Discrete Optimization 9 (2012) 17–28

Contents lists available at SciVerse ScienceDirect

Discrete Optimization

journal homepage: www.elsevier.com/locate/disopt

Ad hoc heuristic for the cover printing problem

David Romero*, Federico Alonso-Pecina

Instituto de Matemáticas, Universidad Nacional Autónoma de México, 62210, Cuernavaca, Mor., Mexico

ARTICLE INFO

Article history:
Received 31 March 2009
Received in revised form 7 March 2011
Accepted 5 October 2011
Available online 15 November 2011

MSC:
90C10
90C27
90C59
90C90

Keywords:
Combinatorial optimization
Integer nonlinear programming
Cover printing
Label printing
Heuristic
Job splitting

ABSTRACT

We address an *NP*-hard combinatorial optimization problem arising in a printing shop. An impression grid is composed by a set of plates. The cover printing problem consists in designing the composition of impression grids, and determining the number of times each grid is to be printed in order to fulfill the demand of different book covers at minimum total printing cost; the latter comes from three fixed costs: for printing one sheet, for producing one plate, and for composing one impression grid. For each cover an unlimited number of plates can be made. To deal with this challenging problem we present an ad hoc heuristic that outperforms all previously proposed approaches, including genetic algorithms, GRASP, and simulated annealing.

1. Introduction

In this paper, we address a combinatorial optimization problem arising in the printing industry. Let $M = \{1, \ldots, m\}$ be a set of different book covers (or advertisements, labels, tracts, etc.) of equal size, and suppose that d_i copies are to be printed of cover i, for $i \in M$. Let $\tilde{d} = (d_1, \ldots, d_m)$ be the requirements vector. Suppose that for each print an unlimited number of identical plates can be made, and that an *impression grid* – also called a master or template – can accommodate a specified number of t plates. The printing process is as follows.

1. Compose an impression grid of t plates (some of them may be identical), and make a certain number of imprints with it. Each imprint produces one large printed sheet of paper which, once properly cut into t parts, yields t copies.
2. Repeat step 1 until all the required copies are made.

From the second grid on, each grid is composed by replacing an arbitrary number of plates from the previous grid. The replaced plates are automatically destroyed and therefore cannot be reused.

The printing cost comes from three fixed costs: C_1 for printing one sheet, C_2 for composing one impression grid (or grid, for short), and C_3 for producing one plate. Thus, the problem consists in determining the number of grids, the composition of each grid (which plates?), and the number of imprints made with each grid, so as to fulfill the copies' requirement at minimum total cost.

* Corresponding author. Tel.: +52 777 3291721; fax: +52 777 3291722.
E-mail addresses: davidr@matcuer.unam.mx (D. Romero), falonso@matcuer.unam.mx (F. Alonso-Pecina).

1572-5286/$ – see front matter
doi:10.1016/j.disopt.2011.10.002

Example. Let $m = 5$ be the number of covers, $t = 4$ the grid size, and $\tilde{d} = (3200, 2500, 3000, 1400, 2050)$ the requirements vector, for a total of 12 150 copies. Suppose a grid is described by a set $\{a_1, a_2, a_3, a_4\}$, where $a_j \in \{1, \dots, 5\}$ for $j = 1, \dots, 4$, are the plates identification. A solution satisfying the requirements with grids $\{2, 4, 3, 3\}$, $\{2, 5, 1, 1\}$, and $\{4, 5, 3, 3\}$, could be: print the first grid 1000 times. Compose the second grid from the first grid by replacing the two plates of cover 3 by two plates of cover 1, and the only plate of cover 4 by one plate of cover 5; print the second grid 1600 times. Finally, print 500 times the third grid. Thus, $1000 + 1600 + 500 = 3100$ imprints are made, $4 \times 3100 = 12\,400$ copies are produced, and ten plates are needed; this yields $3100C_1 + 3C_2 + 10C_3$ total cost and $12\,400 - 12\,150 = 250$ wasted copies, around 2% of wastage.

Note that a better solution (although not necessarily optimal, as this depends of course upon the relationships among costs C_1, C_2, C_3) can immediately be obtained by reversing the order in which grids 2 and 3 are produced, which leads to eight needed plates instead of ten.

The described combinatorial optimization problem was encountered in a Mexican printing shop in 1972, with typical values: $m = 100$, $12 \le t \le 25$, $10\,000 \le d_i \le 100\,000$, $C_2 = 3\,000C_1$, $C_3 = 50C_1$. Since then and with the exception of [1], all known reported investigations on the subject disregard the cost C_3 for producing plates; this will be our approach in the sequel, as it reflects better the realm of modern printing technologies. However, at the conclusion we will suggest a procedure to adopt in case C_3 is not immaterial.

This problem bears some similarity to the cutting stock, the bin packing, and the multiset multicover problems (see for example [2,3]). Although some special cases can be polynomially solved as shown below and in [4], in general this problem is strongly *NP*-hard, as it has been recently established by Ekici et al. [4]. A heuristic centered on the column generation technique of linear programming was outlined more than 30 years ago by Balinski [5], and seemed a good approach to solving it but, unfortunately, it was afterward found to lead to nothing.

To the best of our knowledge, besides some graduate thesis [6–9,1][1] only a handful of papers have been internationally published on the subject. Teghem et al. [10] dealt with a situation originating in a Belgian printing shop, and proposed a solution method that combines the simulated annealing metaheuristic with linear programming techniques. Simulated annealing was also reported by Yiu et al. [13] as a successful heuristic to approximate the optimal solution when the number of grids is prescribed. An approach through genetic algorithms described by Elaoud et al. [11] appeared to improve on the results obtained in [10]. Mohan et al. [14] proposed ad hoc heuristics for a version of the problem that incorporates lower and upper bounds on the number of imprints made by each grid, and in case the number of grids is prescribed. Ekici et al. [4] designed and successfully applied two specific heuristics to 32 real-world instances of an American printing company, including one with as much as 2086 distinct covers. Tuyttens and Vandaele [12] designed and implemented a greedy random adaptative search procedure (GRASP) that was proved to outperform previously proposed simulated annealing and genetic algorithms for several instances with $t = 4$. Also, there is the problem that has arisen in a French printing shop [15], with typical values: $4 \le m \le 18$, $5 \le t \le 12$, $10\,000 \le d_i \le 100\,000$, $C_2 = 10\,000C_1$.

This paper is organized as follows. Section 2 provides a mathematical formulation of the problem having an exponential number of variables. In Section 3 we describe our approach to the problem through both exact and ad hoc heuristic methods. Section 4 is devoted to computational experiments: the proposed methods were evaluated both by comparing our results with those known to us on specific instances, and by extensive testing on randomly generated instances. Finally, in Section 5, together with some final comments, we present a procedure that can be used in case the cost C_3 of producing plates is not immaterial.

2. Mathematical formulation

Recall $M = \{1, \dots, m\}$ is the set of covers, and let $N = \{1, \dots, n\}$ be the set of all possible impression grids, with $n = \binom{m+t-1}{t}$. Consider the integer, non-negative m-by-n matrix $A = \{a_{ij}\}$ where a_{ij} represents the number of plates of cover i in grid j, for $(i, j) \in M \times N$. Obviously $\sum_{i=1}^{m} a_{ij} = t$, for $j \in N$.

