A simple $OPT + 1$ algorithm for cutting stock under the modified integer round-up property assumption

Klaus Jansen
Roberto Solis-Oba

Bericht Nr. 1009
September 2010

A simple $OPT + 1$ algorithm for cutting stock under the modified integer round-up property assumption

Klaus Jansen
Institut für Informatik
Universität zu Kiel, Kiel, Germany
kj@informatik.uni-kiel.de

Roberto Solis-Oba
Department of Computer Science
University of Western Ontario, London, Canada
solis@csd.uwo.ca

Abstract

We present a simple algorithm for the cutting stock problem with a constant number d of object sizes that produces solutions of value at most $OPT+1$ in time $d^{O(d)} \log^7 n$, where OPT is the value of an optimum solution. This algorithm works under the assumption that the modified integer round-up property of Scheithauer and Terno for the cutting stock problem holds.

1 Introduction

In the *cutting stock problem* we are given a set $T = \{T_1, T_2, \ldots, T_d\}$ of object types, where objects of type T_i have positive integer length $s_i \leq 1$. Given an infinite set of unit capacity bins and a set I of n objects containing n_i objects of type T_i, for each $i = 1, \ldots, d$, the problem is to pack I into the minimum possible number of bins. In this paper we consider the version of the problem in which the number d of different object types is constant.

This problem is related to the bin packing problem, where we are also given a set $\mathcal{S}$ of n objects of positive lengths and the goal is to pack $\mathcal{S}$ into the minimum possible number of unit capacity bins. In principle the cutting stock problem and the bin packing problem are equivalent; however, the size of the input for the stock cutting problem might be exponentially smaller than the size of the input for the bin packing problem, as in the bin packing problem we need to specify the length of each individual object, while in the cutting stock problem we only need to specify the length of each object type. This is a critical distinction because a polynomial time algorithm for the bin packing problem is not necessarily a polynomial time algorithm for the cutting stock problem.

Bin packing and cutting stock are classical problems in combinatorial optimization and they have been extensively studied. Both problems are known to be strongly NP-hard and no approximation algorithm for them can have approximation ratio smaller than 3/2 unless P=NP [6].

The cutting stock problem was introduced by Eisemann [4] in 1957 under the name of the "Trim Problem". In 1985 Marcotte [10] showed that an integer program formulation for the cutting stock problem proposed by Gilmore and Gomory in 1961 [7] has the so called integer round-up property when the number d of different object types is 2. Baum and Trotter [1] defined an integer programming problem as having the integer round-up property if its optimum value is equal to the least integer greater than or equal to the optimum value of its linear programming relaxation. In 1982, Orlin proposed a polynomial time algorithm for solving integer programming covering problems satisfying the integer round-up property, thus, showing that the cutting stock problem with 2 different object types is solvable in polynomial time.

In 1976 Berge and Johnson [2] observed that the integer programming formulation of the cutting stock problem of Gilmore and Gomory has a very strong linear programming relaxation, as many instances of the problem satisfy the integer round-up property. Similar observations were made by Diegel and Wäscher and Gau [3, 16]. However, Marcotte [10], Fieldhouse [5] and Scheithauer and Terno [13] proved that the cutting stock problem does not have the integer round-up property. In 1995, Scheithauer and Terno [14] conjectured that the formulation of Gilmore and Gomory for the cutting stock problem satisfies the modified integer round-up property (MIRUP): a linear integer minimization problem has the modified integer round-up property if its optimum value is no larger than the least integer greater than or equal to the optimum value of its linear programming relaxation plus 1. The conjecture has been proven to be true for all instances of the cutting stock problem with $d \leq 6$ [11] and $d = 7$ [15]. Furthermore, there is extensive experimental evidence supporting the conjecture [14].

The best known algorithm for the cutting stock problem with a constant number of object types is presented in [9] and it produces solutions of value at most $OPT + 1$ in time $d^{O(d2^d)} 2^{O(8^d)} \log^{11} n$, where OPT is the value of an optimum solution. In this note we show a simpler and faster algorithm than that presented in [9] that works under the assumption that the modified integer round-up property for cutting stock holds; this algorithm also produces solutions using at most $OPT + 1$ bins.

