INFORMS Journal on Computing
Vol. 22, No. 1, Winter 2010, pp. 131–143
ISSN 1091-9856 | EISSN 1526-5528 | 10 | 2201 | 0131

DOI 10.1287/ijoc.1090.0326

An Inexact Bundle Approach to Cutting-Stock Problems

Krzysztof C. Kiwiel
Systems Research Institute, 01-447 Warsaw, Poland, kiwiel@ibspan.waw.pl

We show that the linear programming relaxation of the cutting-stock problem can be solved efficiently by the recently proposed inexact bundle method. This method saves work by allowing inaccurate solutions to knapsack subproblems. With suitable rounding heuristics, our method solves almost all the cutting-stock instances from the literature.

Key words: nondifferentiable convex optimization; Lagrangian relaxation; integer programming; bundle methods; knapsack problems; cutting stock
History: Accepted by William Cook, former Area Editor for Design and Analysis of Algorithms; received April 2005; revised May 2006, October 2006, September 2008; accepted February 2009. Published online in *Articles in Advance* June 29, 2009.

1. Introduction

The classic Gilmore and Gomory (1961) formulation of the cutting-stock problem (CSP) is usually solved by linear programming (LP)-based column generation, rounding heuristics, and branch and bound; see, e.g., Belov and Scheithauer (2002, 2006), Degraeve and Peeters (2003), Degraeve and Schrage (1999), Vance (1998), and Vanderbeck (1999). Because column generation (CG) applied to its LP relaxation may converge slowly, there is interest in stabilized variants based on LP or quadratic programming (QP) (Ben Amor et al. 2004, Ben Amor and Valério de Carvalho 2005, Briant et al. 2007). Alternatively, the highly efficient hybrid approach of Degraeve and Peeters (2003) generates additional columns by applying subgradient optimization to its Lagrangian relaxation.

In this paper, we show that its LP relaxation can also be solved efficiently by the inexact bundle method of Kiwiel (2006a). This QP-based method saves work by allowing inaccurate solutions to Lagrangian subproblems. For the CSP, each subproblem is a knapsack problem (KP). We give a simple test for *inexact KP solutions* (see §2.2) that works well in practice for a standard branch-and-bound KP solver of Martello and Toth (1990). Furthermore, to avoid the difficulties arising when a bounded KP is transformed into a 0–1 KP (Vanderbeck 2002), we use *relaxed bounds*. Next, by adapting the ideas of Belov and Scheithauer (2002), Holthaus (2002), Stadtler (1990), and Wäscher and Gau (1996) to our inexact framework, we give rounding heuristics that solve *almost all* the CSP instances from the literature; in particular, they perform better than the best heuristics of Wäscher and Gau (1996). In effect, our inexact KP solutions, bound relaxation, and rounding heuristics should be of interest also for other, more traditional CG-based approaches to the CSP.

We now provide a historical perspective for our contributions. Our work was inspired by Briant et al. (2007), where (together with four other applications) the LP relaxation of the CSP was solved by several variants of CG and a standard bundle method. On some CSP instances, bundle was much slower than CG, mostly because its subproblems were more difficult for the KP solver of Vanderbeck (2002). Hence, Claude Lemaréchal suggested the CSP as a testing example for our inexact bundle (Kiwiel 2006a). For technical reasons, instead of the KP solver of Vanderbeck (2002), we used the MT1R procedure of Martello and Toth (1990). Our initial quite disappointing results improved greatly once we used relaxed KP bounds and inexact solutions: our method became much faster in practice than all the algorithms tested in (see §2.2 of Brian et al. 2007 and §5.8 of this paper). Next, we collected more test instances and adapted some rounding heuristics from the literature. The main aim was to appraise our inexact bundle solutions: they are deemed accurate enough if the heuristics solve almost all instances.

We now summarize our findings on admissible inexactness. The relative accuracy in dual-function evaluations is controlled by the tolerance ϵ_r of our KP solver (cf. §2.2). First, for $\epsilon_r = 0$ (i.e., exact bundle), the average computing times are much greater than those for $\epsilon_r = 10^{-5}$ (usually by factors of 30 or more), although the iteration numbers and the heuristic performance are almost the same. Second, the iteration numbers and timings are close for $\epsilon_r = 10^{-3}$, 10^{-4},

and 10^{-5}; however, relative to $\epsilon_r = 10^{-5}$, our heuristics perform much worse for $\epsilon_r = 10^{-3}$ and just marginally worse for $\epsilon_r = 10^{-4}$. Third, further experiments (not reported here for brevity) gave very close results for $\epsilon_r = 10^{-5}$, 10^{-6}, 10^{-7}, and 10^{-8}. In sum, $\epsilon_r = 10^{-5}$ seems to be a good borderline choice. On the other hand, because in the CSP the gap between the primal value and the relaxed dual value is usually less than 1, and either rounding heuristics or branch and bound should "close" this gap, it may seem more appropriate to ensure a given absolute accuracy $\epsilon_a < 1$ in dual-function evaluations (see §5.6.3). Quite suprisingly, our results for a fairly large $\epsilon_a = 0.01$ are very close to those for $\epsilon_r = 10^{-5}$, whereas for $\epsilon_a = 0.05$ our heuristics perform slightly worse.

We thus present the first successful application of our inexact bundle method. Our approach is also useful for the conic bundle variant of Kiwiel and Lemaréchal (2009).

The paper is organized as follows. In §2 we recall the classic CSP model of Gilmore and Gomory (1961) and introduce inexact KP solutions for its Lagrangian relaxation. Our rounding heuristics are given in §3 in a general form suitable for other CSP solvers. The inexact bundle method is reviewed in §4. Our computational results are presented in §5.

2. Lagrangian Relaxation of the CSP

The one-dimensional CSP is to minimize the number of stock pieces of width W used to meet the demands d_i for items to be cut at their widths $w_i \in (0, W]$, for $i = 1, \ldots, m$. The bin-packing problem (BPP) is a special case of the CSP with unit demands.

2.1. The Gilmore-Gomory Model

This classic model is formulated as follows. Denote the set of cutting patterns by

$$P := \{ p \in \mathbb{Z}_+^m \colon wp \le W \}. \tag{1}$$

Let z_p be the number of times pattern p is used. The original model has the form

$$\begin{aligned} \min \ & \sum_{p\in P} z_p \\ \text{s.t.} \ & \sum_{p\in P} p z_p \ge d, \quad z \in \mathbb{Z}_+^{|P|}. \end{aligned} \tag{2a}$$

For Lagrangian relaxation we augment this model with the redundant constraint

$$\sum_{p\in P} z_p \le N, \tag{2b}$$

where N is an upper bound on the optimal value of (2a) (e.g., $N = \sum_i d_i$); this ensures boundedness of the ground set $Z := \{z \in \mathbb{Z}_+^{|P|} \colon \sum_p z_p \le N\}$. Relaxing the demand constraint $\sum_p p z_p \ge d$ with a price vector u yields the Lagrangian $L(z; u) := \sum_p z_p + u(d - \sum_p p z_p)$ and the dual function

$$\theta(u) := \min_{z\in Z} \left\{ L(z; u) = ud + \sum_{p\in P} (1 - up) z_p \right\}. \tag{3}$$

The Lagrangian subproblem above may be solved by finding a solution $p(u)$ of the KP

$$\begin{aligned} p(u) \ &\in \ \operatorname{Arg\,max}\{up \colon p \in P\} \\ &= \operatorname{Arg\,max}\{up \colon wp \le W, \ p \in \mathbb{Z}_+^m\} \end{aligned} \tag{4}$$

and taking $z_{p(u)} = N$ and $z_p = 0$ for $p \ne p(u)$ if $up(u) > 1$, $z = 0$ otherwise, thus producing

$$\theta(u) = ud + N[1 - up(u)]_-, \tag{5}$$

where $[\cdot]_- := \min\{\cdot, 0\}$. Let v_* and v_{LP} denote the optimal values of (2) and its LP relaxation, respectively. It is well-known that v_{LP} coincides with the dual-optimal value

$$\theta_* := \max\{ \theta(u) \colon u \in \mathbb{R}_+^m \}. \tag{6}$$

Experiments show that $\check{u} := w/W$ is a good initial estimate of solutions to the Lagrangian dual (6) (see §4 of Ben Amor and Valério de Carvalho 2005, §2 of Briant et al. 2007). In fact, $\check{u}$ minimizes the relaxed dual function

$$\theta_{\mathrm{LP}}(u) := ud + N[1 - u\underline{p}(u)]_-, \tag{7}$$

where $\underline{p}(u)$ solves the LP relaxation of (4). (Since $\theta_{\mathrm{LP}}(\check{u}) = \check{u}d \le v_* \le N$, we see that $-d = -N(\check{u}d/N)(d/\check{u}d)$ is a subgradient of the second term of (7) at $\check{u}$: $0 \in \partial\theta_{\mathrm{LP}}(\check{u})$.)