Thus the cover printing problem – also referred to as advertisement printing, label printing or job splitting problem (see [14,13,4], respectively) – can be formulated as one of integer nonlinear programming:

$$P = \begin{cases} \text{(min)} & C_1 \sum_{j \in N} x_j + C_2 \sum_{j \in N} y_j & & \\ \text{subject to} & \sum_{j \in N} x_j\, a_{ij} \ge d_i & i \in M, & (1) \\ & x_j(1 - y_j) = 0 & j \in N, & (2) \\ & x_j \ge 0 \text{ and integer} & j \in N, & (3) \\ & y_j \in \{0, 1\} & j \in N. & (4) \end{cases}$$

where x_j and y_j, for $j \in N$, are the decision variables, with $y_j = 1$ if and only if grid j is selected, x_j being the number of its imprints. This formulation, although compact, presents a big challenge: not only are the non-linearity constraints (2)

[1] Graduate thesis [6–9] are cited in [10–12], and were not available to the authors.

together with the integrity constraints (3) and (4) very difficult to deal with, but the number of variables can be tremendously large, even for relatively small values of m and t. In Section 3 we present our approach to Problem P.

Remark 1. There is no optimal solution to P employing more than m grids, and there is no feasible solution to P with less than $\lceil m/t \rceil$ grids.

3. Algorithms

This section deals with the algorithms we developed to approach the cover printing problem. First, in Section 3.1, we consider the cover printing problem with prescribed number of grids, for which we describe Algorithm $\mathcal{G}$, an (exponential) approach to solving it. To find optimal or nearly optimal solutions to small instances of Problem P we propose Algorithm $\mathcal{E}$ in Section 3.2. Then, Section 3.3 is devoted to explain Algorithm $\mathcal{F}$, an exact, polynomial procedure to solve P when both the number of grids is prescribed to two, and $m \in \{2t,\ 2t-1,\ 2t-2\}$. Finally, we present Algorithm $\mathcal{H}$ in Section 3.4, designed to heuristically approach any instance of Problem P, which uses Algorithm $\mathcal{F}$ as a subroutine.

3.1. The cover printing problem with k grids

When the number of grids is prescribed to k the cover printing problem can be stated as

$$P(k) = \begin{cases} \text{(min)} & \sum_{j\in K} x_j & \\ \text{subject to} & \sum_{j\in K} x_j\, b_{ij} \geq d_i & i \in M \\ & \sum_{i\in M} b_{ij} = t, & j \in K \\ & x_j \geq 1 \text{ and integer} & j \in K \\ & b_{ij} \geq 0 \text{ and integer} & (i,j) \in M \times K \end{cases}$$

where $K = \{1, \ldots, k\}$. Here, the decision variables are both x_j for $j \in K$, and the m-by-k "composition matrix" $B = \{b_{ij}\}$. Problem $P(k)$ seems as difficult as Problem P; however, when m and k are small enough an implicit enumeration schema can be devised to solve it.

Clearly, if a feasible solution to $P(k)$ comprises a matrix B and some k-vector, then B belongs to the set Ω of m-by-k integer non-negative matrices $\{b_{ij}\}$ with $\sum_{i\in M} b_{ij} = t$ for $j \in K$, and $\sum_{j\in K} b_{ij} \neq 0$ for $i \in M$. Conversely, every $B \in \Omega$ comprises part of a feasible solution to $P(k)$. Furthermore, for symmetry reasons we can restrict our search to the subset Ω' of matrices in Ω whose columns are in lexicographic descending order.[2] Algorithm $\mathcal{G}$ below incorporates this schema; any subroutine implementing efficiently the Simplex method can be used in step 1(a).

<u>Algorithm $\mathcal{G}$</u>
1. For each $B \in \Omega'$:
   (*a*) Solve the linear programming problem $L(B)$ arising from $P(k)$ when B is assumed constant, and x_j for $j \in K$ are the decision variables.
   (*b*) Set $z(B) \leftarrow \sum_{j\in K} \lceil x_j^*(B) \rceil$, where $(x_1^*(B), \ldots, x_k^*(B))$ denotes an optimal solution to $L(B)$.
2. Form a solution to $P(k)$ with $B^* \in \Omega'$ and $(\lceil x_1^*(B^*) \rceil, \ldots, \lceil x_k^*(B^*) \rceil)$, such that B^* satisfies $z(B^*) = \min_{B\in\Omega'}\{z(B)\}$.

The size of Ω' is exponential in m (we assume $k \leq m \leq kt$). To see this consider first any positive, integer vector $(v_1, \ldots, v_m)$ such that $\sum_{i=1}^m v_i = kt$; then, as established by a more general result (see for instance [16]), $\frac{m!}{(m-k+1)!}$ is a lower bound on the number of non-negative, integer m-by-k matrices where row i sums up to v_i, for $i = 1, \ldots, m$, and each column sums up to t. Furthermore, $\binom{kt-1}{m-1}$ being the number of positive, integer m-vectors whose entries sum up to kt we get $|\Omega'| \geq \frac{m!}{(m-k+1)!}\binom{kt-1}{m-1}$. A trite calculation shows that this figure is exponential in m.

Observe that when the entries $d_1, \ldots, d_m$ are large enough – as usually occurs in practice – Algorithm $\mathcal{G}$ indeed produces optimal or near optimal solutions.

3.2. A procedure for small instances of Problem P

Algorithm $\mathcal{G}$ of Section 3.1 serves as a basis for Algorithm $\mathcal{E}$ – see below –, which produces a solution S^* to Problem P. The rationale of Algorithm $\mathcal{E}$ comes from both Remark 1 and our belief that for most instances of the cover printing problem the

[2] For vectors $\bar{u} = (u_1, \ldots, u_m)$ and $\bar{v} = (v_1, \ldots, v_m)$, vector $\bar{u}$ is lexicographically greater than $\bar{v}$ if there is and index $\hat{\imath}$ such that $u_{\hat{\imath}} > v_{\hat{\imath}}$, and $u_i = v_i$ for every $i \in \{1, \ldots, \hat{\imath} - 1\}$.

cost function $f(k) = kC_2 + z^*(k)C_1$ has a single minimum, where k is the number of grids, and $z^*(k)$ is the value of the true optimal solution to $P(k)$. In view that Algorithm $\mathcal{E}$ has exponential computational complexity – inherited from Algorithm $\mathcal{G}$ – its usefulness is limited to approach very small instances of P.

Algorithm $\mathcal{E}$

$k \leftarrow \lceil m/t \rceil;\ C^* \leftarrow \infty;$
Repeat
  Solve $P(k)$ with Algorithm $\mathcal{G}$, yielding a solution $S(k) = (x_1^*, \ldots, x_k^*)$;
  $C(k) \leftarrow C_2 k + C_1 \sum_{j=1}^{k} x_j^*;$
  If $C(k) < C^*$ **then**
    $C^* \leftarrow C(k);\ S^* \leftarrow S(k);\ k \leftarrow k + 1;$
  Else
    $k \leftarrow m + 1;$
Until $k > m$.

3.3. A polynomial, exact algorithm for a case of $P(2)$

When the number of grids is prescribed to two, Problem P becomes

$$P(2) = \begin{cases} \text{(min)} & x_1 + x_2 & \\ \text{subject to} & x_1\, b_{i1} + x_2\, b_{i2} \geq d_i & i \in M \\ & \sum_{i \in M} b_{ij} = t, & j = 1, 2 \\ & x_j \geq 0 \text{ and integer} & j = 1, 2 \\ & b_{ij} \geq 0 \text{ and integer} & i \in M;\ j = 1, 2 \end{cases}$$

where the decision variables are x_1, x_2, and the m-by-2 composition matrix $B = \{b_{ij}\}$. This particular case of Problem $P(k)$ can be solved to optimality with little (polynomial) effort in case $m \in \{2t,\ 2t-1,\ 2t-2\}$, as we show now.