2 The Algorithm

A *configuration* or *pattern* C_i is a set of objects of total length at most 1, so all objects in a configuration can be packed in a bin. A configuration C_i can be specified with a d-dimensional vector $C_i = (C_{i1}, C_{i2}, \ldots, C_{id})$ in which the j-th entry, C_{ij}, specifies the number of objects of length s_j in C_i. Let $\mathcal{C}$ be the set of all configurations. The cutting stock problem can be formulated as the following integer program, first proposed by Gilmore and Gomory [7].

$$\begin{aligned} \text{IP}: \quad \min \quad & \sum_{C_i \in \mathcal{C}} x_{C_i} \\ \text{s.}t. \quad & \sum_{C_i \in \mathcal{C}} C_{ij} x_{C_i} \geq n_j, \quad \text{for } j = 1, \ldots, d \qquad (1) \\ & x_{C_i} \in \mathbb{Z}_{\geq 0}, \quad \text{for all } C_i \in \mathcal{C} \end{aligned}$$

In this integer program, n_j is the total number of objects of length s_j, and for each configuration C_i, variable x_{C_i} indicates the number of bins storing objects according to C_i. Constraint (1) ensures that all objects are placed in the bins. For any instance I of the cutting stock problem, we let IP(I) and LP(I) be the value of an optimum solution of IP and its linear programming relaxation LP, respectively.

Our algorithm first computes a basic feasible solution for the linear program relaxation LP of IP. Let $x^* = (x_{C_1}, x_{C_2}, \ldots, x_{C_{\mathcal{C}}})$ be this solution. Let $\bar{x}^* = \lfloor x^* \rfloor$ and $\tilde{x}^* = x^* - \bar{x}^*$. We then pack objects into bins according to the integral solution $\bar{x}^*$; this leaves a set $\tilde{I}$ of objects still unpacked. We let $\bar{I}$ denote the set of objects packed so far. We note that $\mathrm{LP}(\bar{I}) = \mathrm{OPT}(\bar{I})$ and $\mathrm{LP}(I) = \mathrm{LP}(\bar{I}) + \mathrm{LP}(\tilde{I})$.

Since linear program LP has d constraints, then basic feasible solution x^* has at most d variables $x^*_{C_j}$ with positive value and, thus, $\mathrm{LP}(\tilde{I}) = \sum_{C_j \in \mathcal{C}} \tilde{x}^*_{C_j} < d$. In other words, the objects in $\tilde{I}$ fit fractionally in a constant number, at most d, of bins. This implies that $\mathrm{OPT}(\tilde{I})$ is also constant. We assume that $\mathrm{LP}(\tilde{I}) \geq 1$ as otherwise x^* would be an optimum integer solution for the problem. We can compute a packing for $\tilde{I}$ into $\mathrm{OPT}(\tilde{I})+1$ bins using a recent algorithm by Jansen et al. [8] in time $2^{O(d \log^2 d)} + O(n)$, this algorithm yields a solution for the cutting stock problem using $OPT(I) + 2$ bins. Below we show a simple algorithm to compute a packing for $\tilde{I}$ in $\lceil \mathrm{LP}(\tilde{I}) \rceil + 1$ bins in time $2^{O(d \log d)}$. Combining this with a packing for $\bar{x}$ as described above gives a solution that uses at most $\mathrm{LP}(\bar{I}) + \lceil \mathrm{LP}(\tilde{I}) \rceil + 1 = \lceil \mathrm{LP}(\bar{I}) + \mathrm{LP}(\tilde{I}) \rceil + 1 = \lceil \mathrm{LP}(I) \rceil + 1 \leq OPT(I) + 1$.

2.1 Packing the Objects in $\tilde{I}$

An object is called *large*, if its length is at least $\delta = /(2\lceil LP(\tilde{I}) \rceil)$; if the length is smaller than δ, the object is called *small*. Let $s_1 > \ldots > s_d$ be the lengths of the objects, let $s_1, \ldots, s_\alpha$ be the lengths of the large objects, and let $s_{\alpha+1}, \ldots, s_d$ be lengths of the small objects. Notice that there are at most $LP(\tilde{I})/\delta = 2(\lceil LP(\tilde{I}) \rceil)^2 < 2d^2$ large objects in instance $\tilde{I}$. Furthermore, since every large object has size at least δ, then each bin can store at most $1/\delta = 2\lceil \mathrm{LP}(\tilde{I}) \rceil \leq 2d$ large objects. Note also that if the MIRUP assumption holds, then there must exist an assignment of large objects to $\lceil \mathrm{LP}(\tilde{I}) \rceil + 1 \leq d + 1$ bins. We can find such an assignment by using the following dynamic programming algorithm.

Every assignment of large objects to a bin can be represented as a vector $(a_1, a_2, \ldots, a_\alpha)$, where a_i is the number of large objects of size s_i assigned to the bin. An assignment of large objects to a set of bins is *feasible* if the large objects fit in the bins. Let us number the bins from 1 to $\lceil \mathrm{LP}(\tilde{I}) \rceil + 1$. Let $\mathcal{A}_1 = \{A^1_1, A^1_2, \ldots, A^1_{r_1}\}$ be the set of vectors representing all feasible assignments of large objects to bin 1. The number of these assignments is at most $(2d)^\alpha \leq (2d)^d$.

