Supplemental Material to: Combinatorial Benders' Cuts for the Strip Packing Problem

This electronic companion is structured as follows. In Section EC.1, we give the proof of Theorem 1. In Section EC.2 we graphically depict two optimal solutions whose structure was commented in the paper. In Section EC.3 we provide additional computational results.

EC.1. Proof of Theorem 1

In order to prove that $y-$check is strongly $\mathcal{NP}-$complete, we give a polynomial transformation from the following problem.

DEFINITION EC.1. 3-PARTITION: Given $\overline{n} = 3m$ items, each having weight $s_j \in Z^+$ $(j = 1, 2, \ldots, \overline{n})$, and a value $B \in Z^+$ such that $\sum_{j=1}^{\overline{n}} s_j = mB$, find a partition of the items into m disjoint subsets $S_1, S_2, \ldots, S_m$ such that $\sum_{j \in S_i} s_j = B$ holds for $i = 1, 2, \ldots, m$, if any.

In the proof of Lemma 1 we used nine items to obtain two $1 \times B$ buckets. Here we want to obtain m $1 \times B$ buckets, using an iterative method that adds nine more items at a time. We start by considering two $5 \times (2B+3)$ rectangles, each obtained by packing nine items as those of Figure 2 (see the 18 hatched items in Figure 3), and we embed them into nine new items following the same scheme used in Lemma 1. In this scheme, however, the new items produce two buckets of width five. In Figure 3 we depict this frame, by drawing in white the new items.

In details, let us call $1', 2', \ldots, 9'$ and $1'', 2'', \ldots, 9''$ the hatched items in the two buckets. Items $1'$ and $1''$ cannot be packed in the same $5 \times (2B+3)$ bucket, because they are too high. As a consequence also items $3'$ and $3''$ must be packed in different buckets. The same reasoning applies to items $2'$ and $2''$, and consequently to items $5'$ and $5''$. Continuing this reasoning one can show that also the remaining items must be packed as in Figure 2. We have thus created two copies of the packing of Figure 2 and four $1 \times B$ empty buckets. We now consider the resulting solution as a single $9 \times (4B+9)$ rectangle and we embed two of them in a frame of other nine items. We continue this process for, say, k times, until we create 2^k $1 \times B$ empty buckets with $2^k \geq m$ (the technical details on the widths, heights and $x-$coordinates of the items are given at the end of this proof).

Let α denote the number of items used to create the 2^k $1 \times B$ buckets. We complete the instance by adding $\overline{n}$ items with $w_j = 1$, $h_j = s_{j-\alpha}$ $(j = \alpha+1, \alpha+2, \ldots, \alpha+\overline{n})$ and p_j^s corresponding to the $x-$coordinate of the empty buckets, and other $2^k - m$ items with $w_j = 1$, $h_j = B$ and the same p_j^s of the previous ones. Each of the last $2^k - m$ items completely fills $2^k - m$ buckets, so leaving exactly m empty buckets. These can be feasibly filled by the remaining items if and only if 3-PARTITION has a feasible solution. Since 3-PARTITION is strongly $\mathcal{NP}-$complete, the same holds for $y-$check.

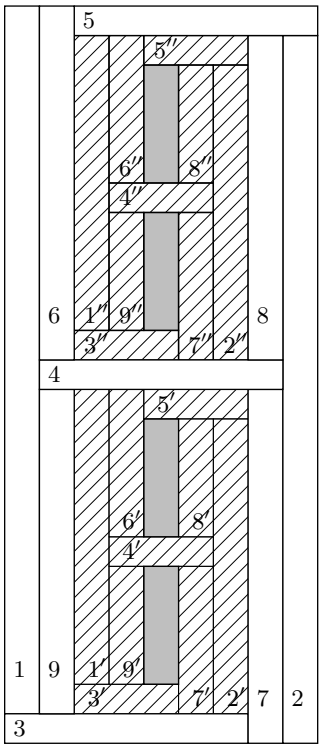

Figure EC.1 Basic frame for the reduction of 3-PARTITION to $y-$check.