Remark 2. Without loss of generality assume $d_1 \geq \cdots \geq d_m$ and $x_1 \geq x_2$. Then, to solve Problem $P(2)$ we can disregard feasible solutions where, for $i < k$, either $b_{i1} = b_{k1}$ and $b_{i2} < b_{k2}$, or $b_{i1} < b_{k1}$ and $b_{i2} = b_{k2}$, or $b_{i1} + b_{i2} = b_{k1} + b_{k2}$ and $b_{i1} < b_{k1}$.

In case $m = 2\,t$, from Remark 2 the only composition matrix to consider – denoted B_0 – is the transpose of $\begin{pmatrix} 1 \cdots 1\, 0 \cdots 0 \\ 0 \cdots 0\, 1 \cdots 1 \end{pmatrix}$. Thus Problem $P(2)$ is optimally solved with B_0 and the optimal solution of

$$\begin{array}{lll} \min & x_1 + x_2 & \\ \text{subject to} & 1x_1 + 0x_2 \geq d_i & (i = 1, \ldots, t) \\ & 0x_1 + 1x_2 \geq d_i & (i = t+1, \ldots, m) \\ & x_1, x_2 \geq 0 \quad \text{and integer,} & \end{array}$$

namely, $x_1^* = d_1, x_2^* = d_{t+1}$.

Now take cases $m = 2\,t - 1$ and $m = 2\,t - 2$. From Remark 2 the only composition matrices worth consideration are shown in Table 1, denoted $B_1, \ldots, B_4$ for case $m = 2\,t - 1$, and $B_5, \ldots, B_{18}$ for case $m = 2\,t - 2$. If ℓ indexes these matrices, then $[x_1^*(\ell),\ x_2^*(\ell)]$ denotes the (easily found) optimal solution to their corresponding integer programming problems.

Thus, from the above considerations an exact procedure to solve $P(2)$ in case $m \in \{2t,\ 2t-1,\ 2t-2\}$ can be readily be built as

Algorithm $\mathcal{F}$

If $m = 2\,t$ **then**
  $x_1^* \leftarrow d_1;\ x_2^* \leftarrow d_{t+1};\ B^* \leftarrow B_0;$
EndIf;
If $m = 2\,t - 1$ **then**
  $Z(\ell) \leftarrow x_1^*(\ell) + x_2^*(\ell)$, for $\ell = 1, \ldots, 4$;
  Let $\ell^* \in \{1, 2, 3, 4\}$ such that $Z(\ell^*) = \min\{Z(1), \ldots, Z(4)\}$;
  $x_1^* \leftarrow x_1^*(\ell^*);\ x_2^* \leftarrow x_2^*(\ell^*);\ B^* \leftarrow B_{\ell^*};$
EndIf;
If $m = 2\,t - 2$ **then**
  $Z(\ell) \leftarrow x_1^*(\ell) + x_2^*(\ell)$, for $\ell = 5, \ldots, 18$;
  Let $\ell^* \in \{5, \ldots, 18\}$ such that $Z(\ell^*) = \min\{Z(5), \ldots, Z(18)\}$;
  $x_1^* \leftarrow x_1^*(\ell^*);\ x_2^* \leftarrow x_2^*(\ell^*);\ B^* \leftarrow B_{\ell^*};$
EndIf.

3.4. Algorithm $\mathcal{H}$

This section presents our main contribution to the subject, namely, an ad hoc heuristic to solve Problem P. Let Θ be the set of feasible solutions to P. For $S \in \Theta$ the reals $C(S)$ and $\pi(S)$ henceforth denote its total cost and the number of

Table 1
The m-by-2 composition matrices for cases $m = 2\,t - 1$ (above) and $m = 2\,t - 2$ (below). Their corresponding optimal solutions $x_1^*(\ell), x_2^*(\ell)$ are also shown.

	B_1	B_2	B_3	B_4
1	2 0	1 1	0 2	1 0
2	1 0	1 0	1 0	1 0
⋮	⋮	⋮	⋮	⋮
$t-1$	1 0	1 0	1 0	1 0
t	0 1	1 0	1 0	1 0
$t+1$	0 1	0 1	1 0	0 2
$t+2$	0 1	0 1	0 1	0 1
⋮	⋮	⋮	⋮	⋮
m	0 1	0 1	0 1	0 1
$x_1^*(\ell)$	$\max\{\lceil d_1/2\rceil, d_2\}$	d_2	d_2	d_1
$x_2^*(\ell)$	d_t	$\max\{d_1 - d_2, d_{t+1}\}$	$\max\{\lceil d_1/2\rceil, d_t\}$	$\max\{\lceil d_{t+1}/2\rceil, d_{t+2}\}$

	B_5	B_6	B_7	B_8	B_9	B_{10}	B_{11}	B_{12}	B_{13}	B_{14}	B_{15}	B_{16}	B_{17}	B_{18}
1	3 0	2 1	1 2	0 3	2 0	2 0	2 0	1 1	1 1	0 2	2 0	1 1	1 0	1 0
2	1 0	1 0	1 0	1 0	2 0	1 1	0 2	1 1	0 2	0 2	1 0	1 0	1 0	1 0
3	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0
⋮							⋮							
$t-2$	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0
$t-1$	0 1	1 0	1 0	1 0	0 1	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0	1 0
t	0 1	0 1	1 0	1 0	0 1	0 1	1 0	1 0	1 0	1 0	0 2	1 0	1 0	1 0
$t+1$	0 1	0 1	0 1	1 0	0 1	0 1	0 1	0 1	1 0	1 0	0 1	0 2	0 3	0 2
$t+2$	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	1 0	0 1	0 1	0 1	0 2
$t+3$	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1
⋮							⋮							
m	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1	0 1

ℓ	$x_1^*(\ell)$	$x_2^*(\ell)$
5	$\max\{\lceil d_1/3\rceil, d_2\}$	d_{t-1}
6	$\max\{d_2, (d_1 - d_t)/2\}$	d_t
7	d_2	$\max\{d_{t+1}, (d_1 - d_2)/2\}$
8	d_2	$\max\{\lceil d_1/3\rceil, d_{t+2}\}$
9	$\max\{\lceil d_1/2\rceil, d_3\}$	d_{t-1}
10	$\max\{\lceil d_1/2\rceil, d_2 - d_t, d_3\}$	d_t
11	$\max\{\lceil d_1/2\rceil, d_3\}$	$\max\{\lceil d_2/2\rceil, d_{t+1}\}$
12	d_3	$\max\{d_1 - d_3, d_{t+1}\}$
13	d_3	$\max\{d_1 - d_3, \lceil d_2/2\rceil, d_{t+2}\}$
14	d_3	$\max\{\lceil d_1/2\rceil, d_{t+3}\}$
15	$\max\{\lceil d_1/2\rceil, d_2\}$	$\max\{\lceil d_t/2\rceil, d_{t+1}\}$
16	d_2	$\max\{d_1 - d_2, \lceil d_{t+1}/2\rceil, d_{t+2}\}$
17	d_1	$\max\{\lceil d_{t+1}/3\rceil, d_{t+2}\}$
18	d_1	$\max\{\lceil d_{t+1}/2\rceil, d_{t+3}\}$

grids forming it, respectively, and $\delta(S) = \max_i \delta_i(S)$, where $\delta_i(S) \geq 0$ is the waste of cover i, for $i = 1, \ldots, m$. Also, let $\Theta(\hat{e}) = \{S \in \Theta \mid \delta(S) \leq \hat{e}\}$ for any given integer $\hat{e}$. Without loss of generality $d_1 \geq \cdots \geq d_m$ is assumed throughout. Let $T = \{1, \ldots, t\}$ be the set of grid sites where plates can be accommodated. For clarity sake in the following algorithms' description we omit unnecessary technical details.