Given the set of vectors $\mathcal{A}_i = \{A^i_1, A^i_2, \ldots, A^i_{r_i}\}$ representing all feasible assignments of large objects to the first i bins, we can find the feasible assignments $\mathcal{A}_{i+1}$ of large objects to the first $i+1$ bins as follows. First, initialize $\mathcal{A}_{i+1}$ to the empty set. Then, for each vector $A^1_j = (a^1_{j1}, a^1_{j2}, \ldots, a^1_{j\alpha}) \in \mathcal{A}_1$ and $A^i_k = (a^i_{k1}, a^i_{k2}, \ldots, a^i_{k\alpha}) \in \mathcal{A}_i$, add $A^1_j + A^i_k = (a^1_{j1} + a^i_{k1}, a^1_{j2} + a^i_{k2}, \ldots, a^1_{j\alpha} + a^i_{k\alpha})$ to $\mathcal{A}_{i+1}$ if $a^1_{jh} + a^i_{kh} \leq \tilde{n}_h$ for each $h = 1, 2, \ldots, \alpha$, where $\tilde{n}_h$ is the number of objects of size s_h in $\tilde{I}$.

Since the number of bins needed to pack $\tilde{I}$ is at most $d+1$ and each bin can store at most $2d$ large objects, then $\tilde{n}_h \leq 2d(d+1)$. Therefore, every set $\mathcal{A}_i$ has at most $(2d(d+1))^\alpha \leq (2d(d+1))^d = d^{O(d)}$ vectors. Therefore, set $\mathcal{A}_{i+1}$ can be obtained from set $\mathcal{A}_i$ in time $d^d \times d^{O(d)} = d^{O(d)}$. From the above discussion, all sets $\mathcal{A}_i$, $i = 1, 2, \ldots, \lceil \mathrm{LP}(\tilde{I}) \rceil + 1$, can be computed in time $(\lceil \mathrm{LP}(\tilde{I})\rceil + 1) \times d^{O(d)} = d^{O(d)}$. If we take any feasible assignment $A \in \mathcal{A}_{\lceil \mathrm{LP}(\tilde{I})\rceil+1}$ of large objects to the $\lceil \mathrm{LP}(\tilde{I})\rceil + 1$ bins, it is not hard to find the assignment $A' \in \mathcal{A}_{\lceil \mathrm{LP}(\tilde{I})\rceil}$ from which A was computed. Applying this process recursively we can get an assignment of large objects to each one of the bins in $d^{O(d)}$ time.

Lemma 2.1 *A feasible assignment of the large objects in $\tilde{I}$ to $\lceil \mathrm{LP}(\tilde{I})\rceil + 1$ bins can be computed in $d^{O(d)}$ time.*

Once an assignment of large objects to bins has been computed, the next step is to place the small objects in the bins. To do this we first calculate the total number of objects of length $s_{\alpha+1}$ that can be placed in the first bin along with the large jobs assigned to this bin by the above dynamic programming algorithm; to do this we compute the total length ℓ_1 of the large objects in the first bin and we let $x = \lfloor (1-\ell_1)/s_{\alpha+1} \rfloor$. Therefore, $\min(x, \tilde{n}_{\alpha+1})$ objects of length $s_{\alpha+1}$ can be placed into the first bin, where $\tilde{n}_{\alpha+1}$ is the number of objects of size $s_{\alpha+1}$ in $\tilde{I}$. If $x \geq \tilde{n}_{\alpha+1}$ then we try to place objects of the next length, $s_{\alpha+2}$ into this bin; otherwise we consider the second bin and proceed in a similar fashion. The time needed to place all the small objects is $O(d)$.

Lemma 2.2 *The above algorithm packs all the small objects in $\tilde{I}$ in the empty space left in the $\lceil \mathrm{LP}\rceil + 1$ bins by the large objects.*

Proof: To show that all the small objects can be packed, let us assume for sake of contradiction that the algorithm cannot pack the small objects into $\lceil \mathrm{LP}(\tilde{I})\rceil + 1$ bins. This means that at least one small object o_i is left unpacked by the above algorithm. Since o_i does not fit into any bin, that means that each bin has objects of total length larger than $1 - \delta$. Using the definition of δ, the total length of the objects in the $\lceil \mathrm{LP}(\tilde{I})\rceil + 1$ bins is then larger than

$$(1 - 1/(2\lceil LP(\tilde{I})\rceil))(\lceil LP(\tilde{I})\rceil + 1) = \frac{(2\lceil LP(\tilde{I})\rceil - 1)}{(2\lceil LP(\tilde{I})\rceil)}(\lceil LP(\tilde{I})\rceil + 1) \geq LP(\tilde{I}),$$

a contradiction. The last inequality follows from

$$(2\lceil LP(\tilde{I})\rceil - 1)(\lceil LP(\tilde{I})\rceil + 1) \geq 2\lceil LP(\tilde{I})\rceil LP(\tilde{I}),$$

which is true since

$$2\lceil LP(\tilde{I})\rceil\lceil LP(\tilde{I})\rceil + \lceil LP(\tilde{I})\rceil - 1 \geq 2\lceil LP(\tilde{I})\rceil LP(\tilde{I}),$$

as $\lceil LP(\tilde{I})\rceil \geq 1$. □

2.2 The Complete Algorithm

A description of the complete algorithm is given below.

