258/2004

Raport Badawczy

Research Report

RB/64/2004

An Inexact Bundle Approach to Cutting Stock Problems

Krzysztof C. Kiwiel

Instytut Badań Systemowych
Polska Akademia Nauk

Systems Research Institute
Polish Academy of Sciences

POLSKA AKADEMIA NAUK

Instytut Badań Systemowych

ul. Newelska 6

01-447 Warszawa

tel.: (+48) (22) 8373578

fax: (+48) (22) 8372772

Kierownik Pracowni zgłaszający pracę:
Prof. dr hab. inż. Krzysztof C. Kiwiel

Warszawa 2004

An inexact bundle approach to cutting-stock problems*

Krzysztof C. Kiwiel†

December 30, 2004

Abstract

We show that the LP 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.

1 Introduction

The cutting-stock problem (CSP) is usually solved by LP-based column generation, rounding heuristics and branch-and-bound; see, e.g., [BeS02, BeS04a, DeP03, DeS99, Van98, Van99]. Since column generation applied to its LP relaxation may converge slowly, there is interest in stabilized variants based on LP or QP [BADF04, BAVdC04, BLM+05]. Alternatively, the highly efficient hybrid approach of [DeP03] 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 [Kiw04]. 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 that works quite well in practice. Further, by using relaxed bounds, we avoid the difficulties arising when a bounded KP is transformed into a 0-1 KP [Van02]. Next, by adapting the ideas of [BeS02, Hol02, Sta90, WäG96] to our inexact framework, we give rounding heuristics that solve almost all the CSP instances from the literature; in particular, they beat the best heuristics of [WäG96].

In effect, our inexact KP solutions, bound relaxation and rounding heuristics should be of interest also for other column generation approaches to the CSP.

Our work was inspired by [BLM+05], where the CSP's Lagrangian relaxation is solved by a standard (exact) bundle method. Thanks to inexact KP solutions, our method is much faster in practice than *all* the algorithms tested in [BLM+05, §2.2] (see Rem. 5.1).

*Research supported by the State Committee for Scientific Research under Grant 4T11A00622.

†Systems Research Institute, Newelska 6, 01–447 Warsaw, Poland (kiwiel@ibspan.waw.pl)

The paper is organized as follows. In §2 we recall the classic CSP model of [GiG61] and introduce inexact KP solutions for its Lagrangian relaxation. Our rounding heuristics are given in §3 in a general form suitable for other approaches. The inexact bundle method is reviewed in §4. Our computational results are presented in §5.

2 Lagrangian relaxation of the CSP

The one-dimensional cutting-stock problem (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 [GiG61] is formulated as follows. Denote the set of *cutting patterns* by

$$P := \left\{ p \in \mathbb{Z}_+^m : wp \le W \right\}. \tag{2.1}$$

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

$$\min \sum_{p \in P} z_p \quad \text{s.t.} \quad \sum_{p \in P} p z_p \ge d, \ z \in \mathbb{Z}_+^{|P|}. \tag{2.2a}$$

For Lagrangian relaxation we augment this model with the redundant constraint

$$\sum_{p \in P} z_p \le N, \tag{2.2b}$$

where N is an upper bound on the optimal value of (2.2a) (e.g., $N = \sum_i d_i$); this ensures boundedness of the *ground set* $Z := \{z \in \mathbb{Z}_+^{|P|} : \sum_p z_p \le N\}$. 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*

$$\theta(u) := \min_{z \in Z} \left\{ L(z; u) = ud + \textstyle\sum_{p \in P} (1 - up) z_p \right\}. \tag{2.3}$$

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

$$p(u) \in \operatorname{Arg\,max} \left\{ up : wp \le W, p \in \mathbb{Z}_+^m \right\} \tag{2.4}$$

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

$$\theta(u) = ud + N \left[1 - up(u) \right]_-, \tag{2.5}$$

where $[\cdot]_- := \min\{\cdot, 0\}$. Let v^{GG} and v_{LP}^{GG} denote the optimal values of (2.2) and its LP relaxation, respectively. It is well known that v_{LP}^{GG} coincides with the dual optimal value

$$\theta_* := \max \left\{ \theta(u) : u \in \mathbb{R}_+^m \right\}. \tag{2.6}$$

Experiments show that $\check{u} := w/W$ is a good initial estimate of solutions to the Lagrangian dual (2.6) [BAVdC04, §4], [BLM+05, §2]. In fact $\check{u}$ minimizes the relaxed dual function

$$\theta_{LP}(u) := ud + N\left[1 - u\underline{p}(u)\right]_-, \tag{2.7}$$

where $\underline{p}(u)$ solves the LP relaxation of (2.4). Indeed, since $\theta_{LP}(\check{u}) = \check{u}d \le v^{GG} \le N$, we see that $-d = -N(\check{u}d/N)(d/\check{u}d)$ is a subgradient of the second term of (2.7) at $\check{u}$.

2.2 Inexact KP solutions

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

$$p \le b \quad \text{with} \quad b_i := \min\{\, d_i, \lfloor W/w_i \rfloor \,\}, \quad i = 1{:}\,m. \tag{2.8}$$

Indeed, adding the bound $p \le b$ to (2.1) and (2.4) does not change v^{GG}, but it may raise v^{GG}_{LP} [NST99]. Then the column generation subproblem (2.4) becomes a bounded KP, which can be turned into a 0-1 KP via the transformation of [MaT90, §3.2]. However, this transformation may duplicate solution representations, thus creating difficulties for 0-1 KP solvers [Van02]. To avoid duplicates, we may use the *relaxed* bound

$$p \le b' \quad \text{with} \quad b'_i := 2^{\lceil \log_2(b_i+1) \rceil} - 1, \quad i = 1{:}\,m, \tag{2.9}$$

which corresponds to replacing d_i in (2.8) by the smallest number $2^j - 1 \ge d_i$, $j \ge 1$ ($2d_i - 1$ in the worst case); the number of transformed variables is the same. We solve the transformed KP by a double precision version of procedure MT1R of [MaT90] (because more recent KP solvers [KPP04] accept integer data only). 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 $\zeta \ge (1 - \epsilon_r)\bar{\zeta}$, where $\zeta := up$ for the incumbent p and $\bar{\zeta}$ is MT1R's upper bound on $up(u)$. Hence, by (2.5), we have the accuracy estimates

$$\underline{\theta}(u) := ud + N\left(1 - \bar{\zeta}\right)_- \le \theta(u) \le \bar{\theta}(u) := ud + N\,(1 - \zeta)_-, \tag{2.10a}$$

$$\bar{\theta}(u) - \underline{\theta}(u) \le N\left(\bar{\zeta} - \zeta\right) \le N\epsilon_r\bar{\zeta}. \tag{2.10b}$$

For a normal exit with an optimal $p = p(u)$, we may replace $\bar{\zeta}$ by ζ and ϵ_r by 0 in (2.10).

3 Heuristic rounding of relaxed solutions

Typical rounding heuristics proceed as follows. A solution $\hat{z}$ of the LP relaxation is rounded into an integer solution $\bar{z}$. A sequential heuristic applied to the *residual* problem (2.2) 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 (2.2) (which is exact if its value is equal to a lower bound on v^{GG}; e.g., $\lceil v^{GG}_{LP} \rceil$). Since 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 [Hol02, STMB01]; however, if the residual problem becomes too small to produce a solution to the original problem, some components of $\bar{z}$ may be decreased [BeS02].

In §3.1 we give a general rounding procedure, which augments the ideas of [BeS02, Hol02] with the oversupply reduction of [Sta90]. As for sequential heuristics, in §3.2 we describe minor (but useful) modifications of the first-fit-decreasing (FFD) of [Chv83] and the heuristics of [BeS04b, Hol02]. Since it pays to call lighter heuristics first, useful combinations of rounding and sequential heuristics are detailed in §3.3.