Algorithm $\mathcal{H}$ below heuristically produces a solution $S^* \in \Theta$ initialized as the solution S°, which is formed with m grids, where grid i is composed by t plates of cover i and $x_i = \lceil d_i/t\rceil$, for $i = 1, \ldots, m$. Also, $\hat{e}$ is the maximum paper wastage allowed for any cover; initialized as d_1 the value of $\hat{e}$ is gradually reduced to zero at every iteration of the outer loop.

<u>Algorithm $\mathcal{H}$</u>
$S^* \leftarrow S^\circ$; $\hat{e} \leftarrow d_1$;
While $\Theta(\hat{e}) \neq \emptyset$ **do**
(*) set $S \leftarrow \min\{S_1, S_2\}$, where S_1 (respectively, S_2) is the heuristic solution to $\min_{S \in \Theta(\hat{e})} \pi(S)$ obtained with Algorithm $\mathcal{H}1$ (respectively, $\mathcal{H}2$);
While there are two grids in S such that:
when excluded from S the number of covers with unsatisfied demand is in the range $[2\,t - 2,\ 2\,t]$, and
when replaced by the grids obtained through Algorithm $\mathcal{F}$ of Section 3.2 a solution $\bar{S}$ arises with $C(\bar{S}) < C(S)$ **do**
$S \leftarrow \bar{S}$
EndWhile;

If $(C(S) < C(S^*))$ **then** $S^* \leftarrow S$;
$\hat{e} \leftarrow \delta(S) - 1$;
EndWhile.

The inner loop of Algorithm $\mathcal{H}$ makes local improvements to solution S through Algorithm $\mathcal{F}$, whenever possible. To perform instruction $(*)$, the core of $\mathcal{H}$, we propose the heuristic procedures $\mathcal{H}1$ and $\mathcal{H}2$ below.

For any given value of $\hat{e}$, Algorithm $\mathcal{H}1$ constructs a solution $S_1 \in \Theta(\hat{e})$ with, say, γ grids, where grid j is to be printed h_j times, for $j = 1, \ldots, \gamma$, with a strategy that aims to minimize γ. Instructions 3 to 15 compose the j-th grid and determine h_j by considering the updated remaining demand $e_1, \ldots, e_m$, once it is assumed that the composed grids $1, \ldots, j-1$ have been printed $h_1, \ldots, h_{j-1}$ times, respectively. This is done in two steps.

The first step computes h_j (instructions 4, 5) as the maximum number of imprints that any possible grid can produce such that the paper wastage (if any) of each cover does not exceed $\hat{e}$; namely, h_j is the optimal solution value of

$$\begin{array}{ll} \max & \xi \\ \text{subject to} & a_i \xi \le e_i + \hat{e} \quad (i = 1, \ldots, m) \\ & \displaystyle\sum_{i=1}^{m} a_i = t \\ & \xi,\ a_i \ge 0 \quad \text{and integer} \end{array}$$

where a_i, for $i = 1, \ldots, m$, are variables too. The second step composes a grid which, once printed ξ times, yields with the aid of Algorithm $\mathcal{L}$ below a maximum number of covers whose remaining demand is completely satisfied (instructions 7 to 10), and at the same time aims to level the remaining demand (instructions 11 to 15). Along the whole process the remaining demand is continuously updated within vector $(e_1, \ldots, e_m)$.

<u>ALGORITHM $\mathcal{H}1$</u>

(1) $(e_1, \ldots, e_m) \leftarrow (d_1, \ldots, d_m)$; $j \leftarrow 0$;
(2) **Repeat**
(3) $j \leftarrow j + 1$; set $\Phi \leftarrow \{ i \in M : e_i > 0 \}$;
(4) find $(i^*, k^*) \in \Phi \times T$ such that
$\lceil e_{i^*}/k^* \rceil = \max_{(i,k) \in \Phi \times T} \{ \lceil e_i/k \rceil : \sum_{v \in \Phi} \lfloor (e_v + \hat{e})/\lceil e_i/k \rceil \rfloor \ge t \}$;
(5) $h_j \leftarrow \lceil e_{i^*}/k^* \rceil$;
(6) $r \leftarrow i^*$; $\theta \leftarrow t$;
(7) **While** $r \le m$ **and** $\theta > 0$ **do**
(8) call Algorithm $\mathcal{L}$;
(9) $r \leftarrow r + 1$;
(10) **EndWhile**;
(11) **While** $\theta > 0$ **do**
(12) find an index $s \in \Phi$ such that $e_s = \max_{i \in \Phi} \{e_i\}$;
(13) put one plate of cover s in grid j;
(14) $e_s \leftarrow e_s - h_j$; $\theta \leftarrow \theta - 1$;
(15) **EndWhile**;
(16) sort vector $(e_1, \ldots, e_m)$ in non increasing order, and
re-index the covers accordingly;
(17) **Until** $e_i \le 0$ for $i = 1, \ldots, m$;
(18) save in S_1 the solution found;

<u>ALGORITHM $\mathcal{L}$</u>

Put $\mu = \min\{ \lfloor \frac{e_r + \hat{e}}{h_j} \rfloor, \theta \}$ plates of cover r in grid j;
$e_r \leftarrow e_r - \mu h_j$; $\theta \leftarrow \theta - \mu$.

When in $\mathcal{H}1$ we replace S_1 with S_2, and instructions 7–15 are replaced with

call Algorithm $\mathcal{L}$;
$r \leftarrow 1$;
While $r \le m$ **and** $\theta > 0$ **do**
call Algorithm $\mathcal{L}$;
$r \leftarrow r + 1$;
EndWhile;

procedure $\mathcal{H}2$ arises which, considering the covers in non increasing order of their remaining demand, simply puts in each grid as many plates as possible. The outer loop of heuristics $\mathcal{H}$ and $\mathcal{H}1$ is performed at most d_1 and m times, respectively. Instruction 4 of $\mathcal{H}1$ takes at most $m^2 t \log(mt)$ time. The total number of times that the two inner loops of $\mathcal{H}1$ are performed is at most m. Instruction 12 of $\mathcal{H}1$ takes $\log(mt)$ time. In regard to heuristic $\mathcal{H}2$, a similar analysis of time can be made. Thus, it is not difficult to see that an efficient implementation of $\mathcal{H}$ yields $O(d_1 m^3 t \log(mt))$ as its computational complexity. On the other hand, in our experiments we have observed that, in general, the average required computer time is very satisfactory (see Section 4.3).

4. Numerical results

The procedures described in Section 3 were implemented on a computer with Xeon 3.4 GHz processor, 2 GB RAM, and Microsoft Visual Studio 2005 compiler. To investigate the efficiency of algorithms $\mathcal{E}$ and $\mathcal{H}$ of Sections 3.2 and 3.4, respectively, we conducted three experiments.

In the first experiment – see Section 4.1 – we tested our algorithms on every available instance considered elsewhere, and on one large instance randomly generated by us. Algorithm $\mathcal{E}$ was applied to eight small instances ($m \leq 15$) obtaining their optimal solutions, most of them having been previously found. Algorithm $\mathcal{H}$ was used on instances whose size made it impractical to apply Algorithm $\mathcal{E}$; when compared with the best previous results of 79 instances its solutions yielded lower cost in 76 of them, equal in one, and higher in only two instances. Moreover, we applied our approach to instances where no grid cost is provided, and instead of looking to minimize cost it is sought to minimize paper wastage when the number of grids is fixed; Algorithm $\mathcal{H}$ improved on the solution of the six considered cases for $m = 18$ and 22. Unfortunately, we could not test our procedures on the 32 real-world instances solved in [4], for their corresponding data were not published.