AlgorithmCUTTINGSTOCK(S, N)
Input: Sets $S = \{s_1, s_2, \ldots, s_d\}$ of object sizes and $N = \{n_1, n_2, \ldots, n_d\}$ of numbers of objects of each type.

1. Compute a basic feasible solution x of LP.

2. Set $\bar{x}^* = \lfloor x^* \rfloor$ and $\tilde{x}^* = x^* = \bar{x}^*$.

3. Consider a packing of the objects into bins according to the integer solution $\bar{x}^*$. Let $\tilde{I}$ be the set of objects that remain unpacked.

4. Use dynamic programming to pack the large objects in $\tilde{I}$ in $\lceil \text{LP} \rceil + 1$ bins.

5. Pack the small objects in $\tilde{I}$ greedily in the $\lceil \text{LP} \rceil + 1$ bins using the empty space left by the large objects.

Theorem 2.1 *There is an algorithm for the cutting stock problem with a constant number d of object types that produces solutions of value at most $OPT + 1$, where OPT is the value of an optimum solution, if the MIRUP conjecture holds. The algorithm runs in time $d^{O(d)} \log^7 n$.*

Proof: We have shown above an algorithm that produces solutions of value at most $OPT + 1$ assuming the MIRUP conjecture for the cutting stock problem is true. Since a basic feasible solution for LP can be computed in time $d^{O(d)} \log^7 n$ (for details see Theorem 1.1 and Lemma 4.3 in [9]), then by Lemma 2.1 the algorithm runs in time $d^{O(d)} + d^{O(d)} \log^7 n = d^{O(d)} \log^7 n$. $\square$

References

[1] S. Baum and L.E. Trotter Jr., Integer rounding for polymatroid and branching optimization problems, *SIAM Journal on Algorithms and Discrete Methods*, 2(4), 416–425, 1981.

[2] C. Berge and E.L. Johnson, Coloring the edges of a hypergraph and linear programming techniques, Research Report CORR 76/4, Department of Combinatorics and Optimization, University of Waterloo, 1976.

[3] A. Diegel, Integer LP solution for large trim problem, Working Paper, University of Natal, 1988,

[4] K. Eisemann, The trim problem, *Management Science*, 3 (3), 1957, 279–284.

[5] M. Fieldhouse, The duality gap in trim problems, *SICUP Bulletin (newsletter of the Special Interest Group on Cutting and Packing*, 5, 1990.

[6] M.R. Garey and D.S. Johnson, *Computers and intractability. A guide to the theory of NP-completeness*, W.H. Freeman and Company, 1979.

[7] P.C. Gilmore and R.E. Gomory, A linear programming approach to the cutting stock problem, *Operations Research*, 9, 1961, 849–859.

[8] K. Jansen, S. Kratsch, D. Marx, and I. Schlotter, Bin packing with fixed number of bins revisited, Proceedings of the 12th Scandinavian Symposium and Workshops on Algorithm Theory, LNCS 6139, 260-272, 2010.

[9] K. Jansen and R. Solis-Oba, An $OPT + 1$ algorithm for the cutting stock problem with a constant number of object lengths, Proceedings of the 14th International Conference on Integer Programming and Combinatorial Optimization, LNCS 6080, 438-449, 2010.

[10] O. Marcotte, The cutting stock problem and integer rounding, *Mathematical Programming*, 33, 1985, 82–92.

[11] C. Nitsche, G. Scheithauser, and J. Terno, New cases of the cutting stock problem having mirup, *Mathematical Methods of Operations Research*, 48 (1), 105–116, 1998.

[12] J.B. Orlin, A polynomial algorithm for integer programming covering problems satisfying the integer round-up property, *Mathematical Programming*, 22, 1982, 231–235.

[13] G. Scheithauer and J. Terno, About the gap between the optimal values of the integer and continuous relaxation one-dimensional cutting stick problem, Operations Research Proceedings 1991, Springer Verlag, 1992.

[14] G. Scheithauer and J. Terno, The modified integer round-up property of the one-dimensional cutting stock problem, *European Journal of Operational research*, 84, 562–571, 1995.

[15] G. Shmonin and A. Sebo, personal communication.

[16] G. Wäscher and T. Gau, Two approaches to the cutting stock problem, IFORS 93, XIII World Conference on Operations Research, 1993.