2.2. Inexact KP Solutions

To strengthen our relaxation, we may consider only proper patterns p such that

$$p \le b \quad \text{with } b_i := \min\{d_i, \lfloor W/w_i \rfloor\}, \ i = 1\colon m. \tag{8}$$

Indeed, adding the bound $p \le b$ to (1) and (4) does not change v_*, but it may raise v_{LP} (Nitsche et al. 1999). Then the CG subproblem (4) becomes a bounded KP, which can be turned into a 0–1 KP via the transformation of Martello and Toth (1990, see §3.2). However, this transformation may duplicate solution representations, thus creating difficulties for 0–1 KP solvers (Vanderbeck 2002). To avoid duplicates, we may use the relaxed bound

$$p \le b' \quad \text{with } b'_i := 2^{\lceil \log_2(b_i+1) \rceil} - 1, \ i = 1\colon m, \tag{9}$$

which corresponds to replacing d_i in (8) by the smallest number of the form $2^j - 1$ with $j \ge 1$ such that $2^j - 1 \ge d_i$ ($2d_i - 1$ in the worst case); the number

of transformed variables is the same. We solve the transformed KP by a double-precision version of the branch-and-bound procedure MT1R of Martello and Toth (1990). To reduce its work, we allow MT1R to find an approximate solution for a given relative accuracy tolerance ϵ_r. Namely, the backtracking step exits if $\zeta \geq (1-\epsilon_r)\bar{\zeta}$, where $\zeta := up$ for the incumbent p and $\bar{\zeta}$ is MT1R's upper bound on the optimal value $up(u)$. Hence, by (5), we have the accuracy estimates

$$\underline{\theta}(u) := ud + N(1-\bar{\zeta})_- \leq \theta(u) \leq \bar{\theta}(u) := ud + N(1-\zeta)_-, \tag{10a}$$

$$\bar{\theta}(u) - \underline{\theta}(u) \leq N(\bar{\zeta}-\zeta) \leq N\epsilon_r\bar{\zeta}. \tag{10b}$$

For a normal exit with an optimal $p = p(u)$, we may replace $\bar{\zeta}$ by ζ and ϵ_r by 0 in (10).

As for our choice of MT1R, we add that Valério de Carvalho (2005) used MT1R as well, Belov and Scheithauer (2006) used a similar branch-and-bound solver, whereas Vanderbeck (1999) and Briant et al. (2007) used the more specialized branch-and-bound solver of Vanderbeck (2002). On the other hand, Degraeve and Peeters (2003) used a similar branch-and-bound solver but with prices multiplied by 10,000 and rounded to integers, without discussing the effects of inexact KP solutions. Furthermore, more-recent KP solvers (Kellerer et al. 2004) accept integer data only; hence, their use with suitable price roundings is left open for a future study. In sum, MT1R is outdated, but we could not find anything better, and we believe that the current results will serve as a useful yardstick for future work with modern KP solvers.

3. Heuristic Rounding of Relaxed Solutions

Typical rounding heuristics for the CSP proceed as follows (see Belov and Scheithauer 2002, 2006; Degraeve and Peeters 2003; Holthaus 2002; Scheithauer et al. 2001; Stadtler 1990; Wäscher and Gau 1996). A solution $\hat{z}$ of the LP relaxation is rounded down into an integer solution $\bar{z} := \lfloor \hat{z} \rfloor$. Next, a sequential heuristic applied to the residual problem (2) with d replaced by $d' := d - \sum_p p\bar{z}_p$ delivers a residual solution $\tilde{z}$. Then the sum $\bar{z} + \tilde{z}$ serves as a possibly inexact solution of (2) (which is exact if its value is equal to a lower bound on v_*, e.g., $\lceil v_{\mathrm{LP}} \rceil$). Because for simple rounding down ($\bar{z} = \lfloor \hat{z} \rfloor$) the residual problem may be too large to be solved optimally by a heuristic, some components of $\bar{z}$ may be increased (Holthaus 2002, Scheithauer et al. 2001); however, if the residual problem becomes too small to produce a solution to the original problem, some components of $\bar{z}$ may be decreased (Belov and Scheithauer 2002).

In §3.1 we give a general rounding procedure, which augments the ideas of Belov and Scheithauer (2002) and Holthaus (2002) with the oversupply reduction of Stadtler (1990). As for sequential heuristics, in §3.2 we describe minor (but useful) modifications of the first-fit-decreasing (FFD) of Chvátal (1983) and the heuristics of Belov and Scheithauer (2007) and Holthaus (2002). Because it pays to call lighter heuristics first, useful combinations of rounding and sequential heuristics are detailed in §3.3.

We add that the rounding procedures of Vanderbeck (1999, §3.7) and Wäscher and Gau (1996, see RSUC) would be difficult to implement in our context. As for sequential heuristics, we also tried the best-fit-decreasing of Chvátal (1983) and the fill-bin heuristics of Vanderbeck (1999), but they did not perform significantly better than FFD in our trials.

3.1. A General Rounding Procedure

Numbering the patterns so that $P = \{p^j\}_{j=1}^n$, we may write (2a) as

$$\begin{aligned} \min \ & \sum_{j=1}^n z_j \\ \text{s.t.} \ & \sum_{j=1}^n p^j z_j \geq d, \quad z \in \mathbb{Z}_+^n. \end{aligned} \tag{11}$$

Given an incumbent solution z^* of (11) (e.g., found by FFD) and a point $\hat{z} \in \mathbb{R}_+^n$ (e.g., found by LP relaxation), the following procedure attempts to improve z^* by calling a heuristic on residual problems derived from rounded variants of $\hat{z}$. Let $e := (1, \ldots, 1) \in \mathbb{R}^n$.

Procedure 1 (Rounding procedure).

Step 1 (*Rounding down*). Set $\bar{z} := \lfloor \hat{z} \rfloor$ and $d' := d - \sum_j p^j \bar{z}_j$. Sort the fractional parts $r_j := \hat{z}_j - \bar{z}_j$ so that $r_{j_1} \geq \cdots \geq r_{j_n}$, and set $\bar{n} := |\{j\colon r_j > 0\}|$.

Step 2 (*Oversupply reduction*). While $d' \not\geq 0$, pick $\bar{\jmath}$ to maximize

$$\sum_{i\colon d'_i < 0} w_i \min\{p_i^j, -d'_i\} \tag{12}$$

over j s.t. $\bar{z}_j > 0$, set $\bar{z}_{\bar{\jmath}} := \bar{z}_{\bar{\jmath}} - 1$ and $d' := d' + p^{\bar{\jmath}}$.

Step 3 (*Partial rounding up*). Set $I := \varnothing$. For $i = 1\colon \bar{n}$, if $p^{j_i} \leq d'$, set $\bar{z}_{j_i} = \bar{z}_{j_i} + 1$, $d' := d' - p^{j_i}$, $I := I \cup \{j_i\}$.

Step 4 (*Heuristic improvement*). Using a heuristic, find a feasible point $\tilde{z}$ for the residual problem (11) with d replaced by d'. If $e\bar{z} + e\tilde{z} < ez^*$, set $z^* := \bar{z} + \tilde{z}$.

Step 5 (*Residual problem extension*). If $I \neq \varnothing$, remove from I its last entry j, set $\bar{z}_j := \bar{z}_j - 1$, $d' := d' + p^j$ and return to Step 4.

If $\hat{z}$ solves the LP relaxation of an equality-constrained CSP, our procedure reduces to the one in Belov and Scheithauer (2002, §2.5); otherwise, Step 2 (as a result of Stadtler 1990, Figure 3) helps. Following Belov and Scheithauer (2002, §5.2), our implementation allows at most 10 returns from Step 5.

One of our heuristics uses the following modification of Step 3, based on the ideas in Holthaus (2002, §3.2).

Step 3′ (*Partial rounding up*). Set $I := \varnothing$, $K := \{j\colon p^j \le d',\ r_j > 0\}$. While $K \neq \varnothing$, pick $\bar{\jmath}$ to maximize $\sum_i p_i^j$ over $j \in K$, set $\bar{z}_{\bar{\jmath}} = \bar{z}_{\bar{\jmath}} + 1$, $d' := d' - p^{\bar{\jmath}}$, $I := I \cup \{\bar{\jmath}\}$, $K := \{j \in K\colon p^j \le d',\ j \neq \bar{\jmath}\}$.

3.2. Sequential Heuristics

We now describe our heuristics for the residual problem (2a) with d replaced by $d' \ge 0$. We assume that $w_1 \ge \cdots \ge w_m$.