In the second experiment Algorithm $\mathcal{H}$ was applied to 60 instances constructed by us for which we could previously establish true global optima as explained in Section 4.2. When the output of Algorithm $\mathcal{H}$ was compared with these known optima it yielded errors from zero to 8.5%, with an overall average error of 3.9%.

Finally, the third experiment was designed to evaluate the performance of Algorithm $\mathcal{H}$ from the point of view of required computer time. We did extensive testing on randomly generated instances of varying size; the results are presented in Section 4.3.

The data for all instances, as well as the best known results and their source can be found in the website www.matcuer.unam.mx/~davidr/cpp.html.

4.1. Testing on specific instances

We started our experiments with instances **I001**–**I006**, named **P1**–**P6** in [12], respectively. For **I001**–**I004**, proposed by Teghem et al. [10] with $m = 3, 4, 5, 8$, Algorithm $\mathcal{E}$ found the true global minima that had been obtained as such in [12], each in less than three seconds of CPU time. With 40 CPU minutes of this exact algorithm we could claim the global optimality of the best reported solutions [12] of **I005** (proposed in [9] with $m = 12$), and **I006** (proposed in [11] with $m = 15$). Besides, heuristic $\mathcal{H}$ was also able to find the optimum of **I006**.

Proceeding further, we considered the ten instances **I007–I016** shown in Tables 2 and 3. Instances **I007–I009** correspond to real world situations [15]; instances **I010** [14] and **I011–I012** [13] slightly differ from the others as no grid cost is provided, and instead of looking to minimize cost it is sought to minimize paper wastage when the number of grids is fixed; **I013**–**I015** were proposed by Tuyttens and Vandaele [12] as **P7**–**P9**, respectively; finally, **I016** reflects a typical situation in a Mexican printing shop, where cover demand was randomly generated by us with uniform distribution.

Our results for **I007**–**I016** are displayed in Tables 4, 5, and Fig. 1. The best previous solutions for **I007**–**I008**, **I010**, **I011**–**I012**, and **I013**–**I015** come from [15,14,13], and [12], respectively. With Algorithm $\mathcal{E}$ and Algorithm $\mathcal{H}$ we obtained the optimal solution of **I007**, improving on previous results. Also, this exact procedure was applied to solve **I010** when the number of grids is fixed to *two* and *three*, allowing us to claim the optimality of the solution proposed in [14] for two grids, and yielding lower paper wastage than Mohan et al. [14] for three grids. On the other hand, Algorithm $\mathcal{H}$ was applied to solve **I008**–**I009** and **I011**–**I016**. For instances **I011**–**I012** we considered three cases in each, depending on the prescribed number of grids. Apart from **I013** this procedure improved on all previous solutions, although we offer no guarantee of global optimality. For instances **I009** and **I016** we had no others' results to compare with. In regard to the running time of Algorithm $\mathcal{H}$, it took 250 s for instance **I016**, and an average of 1.5 s for instances **I001** to **I015**, with a maximum of 3 s.

Finally, Algorithm $\mathcal{H}$ improved on previous results of 74 out of 75 instances (**T001**–**T075**) randomly generated and heuristically solved by Tuyttens and Vandaele [12] with $m = 30, 40, 50$, and five distinct grid costs.

(see www.matcuer.unam.mx/~davidr/coverprinting/Datasets.html.)

4.2. Testing on random instances with known global minimum

To further evaluate the quality of the solutions obtained by Algorithm $\mathcal{H}$ we tested it on 60 non trivial instances randomly generated by us (**E001**–**E060**), and whose true global optima could be previously determined.

Specifically, taking $m = 13$ and $t = 6$ as a first case (denote it [13, 6]) consider an instance of Problem P with some positive C_1 and C_2, and vector demand $D = A^T \times X$, where

$$A = \begin{pmatrix} 1\,0\,0 & 1\,0\,0 & 1\,0\,0 & 1\,1\,0 & 1 \\ 0\,1\,0 & 0\,1\,0 & 0\,1\,0 & 0\,1\,1 & 1 \\ 0\,0\,1 & 0\,0\,1 & 0\,0\,1 & 1\,0\,1 & 1 \end{pmatrix}$$

and X is an arbitrary, positive integer column 3-vector. Clearly, an optimal solution to P in this case is composed by matrix A^T and vector X because: (1) it yields zero wastage, and (2) from Remark 1 there is no feasible solution with less than $\lceil 13/6 \rceil = 3$ grids.

Table 2
Data of instances **I007**–**I012** and **I016**. Instances **I007–I009**, **I010**, and **I011–I012**, come from [15,14,13], respectively.

Instance **I007**		Instance **I008**				Instance **I009**			
$m = 9,\ t = 8$; $C_1 = 0.07,\ C_2 = 700$		$m = 17,\ t = 8$; $C_1 = 0.07,\ C_2 = 700$				$m = 18,\ t = 7$; $C_1 = 0.07,\ C_2 = 700$			
d_1	40 004	d_1	45 340	d_{10}	70 543	d_1	83 672	d_{10}	39 045
d_2	81 721	d_2	32 779	d_{11}	59 686	d_2	47 774	d_{11}	21 944
d_3	38 569	d_3	70 801	d_{12}	51 215	d_3	14 251	d_{12}	41 029
d_4	20 609	d_4	45 543	d_{13}	24 190	d_4	17 441	d_{13}	53 671
d_5	30 183	d_5	92 427	d_{14}	98 958	d_5	53 155	d_{14}	34 494
d_6	58 469	d_6	11 920	d_{15}	50 953	d_6	75 953	d_{15}	76 827
d_7	19 145	d_7	33 181	d_{16}	62 135	d_7	83 543	d_{16}	23 670
d_8	75 308	d_8	69 669	d_{17}	56 990	d_8	37 061	d_{17}	13 956
d_9	40 380	d_9	92 921	d_{18}		d_9	25 687	d_{18}	49 478

	Instance		
	I010	**I011**	**I012**
m	6	18	22
t	4	14	15
d_1	20 900	2200	600
d_2	21 000	200	700
d_3	23 700	500	2350
d_4	25 600	100	850
d_5	31 800	250	625
d_6	32 300	550	800
d_7		550	4100
d_8		550	850
d_9		2500	800
d_{10}		2450	1025
d_{11}		350	4050
d_{12}		1150	3300
d_{13}		3850	950
d_{14}		1400	1050
d_{15}		2700	5300
d_{16}		1400	3750
d_{17}		3050	6300
d_{18}		5550	6275
d_{19}			2275
d_{20}			3650
d_{21}			2650
d_{22}			4850

Instance **I016**							
$m = 100,\ t = 25,\ C_1 = 1,\ C_2 = 3\,000$							
i	d_i	i	d_i	i	d_i	i	d_i
1	36 547	26	54 394	51	89 425	76	18 800
2	80 425	27	57 847	52	76 785	77	35 898
3	39 381	28	89 961	53	76 552	78	78 002
4	79 320	29	66 892	54	35 401	79	18 089
5	48 363	30	79 183	55	78 345	80	83 863
6	41 787	31	39 345	56	44 204	81	17 809
7	83 482	32	13 900	57	78 032	82	49 055
8	46 624	33	41 644	58	30 935	83	90 389
9	15 175	34	69 520	59	58 240	84	73 193
10	32 613	35	71 594	60	35 742	85	23 338
11	54 878	36	36 214	61	97 577	86	45 286
12	97 767	37	41 004	62	86 333	87	83 108
13	34 822	38	42 163	63	70 465	88	91 194
14	57 218	39	92 595	64	62 379	89	69 619
15	82 188	40	63 077	65	12 112	90	59 283
16	97 270	41	60 844	66	76 195	91	21 522
17	77 661	42	11 310	67	40 002	92	97 951
18	27 727	43	99 346	68	37 733	93	33 684
19	14 386	44	29 716	69	70 105	94	63 074
20	77 071	45	21 833	70	52 178	95	55 161
21	96 270	46	90 881	71	75 791	96	33 036
22	76 255	47	77 269	72	58 319	97	57 311
23	70 613	48	33 349	73	17 042	98	64 867
24	98 401	49	56 869	74	87 728	99	73 833
25	46 031	50	93 518	75	92 716	100	78 716