3.1 A general rounding procedure

Numbering the patterns so that $P = \{p^j\}_{j=1}^n$, we may write (2.2a) as

$$\min \sum_{j=1}^{n} z_j \quad \text{s.t.} \quad \sum_{j=1}^{n} p^j z_j \geq d,\ z \in \mathbb{Z}_+^n. \tag{3.1}$$

Given an incumbent solution z^* of (3.1) (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 3.1 (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 $r_{j_1} \geq \ldots \geq r_{j_{\bar{n}}}$, and set $\bar{n} := |\{j : r_j > 0\}|$.

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

$$\sum_{i:d'_i<0} w_i \min\left\{p_i^j, -d'_i\right\} \tag{3.2}$$

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 $I := \emptyset$. For $i = 1{:}\bar{n}$, 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 (3.1) 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 $I \neq \emptyset$, 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 [BeS02, §2.5]; otherwise Step 2 (due to [Sta90, Fig. 3]) helps. Following [BeS02, §5.2], our implementation allows at most ten returns from Step 5.

One of our heuristics uses the following modification of Step 3, based on the ideas in [Hol02, §3.2].

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

3.2 Sequential heuristics

We now describe our heuristics for the residual problem (2.2a) with d replaced by $d' \geq 0$, assuming $w_1 \geq \ldots \geq w_m$.

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

$$p_i := \min\left\{ d''_i, \left\lfloor \left(W - \textstyle\sum_{j<i} w_j p_j \right) / w_i \right\rfloor \right\} \quad \text{for } i = 1\colon m, \tag{3.3}$$

set $\kappa := \min\{\lfloor d''_i / p_i \rfloor : p_i > 0\}$, $\tilde{z}_p := \tilde{z}_p + \kappa$, $d'' := d'' - \kappa p$. The version of [Chv83, p. 208] employs $\kappa \equiv 1$, and hence is less efficient for large demands.

Our modification of the *sequential heuristic procedure* (SHP) of [Hol02, §3.2], given a price vector $\hat{u} \in \mathbb{R}^m$ (e.g., an approximate solution of (2.6)) 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 (3.3) by the bounded KP

$$p \in \operatorname{Arg\,max}\left\{ \bar{u}p : wp \leq W, p \leq d'', p \in \mathbb{Z}^m_+ \right\}. \tag{3.4}$$

Our implementation of the *sequential value correction* (SVC) heuristic of [BeS04b, §2] records the best solution found by calling SHP at most thirty times with $\bar{u}$ modified as follows. Initially $\bar{u}_i := \max\{1, W\hat{u}_i\}$, $i = 1\colon m$. If $wd'' \not\leq W$, then after solving (3.4) and updating d'', for i such that $p_i > 0$, set

$$\bar{u}_i := \left\lceil \gamma_i \bar{u}_i + (W/wp) w_i^{1.04} \right\rceil / (\gamma_i + 1) \quad \text{with} \quad \gamma_i := \Omega_i \left(d'_i + d''_i \right) / p_i, \tag{3.5}$$

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.

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

The remaining heuristics use an extension of Procedure 3.1 with a copy of Step 4 inserted after Step 1; the sequential heuristics employed at these steps are listed below.

Our *periodic* heuristic H1 is called by the bundle method (every twentieth iteration, starting from iteration $k = m + 1$) with the current relaxed solution $\hat{z} := \hat{z}^k$ and the lower bound $\underline{\theta}_k \leq \theta_*$. H1 calls FFD, exiting if $ez^* = \lceil \underline{\theta}_k \rceil$, since then z^* is optimal.

Our *final* heuristics H2, H3 and H4 are called successively upon termination of the boundle method, using the final $\hat{z} := \hat{z}^k$, $\hat{u} := \hat{u}^k$ and $\underline{\theta}_k$. H2 employs 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 $e\bar{z} + e\tilde{z} = \lceil \underline{\theta}_k \rceil$.

4 The inexact proximal bundle method

We now sketch the main features of the inexact bundle method of [Kiw04].

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 (2.10) for $u = u^k$, and let $\zeta_k := \zeta$, $\bar{\zeta}_k := \bar{\zeta}$. Recalling (2.3), define the associated *Lagrangian solution* z^k by setting $z^k_p := N$, $z^k_q := 0$ for $q \neq p$ if $\zeta > 1$, $z^k := 0$ otherwise. Thus we have the lower bound $\underline{\theta}(u^k) \leq \theta(u^k)$ and $L(z^k; u^k) = \bar{\theta}(u)$ in (2.10); in particular,

$$L(z^k; u^k) - \theta(u^k) \leq N\left(\bar{\zeta}_k - \zeta_k\right) \leq N\epsilon_r\bar{\zeta}_k. \tag{4.1}$$

Further, by (2.3), the following *linearization* of θ at u^k majorizes $\theta(u)$ for *all* u:

$$\theta_k(u) := L(z^k; u) = ud + \begin{cases} N\left(1 - up^k\right) & \text{if } z^k \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

Iteration k uses the polyhedral *cutting-plane model* of θ

$$\hat{\theta}_k(\cdot) := \min_{j \in J^k} \theta_j(\cdot) \quad \text{with} \quad k \in J^k \subset \{1, \ldots, k\} \tag{4.3}$$

for finding

$$u^{k+1} := \arg\max \left\{ \hat{\theta}_k(u) - \tfrac{1}{2t_k}|u - \hat{u}^k|^2 : u \in \mathbb{R}^m_+ \right\}, \tag{4.4}$$

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 $k' \leq k$ (usually $\theta^k_{\hat{u}} = \max^k_{j=1} \theta_j(u^j)$). If necessary, the stepsize t_k is increased and u^{k+1} is recomputed until the *predicted ascent* $\hat{\theta}_k(u^{k+1}) - \theta^k_{\hat{u}}$ is sufficiently positive. An *ascent* step to $\hat{u}^{k+1} := u^{k+1}$ with $k' := k + 1$ occurs if

$$\theta_{k+1}(u^{k+1}) - \theta^k_{\hat{u}} \geq \kappa v_k \quad \text{with} \quad v_k := \hat{\theta}_k(u^{k+1}) - \theta^k_{\hat{u}} \tag{4.5}$$

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 (4.3).

If we omitted the quadratic term in (4.4), the resulting cutting-plane method could generate u^{k+1} far from the previous points, and it would require storing all linearizations ($J^k = \{1, \ldots, k\}$ in (4.3)). 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 (4.4) with the QP routine of [Kiw94], 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 $\hat{J}^k := \{j \in J^k : \nu^k_j \neq 0\}$ has at most $m + 1$ elements. We set $J^{k+1} := \hat{J}^k \cup \{k + 1\}$ and then, if necessary, drop from J^{k+1} an index $j \in J^k \backslash \hat{J}^k$ with the largest $\theta_j(u^{k+1})$ to keep $|J^{k+1}| \leq M$ for a fixed $M \geq m + 2$.

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 (2.2) via the *aggregate primal solution*

$$\hat{z}^k := \sum_{j \in J^k} \nu^k_j z^j. \tag{4.6}$$

In other words, $\hat{z}^k_{p^j} = N\nu^k_j$ for nontrivial patterns p^j indexed by $J^k_P := \{j \in \hat{J}^k : z^j \neq 0\}$ (which need not be stored, since they can be recovered from $\nabla\theta_j = d - Np^j$; see (4.2)). Our heuristics also use the *lower bound* $\underline{\theta}_k := \max_{j\le k} \underline{\theta}(u^j)$ on $\theta_* = v^{GG}_{LP}$ (cf. (2.6)).