Our implementation of FFD works as follows. Set $\tilde{z} := 0$, $d'' := d'$. While $d'' \neq 0$, generate the next pattern p by setting

$$p_i := \min\left\{d_i'',\ \left\lfloor \left(W - \sum_{j<i} w_j p_j\right) \Big/ w_i \right\rfloor\right\} \quad \text{for } i = 1\colon m, \tag{13}$$

set $\kappa := \min\{\lfloor d_i''/p_i \rfloor\colon p_i > 0\}$, $\tilde{z}_p := \tilde{z}_p + \kappa$, $d'' := d'' - \kappa p$. The version of Chvátal (1983, p. 208) uses $\kappa \equiv 1$ and hence is less efficient for large demands.

Our modification of the sequential heuristic procedure (SHP) of Holthaus (2002, §3.2), given a price vector $\hat{u} \in \mathbb{R}^m$ (e.g., an approximate solution of (6)) and a price tolerance $u_{\text{tol}} > 0$ for rounding errors (we use $u_{\text{tol}} = 10^{-12}$), sets $\bar{u}_i := \max\{\hat{u}_i, u_{\text{tol}}\}$ for $i = 1\colon m$ and replaces the FFD formula (13) by the bounded KP

$$p \in \operatorname{Arg\,max}\{\bar{u}p\colon wp \le W,\ p \le d'',\ p \in \mathbb{Z}_+^m\}. \tag{14}$$

Our implementation of the sequential value correction (SVC) heuristic of Belov and Scheithauer (2007, §2) records the best solution found by calling SHP at most 30 times with $\bar{u}$ modified as follows. Initially, $\bar{u}_i := \max\{1, W\hat{u}_i\}$, $i = 1\colon m$. If $wd'' \not\le W$, then after solving (14) and updating d'', for i such that $p_i > 0$, set

$$\bar{u}_i := [\gamma_i \bar{u}_i + (W/wp) w_i^{1.04}]/(\gamma_i + 1) \quad \text{with } \gamma_i := \Omega_i(d_i' + d_i'')/p_i \tag{15}$$

for Ω_i picked randomly in $[1/\Omega_i', \Omega_i']$, where Ω_i' is chosen at random in $[1, 1.5]$. An early exit occurs if SHP finds $\tilde{z}$ such that $e\bar{z} + e\tilde{z} = \lceil \theta(\hat{u}) \rceil$, in which case $z^* := \bar{z} + \tilde{z}$ is optimal.

3.3. Combinations of Rounding and Sequential Heuristics

We now give more details on the five heuristics used in our experiments. The heuristics are described as if being called by a general solver for the LP relaxation of (11), which could be any variant of the CG procedure or the bundle method given in §4.

Our initial heuristic H0 calls FFD with $d' = d$ (i.e., on the original problem) to initialize the incumbent $z^* := \tilde{z}$, the upper bound $N := ez^*$, and the lower bound $\underline{\theta}_1 := -\infty$.

Suppose that at iteration $k \ge 1$ of the solver, the following quantities are available: z^* is an incumbent solution of (11), $\hat{z}^k \in \mathbb{R}_+^n$ and $\hat{u}^k \in \mathbb{R}_+^m$ are tentative primal and dual solutions of the LP relaxation, and $\underline{\theta}_k$ is a lower bound on $\theta_* = v_{\text{LP}}$ (cf. (6)). If $ez^* = \lceil \underline{\theta}_k \rceil$, the solver may stop (because z^* is optimal). Otherwise, for iterations k specified below, the remaining heuristics consist of calling an extension of Procedure 1 with a copy of Step 4 inserted after Step 1; the sequential heuristics used at these steps are listed below.

Our periodic heuristic H1 is called by the solver every twentieth iteration, starting from iteration $k = m+1$ (i.e., for $k = m+1, m+21, \ldots,$), with the current relaxed solution $\hat{z} := \hat{z}^k$ and the lower bound $\underline{\theta}_k \le \theta_*$. H1 uses FFD in Procedure 1, exiting if $ez^* = \lceil \underline{\theta}_k \rceil$.

Our final heuristics H2–H4 are called successively upon termination of the solver, using the final $\hat{z} := \hat{z}^k$, $\hat{u} := \hat{u}^k$ and $\underline{\theta}_k$. H2 uses both FFD and SHP, H3 just SHP and the modified Step 3′, whereas H4 uses SVC. Of course, H3 and H4 (or just H4) are not called if H2 (or H3) exits with $ez^* = \lceil \underline{\theta}_k \rceil$, whereas SVC exits when $e\bar{z} + e\tilde{z} = \lceil \underline{\theta}_k \rceil$. The impact of the various heuristics will be discussed in §5.7.

4. The Inexact Proximal Bundle Method

We now sketch the main features of the inexact bundle method of Kiwiel (2006a).

Our method generates trial points $u^k \in \mathbb{R}_+^m$, $k = 1, 2, \ldots,$ at which the dual function θ is evaluated (possibly inexactly) as described in §2.2. Specifically, for each k, set p^k to the (possibly inaccurate) KP solution p satisfying the bounds of (10) for $u = u^k$ and let $\zeta_k := \zeta$, $\bar{\zeta}_k := \bar{\zeta}$. Recalling (3), define the associated Lagrangian solution z^k by setting

$$z_q^k := 0 \quad \text{for } q \neq p^k, \qquad z_{p^k}^k := \begin{cases} N & \text{if } \zeta_k > 1, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

Thus, we have the lower bound $\underline{\theta}(u^k) \le \theta(u^k)$ and $L(z^k; u^k) = \bar{\theta}(u)$ in (10); in particular,

$$L(z^k; u^k) - \theta(u^k) \le N(\bar{\zeta}_k - \zeta_k) \le N\epsilon_r \bar{\zeta}_k. \tag{17}$$

Furthermore, by (3), the following linearization of θ at u^k majorizes $\theta(u)$ for all u:

$$\theta_k(u) := L(z^k; u) = ud + \begin{cases} N(1 - up^k) & \text{if } z^k \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{18}$$

Iteration k uses the polyhedral cutting-plane model of θ

$$\hat{\theta}_k(\cdot) := \min_{j \in J^k} \theta_j(\cdot) \quad \text{with } k \in J^k \subset \{1, \ldots, k\} \tag{19}$$

for finding

$$u^{k+1} := \arg\max\left\{\hat{\theta}_k(u) - \frac{1}{2t_k}|u-\hat{u}^k|^2 \colon u \in \mathbb{R}^m_+\right\}, \quad (20)$$

where $t_k > 0$ is a stepsize that controls the size of $|u^{k+1} - \hat{u}^k|$ and the prox center $\hat{u}^k := u^{k'}$ has the value $\theta^k_{\hat{u}} := \theta_{k'}(u^{k'})$ for some $k' \le k$ (usually $\theta^k_{\hat{u}} = \max_{j=1}^k \theta_j(u^j)$). Because of evaluation errors, we may have $\theta^k_{\hat{u}} > \hat{\theta}_k(\hat{u}^k)$, in which case the *predicted increase*

$$v_k := \hat{\theta}_k(u^{k+1}) - \theta^k_{\hat{u}} \quad (21)$$

may be nonpositive; then t_k is increased and u^{k+1} is recomputed to increase $\hat{\theta}_k(u^{k+1})$ until $v_k \ge |u^{k+1} - \hat{u}^k|^2/2t_k$. An ascent step to $\hat{u}^{k+1} := u^{k+1}$ with $k' := k+1$ occurs if

$$\theta_{k+1}(u^{k+1}) - \theta^k_{\hat{u}} \ge \kappa v_k \quad (22)$$

for a fixed $\kappa \in (0,1)$ (we use $\kappa = 0.1$). Otherwise, a null step $\hat{u}^{k+1} := \hat{u}^k$ improves the next model $\hat{\theta}_{k+1}$ with the new linearization θ_{k+1} as stipulated in (19).

If we omitted the quadratic term in (20), the resulting cutting-plane method could generate u^{k+1} far from the previous points, and it would require storing all linearizations ($J^k = \{1,\dots,k\}$ in (19)). In contrast, the quadratic term usually keeps u^{k+1} close enough to the best point found so far, and it allows limiting the number of stored linearizations.

We solve subproblem (20) with the QP routine of Kiwiel (1994), which finds its multipliers $\{\nu^k_j\}_{j\in J^k} \subset \mathbb{R}_+$, also known as convex weights, such that $\sum_{j\in J^k}\nu^k_j = 1$ and the set $\hat{J}^k := \{j \in J^k \colon \nu^k_j \ne 0\}$ has at most $m+1$ elements. We set $J^{k+1} := J^k \cup \{k+1\}$ and then, if necessary, drop from J^{k+1} an index $j \in J^k \setminus \hat{J}^k$ with the largest $\theta_j(\hat{u}^k)$ to keep $|J^{k+1}| \le M$ for a fixed $M \ge m+2$.

