**4,272.0**	1	−15.7	4,441.4	4,290.8	37	−15.3
49	7,804.4	7,641.8	<1	−2.9	8,369.5	7,192.5	5	−8.6	7,135.5	**6,634.5**	291	−15.7
50	8,169.2	6,879.8	<1	−14.5	7,936.5	6,677.6	7	−17.1	7,197.3	**6,304.4**	306	−21.7
51	7,762.6	7,253.1	<1	−4.3	8,670.4	7,817.0	5	3.2	7,094.3	**6,808.0**	237	−10.1
52	8,297.5	7,992.3	<1	na	9,304.6	8,636.0	5	na	7,637.4	**7,513.4**	220	na
53	7,317.0	6,528.0	<1	−7.8	7,615.5	6,647.8	6	−6.1	6,891.8	**6,164.3**	315	−13.0
54	7,726.1	6,525.7	<1	−14.5	8,160.1	6,808.2	7	−10.8	7,144.5	**6,394.8**	305	−16.2
55	7,661.9	7,043.1	<1	−17.4	8,468.3	7,723.1	6	−9.5	7,204.9	**6,902.4**	237	−19.1
56	7,447.9	**6,968.8**	<1	−11.1	8,462.6	7,927.4	6	1.1	7,163.1	7,115.6	236	−9.3
57	10,643.8	**9,087.1**	<1	−27.6	12,376.1	10,662.2	15	−15.1	10,143.8	9,270.2	393	−26.2
58	10,686.8	9,285.7	<1	−9.8	12,136.8	10,490.9	17	2.0	9,770.9	**8,938.7**	393	−13.1
59	10,310.5	10,161.8	<1	na	13,369.0	11,670.4	15	na	10,183.3	**9,614.4**	352	na
60	11,014.3	10,777.4	<1	na	13,255.4	11,854.2	16	na	10,380.8	**10,043.4**	340	na
61	10,855.8	9,096.8	<1	−13.0	11,382.1	9,556.1	19	−8.6	9,813.4	**8,926.1**	396	−14.6
62	10,733.5	9,415.5	<1	−15.3	10,939.8	9,900.2	18	−10.9	9,692.6	**8,665.3**	427	−22.0
63	10,909.4	10,624.4	<1	−26.4	12,914.2	11,920.7	16	−17.4	10,262.8	**9,902.6**	370	−31.4

Table 6 continued

Inst	Heuristic best				Heuristic roulette				Heuristic matching			
	OF^C (€)	OF^I (€)	T (s)	D (%)	OF^C (€)	OF^I (€)	T (s)	D (%)	OF^C (€)	OF^I (€)	T (s)	D (%)
64	9,846.0	9,743.9	<1	na	13,142.4	11,765.5	15	na	9,774.5	**9,486.6**	346	na
65	14,091.2	12,263.8	<1	−25.1	16,526.1	14,322.7	38	−12.6	12,726.7	**11,961.4**	319	−27.0
66	13,366.6	12,297.3	<1	na	16,154.2	13,742.8	37	na	12,920.2	**11,866.9**	320	na
67	13,537.9	**12,375.2**	<1	na	17,981.6	15,848.1	36	na	13,114.2	12,540.4	307	na
68	13,653.3	13,204.1	<1	na	18,328.1	16,516.4	32	na	12,751.8	**12,407.0**	309	na
69	13,123.7	11,952.4	<1	−15.1	15,411.7	12,905.2	42	−8.4	12,254.0	**10,902.7**	319	−22.6
70	13,431.1	12,259.6	<1	−48.1	15,515.6	12,655.3	39	−46.4	11,621.4	**10,655.4**	321	−54.9
71	13,642.7	12,946.0	<1	na	16,706.5	14,952.9	35	na	12,684.1	**12,358.5**	308	na
72	13,719.1	12,943.9	<1	na	17,036.8	15,639.6	37	na	12,686.4	**12,055.6**	313	na
∅				−11.2				−9.0				−17.2

Table 7 Robustness of multi-pass heuristics for small- and medium-sized instances