We now point out some useful consequences of the convergence analysis in [Kiw04, §5]. The LP relaxation of (2.2) may be written as

$$v^{GG}_{LP} := \min\ \bar{\psi}_0(z) := \sum_{p\in P} z_p \quad \text{s.t.} \quad \bar{\psi}(z) := d - \sum_{p\in P} pz_p \le 0,\ z \in \operatorname{conv} Z. \tag{4.7}$$

Let $\epsilon := \sup_k[\theta_k(u^k) - \theta(u^k)]$ be the maximum evaluation error; by (4.1), we have $\epsilon \le \bar{\epsilon} := N\epsilon_r \sup_k \bar{\zeta}_k$. Consider the set of *$\epsilon$-optimal solutions* of the LP relaxation (4.7):

$$Z_\epsilon := \left\{ z \in \operatorname{conv} Z : \bar{\psi}_0(z) \le v^{GG}_{LP} + \epsilon, \bar{\psi}(z) \le 0 \right\}. \tag{4.8}$$

The limits $\theta^\infty_{\hat{u}} := \lim_k \theta^k_{\hat{u}}$, $\underline{\theta}_\infty := \lim_k \underline{\theta}_k$ satisfy $\theta^\infty_{\hat{u}} \in [v^{GG}_{LP}, v^{GG}_{LP}+\epsilon]$, $\underline{\theta}_\infty \in [\theta^\infty_{\hat{u}} - \bar{\epsilon}, v^{GG}_{LP}]$, and there exists $K \subset \{1,2,\ldots\}$ 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. 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; since 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 $\min\{v_k, |\pi^k| + \alpha_k\} \le \epsilon_{\text{opt}}(1 + |\theta^k_{\hat{u}}|)$, where v_k is given in (4.5), $\pi^k := (\hat{u}^k - u^{k+1})/t_k$, $\alpha_k := v_k - t_k|\pi^k|^2$ and $\epsilon_{\text{opt}} > 0$ is an *optimality tolerance* (cf. [Kiw04, §4.2]). For $\epsilon_{\text{opt}} = 10^{-8}$, $\underline{\theta}_k$ usually agrees with θ_* in at least 8 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 [Van98], the 10 industrial instances of [Van99], and the 20 industrial instances of [DeS99]. In addition, we use the following randomly generated instances: the 4000 instance of [WäG96], the 3360 instances of [DeP03] and the 120 instances of [Van99]. For the BPP, we use the 540 randomly generated instances of [DeP03], and the 160 instances from the BINPACK collection of the OR-Library [Bea90].

The instances of [WäG96] are constructed by the CUTGEN1 generator [GaW95], 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 $R_1, \ldots, R_m \in (0,1)$, the demands $d_i := \lfloor \frac{R_i m\bar{d}}{R_1+\cdots+R_m} \rfloor$ for $i < m$, and $d_m := m\bar{d} - \sum_{i<m} d_i$ (in fact slightly more complicated formulas are used [GaW95]). 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 [DeP03] 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 [DeP03], only $\bar{d} = 50$ is used and the widths are uniformly distributed on $[w_{\min}, cW]$, where $w_{\min} = 500, 1000, 1500$. Both cases have 84 data classes, and 20 instances are generated in each class.

The instances of [Van99] comprise 6 classes with $m = 50$, and 20 instances per class. The first three classes are generated like those of [WäG96] above with $c = 0.25, 0.5, 0.75$ and $\bar{d} = 50$, the next two classes have widths in $[500, 2500]$ and $[500, 5000]$ with $\bar{d} = 50$, and the sixth class has widths in $[500, 5000]$ and $\bar{d} = 100$.

In the BPP instances of [DeP03], $m = 500$ or 1000 weights are uniformly distributed in the intervals $[1, 100]$, $[20, 100]$, $[50, 100]$ as in BPPGEN [ScW97], 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 [DeP03] use $m = 500$, the weight intervals $[1, 10000]$, $[2000, 10000]$, $[5000, 10000]$, and the capacity $W = 10000, 12000, 15000$, again with 20 instances per class.

The BINPACK instances from the OR-Library [Bea90] 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 $m = 120, 250, 500, 1000$. (The classes with $m = 500, 1000$ also appear in the BPP category of [DeP03], but with different instances.) In the *triplet* category, each bin of capacity $W = 1000$ 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 (2.6), 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 (4.2)), 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 (4.4) with the routine of [Kiw94]. 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 $\epsilon_r = 10^{-8}$. For smaller values of ϵ_r, we could not solve even some small instances in reasonable time (e.g., problem 28p0 in Tab. 5.9 took 28.30, 40.99 and 67.13 seconds for $\epsilon_r = 10^{-13}$, 10^{-14} and 0, respectively). For column generation, we used the relaxed bounds of (2.9), because the tighter bounds of (2.8) produced longer computing times. In contrast, SHP employed in (3.4) the natural bounds given by (2.8) 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 follow closely the presentation of [DeP03]. 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 [DeP03] with $\bar{d} = 50$ are reported in Table 5.1. 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 due to discovering that $ez^\star = \lceil \underline{\theta}_k \rceil$ for the incumbent $z^\star$ delivered by H0 or H1 *before* bundle terminated on its own. Recall that H1 is called after H0, H2 after H1, etc., unless $ez^\star = \lceil \underline{\theta}_k \rceil$ occurs earlier. The columns labelled H1 through H4 give the numbers of instances in which the corresponding heuristic found the best primal value $ez^\star$ first (for the remaining instances $ez^\star$ 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* $g := ez^\star - \lceil \underline{\theta}_k \rceil$; we stress that the final gaps never exceeded *one* unit in *all* of our instances.

The averages, maxima and sums in Table 5.1 are taken over 20 instances for each interval, and thus over 80 instances for each "*all*" row. In Table 5.2, there are 60 instances per interval (i.e., 20 instances for each value of the average demand $\bar{d} = 10, 50, 100$), and each "*all*" row gives statistics over the 240 instances used for each value of m. Finally, each row in Table 5.3 reports statistics over 80 instances (obtained from the 20 instances used for each of the four width intervals).

From the "*all*" entries for n_e, H1 through H4 and n_g in Table 5.2, 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 2 problems, and just one out of the 1680 problems was not solved. Note that the best method LR of [DeP03] 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 [DeP03] are presented in Tables 5.4 and 5.5, where each "*all*" row gives statistics over the 240 instances used for each value of m. 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 two out of the 1680 problems were not solved. The rounding heuristic of [DeP03] solved 69.9% of problems, whereas H0 through H2 solved 99.4% of problems.

Comparing the "*all*" rows in Tables 5.1–5.2 and 5.4–5.5, 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

Table 5.1: Small-item-size instances of Degraeve and Peeters (2003), $\bar{d} = 50$

m	int	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
10	[1, 2500]	10.00	38.15	7.95	20	0.00	0.01	19	13	0	0	0	0
	[1, 5000]	10.00	29.45	19.75	31	0.00	0.01	5	7	13	0	0	0
	[1, 7500]	9.95	22.35	19.95	29	0.00	0.01	0	0	8	0	0	0
	[1, 10000]	10.00	19.95	18.10	24	0.00	0.01	3	0	6	1	0	0
	all	9.99	27.47	16.44	31	0.00	0.01	27	20	27	1	0	0
20	[1, 2500]	20.00	78.65	13.00	21	0.00	0.01	20	12	0	0	0	0
	[1, 5000]	19.95	57.10	42.10	56	0.01	0.04	7	7	12	0	0	0
	[1, 7500]	20.00	45.15	42.10	59	0.00	0.01	3	2	8	0	0	0
	[1, 10000]	20.00	38.95	36.45	52	0.00	0.01	4	1	5	0	0	0
	all	19.99	54.96	33.41	59	0.01	0.04	34	22	25	0	0	0
30	[1, 2500]	29.90	116.85	26.40	51	0.01	0.03	20	15	0	0	0	0
	[1, 5000]	29.90	87.30	69.25	91	0.04	0.08	9	9	11	0	0	0
	[1, 7500]	30.00	68.50	65.40	91	0.01	0.02	5	3	8	0	0	0
	[1, 10000]	29.95	60.25	58.15	69	0.01	0.02	4	2	3	0	0	0
	all	29.94	83.22	54.80	91	0.02	0.08	38	29	22	0	0	0
40	[1, 2500]	39.80	153.20	39.65	76	0.02	0.07	19	17	1	0	0	0
	[1, 5000]	39.85	113.20	92.50	121	0.12	0.21	14	14	6	0	0	0
	[1, 7500]	39.90	89.20	92.35	121	0.03	0.05	6	6	6	0	0	0
	[1, 10000]	39.90	76.15	78.25	108	0.02	0.03	3	1	5	0	0	0
	all	39.86	107.94	75.69	121	0.05	0.21	42	38	18	0	0	0
50	[1, 2500]	49.60	190.70	35.75	51	0.02	0.04	20	14	0	0	0	0
	[1, 5000]	49.70	145.30	116.55	171	0.19	0.42	16	16	4	0	0	0
	[1, 7500]	49.75	113.30	124.85	151	0.07	0.12	5	3	13	0	0	0
	[1, 10000]	50.00	99.95	105.30	131	0.04	0.06	1	2	3	0	0	0
	all	49.76	137.31	95.61	171	0.08	0.42	42	35	20	0	0	0
75	[1, 2500]	73.85	285.85	72.10	115	0.07	0.15	19	17	1	0	0	0
	[1, 5000]	74.10	218.00	167.30	256	0.42	1.33	18	18	2	0	0	0
	[1, 7500]	74.80	170.10	196.40	226	0.29	0.61	5	5	7	0	0	0
	[1, 10000]	74.75	145.15	158.75	218	0.12	0.24	3	2	4	0	0	0
	all	74.38	204.78	148.64	256	0.22	1.33	45	42	14	0	0	0
100	[1, 2500]	98.50	374.00	100.50	120	0.13	0.22	20	20	0	0	0	0
	[1, 5000]	99.05	286.45	182.35	261	0.45	1.95	17	17	3	0	0	0
	[1, 7500]	99.30	227.35	272.70	311	0.78	1.27	14	13	5	0	0	0
	[1, 10000]	99.45	194.55	224.65	294	0.31	0.66	5	3	3	0	0	0
	all	99.08	270.59	195.05	311	0.42	1.95	56	53	11	0	0	0