Combining the accumulated Lagrangian solutions $\{z^j\}_{j\in J^k}$ with their weights $\{\nu^k_j\}_{j\in J^k}$, we may estimate solutions to the LP relaxation of (2) via the aggregate primal solution

$$\hat{z}^k := \sum_{j\in J^k} \nu^k_j z^j. \quad (23)$$

In other words (cf. (16)), $\hat{z}^k_{p^j} = N\nu^k_j$ for nontrivial patterns p^j indexed by $J^k_P := \{j \in \hat{J}^k \colon z^j \ne 0\}$ (which need not be stored, because they can be recovered from $\nabla\theta_j = d - Np^j$; see (18)). Our heuristics also use the lower bound $\underline{\theta}_k := \max_{j\le k}\underline{\theta}(u^j)$ on $\theta_* = v_{\mathrm{LP}}$ (cf. (6)).

We now point out some useful consequences of the convergence analysis in Kiwiel (2006a, §5). The LP relaxation of (2) may be written as

$$\begin{aligned} v_{\mathrm{LP}} := \min\ & \bar{\psi}_0(z) := \sum_{p\in P} z_p \\ \text{s.t.}\ & \bar{\psi}(z) := d - \sum_{p\in P} p z_p \le 0, \quad z \in \operatorname{conv} Z. \end{aligned} \quad (24)$$

Let $\epsilon := \sup_k[\theta_k(u^k) - \theta(u^k)]$ be the maximum evaluation error; by (17), we have $\epsilon \le \bar{\epsilon} := N\epsilon_r \sup_k \bar{\zeta}_k$. Consider the set of ϵ-optimal solutions of the LP relaxation (24):

$$Z_\epsilon := \{z \in \operatorname{conv} Z \colon \bar{\psi}_0(z) \le v_{\mathrm{LP}} + \epsilon,\ \bar{\psi}(z) \le 0\}. \quad (25)$$

The limits $\theta^\infty_{\hat{u}} := \lim_k \theta^k_{\hat{u}}$, $\underline{\theta}_\infty := \lim_k \underline{\theta}_k$ satisfy $\theta^\infty_{\hat{u}} \in [v_{\mathrm{LP}}, v_{\mathrm{LP}} + \epsilon]$, $\underline{\theta}_\infty \in [\theta^\infty_{\hat{u}} - \bar{\epsilon}, v_{\mathrm{LP}}]$, and there exists $K \subset \{1,2,\dots\}$ such that $\lim_{k\in K}\bar{\psi}_0(\hat{z}^k) = \theta^\infty_{\hat{u}}$ and $\overline{\lim}_{k\in K}\max_{i=1}^m \bar{\psi}_i(\hat{z}^k) \le 0$; in particular, the bounded sequence $\{\hat{z}^k\}_{k\in K}$ converges to the ϵ-optimal set Z_ϵ. If ϵ_r is small enough, the accuracy observed in practice corresponds to such estimates with ϵ and $\bar{\epsilon}$ determined by the maximum errors $\theta_k(u^k) - \theta(u^k)$ and $\theta(u^k) - \underline{\theta}(u^k)$ that occur for large k; because both errors are at most $N(\bar{\zeta}_k - \zeta_k)$, where the KP gap $\bar{\zeta}_k - \zeta_k$ is usually tiny for large k, small values of ϵ and $\bar{\epsilon}$ can be attained if the algorithm runs long enough.

We stop if $\min\{v_k, |\pi^k| + \alpha_k\} \le \epsilon_{\mathrm{opt}}(1 + |\theta^k_{\hat{u}}|)$, where v_k is given by (21), $\pi^k := (\hat{u}^k - u^{k+1})/t_k$, $\alpha_k := v_k - t_k|\pi^k|^2$ and $\epsilon_{\mathrm{opt}} > 0$ is an optimality tolerance (cf. Kiwiel 2006a, §4.2). For $\epsilon_{\mathrm{opt}} = \epsilon_r = 10^{-8}$, $\underline{\theta}_k$ usually agrees with θ_* in at least eight digits, enough for our purposes.

5. Computational Results

5.1. Data Sets

In our computational experiments, for the CSP we use the 28 industrial instances of Vance (1998), the 10 industrial instances of Vanderbeck (1999), and the 20 industrial instances of Degraeve and Schrage (1999). In addition, we use the following randomly generated instances: the 4,000 instances of Wäscher and Gau (1996), the 3,360 instances of Degraeve and Peeters (2003), and the 120 instances of Vanderbeck (1999). For the BPP, we use the 540 randomly generated instances of Degraeve and Peeters (2003) and the 160 instances from the BINPACK collection of the OR-Library (Beasley 1990).

The instances of Wäscher and Gau (1996) are constructed by the CUTGEN1 generator of Gau and Wäscher (1995), using the following parameter values: the number of orders $m = 10, 20, 30, 40, 50$; the width $W = 10{,}000$; the interval fraction $c = 0.25, 0.5, 0.75, 1$; and the average demand $\bar{d} = 10, 50$. The widths w_i are uniformly distributed integers between 1 and cW. For m uniform random numbers $R_1,\dots,R_m \in (0,1)$, the demands $d_i := \lfloor R_i m\bar{d}/(R_1 + \cdots + R_m)\rfloor$ for $i < m$, and $d_m := m\bar{d} - \sum_{i<m} d_i$ (in fact, slightly more complicated formulas are used by Gau and Wäscher 1995). Duplicate widths are aggregated by summing their demands. Combining the different values for m, c, and $\bar{d}$ results in 40 classes; in each class, 100 instances are generated.

The small-item-size instances of Degraeve and Peeters (2003) are generated similarly for $m = 10, 20, 30, 40, 50, 75, 100$; $c = 0.25, 0.5, 0.75, 1$; and $\bar{d} = 10, 50, 100$, except that $R_1, \ldots, R_m \in (0.1, 0.9)$ for the demand distribution. In the medium-item-size instances of Degraeve and Peeters (2003), only $\bar{d} = 50$ is used, and the widths are uniformly distributed on $[w_{\min}, cW]$, where $w_{\min} = 500, 1{,}000, 1{,}500$. Both cases have 84 data classes, and 20 instances are generated in each class.

The instances of Vanderbeck (1999) comprise six classes with $m = 50$, and 20 instances per class. The first three classes are generated like those of Wäscher and Gau (1996) above, with $c = 0.25, 0.5, 0.75$, and $\bar{d} = 50$, the next two classes have widths in $[500, 2{,}500]$ and $[500, 5{,}000]$ with $\bar{d} = 50$, and the sixth class has widths in $[500, 5{,}000]$ and $\bar{d} = 100$.

In the BPP instances of Degraeve and Peeters (2003), $m = 500$, or 1,000 weights are uniformly distributed in the intervals $[1, 100]$, $[20, 100]$, $[50, 100]$ as in BPPGEN (Schwerin and Wäscher 1997), and the capacity $W = 100, 120, 150$; identical items are aggregated for the corresponding CSPs. In each of the 18 resulting classes, 20 instances are generated. The modified BPP instances of Degraeve and Peeters (2003) use $m = 500$; the weight intervals $[1, 10{,}000]$, $[2{,}000, 10{,}000]$, $[5{,}000, 10{,}000]$; and the capacity $W = 10{,}000, 12{,}000, 15{,}000$, again with 20 instances per class.

The BINPACK instances from the OR-Library (Beasley 1990) comprise two categories. The uniform category has the capacity $W = 150$; m weights uniformly distributed in the interval $[20, 100]$; and 20 instances generated for each value of $m = 120, 250, 500, 1{,}000$. (The classes with $m = 500, 1{,}000$ also appear in the BPP category of Degraeve and Peeters 2003, but with different instances.) In the triplet category, each bin of capacity $W = 1{,}000$ is filled with exactly three items (the first item w' is picked in $[380, 490]$, the second item w'' in $[250, (W - w')/2)$, and the third item equals $W - w' - w''$). There are 20 instances for each value of $m = 60, 120, 249, 501$.

5.2. Implemented Variants

Our codes were programmed in Fortran 77 and run on a notebook PC (Pentium M 755 2 GHz, 1.5 GB RAM) under MS Windows XP.

For solving the dual problem (6), we used a general-purpose bundle code that treats subgradients as dense vectors in double precision. A faster code could exploit the fact that each subgradient of θ has the form $\nabla\theta_k = d$ or $\nabla\theta_k = d - Np^k$ (see (18)), with a common integer part d and an integer-sparse knapsack solution p^k. Ignoring sparsity, our code requires $m \times M$ memory locations for storing up to $M \geq m + 3$ subgradients, and additional workspace of order M^2 for solving the QP subproblem (20) with the routine of Kiwiel (1994). We used $M = m + 3$ to test how "minimal" bundle performs.

The bounded KPs arising in column generation and SHP were solved by the modified version of MT1R (cf. §2.2) with the accuracy tolerance $\epsilon_r = 10^{-5}$ (other choices are discussed in §5.6.2); MT1R's tolerance ϵ was set to 10^{-12}. For column generation, we used the relaxed bounds of (9) because the tighter bounds of (8) produced longer computing times. In contrast, SHP used in (14) the natural bounds given by (8) with d replaced by d''.

