Volume 30 (2), pp. 73–83
http://orion.journals.ac.za

ORiON
ISSN 0529-191-X

Efficient waste reduction algorithms based on alternative underestimates for the modified Wang method

JA Oberholzer* JM Hattingh† T Steyn‡

Received: 26 November 2012; Revised: 7 August 2013; Accepted: 15 January 2013

Abstract

This paper is concerned with wastage reduction in constrained two-dimensional guillotine-cut cutting stock problems, often called trim loss problems. A number of researchers report in the literature on algorithmic approaches to find exact solutions for the trim loss problem. Alternative heuristic functions are investigated and applied to the modified Wang method. This involves the sharpening of underestimates used in the methods heuristic function. Two aspects of these solution approaches are considered and some empirical results are given. The first part considers the feasibility to construct more informed heuristic functions. The second part investigates the role of more informedness on the computational cost of these search processes.

Key words: Cutting stock, trim loss, guillotine cutting.

1 Introduction

The constrained two-dimensional guillotine-cut cutting stock (C2DGC) problem is considered here. An algorithm is introduced that is based on the algorithms proposed by Wang [10] denoted by WA and a modified WA-algorithm by Oliveira and Ferreira [8] denoted by WAM in this paper. These algorithms still form the core of exact trim loss algorithms.

*Department of Computer Science and Information Systems, North-West University, Potchefstroom Campus, Private Bag X6010, Potchefstroom, 2530, South Africa.

†**(Fellow of the Operations Research Society of South Africa)**, Department of Computer Science and Information Systems, North-West University, Potchefstroom Campus, Private Bag X6010, Potchefstroom, 2530, South Africa.

‡Corresponding author: Department of Computer Science and Information Systems, North-West University, Potchefstroom Campus, Private Bag X6010, Potchefstroom, 2530, South Africa, email: tjaart.steyn@nwu.ac.za

http://dx.doi.org/10.5784/30-2-143

The WA method exploits a way of building larger rectangles by joining smaller ones. Wang proposed two algorithms based on this method to solve C2DGC problems. Using this bottom-up rectangle building approach, larger rectangles are gradually generated from the rectangles and rotated versions of the rectangles. They also distinguish between "horizontal" and "vertical" builds in the building process. Every newly generated rectangle may contain trim loss. This trim loss is called internal trim loss for the build. Further trim loss can in general be expected if this build is subsequently combined with other rectangles to form a larger build and the process can be repeated until no further expansion can take place as limited by the dimensions of the stock sheet. Such a build is called a maximal build. The unused part of the stock sheet (if any) also contributes to trim loss. We refer to the latter trim loss as external trim loss. Pertaining to this maximal build the total trim loss of the stock sheet is thus the sum of the internal trim loss for the build and the external trim loss associated with the build.

The WAM algorithm employs techniques of Gilmore and Gomory [4] to (under)estimate the expected total trim loss associated with a build. In the first part of this paper some ideas are investigated to sharpen these (under)estimates to possibly reduce the search space by means of more informed heuristics. The typical search framework is employed where the evaluation function (of a node to be considered in the search) consists of the sum of a cost function and a heuristic function. The evaluation function in this case consists of the "cost" of a build (internal trim loss) plus a heuristic (under)estimate of further "costs" (trim loss external to the build). Some empirical results are given. In the second part of the paper the results are interpreted in the sense that the trade-off is considered between the benefits of informedness and the higher computational cost of the more complex control strategies of the search.

The remainder of the paper is structured as follows. In §2, the problem is presented and current algorithmic approaches used to solve the problem, are discussed. In §3 an algorithm to solve some larger, industry-sized real life problem instances are presented and §4 contains an example illustrating some of these concepts. In §5 empirical tests and results are shown where the effectiveness of the algorithmic approaches is evaluated.

2 Problem representation and algorithmic approaches

The C2DGC problem may be defined as follows. Let S be a stock sheet of length $L \in I^+$ and width $W \in I^+$ where $L \geq W$, and let R be a set of demand rectangles. These rectangles have types r_i $(i = 1, \ldots, n)$, where each type has a demand constraint of $b_i \in I^+$, a length of $\ell_i \in I^+$ and a width of $w_i \in I^+$. Since rotation is allowed in this model, it is assumed that there is a demand constraint b_i concerning the maximal number of rectangles of a certain type r_i to be produced taking equivalence regarding rotation into account. A guillotine cutting pattern with minimum trim loss must thus be determined that produces no more than b_i replicates of rectangle type r_i.