Inst	Heuristic roulette				Heuristic matching			
	OF^I (€)				OF^I (€)			
	Min	Avg	Max	$\frac{(max-min)}{min}$	Min	Avg	Max	$\frac{(max-min)}{min}$
1	731.3	731.3	731.3	0.00	731.3	731.3	731.3	0.00
2	665.3	665.3	665.3	0.00	665.3	665.3	665.3	0.00
3	731.3	731.3	731.3	0.00	731.3	731.3	731.3	0.00
4	1,079.3	1,079.3	1,079.3	0.00	1,079.3	1,079.3	1,079.3	0.00
5	731.3	731.3	731.3	0.00	731.3	731.3	731.3	0.00
6	610.9	610.9	610.9	0.00	610.9	610.9	610.9	0.00
7	731.3	731.3	731.3	0.00	731.3	731.3	731.3	0.00
8	816.8	816.8	816.8	0.00	816.8	816.8	816.8	0.00
9	1,354.5	1,385.4	1,464.8	0.08	1,354.5	1,398.7	1,689.6	0.25
10	1,310.5	1,452.8	2,046.5	0.56	1,310.5	1,427.8	2,114.5	0.61
11	1,202.3	1,392.8	1,841.3	0.53	1,202.3	1,351.9	1,841.3	0.53
12	1,375.8	1,410.1	1,522.8	0.11	1,375.8	1,431.0	1,522.8	0.11
13	1,223.3	1,227.7	1,333.5	0.09	1,223.3	1,227.7	1,333.5	0.09
14	1,242.6	1,245.8	1,322.0	0.06	1,242.6	1,247.3	1,322.0	0.06
15	1,333.5	1,488.9	1,558.9	0.17	1,333.5	1,503.7	1,558.9	0.17
16	1,363.1	1,661.5	1,845.3	0.35	1,363.1	1,694.3	2,021.5	0.48
17	1,904.4	2,112.3	2,491.3	0.31	1,904.4	2,113.1	2,670.0	0.40
18	2,059.2	2,237.7	2,449.8	0.19	2,051.0	2,198.9	2,801.3	0.37
19	2,018.1	2,358.5	2,797.1	0.39	2,018.1	2,193.7	2,508.5	0.24
20	2,028.0	2,523.0	2,915.6	0.44	2,028.0	2,319.9	2,817.6	0.39
21	1,676.3	1,921.8	2,393.3	0.43	1,676.3	1,911.2	2,400.6	0.43
22	1,802.6	2,096.7	2,588.8	0.44	1,832.6	2,012.3	2,378.6	0.30
23	1,818.8	2,361.9	2,919.2	0.61	1,818.8	2,224.2	2,666.3	0.47
24	2,023.4	2,359.0	2,884.9	0.43	2,023.4	2,235.9	2,662.6	0.32
25	2,394.3	2,603.7	3,198.8	0.34	2,366.8	2,526.2	2,921.1	0.23
26	2,563.8	2,801.8	3,069.3	0.20	2,563.8	2,725.0	3,069.3	0.20
27	2,895.0	3,195.3	3,583.9	0.24	2,675.7	3,046.7	3,448.1	0.29
28	2,986.8	3,221.2	3,649.7	0.22	2,940.8	3,164.3	3,407.0	0.16
29	2,172.8	2,544.4	2,997.8	0.38	2,172.8	2,410.4	2,997.8	0.38
30	2,361.3	2,678.7	3,052.3	0.29	2,405.4	2,676.9	3,218.3	0.34
31	2,587.9	2,962.8	3,551.9	0.37	2,587.9	2,834.5	3,424.9	0.32
32	2,675.4	3,194.0	3,643.6	0.36	2,675.4	3,020.2	3,421.4	0.28
33	3,100.4	3,545.9	3,924.4	0.27	3,056.0	3,418.3	3,906.9	0.28
34	3,416.8	3,686.3	4,279.3	0.25	3,304.3	3,571.6	3,927.0	0.19
35	3,435.3	3,973.4	4,643.4	0.35	3,319.0	3,729.9	4,168.3	0.26
36	3,713.3	4,047.4	4,670.7	0.26	3,713.6	3,900.9	4,333.1	0.17
37	3,089.0	3,430.5	3,865.8	0.25	2,973.8	3,299.3	3,624.2	0.22
38	2,898.0	3,458.9	3,911.0	0.35	3,163.9	3,437.8	3,834.0	0.21
39	3,638.9	4,090.9	4,698.3	0.29	3,449.1	3,791.5	4,237.3	0.23