To construct more instances of the cover printing problem with known optimal solution take [25, 8] as a second case, and reason as in case [13, 6] considering now the 4-by-25 matrix

$$\begin{pmatrix} 1\,0\,0\,0 & 1\,0\,0\,0 & 1\,0\,0\,0 & 1\,0\,0\,0 & 1\,0\,0\,0 & 1\,1\,0\,0 & 1 \\ 0\,1\,0\,0 & 0\,1\,0\,0 & 0\,1\,0\,0 & 0\,1\,0\,0 & 0\,1\,0\,0 & 0\,1\,1\,0 & 1 \\ 0\,0\,1\,0 & 0\,0\,1\,0 & 0\,0\,1\,0 & 0\,0\,1\,0 & 0\,0\,1\,0 & 0\,0\,1\,1 & 1 \\ 0\,0\,0\,1 & 0\,0\,0\,1 & 0\,0\,0\,1 & 0\,0\,0\,1 & 0\,0\,0\,1 & 1\,0\,0\,1 & 1 \end{pmatrix},$$

thus obtaining an optimal solution with four grids. Note that each matrix considered in the two described cases has $t/2$ rows and $t(t-2)/2+1$ columns, containing $t-3$ identity matrices of size $t/2$, one matrix with two 1's per row and column, and one column composed by ones. Continuing further, with the previous rationale we can build up matrices for $t = 10, 12, 14, 16$, to get cases [41, 10], [61, 12], [85, 14], and [113, 16], respectively, establishing for each an optimal solution to the cover printing problem with $t/2$ grids.

Our test consisted in applying Algorithm $\mathcal{H}$ to solve ten randomly generated instances (the entries of vector X were generated with uniform distribution in the range $[10\,000, 10\,000 + 2500 \times t]$), with $C_1 = 1$ and $C_2 = 3000$, for each of the six mentioned cases, yielding 60 instances, and then measuring the error of the obtained solutions when compared with the known optima. More precisely, letting z^*_{ij} (respectively, z^H_{ij}) denote the optimal solution value (respectively, the solution value obtained by Algorithm $\mathcal{H}$) that corresponds to the i-th instance generated for case j ($i = 1, \dots, 10;\ j = 1, \dots, 6$), we computed $\rho(i,j) = 100 \times (z^H_{ij} - z^*_{ij})/z^*_{ij}$, as well as $\rho_{\min}(j) = \min_{i=1,\dots,10}\{\rho(i,j)\}$, $\rho_{\max}(j) = \max_{i=1,\dots,10}\{\rho(i,j)\}$, and $\hat{\rho}(j) = \frac{1}{10}\sum_{i=1}^{10}\rho(i,j)$, for $j = 1, \dots, 6$. Our results are displayed in Table 6. We consider all these instances as difficult

Table 3
Data for instances **I013**–**I015** proposed in [12] as **P7**–**P9**, respectively.

Instance **I013** $m = 30,\ t = 4$ $C_1 = 13.44,\ C_2 = 18\,676$				Instance **I014** $m = 40,\ t = 4$ $C_1 = 13.44,\ C_2 = 18\,676$				Instance **I015** $m = 50,\ t = 4$ $C_1 = 13.44,\ C_2 = 18\,676$			
i	d_i	i	d_i	i	d_i	i	d_i	i	d_i	i	d_i
1	1 000	26	26 000	1	700	26	16 100	1	750	26	32 700
2	1 500	27	26 000	2	1 100	27	19 000	2	1 000	27	34 300
3	2 500	28	27 000	3	1 800	28	22 000	3	1 450	28	36 000
4	5 000	29	28 000	4	2 650	29	25 000	4	2 900	29	37 000
5	6 000	30	30 000	5	3 000	30	26 700	5	3 000	30	38 900
6	7 500			6	4 000	31	27 000	6	4 000	31	39 000
7	9 000			7	4 200	32	27 000	7	4 500	32	43 000
8	9 000			8	4 300	33	29 000	8	6 000	33	43 500
9	10 000			9	5 000	34	30 500	9	7 800	34	50 000
10	10 500			10	5 000	35	32 500	10	10 000	35	51 000
11	11 000			11	6 300	36	37 000	11	10 000	36	52 100
12	13 000			12	8 000	37	41 500	12	11 000	37	55 500
13	13 500			13	9 100	38	45 500	13	11 900	38	57 650
14	14 000			14	10 000	39	47 000	14	14 000	39	60 000
15	15 000			15	10 000	40	50 000	15	16 050	40	61 700
16	15 000			16	10 700			16	19 000	41	67 000
17	16 000			17	11 300			17	21 000	42	67 000
18	17 000			18	12 000			18	21 000	43	69 000
19	18 000			19	12 000			19	22 400	44	70 500
20	19 000			20	12 900			20	25 500	45	72 300
21	20 000			21	13 000			21	26 350	46	77 000
22	20 000			22	13 000			22	28 000	47	80 000
23	22 000			23	13 500			23	28 300	48	85 500
24	22 000			24	14 000			24	30 000	49	90 000
25	23 000			25	15 000			25	30 000	50	95 000

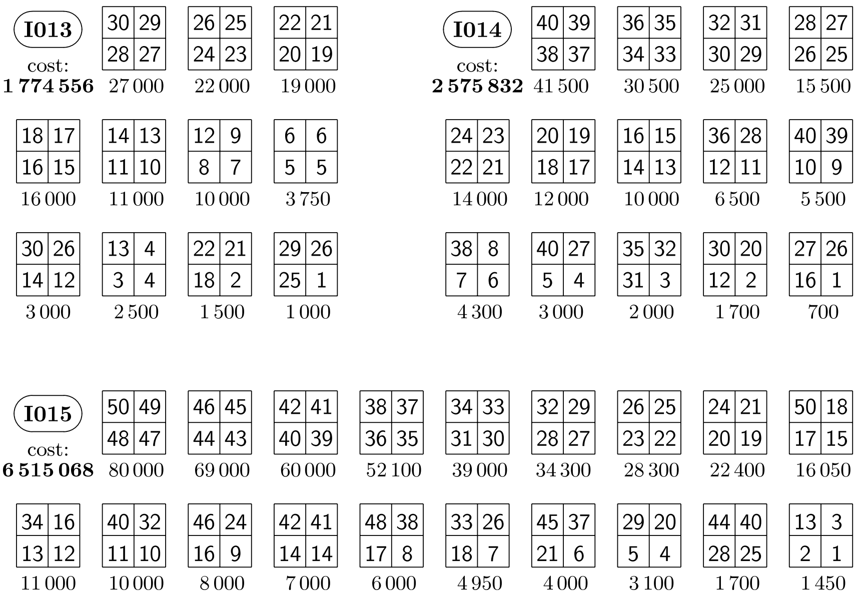

Fig. 1. Solution of Algorithm $\mathscr{H}$ for instances **I013**–**I015**, with 11, 14, and 19 grids, respectively.

because not only their optimal solutions yield zero paper wastage, and they do not have feasible solution with less than $t/2$ grids, but we could not devise a simple procedure to solve them.