Table 5.2: Small-item-size instances of Degraeve and Peeters (2003), $\bar{d} = all$

m	int	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
10	[1, 2500]	10.00	36.70	8.02	28	0.00	0.01	54	30	5	0	0	0
	[1, 5000]	9.98	28.80	16.13	31	0.00	0.01	22	12	33	0	0	0
	[1, 7500]	9.98	22.10	18.17	29	0.00	0.01	19	4	18	0	0	0
	[1, 10000]	10.00	19.48	18.25	31	0.00	0.01	18	3	14	1	0	0
	all	9.99	26.77	15.14	31	0.00	0.01	113	49	70	1	0	0
20	[1, 2500]	19.95	74.15	14.92	61	0.00	0.02	58	31	2	0	0	0
	[1, 5000]	19.90	56.25	37.57	56	0.01	0.04	30	25	29	0	0	0
	[1, 7500]	19.97	43.53	41.33	69	0.00	0.01	16	6	21	0	0	0
	[1, 10000]	20.00	38.57	36.23	52	0.00	0.01	16	2	12	0	0	0
	all	19.95	53.13	32.51	69	0.01	0.04	120	64	64	0	0	0
30	[1, 2500]	29.85	110.18	22.02	51	0.01	0.03	59	36	1	0	0	0
	[1, 5000]	29.88	84.18	62.42	91	0.04	0.22	35	33	24	0	0	1
	[1, 7500]	29.95	66.50	65.53	91	0.01	0.03	16	10	19	0	0	0
	[1, 10000]	29.95	58.18	57.65	83	0.01	0.02	20	6	13	0	0	0
	all	29.91	79.76	51.90	91	0.02	0.22	130	85	57	0	0	1
40	[1, 2500]	39.75	146.80	31.82	79	0.02	0.13	56	36	4	0	0	0
	[1, 5000]	39.87	111.88	80.00	134	0.09	0.35	44	42	16	0	0	0
	[1, 7500]	39.92	88.75	93.35	121	0.03	0.10	20	13	19	0	0	0
	[1, 10000]	39.87	74.78	76.62	108	0.02	0.03	14	7	14	0	0	0
	all	39.85	105.55	70.45	134	0.04	0.35	134	98	53	0	0	0
50	[1, 2500]	49.60	182.70	32.93	71	0.02	0.06	60	37	0	0	0	0
	[1, 5000]	49.65	140.88	106.97	181	0.20	0.66	41	40	19	0	0	0
	[1, 7500]	49.82	110.80	122.42	151	0.07	0.13	19	16	24	0	0	0
	[1, 10000]	49.93	94.27	98.40	131	0.03	0.06	14	9	12	0	0	0
	all	49.75	132.16	90.18	181	0.08	0.66	134	102	55	0	0	0
75	[1, 2500]	73.82	271.55	58.00	115	0.06	0.16	58	42	2	0	0	0
	[1, 5000]	74.23	209.83	157.35	256	0.53	2.00	49	49	11	0	0	0
	[1, 7500]	74.67	165.52	190.80	239	0.26	0.61	31	26	16	0	0	0
	[1, 10000]	74.72	142.38	160.85	227	0.12	0.27	10	4	15	0	0	0
	all	74.36	197.32	141.75	256	0.24	2.00	148	121	44	0	0	0
100	[1, 2500]	98.13	359.88	74.17	152	0.10	0.22	59	42	1	0	0	0
	[1, 5000]	98.90	280.47	166.03	277	0.44	2.81	52	49	8	0	0	0
	[1, 7500]	99.18	221.73	268.80	311	0.76	1.54	37	35	14	0	1	0
	[1, 10000]	99.45	191.37	224.30	294	0.29	0.66	17	10	11	0	0	0
	all	98.92	263.36	183.33	311	0.40	2.81	165	136	34	0	1	0

Table 5.3: Small-item-size instances of Degraeve and Peeters (2003), *int* = *all*

m	$\bar{d}$	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
10	10	10.00	24.95	10.77	26	0.00	0.01	69	14	6	0	0	0
	50	9.99	27.47	16.44	31	0.00	0.01	27	20	27	1	0	0
	100	9.99	27.89	18.21	31	0.00	0.01	17	15	37	0	0	0
20	10	19.94	48.60	25.93	58	0.00	0.01	57	16	9	0	0	0
	50	19.99	54.96	33.41	59	0.01	0.04	34	22	25	0	0	0
	100	19.94	55.81	38.20	69	0.01	0.03	29	26	30	0	0	0
30	10	29.91	73.34	43.33	91	0.01	0.22	57	28	5	0	0	1
	50	29.94	83.22	54.80	91	0.02	0.08	38	29	22	0	0	0
	100	29.88	82.73	57.59	91	0.02	0.13	35	28	30	0	0	0
40	10	39.83	96.79	56.89	109	0.02	0.13	59	29	9	0	0	0
	50	39.86	107.94	75.69	121	0.05	0.21	42	38	18	0	0	0
	100	39.86	111.94	78.76	134	0.05	0.35	33	31	26	0	0	0
50	10	49.70	121.30	75.41	151	0.05	0.36	59	35	6	0	0	0
	50	49.76	137.31	95.61	171	0.08	0.42	42	35	20	0	0	0
	100	49.79	137.88	99.51	181	0.11	0.66	33	32	29	0	0	0
75	10	74.41	181.36	121.24	216	0.18	1.15	61	41	7	0	0	0
	50	74.38	204.78	148.64	256	0.22	1.33	45	42	14	0	0	0
	100	74.29	205.83	155.38	239	0.32	2.00	42	38	23	0	0	0
100	10	98.90	243.64	154.60	310	0.28	1.69	62	35	4	0	0	0
	50	99.08	270.59	195.05	311	0.42	1.95	56	53	11	0	0	0
	100	98.78	275.86	200.32	303	0.49	2.81	47	48	19	0	1	0

average solution times grow significantly, and the maximum solution times jump up, most spectacularly on the instances with width interval $[1500, 2500]$. This is due to the poor performance of our knapsack solver on these instances. Similar slowdowns on this interval were reported in [DeP03, Tab. 4a] already for $m = 20$, i.e., even for smaller problems.