Our implementation of the rounding procedure of §3.1 is slower than necessary because the patterns are recovered as $p^j = (d - \nabla\theta_j)/N$, instead of being stored separately.

5.3. Results for the Cutting-Stock Problem

To ease comparisons, we closely follow the presentation of Degraeve and Peeters (2003). Every data class is identified by three parameters: the number of items m, the interval in which the widths are distributed denoted by *int*, and the average demand $\bar{d}$. An indicator "*all*" for any of these parameters means that the reported results are aggregated over all relevant values for that particular parameter. If a parameter is constant for all instances represented in a table, its value is indicated in the table heading.

Our results for the small-item-size instances of Degraeve and Peeters (2003) with $int = all$, $\bar{d} = all$ are reported in Table 1; full details are given in

Table 1 **Small-Item-Size Instances of Degraeve and Peeters (2003), $int = all$, $\bar{d} = all$**

m	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
10	9.99	26.77	15.14	31	0.00	0.01	113	49	70	1	0	0
20	19.95	53.13	32.51	69	0.01	0.04	120	64	64	0	0	0
30	29.91	79.76	51.90	91	0.02	0.22	130	85	57	0	0	1
40	39.85	105.55	70.41	134	0.04	0.36	134	98	53	0	0	0
50	49.75	132.16	90.20	181	0.08	0.66	134	102	55	0	0	0
75	74.36	197.32	141.82	256	0.24	2.00	149	122	43	0	0	0
100	98.92	263.36	183.88	311	0.40	2.81	165	136	34	0	1	0

Table 2 **Medium-Item-Size Instances of Degraeve and Peeters (2003), *int* = *all*, $\bar{d} = 50$**

m	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
10	9.98	23.09	17.52	29	0.00	0.02	54	48	112	0	1	0
20	19.95	45.58	35.05	58	0.01	0.10	68	50	114	0	1	0
30	29.84	68.47	53.71	93	0.02	0.16	73	73	105	0	0	0
40	39.78	90.65	69.94	120	0.03	0.58	70	63	110	0	0	0
50	49.64	113.69	88.76	156	0.06	0.90	74	65	118	1	1	1
75	74.08	169.10	137.04	232	0.37	8.60	82	73	105	0	0	1
100	98.45	226.07	184.45	295	1.43	62.18	73	72	117	0	4	0

Tables 9–11 in the Online Supplement (available at http://joc.pubs.informs.org/ecompanion.html). The columns m_{av} and m'_{av} give the average numbers of items and variables in the associated 0–1 knapsack subproblems. The columns i_{av} and i_{mx} report the average and maximum numbers of iterations of the bundle code. The columns t_{av} and t_{mx} give the average and maximum running times in wall-clock seconds. The column n_e lists the numbers of "early" terminations because of discovering that $ez^* = \lceil \underline{\theta}_k \rceil$ for the incumbent z^* delivered by H0 or H1 *before* bundle terminated on its own. Recall that H1 is called after H0, H2 after H1, etc., unless $ez^* = \lceil \underline{\theta}_k \rceil$ occurs earlier. The columns labelled H1 through H4 give the numbers of instances in which the corresponding heuristic found the best primal value ez^* first (for the remaining instances, ez^* was found by H0); a zero entry means that heuristic was not called or did not contribute usefully. The final column n_g reports the numbers of instances with a nonzero final *gap* $g := ez^* - \lceil \underline{\theta}_k \rceil$; we stress that the final gaps never exceeded *one* unit in *all* of our instances. The averages, maxima, and sums in Table 1 are taken over the 240 instances used for each value of m.

From the entries for n_e, H1 through H4 and n_g in Table 1, we see that early termination occured on between 47% and 69% of problems, H0 and H1 solved between 70% and 85% of problems, H2 solved almost all the remaining problems, H3 and H4 helped in solving two problems, and just one out of the 1,680 problems was not solved. Note that the best method LR of Degraeve and Peeters (2003) also could not solve one instance within 15 minutes (two instances within 6 minutes), and its FFD-based rounding heuristic solved 91.6% of problems, whereas our "lighter" heuristics H0 through H2 solved 99.8% of problems.

Our results for the medium-item-size instances of Degraeve and Peeters (2003) are presented in Table 2, where each row gives statistics over the 240 instances used for each value of m (see Tables 12 and 13 in the Online Supplement for more details). Early termination occured on between 22% and 35% of problems, H0 and H1 solved between 49% and 56% of problems, H2 solved almost all the remaining problems, H3 solved one problem, H4 solved 7 problems, and just 2 out of the 1,680 problems were not solved. The rounding heuristic of Degraeve and Peeters (2003) solved 69.9% of problems, whereas H0 through H2 solved 99.4% of problems.

Comparing Tables 1 and 2, we see that the average and maximum solution times are quite similar in the small- and medium-size-item cases for problem sizes m up to 50. However, for $m = 75$ and 100, in the medium-size-item case, the average solution times grow significantly, and the maximum solution times jump up, most spectacularly on the instances with width interval $[1{,}500, 2{,}500]$; see Table 13 in the Online Supplement. This is due to the poor performance of our knapsack solver on these instances. Similar slowdowns on this interval were reported in Degraeve and Peeters (2003, Table 4a) already for $m = 20$, i.e., even for smaller problems.

To save space, Table 3 presents only aggregate results on the instances of Wäscher and Gau (1996), with each row giving statistics over the 800 instances used for each value of m. Here, our main point is that only 3 out of 4,000 (0.075%) problems were not solved. Our "lighter" heuristics H0 through H2 solved 99.7% of problems, whereas the two best (and

Table 3 **CSP Instances of Wäscher and Gau (1996), *int* = *all*, $\bar{d}$ = *all***

m	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
10	9.99	25.37	14.27	35	0.00	0.02	449	134	192	0	0	0
20	19.96	50.46	30.73	61	0.01	8.35	485	240	183	0	2	0
30	29.90	75.72	48.18	105	0.01	0.13	503	281	161	0	1	0
40	39.84	100.10	65.06	123	0.04	3.31	502	313	160	0	2	2
50	49.73	125.22	84.75	171	0.07	0.46	526	341	138	0	4	1
all	29.88	75.37	48.60	171	0.03	8.35	2,465	1,309	834	0	9	3

Table 4 CSP Instances of Vanderbeck (1999), $m = 50$

$\bar{d}$	*int*	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
50	[1, 2,500]	49.40	185.30	47.40	71	0.03	0.05	20	18	0	0	0	0
50	[1, 5,000]	49.65	143.05	114.05	151	0.20	0.34	13	13	7	0	0	0
50	[1, 7,500]	49.75	110.00	111.85	144	0.06	0.11	6	5	8	0	0	0
50	[500, 2,500]	49.40	166.10	57.05	77	0.03	0.05	14	14	6	0	0	0
50	[500, 5,000]	49.70	128.20	103.65	114	0.14	0.27	11	11	9	0	0	0
100	[500, 5,000]	49.70	129.25	104.40	131	0.14	0.32	8	8	12	0	0	0

more complicated) heuristics RSUC and CSTAOPT of Wäscher and Gau (1996) solved 98.0% and 92.7% of problems, respectively (99.6% if they had been applied together). The fairly large maximum solution time in Table 3 stemmed from a single knapsack subproblem.

Table 4 gives our results for the six data classes of Vanderbeck (1999) with $m = 50$ and 20 instances per row. Because we used the original instances, the results are not identical to those in Tables 9 and 13 in the Online Supplement, but the performance of H0 through H2 is similar; in fact, H0 through H2 suffice for solving *all* the CSP instances used by Vanderbeck (1999).

Quite suprisingly, all the industrial instances we could find in the literature turned out to be easy for our method: they were solved in a fraction of a second (see Tables 14–16 in the Online Supplement).

5.4. Results for the Bin-Packing Problem

Following Degraeve and Peeters (2003), in the next three tables we present our results for the BPP. Table 5 gives our results for the BPP instances of Degraeve and Peeters (2003) (20 instances per row). All the 360 instances were solved (H4 helped once).

Table 6 reports results for the BINPACK instances from the OR-Library (Beasley 1990) (20 instances per row). The first four uniform classes were solved by calling H4 just once. However, only 19 out of the 80 triplet instances were solved (with H4 helping on one instance). The remaining instances had unit gaps; the "Gap (%)" column gives averages of *relative* gaps $(ez^* - \lceil \underline{\theta}_k \rceil)/\lceil \underline{\theta}_k \rceil$. We add that for the CSP instances of §5.3, the running times of H4 were not excessive, and H4 was called quite infrequently anyway. In contrast, on the triplet classes t249 and t501, the use of H4 increased the running times substantially, as illustrated in Table 7 (the influence of H3 could be ignored). Note that the triplet classes are quite difficult for traditional LP relaxation (Degraeve and Peeters 2003, Table 12) in the Online Supplement.