Table 7 continued

Inst	Heuristic roulette				Heuristic matching			
	OF^I (€)				OF^I (€)			
	Min	Avg	Max	$\frac{(max-min)}{min}$	Min	Avg	Max	$\frac{(max-min)}{min}$
40	3,771.8	4,285.8	5,052.4	0.34	3,613.9	3,929.9	4,282.0	0.18
∅				0.25				0.23

asterisk. For the small- and medium-sized instances, the initial solution could be improved for 19 instances. By providing an initial solution, the MIP gaps of model (PS) could be reduced for 9 instances. However, for 7 instances larger MIP gaps were obtained. For the large-sized instances, only one initial solution could be improved. The impact on the MIP gaps of model (PS) for large-sized instances cannot be evaluated properly as there is only one instance for which model (PS) found a feasible solution. Overall, providing an initial solution does not increase the performance of model (PS) substantially.

Next, we turn to the heuristic approach. In Tables 5 and 6, we present the results of the heuristic under the three alternative selection strategies for the small- and medium-sized, and the large-sized instances, respectively. For each selection strategy and each instance, we report the objective function value that was obtained after the construction (column OF*C*) and after the improvement phase (column OF*I*), respectively. For the multi-pass implementations, we applied the improvement phase after each pass and report the best solution found within the 50 passes after the construction and after the improvement phase. For each instance we set the best objective function value(s) in bold face. In addition, we report for each strategy the total CPU time required T, and the relative deviation D to the best solution found by the MBLP models. The CPU time reported for the multi-pass implementations includes all 50 passes.

Both the roulette-wheel based and the matching-based selection strategy outperform the greedy selection strategy for the small- and medium-sized instances (cf. Table 5). Both approaches devise for 23 of the 24 small-sized instances an optimal solution within short CPU times. The cost reduction achieved by the improvement phase tends to increase in the number of designs. From Table 6, we conclude that for the large-sized instances, all solutions found by the matching-based strategy have smaller objective function values than the best solutions found by either MBLP formulation. The relative advantage of the matching-based selection strategy tends to increase in the number of designs. Furthermore, for large-sized instances, the roulette-wheel based selection strategy is generally outperformed by the greedy selection strategy. One reason for that may be that the large portion of the roulette wheel which represents mergers that result in moderate or low savings deteriorates the performance of this selection strategy.

To evaluate the robustness of the multi-pass implementations, we report in Tables 7 and 8 the minimum, the average, and the maximum objective function value after the improvement phase. In columns five and nine of these two tables, we