The WA-algorithm [10] given in Algorithm 1 terminates with an optimal (exact) solution to the C2DGC problem. Note that when the WA-algorithm returns to Step 1(b) (from Step 4) with an increased β value, it discards all previous builds and information. Note also that the WA-algorithm considers storage of a rectangle in Step 2 (a)(ii) using only

internal trim loss.

Algorithm 1: WA-Algorithm

1. (a) Choose a value of β, $0 \leq \beta < 1$ and an increment $\delta > 0$. Typically choose δ close to zero, for instance $\delta = 0.01$.
   (b) Set $L^{(0)} = R$ and set $k = 1$.
2. (a) Compute $F^{(k)}$ as the set of all rectangles T satisfying:
      i. T results from a feasible horizontal or vertical build of two rectangles from $L^{(k-1)}$.
      ii. The internal trim loss of T does not exceed $\beta \cdot L \cdot W$.
      iii. T does not violate the demand of constraints $b_1, \ldots, b_n$.
   (b) Set $L^{(k)} = L^{(k-1)} \cup F^{(k)}$.
3. If $L^{(k)} \neq L^{(k-1)}$, set $k \leftarrow k + 1$ and go to Step 2.
4. For each maximal build of $L^{(k)}$ compute the total trim loss and select the build with the smallest total trim loss. If it is less than or equal to $\beta \cdot L \cdot W$, it is optimal and the algorithm terminates. Otherwise, increase β by δ and go to Step 1(b).

In Oliveira and Ferreira [8] the same representation and building process as the WA method is used with the main difference that the requirement in Step 2 (a)(ii) is changed. They require that the internal trim loss of T plus an underestimate of further trim loss incurred by expanding builds based on T should not exceed $\beta \cdot L \cdot W$. The underestimate of Oliveira and Ferreira is based on the solution of a two-dimensional knapsack function as described by Gilmore and Gomory [4]. Daza *et al.* [3] propose a different way of calculating the underestimates that is motivated by the desire to find a monotone and admissible heuristic for the search procedure.

Neither Oliveira and Ferreira or Daza *et al.* discuss the possible re-use of previous build information in the design of the algorithms. In this paper the use of build information to produce sharper underestimates is considered.

3 Algorithms based on alternative heuristic functions

The WAM-algorithm [8] employs at every coordinate of a lookup table (as used in their evaluation function to identify builds that should be stored) underestimates of further waste using the underestimates produced by solving an unbounded two-dimensional knapsack function. In this section a procedure is explored that can be used to produce underestimates either by superimposing it on the estimates provided by the Gilmore and Gomory method [4] or using it in a standalone fashion using other easily obtained underestimates. These algorithms are based on the information obtained by the application of the WA-algorithm to a portion of the stock sheet and generating underestimates as part of the process. The details of the proposed PSSP-algorithm (partial stock sheet propagation algorithm) are given in Algorithm 2. The PSSP-algorithm is given as an example of methods that can be employed to improve the informedness of heuristic underestimates like that of Gilmore and Gomory.

In this paper the following four approaches are considered although there may be others that can be devised:

Algorithm 2: The PSSP-algorithm

1. Choose a value of β, $0 \leq \beta < 1$ and a value for δ. (Normally close to zero).
2. Initialize an array A with dimensions W and L by using underestimates of waste at every coordinate. Specifically the unbounded two-dimensional knapsack function of Gilmore and Gomory [4] can be used. Fill each cell of A with the value obtained. Note that any legitimate underestimates will also suffice (like underestimates of zero).
3. Choose S such that $0 < S \leq 1$ (§5 discusses possible choices).
4. Consider the partial stock sheet with dimensions $\lfloor S \cdot W \rfloor$ and L. Apply the WA-algorithm to the "stock sheet" with the current β-value. When Step 4 of the WA-algorithm is reached (for the first time) $L^{(k)}$ has elements containing internal waste that correspond to certain cells(x, y). Designate these as waste$(L_i^{(k)}, x, y)$ where $L_i^{(k)}$ is the i^{th} with dimensions x, y.
5. Initialize array B as follows:
   If no waste$(L_i^{(k)}, 1, 1)$ in the list exists, define waste$(L_1^{(k)}, 1, 1) = 1$
   For $x = 1, \ldots, \lfloor S \cdot W \rfloor$ and $y = 1, \ldots, L$ **do:**
   Define waste$(L_1^{(k)}, x, y) = \lfloor \beta \cdot L \cdot \lfloor S \cdot W \rfloor \rfloor + 1$ if no waste$(L_i^{(k)}, x, y)$ exists in the list and $(x, y) \neq (1, 1)$.
   Set $B(x, y) = \min_i \left[\text{waste}(L_i^{(k)}, x, y)\right]$.
6. Calculate the minimum total trim loss for the subsheet with dimensions (x, y) for each (x, y) by executing the following propagation scheme on the elements of the array B (in a breadth first manner):
   $B(1, y) = \min[B(1, y), B(1, y - 1) + 1]$ for $y = 2, \ldots, L$.
   For $x = 2, \ldots, \lfloor S \cdot W \rfloor$ **do:**
   $B(x, 1) = \min[B(x, 1), B(x - 1, 1) + 1]$
   $B(x, y) = \min[B(x, y), B(x - 1, y) + y, B(x, y - 1) + x]$ for $y = 2, \ldots, L$.
7. Update the underestimates as represented in array A:
   For $x = 1, \ldots, \lfloor S \cdot W \rfloor$ and $y = 1, \ldots, L$ **do:**
   Set $A(x, y) = \max(B(x, y), A(x, y))$.
8. To add symmetrical builds to the lookup table, do the following:
   For $x = \lfloor S \cdot W \rfloor + 1, \ldots, W$ and $y = 1, \ldots, \lfloor S \cdot W \rfloor$
   $A(x, y) = A(y, x)$
9. Solve the original trim loss problem for the stock sheet with dimensions W and L with the WAM-algorithm using the tighter lower bounds stored in array A and the current β-value; and
10. If an optimal solution was found for the problem instance (best build is found where totaltrimloss $\leq \beta \cdot L \cdot W$), stop. Otherwise increase β by δ and return to Step 4.

1. WA as the original Wang-algorithm (Wang [10]);
2. WAM(GG) as introduced by Oliveira and Ferreira [8] with Gilmore-Gomory underestimates;
3. WAM(GG + PSSP). The same as WAM(GG) with estimates enhanced according to the PSSP method and
4. WAM(0 + PSSP). Initial underestimates of zero enhanced by the PSSP method.

Note that the choice of S in Step 3 strives to find a trade-off between the volume of calculations and effective underestimate improvement in as large a portion of the lookup table as possible. For small problems it should be feasible to choose $S = 1$. The underestimates obtained in Steps 7 and 8 are used as a more informed heuristic in the WAM-algorithm (Step 9) and as such it prunes the search space. In the rest of this paper the trade-off of additional work to improve underestimates and the beneficial effects of a pruned search space are investigated. In the proposition below, the validity of the underestimates generated

by the PSSP-algorithm are formally considered.

Proposition 1
For each choice β^* of β (assigned in Steps 1 or 10 of the PSSP-algorithm) Step 7 registers a sequence of non-decreasing underestimates for total trim-loss at every coordinate (x, y) of A.

Proof: If for a particular (x, y) it follows that $B(x, y) \leq \beta^* \cdot L \cdot \lfloor S \cdot W \rfloor$ at Step 5, it is a candidate solution to the trim-loss problem corresponding to the partial sheet (x, y). There is, however, the possibility that a build(s) exists for a partial sheet (strictly) contained in the (x, y) partial sheet with the associated (internal) waste less than or equal to $\beta^* \cdot L \cdot \lfloor S \cdot W \rfloor$ and the total trim-loss calculated with respect to the partial sheet (x, y) also less than or equal to $\beta^* \cdot L \cdot \lfloor S \cdot W \rfloor$. Such a build will then be a candidate for the optimal cutting pattern for (x, y). The detection of such builds is accomplished by Step 6. The $A(x, y)$ is updated in Step 7 (and 8 for symmetry). This is only done if $B(x, y) > A(x, y)$. This condition guarantees optimality of the minimal waste for the (x, y) partial sheet since the argument is precisely that of Wang [10]. For the $B(x, y)$ calculated in Steps 4, 5 and 6 there are only two possibilities.