Table 8 presents our results for the modified BPP classes of Degraeve and Peeters (2003) (20 instances per row as described in §5.1). Just one out of the 180

Table 5 BPP Instances of Degraeve and Peeters (2003)

m	W	*int*	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
500	100	[1, 100]	99.35	167.20	184.10	221	0.06	0.09	12	1	1	0	0	0
		[20, 100]	80.75	116.00	111.50	123	0.02	0.03	10	2	0	0	0	0
		[50, 100]	51.00	52.00	56.60	63	0.00	0.01	15	0	0	0	0	0
	120	[1, 100]	99.65	181.85	37.05	195	0.29	3.79	17	1	0	0	0	0
		[20, 100]	80.85	131.20	132.80	146	0.03	0.04	14	6	0	0	0	0
		[50, 100]	51.00	62.00	56.55	61	0.00	0.01	13	0	0	0	0	0
	150	[1, 100]	99.45	201.55	1.00	1	0.00	0.00	20	0	0	0	0	0
		[20, 100]	80.85	151.65	86.55	102	0.01	0.02	14	14	5	0	1	0
		[50, 100]	51.00	77.00	64.80	72	0.01	0.01	12	0	0	0	0	0
1,000	100	[1, 100]	100.00	183.65	199.20	230	0.07	0.11	12	1	1	0	0	0
		[20, 100]	81.00	117.95	114.25	133	0.02	0.02	14	4	1	0	0	0
		[50, 100]	51.00	52.00	57.35	64	0.00	0.01	9	0	0	0	0	0
	120	[1, 100]	100.00	202.20	25.00	181	0.01	0.04	20	3	0	0	0	0
		[20, 100]	81.00	132.95	143.40	167	0.03	0.04	10	3	2	0	0	0
		[50, 100]	51.00	62.00	56.90	62	0.00	0.01	11	0	0	0	0	0
	150	[1, 100]	100.00	226.15	7.00	121	0.00	0.03	20	1	0	0	0	0
		[20, 100]	81.00	154.90	86.85	101	0.01	0.02	11	11	9	0	0	0
		[50, 100]	51.00	77.00	67.25	77	0.01	0.01	10	0	0	0	0	0

Table 6 BINPACK Uniform and Triplet Instances

Name	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	Gap (%)	n_g
u120	63.20	88.75	48.60	89	0.00	0.01	20	14	0	0	0	0.0	0
u250	77.25	129.00	86.40	122	0.01	0.03	19	19	1	0	0	0.0	0
u500	80.80	151.05	85.90	113	0.01	0.04	16	16	3	0	1	0.0	0
u1000	81.00	155.00	86.30	97	0.01	0.02	12	12	8	0	0	0.0	0
t60	49.95	58.80	40.20	56	0.01	0.04	0	1	19	0	0	1.5	6
t120	86.15	110.75	72.70	91	0.06	0.09	0	1	18	0	1	2.0	16
t249	140.10	199.15	126.70	146	0.26	0.37	0	1	19	0	0	1.2	20
t501	194.25	315.40	167.40	189	0.67	1.14	0	0	20	0	0	0.6	19

problems was not solved (H4 helped on one problem). The transformation into a CSP reduced the number of items by at most 5% on average. For almost 500 variables, the large iteration numbers and running times are not too suprising.

5.5. Impact of Tighter Knapsack Bounds

The results of §5.3 were obtained for the relaxed bounds of (9). Using the tighter bounds of (8) allowed us to solve just two more instances at the expense of longer running times (see Tables 17–19 in the Online Supplement). To save space, from now on we employ the standard set of the 7,360 instances from Tables 1–3 to evaluate our heuristics and its reduced subset with $m \geq 30$ (4,800 instances) for performance profiles (Dolan and Moré 2002), with zero running times replaced by 0.001 because of the poor resolution of our timer. The performance profile of tighter versus relaxed bounds is given in Figure 1; it plots the portion of instances $\rho_s(\tau)$ on which a particular variant was not slower than the fastest variant by more than a given ratio τ.

5.6. Impact of Evaluation Errors

5.6.1. Comparison with Exact Bundle. When the dual-objective evaluations happen to be exact, our code runs essentially like the standard bundle of Feltenmark and Kiwiel (2000). Figure 2 gives the performance profile of inexact bundle ($\epsilon_r = 10^{-5}$) with relaxed bounds versus exact bundle ($\epsilon_r = 0$) with relaxed or tighter bounds. Referring to Tables 22–27 in the Online Supplement for details, we only note that the running times increased quite dramatically (usually at least 30 times) in the exact case, although the iteration numbers and the performance of our heuristics did not change significantly.

5.6.2. Other Choices of the Relative Error Tolerance. In the initial version of this paper we used the accuracy tolerance $\epsilon_r = 10^{-8}$; the results were very close to those in Tables 1–18 (where $\epsilon_r = 10^{-5}$). Figure 3 gives the performance profile for $\epsilon_r = 10^{-5}$, 10^{-4}, and 10^{-3} (see also Tables 28–33 in the Online Supplement). Here, $\epsilon_r = 10^{-4}$ did not improve on our standard choice of $\epsilon_r = 10^{-5}$ (giving one more gap in Table 28 in the Online Supplement), whereas $\epsilon_r = 10^{-3}$ was too large, causing our heuristics to fail more frequently (168 more gaps in Tables 31–33 in the Online Supplement).

Further insight may be gained as follows. By (10), the absolute error in evaluating θ is bounded by $N\epsilon_r$ once $\bar{\zeta}$ gets close to 1. The upper bound $N := ez^*$ delivered by FFD (cf. §3.3) is usually close to the optimal primal value v_*. Typical instances have the integer round-up property $\lceil \theta_* \rceil = v_*$, but our heuristics fail if we can't find a lower bound $\underline{\theta}_k > v_* - 1$. Thus, we may expect failures when the absolute errors get close to $N\epsilon_r > 1$. Now, in Tables 31–33 in the Online Supplement the average values of v_* and N grow linearly with m, reaching order 5,000, 2,875, and 1,250 for the final classes, where $N\epsilon_r > 1$ for $\epsilon_r = 10^{-3}$; thus, the small percentage of failures suggests that the actual errors tended to be smaller than their upper bounds.

5.6.3. Absolute Error Tolerances. In view of the discussion in §5.6.2, we also considered choosing ϵ_r so that the evaluation errors did not exceed a given absolute error tolerance $\epsilon_a < 1$ (with SHP using $\epsilon_r = 10^{-5}$ as in §5.3). Specifically, for evaluating θ we used $\epsilon_r := \epsilon_a/N$. Figure 4 gives the performance profile for $\epsilon_a = 0.01$ and 0.05 versus the standard $\epsilon_r = 10^{-5}$ (see also Tables 34–39 in the Online Supplement). Our results for $\epsilon_a = 0.01$ were very close to those for $\epsilon_r = 10^{-5}$, whereas $\epsilon_a = 0.05$ was too large, causing our heuristics

Table 7 BINPACK Triplet Instances Without H3 and H4

Name	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	Gap (%)	n_g
t60	49.95	58.80	40.20	56	0.00	0.01	0	1	19	1.5	6
t120	86.15	110.75	72.70	91	0.01	0.02	0	1	19	2.1	17
t249	140.10	199.15	126.70	146	0.04	0.06	0	1	19	1.2	20
t501	194.25	315.40	167.40	189	0.08	0.10	0	0	20	0.6	19

Table 8 Modified BPP Instances of Degraeve and Peeters (2003)

W	*int*	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
10,000	[1, 10,000]	488.65	494.05	1,484.40	1,737	34.95	48.35	14	3	0	0	0	0
	[2,000, 10,000]	485.15	490.20	800.70	916	7.05	9.87	15	1	0	0	0	0
	[5,000, 10,000]	474.75	474.80	457.70	480	1.15	1.35	16	0	0	0	0	0
12,000	[1, 10,000]	486.95	494.55	817.90	1,732	25.89	58.02	18	7	1	0	0	0
	[2,000, 10,000]	484.75	492.20	1,157.90	1,328	15.00	21.33	18	2	0	0	0	0
	[5,000, 10,000]	475.95	480.35	520.75	550	2.20	2.64	15	0	0	0	0	0
15,000	[1, 10,000]	487.90	497.15	293.60	1,171	8.00	67.00	18	6	0	0	1	1
	[2,000, 10,000]	482.70	494.25	805.05	1,144	16.19	29.37	16	16	4	0	0	0
	[5,000, 10,000]	475.25	486.95	691.50	786	5.14	6.31	13	0	0	0	0	0

to fail more frequently (16 more gaps in Tables 37–39 in the Online Supplement).