To save space, Table 5.6 presents only aggregate results on the instances of [WäG96], with each row giving statistics over the 800 instances used for each value of m. Here our main point is that only three out of 4000 (0.075%) problems were not solved. Our "lighter" heuristics H0 through H2 solved 99.7% of problems, whereas the two best (and more complicated) heuristics RSUC and CSTAOPT of [WäG96] solved 98.0% and 92.7% of problems, respectively (99.6% if they had been applied together). The fairly large maximum solution time in Tab. 5.6 stemmed from a single knapsack subproblem.

Table 5.7 gives our results for the 6 data classes of [Van99] with $m = 50$ and 20 instances per row. (Since we used the original instances, the results are not identical to those in Tabs. 5.1 and 5.5.) These instances are fairly easy, never requiring H3 or H4; in fact H0 through H2 suffice for solving *all* the CSP instances used in [Van99].

Quite suprisingly, all the industrial instances we could find in the literature turned out to be easy for our method. Tables 5.8–5.10 give our results for the industrial instances of [Van98] (as numbered in [DeP03, Tab. 7]), [Van99, Tab. 1] and [DeS99] (as named in [DeP03, Tab. 9]). The final column identifies the heuristic which delivered the optimal solution; in other words, H0 through H2 solved all these instances except for a single instance solved by H3.

Table 5.4: Medium-item-size instances of Degraeve and Peeters (2003), $\bar{d} = 50$

m	int	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
10	[500, 2500]	10.00	33.80	13.65	23	0.00	0.02	13	13	7	0	0	0
	[1000, 2500]	10.00	31.40	15.45	25	0.00	0.01	9	9	11	0	0	0
	[1500, 2500]	9.90	29.70	16.40	25	0.00	0.01	4	4	16	0	0	0
	[500, 5000]	10.00	28.15	19.10	25	0.00	0.01	5	5	13	0	1	0
	[1000, 5000]	10.00	24.70	17.30	22	0.00	0.01	5	7	12	0	0	0
	[1500, 5000]	9.90	22.50	17.40	23	0.00	0.01	2	5	15	0	0	0
	[500, 7500]	10.00	20.35	20.00	29	0.00	0.01	3	1	10	0	0	0
	[1000, 7500]	10.00	19.00	18.65	24	0.00	0.01	1	0	8	0	0	0
	[1500, 7500]	10.00	17.40	19.25	25	0.00	0.01	0	0	8	0	0	0
	[500, 10000]	10.00	18.30	18.95	28	0.00	0.01	5	0	3	0	0	0
	[1000, 10000]	10.00	16.50	17.35	22	0.00	0.00	3	2	4	0	0	0
	[1500, 10000]	10.00	15.30	16.75	20	0.00	0.01	4	2	5	0	0	0
	all	9.98	23.09	17.52	29	0.00	0.02	54	48	112	0	1	0
20	[500, 2500]	20.00	67.00	26.85	58	0.01	0.09	16	16	4	0	0	0
	[1000, 2500]	19.95	63.30	33.95	58	0.02	0.11	13	13	7	0	0	0
	[1500, 2500]	19.75	59.25	39.00	49	0.02	0.03	10	10	10	0	0	0
	[500, 5000]	19.95	52.40	39.85	45	0.01	0.02	3	3	17	0	0	0
	[1000, 5000]	19.95	48.30	36.35	41	0.01	0.02	2	2	18	0	0	0
	[1500, 5000]	19.95	45.75	34.50	41	0.00	0.01	1	2	18	0	0	0
	[500, 7500]	19.95	40.50	39.70	52	0.00	0.01	3	1	7	0	0	0
	[1000, 7500]	20.00	37.55	37.90	44	0.00	0.01	4	2	10	0	0	0
	[1500, 7500]	20.00	34.30	34.05	41	0.00	0.01	3	1	8	0	1	0
	[500, 10000]	20.00	34.80	33.90	48	0.00	0.01	5	0	4	0	0	0
	[1000, 10000]	19.95	32.40	32.65	45	0.00	0.01	4	0	5	0	0	0
	[1500, 10000]	20.00	31.35	31.95	40	0.00	0.01	4	0	6	0	0	0
	all	19.95	45.58	35.05	58	0.01	0.11	68	50	114	0	1	0
30	[500, 2500]	29.65	100.25	39.35	51	0.01	0.04	16	16	4	0	0	0
	[1000, 2500]	29.85	95.30	40.65	55	0.01	0.04	11	11	9	0	0	0
	[1500, 2500]	29.50	88.50	59.05	93	0.06	0.16	10	10	10	0	0	0
	[500, 5000]	30.00	79.65	61.00	71	0.03	0.06	6	6	14	0	0	0
	[1000, 5000]	29.80	72.95	57.75	71	0.02	0.11	9	9	11	0	0	0
	[1500, 5000]	29.60	66.70	53.80	65	0.01	0.02	6	8	12	0	0	0
	[500, 7500]	29.95	62.25	63.90	75	0.01	0.03	4	3	7	0	0	0
	[1000, 7500]	30.00	55.60	58.95	71	0.01	0.02	3	2	10	0	0	0
	[1500, 7500]	29.90	51.75	55.85	69	0.01	0.01	1	1	10	0	0	0
	[500, 10000]	29.95	52.30	52.00	67	0.01	0.01	2	2	6	0	0	0
	[1000, 10000]	29.95	49.85	53.05	64	0.01	0.01	3	3	7	0	0	0
	[1500, 10000]	29.95	46.55	49.15	61	0.01	0.01	2	2	5	0	0	0
	all	29.84	68.47	53.71	93	0.02	0.16	73	73	105	0	0	0
40	[500, 2500]	39.75	135.85	46.45	61	0.02	0.04	15	15	5	0	0	0
	[1000, 2500]	39.65	125.35	50.60	68	0.02	0.03	11	11	9	0	0	0
	[1500, 2500]	39.30	117.90	66.90	101	0.15	0.58	9	9	11	0	0	0
	[500, 5000]	39.85	103.30	82.05	101	0.07	0.11	9	9	11	0	0	0
	[1000, 5000]	39.80	95.50	76.10	99	0.05	0.09	6	7	13	0	0	0
	[1500, 5000]	39.75	90.85	71.70	84	0.03	0.05	4	4	16	0	0	0
	[500, 7500]	39.80	81.05	88.40	120	0.02	0.04	1	0	11	0	0	0
	[1000, 7500]	39.90	73.90	81.00	96	0.02	0.03	3	2	9	0	0	0
	[1500, 7500]	39.90	71.35	76.05	89	0.01	0.02	3	2	10	0	0	0
	[500, 10000]	39.85	68.80	71.15	93	0.01	0.02	2	2	6	0	0	0
	[1000, 10000]	39.90	63.50	65.55	80	0.01	0.02	2	0	5	0	0	0
	[1500, 10000]	39.90	60.40	63.30	82	0.01	0.02	4	1	5	0	0	0
	all	39.78	90.65	69.94	120	0.04	0.58	69	62	111	0	0	0