Table 8 Robustness of multi-pass heuristics for large-sized instances

Inst	Heuristic roulette				Heuristic matching			
	OF^I (€)				OF^I (€)			
	Min	Avg	Max	$\frac{(max-min)}{min}$	Min	Avg	Max	$\frac{(max-min)}{min}$
41	3,951.0	4,307.1	4,827.5	0.22	3,844.3	4,161.8	4,546.9	0.18
42	4,048.0	4,480.9	5,193.0	0.28	3,693.1	4,290.3	4,905.7	0.33
43	4,854.4	5,225.5	5,777.6	0.19	4,450.0	4,797.0	5,113.8	0.15
44	4,246.1	4,801.4	5,607.6	0.32	4,097.2	4,353.2	4,797.8	0.17
45	3,757.0	4,087.0	4,505.9	0.20	3,616.1	3,965.2	4,327.2	0.20
46	3,893.0	4,313.2	4,861.4	0.25	3,868.0	4,136.8	4,520.9	0.17
47	4,312.5	4,758.8	5,181.9	0.20	3,903.9	4,146.7	4,492.9	0.15
48	4,272.0	5,071.2	5,665.9	0.33	4,290.8	4,616.4	5,020.1	0.17
49	7,192.5	7,937.5	8,916.9	0.24	6,634.5	7,213.1	7,800.8	0.18
50	6,677.6	7,396.8	8,161.2	0.22	6,304.4	6,748.9	7,055.3	0.12
51	7,817.0	8,853.4	10,085.0	0.29	6,808.0	7,172.5	7,603.5	0.12
52	8,636.0	9,431.3	10,417.9	0.21	7,513.4	7,889.0	8,247.3	0.10
53	6,647.8	7,310.2	8,129.5	0.22	6,163.1	6,691.1	7,164.6	0.16
54	6,808.2	7,454.2	8,258.7	0.21	6,394.8	6,806.4	7,102.2	0.11
55	7,723.1	8,731.8	10,124.9	0.31	6,963.6	7,311.5	7,803.8	0.12
56	7,927.4	8,827.2	9,761.7	0.23	7,115.6	7,438.7	7,896.2	0.11
57	10,662.2	11,663.1	13,294.0	0.25	9,270.2	9,729.2	10,396.8	0.12
58	10,490.9	11,512.9	12,609.7	0.20	8,938.7	9,505.8	10,044.6	0.12
59	11,670.4	13,116.2	14,625.7	0.25	9,614.4	10,191.9	10,696.8	0.11
60	11,854.2	13,465.1	14,706.4	0.24	9,935.2	10,351.0	10,907.5	0.10
61	9,556.1	10,788.3	12,037.4	0.26	8,926.1	9,396.5	10,126.0	0.13
62	9,900.2	10,784.7	11,931.7	0.21	8,665.3	9,180.7	9,768.3	0.13
63	11,920.7	13,465.6	15,191.7	0.27	9,902.6	10,532.4	11,006.4	0.11
64	11,765.5	13,174.0	14,762.2	0.25	9,486.6	9,879.9	10,236.4	0.08
65	14,322.7	15,510.8	17,396.9	0.21	11,835.3	12,391.6	12,924.9	0.09
66	13,742.8	15,077.3	16,749.9	0.22	11,894.3	12,500.6	13,076.9	0.10
67	15,848.1	17,710.1	20,428.1	0.29	12,540.4	13,150.2	13,795.6	0.10
68	16,516.4	17,894.8	19,411.0	0.18	12,527.9	13,041.3	13,637.2	0.09
69	12,905.2	14,099.3	15,650.1	0.21	10,902.7	11,491.3	12,276.6	0.13
70	12,655.3	14,848.1	16,406.1	0.30	10,655.4	11,410.3	12,132.4	0.14
71	14,952.9	17,216.1	19,035.3	0.27	12,416.6	12,729.0	13,264.6	0.07
72	15,639.6	17,215.5	18,934.7	0.21	12,063.2	12,594.1	13,209.3	0.10
∅				0.24				0.13

state the relative gap between the minimum and the maximum objective function value for each instance. The results indicate that the matching-based selection strategy is more robust, especially for large-sized instances.

7 Conclusions

Within the context of a real-world offset printing process for manufacturing napkin pouches, we have analyzed the problem of determining a number of printing plates, a feasible allocation of designs to the slots of these plates, and a number of rotations of each plate such that a given customer demand is fulfilled for all designs and the total overproduction and setup costs are minimized. First, we have formulated this problem as a mixed-binary linear program, which we have then extended by additional constraints to remove symmetric solutions from the search space. Second, we have developed a heuristic solution approach, which consists of a construction phase and an improvement phase; the construction phase is related to matching-based savings heuristics for the vehicle routing problem. Our computational results indicate that within reasonable CPU times, optimal solutions to small-sized instances can be determined using the MBLP model with symmetry-breaking constraints. For medium- and large-sized instances, the model without symmetry-breaking constraints performs better; however, no feasible solution is found for approximately half of the large-sized instances, and rather large MIP gaps are observed. Using the multi-pass savings-based heuristic, we could devise optimal solutions for all small-sized instances; for medium and large-sized instances, the heuristic outperforms the MBLP model considerably.