- $B(x, y) \leq \beta^* \cdot L \cdot \lfloor S \cdot W \rfloor$ in which case $B(x, y)$ is the optimal (least waste) solution for the partial stock sheet (x, y) (according to the Wang [10] argument), or
- $B(x, y) = \lfloor \beta^* \cdot L \cdot \lfloor S \cdot W \rfloor \rfloor + 1$, which is true since in such a case no solution exists with waste is less than or equal to $\beta^* \cdot L \cdot \lfloor S \cdot W \rfloor$.

These facts establish that $B(x, y)$ are underestimates of waste for the partial stock sheet (x, y). Subsequent underestimates are non-decreasing in the light of Step 7. ■

4 An example of the lookup table calculation in the PSSP-algorithm

Consider a stock plate with width 5 and length 5 units. It has two demand rectangles. The first has a length, width and upperbound of 1 unit, while the second has a width of 2, length of 3 and an upperbound of 5. This example (EP 1) will be used to demonstrate a typical iteration of the PSSP-algorithm to generate a lookup table of underestimates.

Figure 1 shows the possible builds that are generated by the WA-algorithm with a β-value of 0.24 and a partial stock sheet with dimensions (3,5). (The maximal builds are e, g and h).

A typical iteration of Algorithm 2 to demonstrate the generation of a lookup table of underestimates by utilizing the patterns generated (Figure 1) is shown below.

Step 1. A β-value of 0.24 is arbitrarily chosen for illustrative purposes (through previous experimentation it was found that this β-value is required for the algorithm to generate an optimal solution). It should be noted that any other β-value could also have been chosen with $0 \leq \beta < 1$; Choose δ such that $\delta \cdot L \cdot \lfloor S \cdot W \rfloor = 1$ (to make calculations simple).

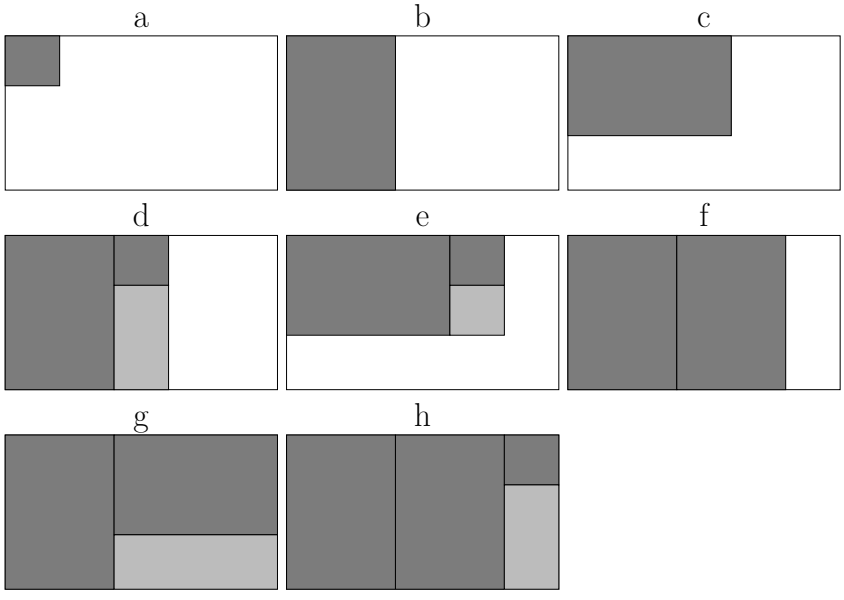


Figure 1: *Builds for the example problem instance (EP1).*

Step 2. Initialize array A with underestimates calculated by the Gilmore and Gomory knapsack [4]. For this problem instance, because a demand piece with dimensions $(1, 1)$ exists, the knapsack generates underestimates of 0 at each dimension (because the knapsack assumes that there is no bound on the number of demand pieces that can be used). One could also initialize A with underestimates of zero without executing the Gilmore and Gomory knapsack for any problem instance, which would trivially also represent feasible underestimates. (Figure 2(i)).