Details on sizes and coordinates of the items. We constructed the instance by adding for k times nine items as those depicted in Figure 2. Let (i,j) denote the item of "type" i $(i=1,2,\ldots,9)$ used at iteration j $(j=1,2,\ldots,k)$. The items of type 1, 2, 6, 7, and 8 have width one, but height depending on the iteration, the items of type 3, 4, and 5 have height one, but width depending on the iteration. In particular:

$$
\begin{aligned}
&w_{(i,j)}=1,\ h_{(i,j)}=\ 2^jB+3(2^j-1)-1 && i=1,2\\
&w_{(i,j)}=1,\ h_{(i,j)}=h_{(1,j)}/2 && i=6,7\\
&w_{(i,j)}=1,\ h_{(i,j)}=h_{(6,j)}-1 && i=8,9\\
&h_{(i,j)}=1,\ w_{(i,j)}=\begin{cases}3 & \text{for } j=1\\ 3+4(j-1) & \text{for } j>1\end{cases} && i=3,4,5.
\end{aligned}
$$

The $x-$coordinate of an item (i,j) is:

$$
\begin{aligned}
&2(k-j) && i=1,3\\
&1+2(k-j) && i=4,6,9\\
&2+2(k-j) && i=5\\
&2^k-2(k-j) && i=2\\
&2^k-1-2(k-j) && i=7,8.
\end{aligned}
$$

This information concludes the proof. □

EC.2. Graphical representation of two optimal solutions

In Figure EC.2 we depict an optimal solution of instance cgcut03 by Christofides and Whitlock (1977), and in Figure EC.3 an optimal solution of instance gcut04 by Beasley (1985). Both figures are scaled, so that a unit on the height is one half of a unit on the width.

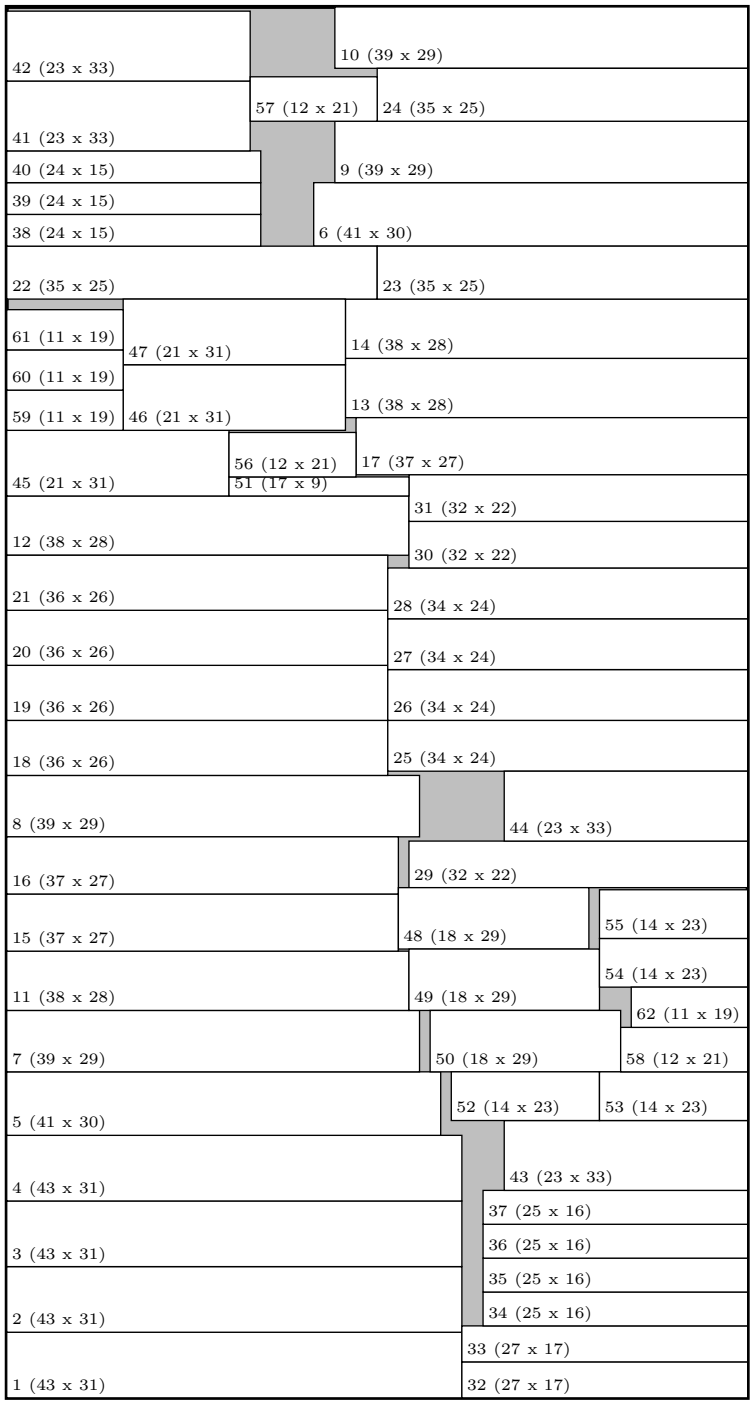

Figure EC.2 Optimal solution of instance cgcut03 (W=70, $z = 656$).