Table 5.5: Medium-item-size instances of Degraeve and Peeters (2003), $\bar{d} = 50$

m	int	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
50	[500, 2500]	49.60	170.70	53.60	71	0.03	0.04	20	20	0	0	0	0
	[1000, 2500]	49.20	155.95	64.55	83	0.03	0.04	8	8	12	0	0	0
	[1500, 2500]	49.00	147.00	75.05	109	0.24	0.89	2	3	17	0	0	0
	[500, 5000]	49.45	127.80	104.05	128	0.15	0.26	10	10	10	0	0	0
	[1000, 5000]	49.80	121.15	92.75	108	0.08	0.14	8	8	12	0	0	0
	[1500, 5000]	49.55	114.20	92.40	123	0.06	0.09	4	4	16	0	0	0
	[500, 7500]	49.75	104.90	120.35	156	0.05	0.10	4	3	9	0	1	0
	[1000, 7500]	49.70	93.70	106.30	125	0.04	0.06	2	2	14	1	0	0
	[1500, 7500]	49.90	86.65	96.20	112	0.03	0.07	4	3	7	0	0	1
	[500, 10000]	49.85	85.30	93.25	118	0.03	0.05	4	2	8	0	0	0
	[1000, 10000]	50.00	83.05	89.65	111	0.02	0.04	4	0	9	0	0	0
	[1500, 10000]	49.90	73.90	77.00	108	0.01	0.02	4	2	4	0	0	0
	all	49.64	113.69	88.76	156	0.06	0.89	74	65	118	1	1	1
75	[500, 2500]	73.65	251.20	82.85	102	0.06	0.08	14	14	6	0	0	0
	[1000, 2500]	73.40	232.30	88.60	101	0.05	0.06	5	5	15	0	0	0
	[1500, 2500]	71.85	215.55	102.25	164	2.72	8.56	3	3	17	0	0	0
	[500, 5000]	74.10	193.55	163.50	216	0.50	0.98	9	9	11	0	0	0
	[1000, 5000]	74.00	181.05	143.95	175	0.29	0.46	12	12	8	0	0	0
	[1500, 5000]	74.15	167.65	141.75	205	0.18	0.41	10	11	9	0	0	1
	[500, 7500]	74.70	154.65	181.65	232	0.20	0.50	7	7	11	0	0	0
	[1000, 7500]	74.50	138.80	171.25	201	0.12	0.16	6	7	5	0	0	0
	[1500, 7500]	74.50	128.95	153.60	178	0.09	0.15	6	3	7	0	0	0
	[500, 10000]	74.70	129.85	146.80	180	0.09	0.13	6	0	3	0	0	0
	[1000, 10000]	74.65	120.65	137.85	166	0.07	0.12	3	2	4	0	0	0
	[1500, 10000]	74.70	114.95	130.40	180	0.05	0.10	2	1	8	0	0	0
	all	74.08	169.10	137.04	232	0.37	8.56	83	74	104	0	0	1
100	[500, 2500]	98.10	336.05	104.35	119	0.08	0.12	13	13	7	0	0	0
	[1000, 2500]	96.30	303.40	105.95	116	0.07	0.14	4	4	16	0	0	0
	[1500, 2500]	95.55	286.65	128.35	210	13.50	61.84	3	3	17	0	0	0
	[500, 5000]	98.70	263.50	204.25	260	0.81	2.06	15	15	5	0	0	0
	[1000, 5000]	98.65	240.15	197.50	220	0.72	1.00	9	9	11	0	0	0
	[1500, 5000]	98.70	225.25	189.35	269	0.48	0.73	8	8	10	0	2	0
	[500, 7500]	99.15	202.85	252.35	295	0.43	0.54	4	6	10	0	0	0
	[1000, 7500]	99.40	188.45	240.60	295	0.31	0.40	8	8	10	0	1	0
	[1500, 7500]	98.70	174.60	212.80	236	0.22	0.33	1	1	14	0	1	0
	[500, 10000]	99.40	174.70	210.00	272	0.23	0.40	1	0	7	0	0	0
	[1000, 10000]	99.45	163.90	192.15	235	0.17	0.43	3	2	4	0	0	0
	[1500, 10000]	99.25	153.35	174.25	208	0.12	0.16	3	2	7	0	0	0
	all	98.45	226.07	184.32	295	1.43	61.84	72	71	118	0	4	0

Table 5.6: 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	8.36	485	240	183	0	2	0
30	29.90	75.72	48.18	105	0.01	0.14	503	281	161	0	1	0
40	39.84	100.10	65.05	123	0.04	3.29	503	314	159	0	2	2
50	49.73	125.22	84.75	171	0.07	0.47	526	341	138	0	4	1
all	29.88	75.37	48.60	171	0.03	8.36	2466	1310	833	0	9	3

Table 5.7: 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, 2500]	49.40	185.30	47.40	71	0.03	0.06	20	18	0	0	0	0
50	[1, 5000]	49.65	143.05	114.05	151	0.20	0.34	13	13	7	0	0	0
50	[1, 7500]	49.75	110.00	111.85	144	0.06	0.11	6	5	8	0	0	0
50	[500, 2500]	49.40	166.10	56.80	77	0.03	0.04	14	14	6	0	0	0
50	[500, 5000]	49.70	128.20	103.65	114	0.14	0.27	11	11	9	0	0	0
100	[500, 5000]	49.70	129.25	104.40	131	0.14	0.33	8	8	12	0	0	0

5.4 Impact of tighter knapsack bounds

The results of §5.3 were obtained for the relaxed bounds of (2.9). Using the tighter bounds of (2.8) allowed us to solve just two more instances at the expense of longer running times. To save space, the following tables and remarks list only data classes on which the tightening of KP bounds mattered most, giving more details for larger problem sizes.

Concerning Tables 5.11–5.12, the good news is that tighter bounds allowed us to solve *all* the small-item-size instances of [DeP03], and *all but one* of the medium-item-size instances of [DeP03]. Unfortunately the running times grew substantially relative to Tabs. 5.2 and 5.5. On the small-item-size instances, for $m \geq 40$ the average running times grew by about 150% (mostly from increasing by about 200% on width interval $[1, 5000]$). On the medium-item-size instances, the average running times grew by 200%, 217%, 303% and 446% for $m = 40, 50, 75$ and 100, increasing by 367–531% on width interval $[1500, 2500]$, 157–223% on $[500, 5000]$, and 140–179% on $[1000, 5000]$; for $m = 100$, they went up by 67–156% on four other intervals. The iteration numbers were about the same. The increase in running times can be attributed to the knapsack solver (which made more than two million backtrackings on some subproblems).

For the instances of [WäG96], the same 3997 out of 4000 instances were solved, but relative to Tab. 5.6, for $m = 40$ and 50 the average running times grew by 100% and 143%.

For the instances of [Van99], relative to Tab. 5.7, the average running times grew by between 67% and 200%; their sum increased by 175%.

Table 5.8: Industrial CSP instances of Vance (1998)

inst.	m	m'	i	t	n_e	Hi
1	2	7	1	0.00	1	H0
2	2	8	3	0.00	1	H1
3	3	11	1	0.00	1	H0
4	5	17	1	0.00	1	H0
5	14	50	15	0.01	1	H1
6	5	19	1	0.00	1	H0
7	4	14	1	0.00	1	H0
8	7	27	10	0.00	0	H2
9	11	46	12	0.00	1	H1
10	3	9	2	0.00	1	H0
11	2	7	1	0.00	1	H0
12	6	23	1	0.00	1	H0
13	2	9	1	0.00	1	H0
14	3	11	4	0.00	1	H1
15	7	20	8	0.00	1	H1
16	4	9	1	0.00	1	H0
17	12	42	24	0.00	0	H2
18	14	44	15	0.00	1	H1
19	5	15	13	0.00	0	H2
20	11	31	21	0.00	0	H2
21	9	27	16	0.00	0	H2
22	8	25	16	0.00	0	H2
23	7	20	8	0.00	1	H1
24	7	22	13	0.00	0	H2
25	12	39	13	0.00	1	H1
26	6	18	7	0.00	1	H1
27	12	40	13	0.00	1	H1
28	18	48	1	0.00	1	H0

Table 5.9: Industrial CSP instances of Vanderbeck (1999)

inst.	name	m	m'	i	t	n_e	Hi
1	7p18	7	22	13	0.00	0	H2
2	11p4	11	46	12	0.01	1	H1
3	12p19	12	39	13	0.00	1	H1
4	14p12	14	50	15	0.01	1	H1
5	d16p6	16	34	17	0.00	1	H1
6	25p0	25	80	66	0.07	1	H1
7	28p0	28	102	47	0.02	0	H2
8	30p0	26	86	27	0.00	1	H1
9	d33p20	23	53	24	0.07	1	H1
10	d43p21	32	74	33	0.04	1	H1