Step 3. For demonstrative purposes, a value of 0.6 is chosen for S $(0 < S \leq 1)$. This results in a partial stock sheet with dimensions $(\lfloor S \cdot W \rfloor, L) = (3, 5)$.

Step 4. Solve the problem with a partial stock sheet of dimensions $\lfloor S \cdot W \rfloor$ and L (for the original demand pieces) with the WA-algorithm (Figure 2(ii)). See Figure 1 for possible builds.

Step 5. Initialize array B. Choose the smallest internal trim loss build for each dimension in the partial stock sheet from the list of builds generated in Step 4 and write it to array B, if such a build exists. Otherwise, set the value to $\lfloor \beta \cdot L \cdot \lfloor S \cdot W \rfloor \rfloor + 1$ (Figure 2(iii)).

Step 6. Execute the propagation scheme in the PSSP-algorithm. The data in each cell in Figure 2(iv) shows the intermediate calculations. The number at the bottom of each cell gives the underestimate.

Step 7. Update the lookup table (A) where possible with improved under-estimates.

Step 8. Symmetrical equivalent values are also updated in the final full lookup table (Figure 2(v) and 2(vi)).

If the Gilmore-Gomory underestimates in Figure 2(i) is compared to the "sharper" underestimates calculated with the PSSP-algorithm given in Figure 2(vi) it is found that many

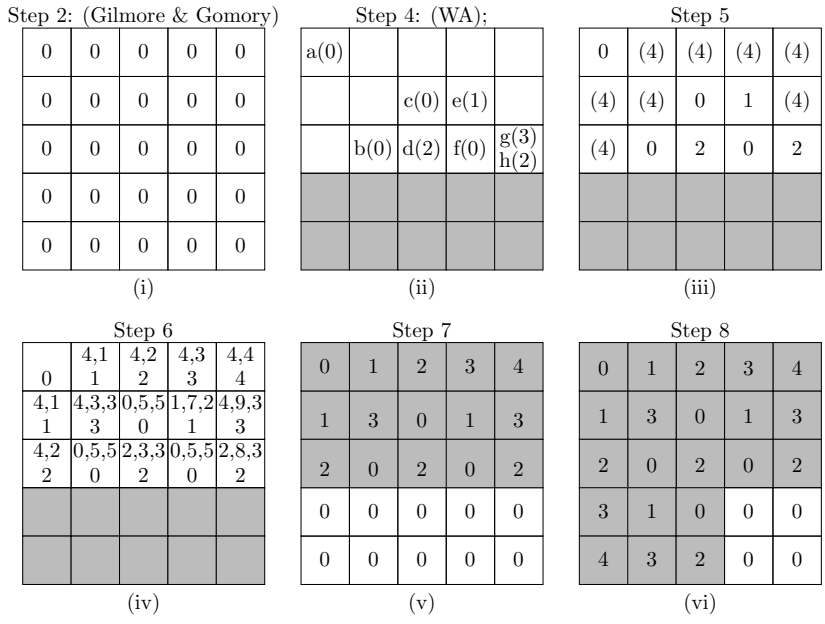


Figure 2: *Demonstration of the generation of underestimates.*

of the estimates are more "informed."

5 Empirical results

Different data sets exist in literature to compare algortihms. The first eight problem instances used in this paper are from Daza *et al.* [3] and are given in Table 1.