As described in the paper, these two solutions are characterized by complex non-guillotine structures, that create large holes and make difficult the computation of both the lower and the upper bound. The solutions that we obtained on all other instances, either proven optimal or heuristic, are available for download on our web site `www.or.unimore.it\resources\SPP.html`.

Figure EC.3 Optimal solution of instance gcut04 (W=250, $z = 2995$).

EC.3. Additional Computational Results

In Table EC.1 we give the results of our algorithm on the "easy" benchmark sets ngcut, ht and beng. As done in the paper, in column "opt" we report a "*" if the instance is solved to proven optimality, in column "sec" we give the computational time, and in column z we report the solution value found by BLUE. We recall that we compare with: MMV03 = Martello et al. (2003), BKC07 = the most performing algorithm (DA) by Bekrar et al. (2007), APT09 = Alvarez-Valdes et al. (2009), KINYN09 = the most performing algorithm (G-STAIRCASE) by Kenmochi et al. (2009), BM10 = Boschetti and Montaletti (2010), CO11 = the most performing algorithm (DS) by Castro and Oliveira (2011), and AIT12 = Arahori et al. (2012).

Table EC.1 Results and comparison on nguct, ht, and beng instances.

			MMV03 0.8GHz t.l.=3600s		BKC07 1.7GHz t.l.=3600s		APT09 2GHz t.l.=1200s		KINYN09 3GHz t.l.=3600s		BM10 1.6GHz t.l.=1200s		CO11 2.5GHz t.l.=3600s		AIT12 3.3GHz t.l.=3600s		BLUE 2.33GHz t.l.=1200s		
name	n	W	opt	sec	opt	sec	opt	sec	opt	sec	opt	sec	opt	sec	opt	sec	z	opt	sec
ngcut01	10	10	*	0.05	*	23.20	*	2.20	*	0.39	*	0.08	*	81.60	*	0.00	23	*	0.19
ngcut02	17	10	*	11.31	*	1052.72	*	3.10		*t.lim.*	*	0.47		*t.lim.*	*	0.07	30	*	0.08
ngcut03	21	10	*	27.01	*	519.70	*	0.00	*	0.10	*	1.62	*	12.60	*	0.00	28	*	0.03
ngcut04	7	10	*	0.00	*	0.02	*	0.00	*	0.14	*	0.14	*	1.22	*	0.00	20	*	0.04
ngcut05	14	10	*	0.00	*	119.58	*	0.00	*	0.07	*	0.34	*	3.18	*	0.00	36	*	0.02
ngcut06	15	10	*	727.20	*	1079.26	*	4.60	*	147.31	*	0.84		*t.lim.*	*	0.16	31	*	0.41
ngcut07	8	20	*	0.00	*	0.00	*	0.00	*	0.10	*	0.38	*	0.98	*	0.00	20	*	0.01
ngcut08	13	20	*	53.09	*	178.06	*	3.50	*	0.50	*	15.39	*	29.50	*	0.06	33	*	0.36
ngcut09	18	20		*t.lim.*	*	1269.83	*	58.10	*	1971.64	*	286.55		*t.lim.*	*	3.48	50	*	0.67
ngcut10	13	30	*	0.18	*	1152.83	*	2.60	*	113.98	*	6.58		*t.lim.*	*	0.01	80	*	0.06
ngcut11	15	30	*	483.01	*	733.18	*	13.80	*	7.71	*	107.47		*t.lim.*	*	0.04	52	*	0.44
ngcut12	22	30	*	0.00	*	866.32	*	0.00		*t.lim.*	*	1.41		*t.lim.*	*	0.00	87	*	0.03
tot opt/avg sec			11	118.35	**12**	582.89	**12**	7.33	10	224.19	**12**	35.11	6	21.51	**12**	0.32		**12**	0.20
ht01	16	20	*	10.84	*	0.00	*	0.00	*	0.07	*	3.84	*	2.08	*	0.00	20	*	0.02
ht02	17	20	*	3043.25	*	378.81	*	0.40	*	0.07	*	149.98	*	5.28	*	0.00	20	*	0.25
ht03	16	20	*	500.75	*	197.53	*	0.10	*	0.10	*	1.22	*	2.51	*	0.00	20	*	0.03