Table 5.10: Industrial CSP instances of Degraeve and Schrage (1999)

name	m	m'	i	t	n_e	Hi
DS01	41	90	62	0.05	1	H1
DS02	40	89	71	0.04	0	H2
DS03	26	56	47	0.00	1	H1
DS04	14	29	20	0.01	0	H3
DS05	18	33	31	0.00	0	H2
DS06	71	149	132	0.28	1	H1
DS07	14	41	15	0.01	1	H1
DS08	35	58	47	0.00	1	H1
DS09	35	86	36	0.05	1	H1
DS10	46	98	102	0.06	1	H1
DS11	42	89	43	0.00	1	H1
DS12	53	110	97	0.03	1	H1
DS13	22	47	30	0.00	1	H1
DS14	29	45	40	0.00	0	H0
DS15	43	78	55	0.00	1	H1
DS16	8	24	13	0.01	0	H2
DS17	37	111	36	0.00	0	H2
DS18	16	54	17	0.00	1	H1
DS19	23	67	43	0.02	0	H2
DS20	11	41	18	0.04	0	H2

Table 5.11: Small-item-size instances with tight KP bounds, $\bar{d} = all$

m	int	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
40	$[1, 5000]$	39.87	111.88	80.40	141	0.28	1.38	44	42	16	0	0	0
	all	39.85	105.55	70.63	141	0.10	1.38	126	93	58	0	0	0
50	$[1, 5000]$	49.65	140.88	107.18	171	0.60	2.20	41	40	19	0	0	0
	all	49.75	132.16	90.49	171	0.20	2.20	131	100	57	0	0	0
75	$[1, 5000]$	74.23	209.83	158.25	236	1.63	6.29	52	52	8	0	0	0
	$[1, 7500]$	74.67	165.52	189.22	249	0.53	1.79	28	26	16	0	0	0
	$[1, 10000]$	74.72	142.38	160.50	231	0.19	0.49	15	6	13	0	0	0
	all	74.36	197.32	141.57	249	0.61	6.29	152	125	40	0	0	0
100	$[1, 2500]$	98.13	359.88	74.53	140	0.16	0.42	59	42	1	0	0	0
	$[1, 5000]$	98.90	280.47	165.08	281	1.34	11.15	52	49	7	0	1	0
	$[1, 7500]$	99.18	221.73	268.20	319	1.54	4.55	28	27	23	0	0	0
	$[1, 10000]$	99.45	191.37	225.10	301	0.51	1.68	16	9	11	0	1	0
	all	98.92	263.36	183.23	319	0.89	11.15	155	127	42	0	2	0

Table 5.12: Medium-item-size instances with tight KP bounds, $\bar{d} = 50$

m	*int*	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}	n_e	H1	H2	H3	H4	n_g
30	[1500, 2500]	29.50	88.50	58.50	91	0.30	0.81	12	12	8	0	0	0
	all	29.84	68.47	53.63	91	0.05	0.81	78	77	100	0	1	0
40	[1500, 2500]	39.30	117.90	67.50	101	0.82	3.75	9	9	11	0	0	0
	[500, 5000]	39.85	103.30	82.20	101	0.18	0.32	9	9	11	0	0	0
	all	39.78	90.65	70.05	116	0.12	3.75	69	63	110	0	0	0
50	[1500, 2500]	49.00	147.00	75.00	109	1.12	4.80	4	4	16	0	0	0
	[500, 5000]	49.45	127.80	103.95	128	0.40	0.76	11	11	9	0	0	0
	[1000, 5000]	49.80	121.15	93.15	110	0.20	0.34	8	8	12	0	0	0
	all	49.64	113.69	88.92	154	0.19	4.80	83	71	111	0	3	1
75	[1500, 2500]	71.85	215.55	101.75	159	14.03	53.75	3	3	17	0	0	0
	[500, 5000]	74.10	193.55	164.80	216	1.43	3.15	10	10	9	1	0	0
	[1000, 5000]	74.00	181.05	144.05	175	0.81	1.35	12	12	8	0	0	0
	[1500, 5000]	74.15	167.65	142.70	205	0.44	0.67	9	9	11	0	0	0
	[500, 7500]	74.70	154.65	180.45	216	0.36	0.74	11	11	7	0	0	0
	all	74.08	169.10	137.08	216	1.49	53.75	80	71	106	1	0	0
100	[1500, 2500]	95.55	286.65	128.05	199	85.18	413.15	3	3	17	0	0	0
	[500, 5000]	98.70	263.50	206.25	260	2.62	7.87	17	17	3	0	0	0
	[1000, 5000]	98.65	240.15	197.00	220	1.97	2.78	9	9	11	0	0	0
	[1500, 5000]	98.70	225.25	189.80	263	1.23	1.79	7	7	12	0	1	0
	[500, 7500]	99.15	202.85	255.25	288	0.75	1.21	4	4	12	0	0	0
	[1000, 7500]	99.40	188.45	238.95	293	0.52	0.83	7	9	10	0	0	0
	[1000, 10000]	99.45	163.90	193.25	246	0.31	1.93	2	0	6	0	0	0
	all	98.45	226.07	184.73	293	7.81	413.15	75	73	118	0	2	0

5.5 Results for the bin-packing problem

Following [DeP03], in the next three tables we present our results for the BPP.

Table 5.13 gives our results for the BPP instances of [DeP03] (20 instances per row). All the 360 instances were solved (without requiring H3 and H4).

Table 5.14 reports results for the BINPACK instances from the OR-Library [Bea90] (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 5.15 (the influence of H3 could be ignored). Note that the triplet classes are quite difficult for traditional LP relaxation [DeP03, Tab. 12].

Table 5.16 presents our results for the modified BPP classes of [DeP03] (20 instances per row). Just one out of the 180 problems was not solved (H4 helped on two problems). 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.

Table 5.13: 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	0.28	3.71	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	0.01	20	0	0	0	0	0
		[20, 100]	80.85	151.65	86.70	102	0.01	0.02	14	14	6	0	0	0
		[50, 100]	51.00	77.00	64.80	72	0.00	0.01	12	0	0	0	0	0
1000	100	[1, 100]	100.00	183.65	199.90	230	0.07	0.11	12	1	1	0	0	0
		[20, 100]	81.00	117.95	114.25	133	0.02	0.03	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	24.65	174	0.01	0.05	19	2	1	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	0.02	20	1	0	0	0	0
		[20, 100]	81.00	154.90	86.65	102	0.01	0.02	12	12	8	0	0	0
		[50, 100]	51.00	77.00	67.25	77	0.01	0.01	10	0	0	0	0	0

Table 5.14: 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	0.01	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	85.85	113	0.01	0.04	16	16	3	0	1	0.0%	0
u1000	81.00	155.00	86.20	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	72.60	91	0.06	0.09	0	0	19	0	1	2.0%	16
t249	140.10	199.15	125.25	145	0.26	0.33	0	0	20	0	0	1.2%	20
t501	194.25	315.40	166.90	194	0.63	0.89	0	0	20	0	0	0.6%	19

Table 5.15: 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	72.60	91	0.01	0.02	0	0	20	2.1%	17
t249	140.10	199.15	125.25	145	0.04	0.06	0	0	20	1.2%	20
t501	194.25	315.40	166.90	194	0.08	0.11	0	0	20	0.6%	19

Table 5.16: 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
10000	[1, 10000]	488.65	494.05	1484.40	1737	35.03	48.50	14	3	0	0	0	0
	[2000, 10000]	485.15	490.20	800.70	916	7.07	9.89	15	1	0	0	0	0
	[5000, 10000]	474.75	474.80	457.70	480	1.16	1.36	16	0	0	0	0	0
12000	[1, 10000]	486.95	494.55	815.90	1732	25.86	58.14	18	7	1	0	0	0
	[2000, 10000]	484.75	492.20	1157.90	1328	15.04	21.37	18	2	0	0	0	0
	[5000, 10000]	475.95	480.35	520.75	550	2.22	2.65	15	0	0	0	0	0
15000	[1, 10000]	487.90	497.15	285.10	1171	7.99	67.07	17	5	0	0	2	1
	[2000, 10000]	482.70	494.25	805.05	1144	16.11	29.28	16	16	4	0	0	0
	[5000, 10000]	475.25	486.95	691.50	786	5.13	6.29	13	0	0	0	0	0