Problem	Stock plate (W, L)	Demand rectangles(w, ℓ, b)
P1	(10,15)	(1,2,1); (2,3,2); (3,3,2); (3,4,5); (2,8,3); (3,7,1); (4,8,2)
P2	(40,70)	(9,17,1); (11,19,4); (12,21,3); (14,23,4); (15,24,1); (15,24,2); (16,25,4); (17,27,2); (18,29,3), (21,31,3); (22,32,2); (23,33,3); (24,34,2); (25,35,2); (26,36,1); (27,37,1); (28,38,1); (29,39,1); (30,41,1); (31,43,1).
P3	(40,70)	(5,29,1); (9,39,4); (9,55,1); (15,39,1); (11,16,2); (21,23,3); (14,29,4); (16,19,3); (9,36,2); (4,22,2).
P4	(40,70)	(18,22,2); (10,40,1); (13,27,3); (18,23,2); (8,29,4); (4,16,1); (9,47,1); (19,19,4); (13,16,2); (16,36,4).
P5	(4,8)	(1,2,1); (3,3,3); (1,4,2); (2,2,3).
P6	(12,30)	(3,6,3); (4,7,2); (5,7,1); (2,8,3); (5,8,3); (6,9,2); (7,9,1); (4,10,4); (5,15,3); (6,19,2); (4,8,2); (4,9,1).
P7	(32,47)	(13,14,1); (11,15,2); (7,14,30); (12,20,4); (11,17,3); (14,16,3); (8,24,2); (8,17,3); (13,16,1); (16,19,2).
P8	(50,55)	(11,13,3); (17,28,3); (23,23,3); (16,18,3); (15,18,1); (18,23,1); (15,24,4); (16,24,2); (27,29,2); (10,31,4).

Table 1: *The eight problem instances from Daza* et al. *[3].*

Table 2 displays data for each problem that were solved using the WAM(GG)-algorithm

as introduced by Oliveira and Ferreira [8]. Furthermore, it also displays the results for the WAM(GG+PSSP)-algorithm, *i.e.* each problem were solved using the WAM(GG)-algorithm in Algorithm 2, Step 9 with updated Gilmore-Gomory underestimate values (PSSP) based on the solutions of the partial sub-problem.

The partial sub-problems were solved using the original demand rectangles and a stock sheet of the dimensions $(\lfloor S \cdot W \rfloor, L)$. The choice of S could conceivably be handled in a few different ways. Through empirical tests in Oberholzer [7] it was found that $S = 0.25$, is generally a good choice.

	P1	P2	P3	P4	P5	P6	P7	P8
WAM(GG) Time (sec)	0.62	0.67	0.74	0.55	0.14	0.64	0.56	0.45
WAM(GG+PSSP) Total time (sec)	—	0.64	0.73	0.53	—	—	0.59	0.45
Trim-loss	0	29	43	31	0	0	8	34

Table 2: *Sharper estimates on small problem instances (P1 to P8).*

Most authors, including Daza *et al.* [3] and Christofides and Whitlock [2] report empirical results for the WA- and WAM-algorithms in general for small, textbook-sized problem instances. For the test problems $(\text{P1}, \ldots, \text{P8})$ considered it is clear that the problems can be solved very easily and that there is very little merit to further calculate tighter underestimates using algorithms based on the PSSP ideas. It is uncertain how well these methods scale when larger problems are used. In the last part of this section some results are presented based on the solution of larger, industry-sized problem instances with algorithms based on the WA-, WAM- and PSSP-methods. The problem instances were obtained from a local corporation that manufactures and sells glass. Table 3 summarizes these problem instances.

Problem	Stock plate (W, L)	Demand rectangles
PG1	(2 000,2 800)	(290,1 440,3); (585,955,1); (560,925,17); (290,950,12); (956,1 195,2); (1 195,1 440,5); (1 440,1 490,1)
PG2	(2 550,3 210)	(1 130,1 150,108); (894,1 130,162); (889,1 264,108); (979,1 332,108); (1 064,1 086,108); (804,1 264,108); (753,1 330,54)
PG3	(1 500,2 125)	(290,1 440,3); (585,955,1); (560,925,17); (290,950,12); (955,1 195,2); (1 140,1 195,5); (1 140,1 490,1).
PG4	(1 000,1 500)	(129,290,20); (355,585,4); (560,925,17); (290,950,12); (395,555,2); (650,796,5); (200,324,10).

Table 3: *Set of four C2DGC problem instances from a glass merchant.*

Table 4 shows the results obtained when the larger problem instances are solved with the WA-, WAM(GG)- and PSSP-algorithms. The partial sub-problems for the PSSP-algorithm are once again solved using the original demand rectangles and a stock sheet of the dimensions $(\lfloor S \cdot W \rfloor, L)$. The results for the partial sub-problems are not shown separately in Table 4, but the PSSP totals incorporate the calculation effort.