ht04	25	40	*	8.26	*	874.05	*	0.10	*	0.11	*	611.70	*	91.80	*	0.00	15	*	0.06
ht05	25	40	*	20.29	*	571.65	*	0.10	*	0.06	*	300.95	*	20.40	*	0.00	15	*	0.06
ht06	25	40	*	16.94	*	0.00	*	1.40	*	0.06	*	25.79	*	18.30	*	0.00	15	*	0.05
ht07	28	60		*t.lim.*		*n.a.*	*	1.80	*	0.10	*	654.56	*	3771.00	*	0.01	30	*	0.06
ht08	29	60		*t.lim.*		*n.a.*		*t.lim.*	*	76.97	*	732.05		*t.lim.*	*	23.77	30	*	57.66
ht09	28	60	*	0.00	*	0.00	*	8.70	*	0.13	*	669.90		*t.lim.*	*	0.00	30	*	0.07
tot opt/avg sec			7	514.33	7	288.86	8	1.58	**9**	8.63	**9**	350.00	7	558.77	**9**	2.64		**9**	6.47
beng01	20	25	*	911.37	*	608.41	*	5.50	*	0.93	*	26.95			*	0.25	30	*	0.67
beng02	40	25		*t.lim.*		*t.lim.*	*	0.40	*	22.89	*	72.55			*	5.43	57	*	0.57
beng03	60	25		*t.lim.*		*t.lim.*	*	0.50	*	0.32	*	69.42			*	0.02	84	*	0.21
beng04	80	25		*t.lim.*		*t.lim.*	*	3.30		*t.lim.*	*	198.22			*	0.01	107	*	1.46
beng05	100	25	*	500.62		*t.lim.*	*	0.10	*	0.31	*	73.42			*	0.03	134	*	0.46
beng06	40	40		*t.lim.*		*t.lim.*	*	0.10	*	0.29	*	82.41			*	0.00	36	*	0.18
beng07	80	40	*	0.56		*t.lim.*	*	0.10	*	0.18	*	607.33			*	0.04	67	*	2.47
beng08	120	40	*	500.54		*t.lim.*	*	0.10	*	2.67	*	93.73			*	0.01	101	*	0.72
beng09	160	40	*	0.03	*	0.00	*	0.10	*	2.38	*	76.03			*	3.65	126	*	1.04
beng10	200	40	*	0.03		*n.a.*	*	0.10	*	6.52	*	82.85			*	0.25	156	*	1.60
tot opt/avg sec			6	318.86	2	304.21	**10**	1.03	9	4.05	**10**	138.29			**10**	0.97		**10**	0.94

References

Alvarez-Valdes, R., F. Parreño, J. Tamarit. 2009. A branch and bound algorithm for the strip packing problem. *OR Spectrum* **31** 431–459.

Arahori, Y., T. Imamichi, H. Nagamochi. 2012. An exact strip packing algorithm based on canonical forms. *Computers & Operations Research* **39** 2991–3011.

Beasley, J. E. 1985. Algorithms for unconstrained two-dimensional guillotine cutting. *Journal of the Operational Research Society* **36** 297–306.

Bekrar, A., I. Kacem, C. Chu. 2007. A comparative study of exact algorithms for the two dimensional strip packing problem. *Journal of Industrial and Systems Engineering* **1** 151 – 170.

Boschetti, M.A., L. Montaletti. 2010. An exact algorithm for the two-dimensional strip-packing problem. *Operations Research* **58** 1774–1791.

Castro, P.M., J.F. Oliveira. 2011. Scheduling inspired models for two-dimensional packing problems. *European Journal of Operational Research* **215** 45 – 56.

Christofides, N., C. Whitlock. 1977. An algorithm for two-dimensional cutting problems. *Operations Research* **25** 30–44.

Kenmochi, M., T. Imamichi, K. Nonobe, M. Yagiura, H. Nagamochi. 2009. Exact algorithms for the two-dimensional strip packing problem with and without rotations. *European Journal of Operational Research* **198** 73 – 83.

Martello, S., M. Monaci, D. Vigo. 2003. An exact approach to the strip packing problem. *INFORMS Journal on Computing* **15** 310–319.