Table 5.17: Comparison of running times with Degraeve and Peeters (2003), $int = all$

	Tab. 5.1			Tab. 5.2		Tabs. 5.4–5.5	
m	HR	LR	BR	LR	BR	LR	BR
30	0.17	0.10	0.02	0.10	0.02	0.29	0.02
40	0.44	0.21	0.05	0.21	0.04	0.71	0.04
50	0.74	0.38	0.08	0.37	0.08	1.45	0.06
75	5.03	0.81	0.22	2.18	0.24	9.57	0.37
100	10.14	2.99	0.42	2.63	0.40	21.08	1.43

5.6 Comparisons with other procedures from the literature

In Table 5.17 we compare the average running times of our *bundle relaxation* code BR with the two best procedures HR and LR of [DeP03] on the instances used for Tabs. 5.1, 5.2 and 5.4–5.5. The times for HR and LR obtained on a Pentium Pro 200 MHz were extracted from [DeP03, Tabs. 1–4b]. Two points should be noted. First, both HR and LR employed an industrial LP solver (much more sophisticated than our dense QP solver), and LR additionally used subgradient optimization. Second, due to lacking knowledge, let's assume that the machine of [DeP03] was ten times slower than ours. Then Table 5.17 suggests that on the small-item-size instances BR was comparable in speed with HR (about twice slower than LR), while on the medium-item-size instances BR could beat LR. Similarly, in view of Tab. 5.6 and [DeP03, Tab. 10], on the instances of [WäG96] BR was as fast as HR (twice slower than LR), whereas Tab. 5.7 and [DeP03, Tab. 5a] indicate that on the instances of [Van99] BR was comparable with HR, and sometimes faster than LR. On the industrial instances of [DeS99] (cf. Tab. 5.10 and [DeP03, Tab. 9]), BR behaved like HR (sometimes better than LR).

When the dual objective evaluations happen to be exact, our BR code runs essentially like the standard bundle method used in [FeK00]. Therefore, we now summarize our results for exact KP solutions ($\epsilon_r = 0$) relative to Tab. 5.2 (where $\epsilon_r = 10^{-8}$); similar features were observed on other instances. First, the iteration numbers and the performance of our

Table 5.18: Industrial and random CSP instances of Briant et al. (2004)

instance source	m_{av}	m'_{av}	i_{av}	i_{mx}	t_{av}	t_{mx}
[BLM+05, Tab. 2.1]	18.00	56.89	30.00	69	0.02	0.10
[BLM+05, Tab. 2.2]	49.70	129.25	108.15	141	0.15	0.40
[BLM+05, Tab. 2.3]	18.00	57.89	26.44	40	0.08	0.51
[BLM+05, Tab. 2.4]	49.70	129.60	119.35	154	0.34	0.47

heuristics did not change significantly. (In other words, the errors occuring in the inexact case were small enough to be accommodated gracefully by our code.) Second, the running times increased quite dramatically: for $m = 30, 40, 50, 75$ and 100, the "*all*" average times grew by factors of 27.5, 60.6, 13.5, 31.7 and 33.2, respectively.

Finally, we compare our running times with those in [BLM+05, §2.2].

Remark 5.1. Our timings for u120 and u250 in Tab. 5.14 and for t120 and t249 in Tab. 5.15 were about the same without H1 through H4. Relative to [BLM+05, Tab. 2.5] (where the machine used was about twice slower than ours), our running times were shorter at least 57 times for u120/u250, 284 times for t120, and 227 times for t249. Table 5.18 gives our results for the remaining CSP instances of [BLM+05, §2.2]. Relative to [BLM+05, Tabs. 2.1–2.4], our running times were shorter 12, 25, 32 and 27 times, respectively.

Acknowledgment. I would like to thank 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.

References

[BADF04] H. Ben Amor, J. Desrosiers and A. Frangioni, *Stabilization in column generation*, Tech. Report G-2004-62, GERAD, Montreal, 2004.

[BAVdC04] H. Ben Amor and J. M. Valério de Carvalho, *Cutting stock problems*, Tech. Report G-2004-30, GERAD, Montreal, 2004.

[Bea90] J. E. Beasley, *OR-Library: Distributing test problems by electronic mail*, J. Oper. Res. Soc. **41** (1990) 1069–1072.

[BeS02] G. Belov and G. Scheithauer, *A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths*, European J. Oper. Res. **141** (2002) 274–294.

[BeS04a] ———, *A branch-and-cut-and-price algorithm for one- and two-dimensional two-stage cutting problems*, European J. Oper. Res. ? (2004). To appear.

[BeS04b] ———, *Setup and open stacks minimization in one-dimensional stock cutting*, INFORMS J. Comput. ? (2004). To appear.

[BLM+05] O. Briant, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot and F. Vanderbeck, *Comparison of bundle and classical column generation*, Research Report RR-5453, INRIA, Montbonnot, France, 2005.

[Chv83] V. Chvátal, *Linear Programming*, Freeman, New York, N.Y., 1983.

[DeP03] Z. Degraeve and M. Peeters, *Optimal integer solutions to industrial cutting stock problems: Part 2: Benchmark results*, INFORMS J. Comput. **15** (2003) 58–81.

[DeS99] Z. Degraeve and L. Schrage, *Optimal integer solutions to industrial cutting stock problems*, INFORMS J. Comput. **11** (1999) 406–419.

[FeK00] S. Feltenmark and K. C. Kiwiel, *Dual applications of proximal bundle methods, including Lagrangian relaxation of nonconvex problems*, SIAM J. Optim. **10** (2000) 697–721.

[GaW95] T. Gau and G. Wäscher, *CUTGEN1: A problem generator for the standard one-dimensional cutting stock problem*, European J. Oper. Res. **84** (1995) 572–579.

[GiG61] P. C. Gilmore and R. E. Gomory, *A linear programming approach to the cutting-stock problem*, Oper. Res. **9** (1961) 849–859.

[Hol02] O. Holthaus, *Decomposition approaches for solving the integer one-dimensional cutting stock problem with different types of standard lengths*, European J. Oper. Res. **141** (2002) 295–312.

[Kiw94] K. C. Kiwiel, *A Cholesky dual method for proximal piecewise linear programming*, Numer. Math. **68** (1994) 325–340.

[Kiw04] ———, *A proximal bundle method with approximate subgradient linearizations*, SIAM J. Optim. **?** (2004). To appear.

[KPP04] H. Kellerer, U. Pferschy and D. Pisinger, *Knapsack Problems*, Springer, Berlin, 2004.

[MaT90] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, New York, 1990.

[NST99] C. Nitsche, G. Scheithauer and J. Terno, *Tighter relaxations for the cutting stock problem*, European J. Oper. Res. **112** (1999) 654–663.

[ScW97] P. Schwerin and G. Wäscher, *The bin-packing problem: A problem generator and some numerical experiments*, Int. Trans. Oper. Res. **4** (1997) 337–389.

[Sta90] H. Stadtler, *One-dimensional cutting stock problem in the aluminium industry and its solution*, European J. Oper. Res. **44** (1990) 209–223.

[STMB01] G. Scheithauer, J. Terno, A. Müller and G. Belov, *Solving one-dimensional cutting stock problems exactly using a cutting plane algorithm*, J. Oper. Res. Soc. **52** (2001) 1390–1401.

[Van98] P. H. Vance, *Branch and price algorithms for the one-dimensional cutting stock problem*, Comput. Optim. Appl. **9** (1998) 212–228.

[Van99] F. Vanderbeck, *Computational study of a column generation algorithm for bin packing and cutting stock problems*, Math. Programming **86** (1999) 565–594.

[Van02] ———, *Extending Dantzig's bound to the bounded multiple-class binary knapsack problem*, Math. Programming **94** (2002) 125–136.

[WäG96] G. Wäscher and T. Gau, *Heuristics for the integer one-dimensional cutting stock problem*, OR Spectrum **18** (1996) 131–144.