4.3. Evaluating the needed computer time

For each of twelve selected combinations of $m \in \{10, 25, 50, 100\}$ and $t \in \{6, 8, 12, 16, 20, 25\}$, we created ten instances where the demand of each cover was randomly generated with uniform distribution in the range [10 000, 100 000]. Then we computed the running time of Algorithm $\mathscr{H}$ for each of these 120 instances (**R001**–**R120**), with $C_1 = 1$

Table 4
Solutions of Algorithm $\mathcal{H}$ for instances **I007–I012**. Daggers indicate the best previous results, which come from [15,14,13], for instances **I007–I008**, **I010**, and **I011–I012**, respectively.

Instance	Solution value	Grid	Imprints	Cover 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
I007	5283.53	1	40 861	1 2 1 0 1 0 0 2 1
	†5492.83	2	14 618	0 0 0 2 0 4 2 0 0
I008	11 475.52	1	70 801	0 0 1 0 1 0 0 1 1 1 1 0 0 0 0 1 1
	†12 295.22	2	51 215	1 1 0 1 0 0 1 0 0 0 0 1 0 2 1 0 0
		3	11 920	0 0 0 0 2 1 0 0 2 0 0 0 3 0 0 0 0
I009	11 191.60	1	53 671	0 1 0 0 1 1 1 0 0 0 0 0 1 0 1 0 0 1
		2	34 494	1 0 0 0 0 0 1 1 1 1 0 1 0 1 0 0 0 0
		3	24 589	2 0 0 1 0 1 0 0 0 0 1 0 0 0 1 1 0 0
		4	7 126	0 0 2 0 0 0 0 1 0 1 0 1 0 0 0 0 2 0
I010 with 3 grids	1.09%	1	21 000	1 1 0 0 1 1
		2	5 450	0 0 2 0 2 0
	†2.64%	3	12 800	0 0 1 2 0 1
I011 with 3 grids	5.119%	1	1400	1 0 0 0 0 0 0 0 1 1 0 0 2 0 2 1 2 4
		2	575	1 0 0 0 0 1 1 1 2 2 0 2 2 2 0 0 0 0
	†5.854%	3	225	1 1 3 1 2 0 0 0 0 0 2 0 0 2 0 0 2 0
I011 with 4 grids	0.771%	1	1 400	1 0 0 0 0 0 0 0 1 1 0 0 2 1 1 1 2 4
		2	550	1 0 0 0 0 1 1 1 2 2 0 2 2 0 2 0 0 0
	†3.243%	3	125	2 0 4 0 2 0 0 0 0 0 3 0 0 0 1 0 2 0
		4	034	0 6 0 3 0 0 0 0 0 0 0 2 0 0 3 0 0 0
I011 with 5 grids		1	1 388	1 0 0 0 0 0 0 0 1 1 0 0 2 1 1 1 2 4
	0.437%	2	537	1 0 0 0 0 1 1 1 2 2 0 2 2 0 2 0 0 0
		3	125	2 0 4 0 2 0 0 0 0 0 3 0 0 0 1 0 2 0
	†1.559%	4	40	0 5 0 3 0 0 0 0 1 0 0 2 0 0 3 0 0 0
		5	12	3 0 0 0 0 2 2 2 0 0 0 0 0 2 0 1 2 0
I012 with 3 grids	6.392%	1	2 650	0 0 1 0 0 0 1 0 0 0 1 1 0 0 2 1 2 2 0 1 1 2
		2	1 138	0 0 0 1 0 1 1 1 1 1 1 0 1 1 0 1 1 1 2 1 0 0
	†8.096%	3	262	3 3 0 0 3 0 2 0 0 0 1 3 0 0 0 0 0 0 0 0 0 0
I012 with 4 grids	2.452%	1	2 425	0 0 1 0 0 0 1 0 0 0 1 1 0 0 2 1 2 2 0 1 1 2
		2	950	0 0 0 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 2 1 0 0
	†4.235%	3	400	1 2 0 0 2 0 2 0 2 0 2 0 0 0 1 1 1 1 0 0 0 0
		4	125	2 0 0 0 0 0 0 0 0 1 0 0 0 1 1 0 1 1 3 3 2 0
I012 with 5 grids		1	2 350	0 0 1 0 0 0 1 0 0 0 1 1 0 0 2 1 2 2 0 1 1 2
	0.482%	2	875	0 0 0 1 0 0 2 1 0 1 1 1 1 1 0 1 1 1 2 1 0 0
		3	400	1 1 0 0 1 2 0 0 2 0 2 0 0 0 1 1 1 1 1 1 0 0
	†2.468%	4	150	1 2 0 0 1 0 0 0 0 1 0 0 0 1 1 0 2 2 1 0 2 1
		5	50	1 0 0 0 2 0 0 0 0 0 1 2 2 1 1 3 1 0 0 1 0 0

and $C_2 = 3000$, obtaining reasonable results. Table 7 shows the minimum, average, and maximum CPU time needed by Algorithm $\mathcal{H}$ when solving the ten instances for each selected combination.

5. Discussion

For the cover printing problem – in which the cost for producing plates is disregarded – we have proposed a mathematical programming formulation in Section 2, and several solution procedures in Section 3. These methods were tested with all specific instances known to us as well as with randomly generated instances of size up to $m = 113$. The results shown in Section 4 indicate a clear superiority of our approach over those proposed elsewhere, whenever we had data to compare with. We hope that our investigation will be an incentive to discover better discrete optimization techniques for this challenging problem.

One final word. In case we want to solve the cover printing problem taking into account the cost of plates we propose the following procedure: first use the methods of Section 3 to solve the problem without the cost of plates, and denote K the set of grids obtained. Then form a complete non directed graph G whose set of vertices corresponds to K, and for $x, y \in K$ the length of edge (x, y) is the number of plates that one would need to replace in grid x to obtain grid y. Finally, process the grids in K in the order $\tilde{b} = (b_1, b_2, \ldots, b_{|K|})$, where $\tilde{b}$ is a Hamiltonian path of minimum length in graph G. This procedure minimizes the number of required plates – and hence the cost – once the set of grids and number of imprints has been found. Although no polynomial algorithm is known to find an optimal Hamiltonian path, a branch-and-bound technique would yield a solution in small time for the typical instance size encountered in printing shops.

Table 5
Solution of Algorithm $\mathscr{H}$ for instance **I016**, yielding 8 grids, 2.40% wastage, and 265 918 total cost.

Grid	1	2	3	4	5	6	7	8
imprints	82 188	63 077	41 004	18 800	17 315	12 135	5540	1859