Considering the results in Table 4, it is clear that the WAM(GG)-algorithm does not perform well for problem instances with larger stock sheets. The reason for this is that if, for instance, a lookup table of underestimates must be calculated using the Gilmore-Gomory approach for problem PG2, an underestimate for each cell of the matrix must

be calculated. This translates to a staggering $2\,550 \times 3\,210 = 8\,185\,500$ underestimates! Table 4 further shows that to compute these values for PG2, 454.28 seconds of processing time is required. To solve problem PG2 with the WA-algorithm, requires only 9.87 seconds. This shows that the WAM-algorithm does not scale well. It should be noted, though, that after the underestimates are available, the solving process completion is very fast for the WAM- algorithm. Therefore, the underestimates obtained through the unbounded two-dimensional knapsack are valuable if available, but it generally takes long to calculate. This implies that if another method, that executed faster, could be devised to calculate underestimates; these resulting initial underestimates would also be of value.

Problem	Algorithm	Generated nodes (N)	Stored nodes (L)	Processing time (sec)	Trim loss
PG1	Total: WA	1 766 454	1 974	5.31	96 525
	GG Table			274.67	
	Total: WAM(GG)	14 928	160	275.59	96 525
	Total: WAM(0+PSSP)	660 566	1 166	3.27	96 525
PG2	Total: WA	8 244 610	3 756	9.87	302 086
	GG Table			454.28	
	Total: WAM(GG)	8 866	151	456.00	302 086
	Total: WAM(0+PSSP)	633 418	1 131	5.60	302 086
PG3	Total: WA	133 946	559	0.94	72 175
	GG Table			139.06	
	Total: WAM(GG)	2 118	71	139.48	72 175
	Total: WAM(0+PSSP)	45 840	357	0.87	72 175
PG4	Total: WA	934 468	1 504	4.12	8 610
	GG Table			55.15	
	Total: WAM(GG)	20 200	156	55.38	8 610
	Total: WAM(0+PSSP)	263 086	1 030	1.93	8 610

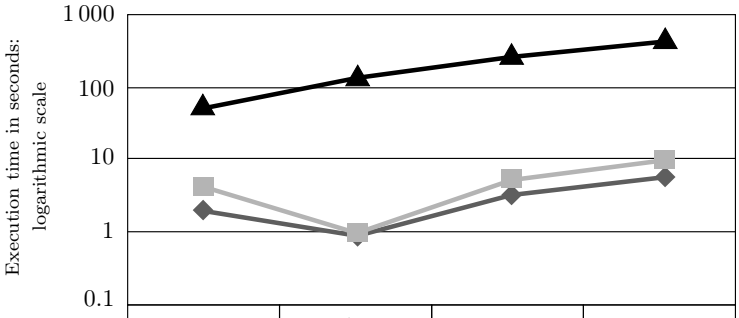
Table 4: *A summary of the results when the PSSP-algorithm was applied to problems from industry.*

For the PSSP-algorithm, the results in Table 4 were obtained by simply using initial underestimates of 0 to initialize the PSSP-algorithm, thereby eliminating the need for costly calculations of underestimates.

The results show that by eliminating the underestimate calculation using the Gilmore-Gomory approach, processing time decreases significantly. Lastly, if the calculation time for the underestimate table is not considered, it should be noted that the WAM(GG)-algorithm completes very fast. Therefore, if a method can be devised where underestimates (not necessarily as good as the Gilmore-Gomory estimates but better than 0) are produced quickly, the PSSP-algorithm might be enhanced.

Figure 3 shows that the PSSP-algorithm with initial underestimates of zero does indeed scale well for problem instances where larger stock sheets are concerned.

An observation illustrated in Figure 4, adapted from Luger and Stubblefield [5], applies to the modified Wang- and PSSP-algorithms and their respective mechanisms to generate underestimates with various levels of informedness. Referring to Figure 4, the total cost of solving the given problem must be minimized. To accomplish this, a strategy to calculate



	1 500 000 PG4	3 187 500 PG3	5 600 000 PG1	8 185 500 PG2
WAM (0+PSSP)	1.93	0.87	3.27	5.6
WA	4.12	0.94	5.31	9.87
WAM (GG)	55.38	139.48	275.59	456

Figure 3: *Scalability of the PSSP-algorithm with initial underestimates of zero.*

underestimates must be chosen that does not have calculation times that spiral out of control, and yet delivers results that can decrease the application cost of the algorithm used as solver. The modified Wang method fails here, according to Figure 3, where the control strategy becomes costly as the stock sheet size increases. The PSSP-algorithm, on the other hand, succeeds in generating effective underestimates at a reasonable cost.