5.6.4. More Inexact Null Steps. We now consider a modification in which our KP solver exits once at least `bkmin` backtrackings have occured, for a given parameter `bkmin`, and the incumbent value ζ satisfies

$$\zeta > 1 + (u^{k+1}d - \theta^k_{\hat{u}} - \kappa v_k)/N, \tag{26}$$

so that $\zeta_{k+1} := \zeta$ yields a null step; cf. (22) (normally, $u^{k+1}d > \theta^k_{\hat{u}} + \kappa v_k$ and (26) holds iff (22) fails). Such "more inexact" null steps may save KP work, but shallower cuts may yield slower convergence; see Kiwiel (2006b, §4.2) for a general discussion of relaxed null steps. Figure 5 gives the performance profile for `bkmin` $= 0$, 1,000, and ∞ with $\epsilon_r = 10^{-5}$ (see also Tables 40–45 in the Online Supplement). Relative to the standard `bkmin` $= \infty$, for `bkmin` $= 0$ the average iteration numbers grew by 59%–114% on the largest instances, and four more gaps occured. In contrast, for `bkmin` $= 1{,}000$ the average iteration numbers grew by only 5%–13% on the largest instances, the solution times decreased noticeably, and three gaps disappeared. On the other hand, the maximum iteration numbers increased substantially on the larger instances, giving some cause for concern.

5.6.5. A Discussion of Error Tolerances. Although in general one may expect trade-offs between the accuracy of subproblem solutions and the speed of convergence, for the CSP such trade-offs may have little practical impact, because Tables 9–30 (in the Online Supplement) exhibit fairly small variations in iteration numbers and computing times for "reasonable" accuracy tolerances. Therefore, we would not expect much gain from dynamic tolerance adjustment: loose at the beginning and progressively decreasing.

We add that dynamic handling of the accuracy may be important in general, especially if the oracle's work depends "continuously" on the accuracy required. However, this need not be the case for our MT1R, which seems to have the following properties: (1) its work explodes on some subproblems when the accuracy required is "too high," and (2) its work does not vary much otherwise. Thus, the main point is to

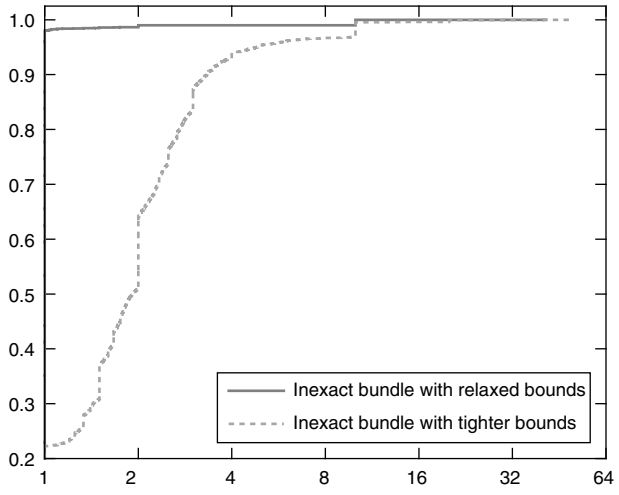

Figure 1 Performance Profile for Inexact Bundle with Tight vs. Relaxed Bounds

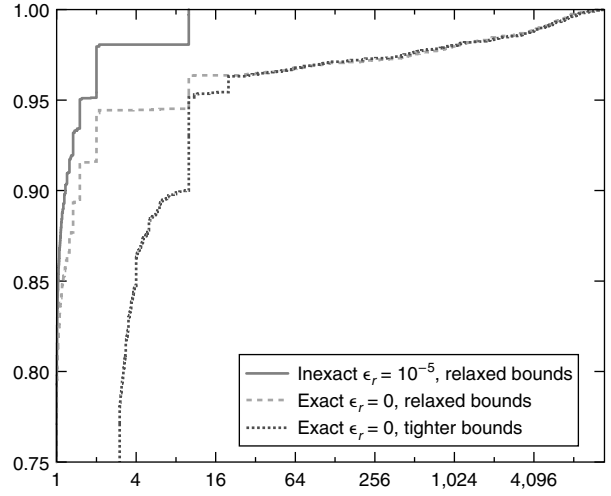

Figure 2 Performance Profile for Inexact Bundle with Relaxed Bounds vs. Exact Bundle with Tight/Relaxed Bounds

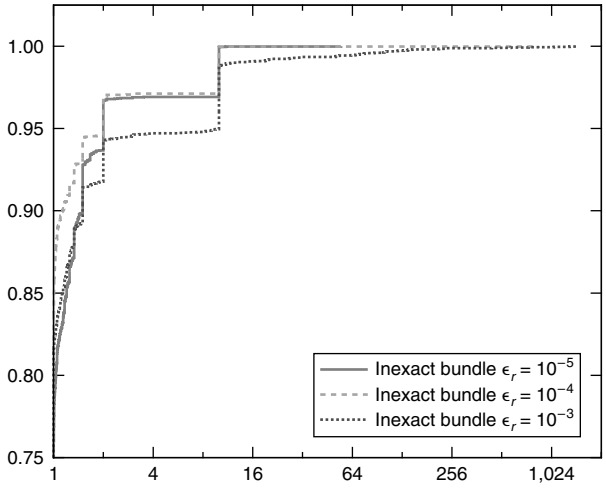

Figure 3 Performance Profile for Relative Error Tolerances

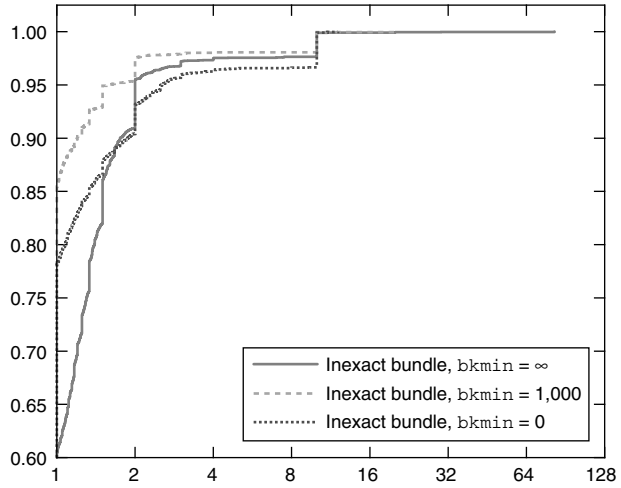

Figure 5 Performance Profile for `bkmin`

avoid accuracies that are "too high" or "too low" for the dual solver to succeed, whereas for all "intermediate" accuracies, the solution time should not vary significantly (unless smaller accuracies affect the iteration numbers "more than proportionally"). We conjecture that similar effects are likely to hold for other integer-programming applications with branch-and-bound oracles that deliver relatively good incumbents quickly.

5.7. Impact of Various Heuristics

For the 7,538 CSP instances reported in Tables 1–4 and 14–16, our heuristics H3 and H4 helped in solving 3 and 21 problems, respectively, and 6 problems were not solved. When H3 was switched off, H4 solved the three instances previously solved by H3 with the same timings. Thus, H3 could be omitted, but it might become more useful on other instances. On the other hand, it is worth observing that when both H3 and H4 were switched off, our "lighter" heuristics H1 and H2 performed quite well, solving 99.64% of problems.

In an attempt to assess the importance of the combination of oversupply reduction (Step 2 of Procedure 1), rounding up (Step 3), and residual problem extension (Step 5), we tested a version of the residual rounding heuristic named H5 that simply rounds the final relaxed primal solution down, and performs FFD on the residual problem to augment the rounded down solution. With Steps 2, 3, and 5 of the rounding procedure omitted, this heuristic H5 was able to optimally solve only 87.01% of the standard instances, as opposed to 99.64% for the default implementation of H0, H1, and H2 (see Tables 46–48 in the Online Supplement). Thus these steps (in tandem) are very important to its overall success.

Our next improvement on H5, named H6, consists in calling Procedure 1 with only Step 2 omitted, and Step 4 using FFD. H6 performs much better than H5, solving 96.00% of problems (see Tables 49–51 in the Online Supplement). Thus, the rounding procedure of Belov and Scheithauer (2002) may yield significant improvements also for FFD.

Finally, we note that H2 and H4 improve on H6 by using Step 2 of Procedure 1 and either SHP or SVC in addition to FFD at Step 4. Specifically, H1 and H2 solved 99.64% of problems, and together with H4 they solved 99.92% of problems. To save space, the results for H2 alone are omitted.