i	Grid 1 2 3 4 5 6 7 8	i	Grid 1 2 3 4 5 6 7 8	i	Grid 1 2 3 4 5 6 7 8	i	Grid 1 2 3 4 5 6 7 8
1	0 0 0 2 0 0 0 0	26	0 0 1 0 0 1 0 1	51	1 0 0 0 0 0 1 1	76	0 0 0 1 0 0 0 0
2	1 0 0 0 0 0 0 0	27	0 0 1 0 1 0 0 0	52	0 1 0 0 0 1 0 1	77	0 0 0 2 0 0 0 0
3	0 0 1 0 0 0 0 0	28	1 0 0 0 0 0 2 0	53	0 1 0 0 0 1 0 1	78	0 1 0 0 1 0 0 0
4	1 0 0 0 0 0 0 0	29	0 1 0 0 0 0 1 0	54	0 0 0 2 0 0 0 0	79	0 0 0 1 0 0 0 0
5	0 0 1 0 0 0 1 1	30	1 0 0 0 0 0 0 0	55	0 1 0 0 1 0 0 0	80	1 0 0 0 0 0 0 1
6	0 0 1 0 0 0 0 1	31	0 0 1 0 0 0 0 0	56	0 0 1 0 0 0 1 0	81	0 0 0 0 1 0 0 1
7	1 0 0 0 0 0 0 1	32	0 0 0 0 0 1 0 1	57	0 1 0 0 1 0 0 0	82	0 0 1 0 0 0 2 0
8	0 0 1 0 0 0 1 1	33	0 0 1 0 0 0 0 1	58	0 0 0 1 0 1 0 0	83	1 0 0 0 0 0 2 0
9	0 0 0 0 1 0 0 0	34	0 1 0 0 0 0 1 1	59	0 0 1 0 1 0 0 0	84	0 1 0 0 0 1 0 0
10	0 0 0 1 0 1 0 1	35	0 1 0 0 0 1 0 0	60	0 0 0 2 0 0 0 0	85	0 0 0 1 0 0 1 0
11	0 0 1 0 0 1 0 1	36	0 0 0 2 0 0 0 0	61	1 0 0 0 1 0 0 0	86	0 0 1 0 0 0 1 0
12	1 0 0 0 1 0 0 0	37	0 0 1 0 0 0 0 0	62	1 0 0 0 0 0 1 0	87	1 0 0 0 0 0 0 1
13	0 0 0 2 0 0 0 0	38	0 0 1 0 0 0 0 1	63	0 1 0 0 0 0 1 1	88	1 0 0 0 0 1 0 0
14	0 0 1 0 1 0 0 0	39	1 0 0 0 0 1 0 0	64	0 1 0 0 0 0 0 0	89	0 1 0 0 0 0 1 1
15	1 0 0 0 0 0 0 0	40	0 1 0 0 0 0 0 0	65	0 0 0 0 0 1 0 0	90	0 0 1 1 0 0 0 0
16	1 0 0 0 1 0 0 0	41	0 1 0 0 0 0 0 0	66	0 1 0 0 0 1 0 1	91	0 0 0 1 0 0 1 0
17	0 1 0 0 1 0 0 0	42	0 0 0 0 0 1 0 0	67	0 0 1 0 0 0 0 0	92	1 0 0 0 1 0 0 0
18	0 0 0 1 0 1 0 0	43	1 0 0 0 1 0 0 0	68	0 0 1 0 0 0 0 0	93	0 0 0 1 1 0 0 0
19	0 0 0 0 1 0 0 0	44	0 0 0 1 0 1 0 0	69	0 1 0 0 0 0 1 1	94	0 1 0 0 0 0 0 0
20	0 1 0 0 0 1 0 1	45	0 0 0 1 0 0 1 0	70	0 0 1 0 0 1 0 0	95	0 0 1 0 1 0 0 0
21	1 0 0 0 0 1 1 0	46	1 0 0 0 0 1 0 0	71	0 1 0 0 0 1 0 1	96	0 0 0 1 1 0 0 0
22	0 1 0 0 0 1 0 1	47	0 1 0 0 1 0 0 0	72	0 0 1 0 1 0 0 0	97	0 0 1 0 1 0 0 0
23	0 1 0 0 0 0 2 0	48	0 0 0 1 1 0 0 0	73	0 0 0 0 1 0 0 0	98	0 1 0 0 0 0 0 1
24	1 0 0 0 1 0 0 0	49	0 0 1 0 1 0 0 0	74	1 0 0 0 0 0 1 0	99	0 1 0 0 0 1 0 0
25	0 0 1 0 0 0 1 0	50	1 0 0 0 0 1 0 0	75	1 0 0 0 0 1 0 0	100	1 0 0 0 0 0 0 0

Table 6
Minimum, average, and maximum error—$\rho_{\min}(j)$, $\hat{\rho}(j)$, and $\rho_{\max}(j)$, respectively—of Algorithm $\mathscr{H}$ solutions with respect to the global optimum of ten random instances for cases $j = 1, \ldots, 6$. Figures in percent.

j	1	2	3	4	5	6
m	13	25	41	61	85	113
t	6	8	10	12	14	16
$\rho_{\min}(j)$	0.0	0.3	3.3	2.4	0.9	0.9
$\hat{\rho}(j)$	2.9	5.1	4.8	4.7	3.2	2.9
$\rho_{\max}(j)$	5.9	8.5	6.3	6.6	4.1	3.8

Table 7
Minimum, average, and maximum time (in seconds, rounded to nearest integer) required by Algorithm $\mathscr{H}$ to solve 10 random instances of 12 selected combinations of t and m.

m	10	10	25	25	25	50	50	50	100	100	100	100
t	6	8	8	12	16	12	16	20	12	16	20	25
min	0	0	0	0	0	9	10	13	87	136	166	210
avg	1	0	1	2	1	20	14	15	119	189	225	270
max	2	1	2	6	2	29	22	20	155	249	320	321

References

[1] L.E. Carrera, Algoritmo para encontrar el óptimo a una simplificación de un problema combinatorio presente en la industria editorial, Graduate Thesis, Mathematics Department, Cinvestav, Mexico City, 2006 (in Spanish).
[2] G. Wäscher, H. Haussner, H. Schumann, An improved typology of cutting and packing problems, European Journal of Operational Research 183 (2007) 1109–1130.
[3] Q.-S. Hua, Y. Wang, D. Yu, F.C.M. Lau, Dynamic programming based algorithms for set multicover and multiset multicover problems, Theoretical Computer Science 411 (2010) 2467–2474.
[4] A. Ekici, O. Ergun, P. Keskinocak, M.G. Lagoudakis, Optimal job splitting on a multi-slot machine with applications in the printing industry, Naval Research Logistics 57 (2010) 237–251.
[5] M. Balinski, Personal communication.
[6] M. Naya, Problème du mariage des couvertures posé par la S.A. Casterman, Graduate Thesis, Faculté Polytechnique de Mons, Belgium, 1990.
[7] C. Antoniadis, Problème du mariage des couvertures: résolution par la méthode du recuit simulé, Graduate Thesis, Université de Mons–Hainaut, Belgium, 1992.
[8] C. Carrein, Problème du mariage des couvertures par la méthode tabou, Graduate Thesis, Faculté Polytechnique de Mons, Belgium, 1994.
[9] G. Fasbender, A branch and price algorithm for the book cover printing problem, Graduate Thesis, Université Libre de Bruxelles, Belgium, 2000.

[10] J. Teghem, M. Pirlot, C. Antoniadis, Embedding of linear programming in a simulated annealing algorithm for solving a mixed integer production planning problem, Journal of Computational and Applied Mathematics 64 (1995) 91–102.
[11] S. Elaoud, J. Teghem, B. Bouaziz, Genetic algorithms to solve the cover printing problem, Computers and Operations Research 34 (2007) 3346–3361.
[12] D. Tuyttens, A. Vandaele, Using a greedy random adaptative search procedure to solve the cover printing problem, Computers and Operations Research 37 (2010) 640–648.
[13] K.F.C. Yiu, K.L. Mak, H.Y.K. Lau, A heuristic for the label printing problem, Computers and Operations Research 34 (2007) 2576–2588.
[14] S.R. Mohan, S.K. Neogy, A. Seth, N.K. Garg, S. Mittal, An optimization model to determine master designs and runs for advertisement printing, Journal of Mathematical Modelling and Algorithms 6 (2007) 259–271.
[15] L. Trilling, Personal communication.
[16] V. Klee, C. Witzgall, Facets and vertices of transportation polytopes, in: Mathematics of the Decision Sciences, American Mathematical Society, 1970, (Part 1).