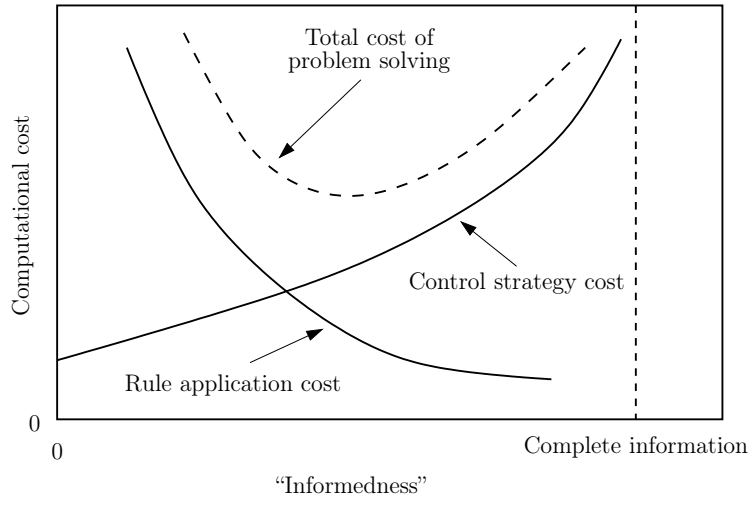


Figure 4: *A plot of the different computational costs as a function of informedness.*

6 Conclusion

This paper introduces alternative algorithms called partial stock sheet propagation (PSSP) algorithms, which aim at improving the lower bounds as produced by the unbounded two-dimensional knapsack function of Gilmore and Gomory. Improvements on execution time,

generated nodes and stored nodes were illustrated through the results of some numerical experiments. For larger, more complex, industry type of problems, the Gilmore and Gomory estimates were found to be computationally costly to produce. The simple approach to start with underestimates of zero and to improve them using the PSSP algorithm described in this paper turned out to be effective. However, the main practical result found in this research was that the easy to find less informed heuristic functions performed well (computationally) on more complex problems in producing exact trim loss solutions. For small problems the computational cost of most of the approaches is comparable.

References

[1] Christofides N & Hadjiconstantinou E, 1995, *An exact algorithm for orthogonal 2-D cutting problems using guillotine cuts*, European Journal of Operational Research, **83(1)**, pp. 21–38.

[2] Christofides N & Whitlock C, 1977, *An algorithm for two-dimensional cutting problems*, Operations Research, **25(1)**, pp. 30–44.

[3] Daza VP, De Alvarenga AG & Diego J, 1995, *Exact solutions for constrained two-dimensional cutting problems*, European Journal of Operations Research, **84(3)**, pp. 633–644.

[4] Gilmore PC & Gomory RE, 1966, *The theory and computation of knapsack functions*, Operations Research, **14(6)**, pp. 1045-1075.

[5] Luger GF & Stubblefield WA, 1991, *Artificial intelligence: Structures and strategies for complex problem solving*, 2nd Edition, The Benjamin-Cummings Publishing Company Inc, Redwood City (CA).

[6] Morabito R & Garcia V, 1998, *The cutting stock problem in a hardboard industry: A case study*, Computers & Operations Research, **25(6)**, pp. 469–485.

[7] Oberholzer JA, 2003, *Implementing artificial intelligence search methods to solve constrained two-dimensional guillotine-cut cutting stock problems*, PhD Thesis, Potchefstroom University for CHE, Potchefstroom.

[8] Oliveira JF & Ferreira JS, 1990, *An improved version of Wang's algorithm for two-dimensional cutting problems*, European Journal of Operational Research, **44(2)**, pp. 256–266.

[9] Vasko FJ, 1989, *A computational improvement to Wang's two-dimensional cutting stock algorithm*, Computers and Industrial Engineering, **16(1)**, pp. 109–115.

[10] Wang PY, 1983, *Two algorithms for Constrained two-dimensional cutting stock problems*, Operations Research, **31(3)**, pp. 573–586.