The real business world situation discussed and the solution approaches proposed in this paper open up an interesting area for the application of combinatorial optimization in production planning and control. An important area of future research is the development of hybrid solution approaches, i.e., to combine the MBLP model formulation presented in this paper with an heuristic search strategy. In addition, alternative mixed-integer linear programming formulations should be investigated; one possible direction would be to generate a set of promising plate patterns, and to use linear programming for selecting a feasible subset of these plate patterns with minimal total cost.

References

Altinkemer K, Gavish B (1991) Parallel savings based heuristics for the delivery problem. Oper Res 39:456–469

Clarke G, Wright J (1964) Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 12:568–581

Degraeve Z, Vandebroek M (1998) A mixed integer programming model for solving a layout problem in the fashion industry. Manag Sci 44:301–310

Degraeve Z, Gochet W, Jans R (2002) Alternative formulations for a layout problem in the fashion industry. Eur J Oper Res 143:80–93

Ekici A, Ergun Ö, Keskinocak P, Lagoudakis MG (2010) Optimal job splitting on a multi-slot machine with applications in the printing industry. Naval Res Logist 57:237–251

Elaoud S, Teghem J, Bouazizc B (2007) Genetic algorithms to solve the cover printing problem. Comput Oper Res 34:3346–3361

Jans R (2009) Solving lot-sizing problems on parallel identical machines using symmetry-breaking constraints. Inf J Comput 21:123–136

Jans R, Desrosiers J (2013) Efficient symmetry breaking formulations for the job grouping problem. Comput Oper Res 40:1132–1142
Martens J (2004) Two genetic algorithms to solve a layout problem in the fashion industry. Eur J Oper Res 154:304–322
Mohan SR, Neogy SK, Seth A, Garg NK, Mittal S (2007) An optimization model to determine master designs and runs for advertisement printing. J Math Model Algorithms 6:259–271
Peeters M, Degraeve Z (2004) The co-printing problem: a packing problem with a color constraint. Oper Res 52:623–638
Romero D, Alonso-Pecina F (2012) Ad hoc heuristic for the cover printing problem. Discret Optim 9:17–28
Rose DM, Shier DR (2007) Cut scheduling in the apparel industry. Comput Oper Res 34:3209–3228
Teghem J, Pirlot M, Antoniadis C (1995) Embedding of linear programming in a simulated annealing algorithm for solving a mixed integer production planning problem. J Comput Appl Math 64:91–102
Tuyttens D, Vandaele A (2010) Using a greedy random adaptative search procedure to solve the cover printing problem. Comput Oper Res 37:640–648
Wark P, Holt J (1994) A repeated matching heuristic for the vehicle routing problem. J Oper Res Soc 45:1156–1167
Yiu KFC, Mak KL, Lau HYK (2007) A heuristic for the label printing problem. Comput Oper Res 34:2576–2588

Philipp Baumann received his BSc, MSc and PhD in Business Administration from the University of Bern (Switzerland) in 2007, 2009, and 2013, respectively. Currently he is a postdoctoral fellow at the Department of Industrial Engineering and Operations Research at the University of California, Berkeley. His research interests include optimization in finance, project management and project scheduling, production planning and control, data mining, and network-based optimization. He has published in European Journal of Operational Research, International Journal of Production Research, and Mathematical Methods of Operations Research.

Salome Forrer received her BSc in Business Administration from the University of Bern (Switzerland) in 2012 and her MSc in Finance from the University of Rochester (New York) in 2014. Currently she is a research and teaching assistant at the Department of Business Administration at the University of Bern. Her research focuses on optimization in finance and on combinatorial optimization in operations management.

Norbert Trautmann received his BSc and MSc in Business Engineering/Management Science and his PhD in Business and Economics from the University of Karlsruhe (Germany) in 1994, 1997, and 2000, respectively. Currently he holds the Chair in Quantitative Methods in Business Administration at the University of Bern (Switzerland). His research interests are in combinatorial optimization, project management and project scheduling, production planning and control, and portfolio optimization. He has published in European Journal of Operational Research, International Journal of Production Research, Journal of Scheduling, Mathematical Methods of Operations Research, and OR Spectrum.