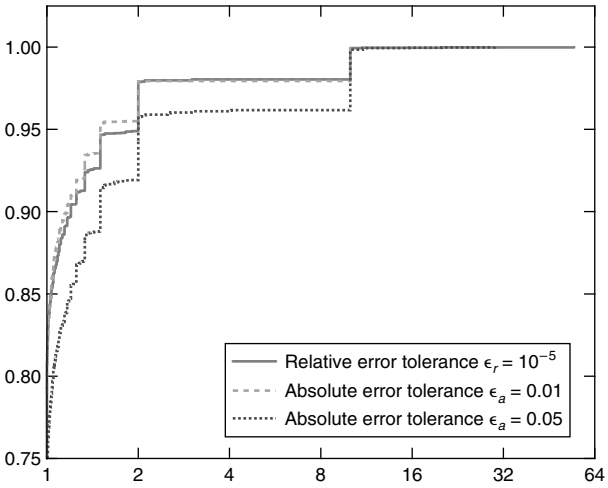

Figure 4 Performance Profile for Absolute Error Tolerances

5.8. Comparisons with Other Procedures from the Literature

In view of (5)–(6), our algorithm may be regarded as an exact penalty method for the constrained problem of maximizing ud s.t. $up(u) \le 1$, $u \ge 0$. This problem can also be solved by the conic variant of Kiwiel and Lemaréchal (2009). Figure 6 gives the performance profile for the conic versus penalty variant. The conic

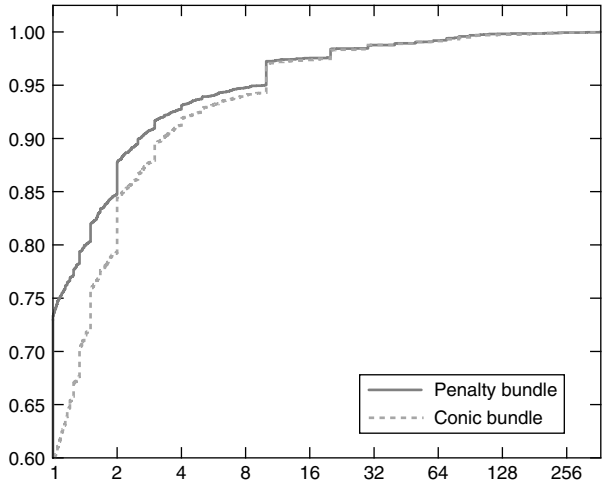

Figure 6 Performance Profile for Conic vs. Penalty Bundle

variant was slightly slower and gave one more gap on the standard set; cf. Tables 52–54 (in the Online Supplement).

The comparison in Kiwiel and Lemaréchal 2009, §7.4) of the conic variant with the procedures of Degraeve and Peeters (2003) in terms of the numbers of oracle calls carries over to the penalty variant as well, because both variants behaved similarly. Although proper timing comparisons are not available, Table 55 (in the Online Supplement), suggests that our code may compete with the procedures of Degraeve and Peeters (2003), at least on some instances.

Finally, we add that Table 60 (in the Online Supplement) shows that our standard variant (with $\epsilon_r = 10^{-5}$ and relaxed bounds) is much faster than the algorithms tested in Briant et al. (2007, §2.2), with speedups of at least 8 for the smallest instances, and of order 11–90 for the larger instances.

6. Conclusions

For cutting-stock problems, we have shown that an inexact bundle approach to solving the LP relaxation, coupled with rounding heuristics, is a method that is able to effectively solve many cutting-stock instances from the literature. By solving the KP subproblems only to a relative accuracy of $\epsilon_r = 10^{-5}$ we get (almost uniformly) speedup of the order of at least 30 on average in larger instances. Although our heuristics combine several well-known ideas from the literature, our two "lighter" heuristics, H1 and H2, performed suprisingly well, solving 99.64% of standard test problems, and together with our "heavier" heuristic, H4, they solved 99.92% of problems.

Acknowledgments

The author would like to thank the associate editor and the two anonymous referees for helpful comments. Further, the author is grateful to G. Belov, Z. Degraeve, M. Peeters, D. Pisinger, G. Scheithauer, and L. Schrage for extensive discussions, and F. Vanderbeck and G. Wäscher for sharing their instances. Special thanks go to C. Lemaréchal for inspiring this work. This research was supported by the INRIA New Investigation Grant "Convex Optimization and Dantzig-Wolfe Decomposition."

References

Beasley, J. E. 1990. OR-Library: Distributing test problems by electronic mail. *J. Oper. Res. Soc.* **41**(11) 1069–1072.

Belov, G., G. Scheithauer. 2002. A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. *Eur. J. Oper. Res.* **141**(2) 274–294.

Belov, G., G. Scheithauer. 2006. A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting. *Eur. J. Oper. Res.* **171**(1) 85–106.

Belov, G., G. Scheithauer. 2007. Setup and open stacks minimization in one-dimensional stock cutting. *INFORMS J. Comput.* **19**(1) 27–35.

Ben Amor, H., J. M. Valério de Carvalho. 2005. Cutting stock problems. G. Desaulniers, J. Desrosiers, M. M. Salomon, eds. *Column Generation*. Springer, New York, 131–161.

Ben Amor, H., J. Desrosiers, A. Frangioni. 2004. Stabilization in column generation. Technical Report G-2004-62, GERAD, Montréal.

Briant, O., C. Lemaréchal, P. Meurdesoif, S. Michel, N. Perrot, F. Vanderbeck. 2007. Comparison of bundle and classical column generation. *Math. Programming* **113**(2) 299–344.

Chvátal, V. 1983. *Linear Programming*. Freeman, New York.

Degraeve, Z., M. Peeters. 2003. Optimal integer solutions to industrial cutting-stock problems: Part 2, benchmark results. *INFORMS J. Comput.* **15**(1) 58–81.

Degraeve, Z., L. Schrage. 1999. Optimal integer solutions to industrial cutting stock problems. *INFORMS J. Comput.* **11**(4) 406–419.

Dolan, E. D., J. J. Moré. 2002. Benchmarking optimization software with performance profiles. *Math. Programming* **91**(2) 201–213.

Feltenmark, S., K. C. Kiwiel. 2000. Dual applications of proximal bundle methods, including Lagrangian relaxation of nonconvex problems. *SIAM J. Optim.* **10**(3) 697–721.

Gau, T., G. Wäscher. 1995. CUTGEN1: A problem generator for the standard one-dimensional cutting stock problem. *Eur. J. Oper. Res.* **84**(3) 572–579.

Gilmore, P. C., R. E. Gomory. 1961. A linear programming approach to the cutting-stock problem. *Oper. Res.* **9** 849–859.

Gilmore, P. C., R. E. Gomory. 1963. A linear programming approach to the cutting-stock problem (Part II). *Oper. Res.* **11**(6) 863–888.

Holthaus, O. 2002. Decomposition approaches for solving the integer one-dimensional cutting stock problem with different types of standard lengths. *Eur. J. Oper. Res.* **141**(2) 295–312.

Kellerer, H., U. Pferschy, D. Pisinger. 2004. *Knapsack Problems*. Springer, Berlin.

Kiwiel, K. C. 1994. A Cholesky dual method for proximal piecewise linear programming. *Numer. Math.* **68** 325–340.

Kiwiel, K. C. 2006a. A proximal bundle method with approximate subgradient linearizations. *SIAM J. Optim.* **16**(4) 1007–1023.

Kiwiel, K. C. 2006b. A proximal-projection bundle method for Lagrangian relaxation, including semidefinite programming. *SIAM J. Optim.* **17**(4) 1015–1034.

Kiwiel, K. C., C. Lemaréchal. 2009. An inexact bundle variant suited to column generation. *Math. Programming* **118**(1) 177–206.

Martello, S., P. Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, New York.

Nitsche, C., G. Scheithauer, J. Terno. 1999. Tighter relaxations for the cutting stock problem. *Eur. J. Oper. Res.* **112**(3) 654–663.

Scheithauer, G., J. Terno, A. Müller, G. Belov. 2001. Solving one-dimensional cutting stock problems exactly using a cutting plane algorithm. *J. Oper. Res. Soc.* **52**(2) 1390–1401.

Schwerin, P., G. Wäscher. 1997. The bin-packing problem: A problem generator and some numerical experiments. *Internat. Trans. Oper. Res.* **4** 337–389.

Stadtler, H. 1990. A one-dimensional cutting stock problem in the aluminium industry and its solution. *Eur. J. Oper. Res.* **44**(2) 209–223.

Valério de Carvalho, J. M. 2005. Using extra dual cuts to accelerate column generation. *INFORMS J. Comput.* **17**(2) 175–182.

Vance, P. H. 1998. Branch and price algorithms for the one-dimensional cutting stock problem. *Comput. Optim. Appl.* **9**(3) 212–228.

Vanderbeck, F. 1999. Computational study of a column generation algorithm for bin packing and cutting stock problems. *Math. Programming* **86**(3) 565–594.

Vanderbeck, F. 2000. On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Oper. Res.* **48**(1) 111–128.

Vanderbeck, F. 2002. Extending Dantzig's bound to the bounded multiple-class binary knapsack problem. *Math. Programming* **94**(1) 125–136.

Wäscher, G., T. Gau. 1996. Heuristics for the integer one-dimensional cutting stock problem. *OR Spectrum* **18**(3) 131–144.