A Note on Dual Approximation Algorithms for Class Constrained Bin Packing Problems*

E. C. Xavier† F. K. Miyazawa†

Abstract

In this paper we present a dual approximation scheme for the class constrained shelf bin packing problem. In this problem, we are given bins of capacity 1, and n items of Q different classes, each item e with class c_e and size s_e. The problem is to pack the items into bins, such that two items of different classes packed in a same bin must be in different shelves. Items in a same shelf are packed consecutively. Moreover, items in consecutive shelves must be separated by shelf divisors of size d. In a shelf bin packing problem, we have to obtain a shelf packing such that the total size of items and shelf divisors in any bin is at most 1. A dual approximation scheme must obtain a shelf packing of all items into N bins, such that, the total size of all items and shelf divisors packed in any bin is at most $1 + \varepsilon$ for a given $\varepsilon > 0$ and N is the number of bins used in an optimum shelf bin packing problem. Shelf divisors are used to avoid contact between items of different classes and can hold a set of items until a maximum given weight. We also present a dual approximation scheme for the class constrained bin packing problem. In this problem, there is no use of shelf divisors, but each bin uses at most C different classes.

Key Words: Bin Packing, Approximation Algorithms. **MSC number:** 68W25

1 Introduction

In this paper we study class constrained bin packing problems, that are generalizations of the well known NP-hard bin packing problem. We first consider the class constrained shelf bin packing (CCSBP) problem. In this problem we are given a tuple $I = (L, s, c, Q, d, \Delta)$, where $L = (1, \ldots, n)$ is a list of n items, each item $e \in L$ with size $0 < s_e \leq 1$ and class $c_e \in \{1, \ldots, Q\}$, d is the size of a shelf division, Δ is the maximum size of a shelf and the bins size is 1.

Given a list or set of items S we denote by $s(S)$ the total size of items in S, *i.e.* $s(S) = \sum_{e \in S} s_e$.

[1]This research was partially supported by FAPESP (proc. 2008/01490-3), FAEPEX (proc. 31608) and CNPQ (proc. 478470/06-1, 472504/07-0, 306624/07-9).

[2]{ecx,fkm}@ic.unicamp.br — Institute of Computing — University of Campinas — UNICAMP, P.O. Box 6176, 13083-970, Campinas, SP, Brazil.

A *shelf packing* $\mathcal{P}$ of an instance I for the CCSBP problem is a packing of the items in a set of bins $\mathcal{P} = \{P_1, \ldots, P_k\}$, where the items packed in a bin $P_i \in \mathcal{P}$ are partitioned into shelves $\{S_1^i, \ldots, S_{q_i}^i\}$ such that for each shelf S_j^i we have $s(S_j^i) \leq \Delta$, all items in S_j^i are of the same class and $\sum_{j=1}^{q_i}(s(S_j^i) + d) \leq 1$. The problem is to find a shelf packing that uses the minimum number of bins. The problem applies when some items cannot be stored in a same shelf (like foods and chemical products) and therefore, they must be separated by shelf divisors.

We also consider the class constrained bin packing problem, which we denote by CCBP. In this problem we are given a tuple $I = (L, s, c, C, Q)$ where $L = (1, \ldots, n)$ is a list of n items, each item $e \in L$ with size $0 < s_e \leq 1$ and class $c_e \in \{1, \ldots, Q\}$, and a set of bins, each one with capacity 1 and C compartments. A packing for instance I is a set of bins $\mathcal{P} = \{P_1, \ldots, P_k\}$ such that the number of different classes of items packed in each bin P_i is at most C and the total items size in each bin is at most 1. The problem is to find a packing of instance I that uses the minimum number of bins.

In both problems we assume that Q, the number of different classes in the input instance, is bounded by a constant. The bin packing problem is a particular case of CCBP and CCSBP when there is only one class, $d = 0$ and $\Delta = 1$. In Figure 1 we present an example of the two types of packings considered.

Given an algorithm $\mathcal{A}$ for the CCBP or CCSBP problem and an instance I, we denote by $\mathcal{A}(I)$ the number of bins used by the algorithm to pack this instance. We denote by $\mathrm{OPT}(I)$ the number of bins used by an optimum solution to pack the instance I. In both notations the problem considered will be clear from the context. Given an integer t, we denote by $[t]$ the set $\{1, \ldots, t\}$.

In [5], Hochbaum and Shmoys presented the concept of dual approximation algorithms where one has to find an infeasible optimal solution, and the quality of the algorithm is measured by how infeasible is the generated solution. There are some cases where the restrictions of the problem are flexible in practice and the concept of dual approximation algorithms can be applied.

A dual polynomial time approximation scheme (dual PTAS) for the CCSBP problem is an algorithm that, for all instances I, produces solutions that uses at most $\mathrm{OPT}(I)$ bins, each bin with size at most $(1 + O(\varepsilon))$ and each shelf of the bin with size at most $(1 + O(\varepsilon))\Delta$. A dual PTAS for the CCBP problem is an algorithm that, for all instances I, it produces solutions that uses at most $\mathrm{OPT}(I)$ bins, each bin with size at most $(1 + O(\varepsilon))$. In both cases ε is a fixed parameter given to the algorithm.

Woeginger [16] presented general properties in order to guarantee the existence of approximation schemes by dynamic programming algorithms for some problems.

Packing problems with class constraints have many applications in multimedia storage systems, resource allocation [15, 11, 4, 8, 14, 17, 13, 3, 19] and in operations research like manufacturing systems [7, 10, 1]. The CCSBP problem appears in the iron and steel industry [2, 9, 6, 20, 18].

The CCSBP problem admits an asymptotic polynomial time approximation scheme [20]. A knapsack version of this problem also admits a PTAS [18]. This paper is the first one to present a dual PTAS for the CCSBP problem.

We also present a dual PTAS for the CCBP problem. Notice that a dual approximation scheme for the CCBP problem was first presented by Shachnai and Tamir [12] also considering that the number of

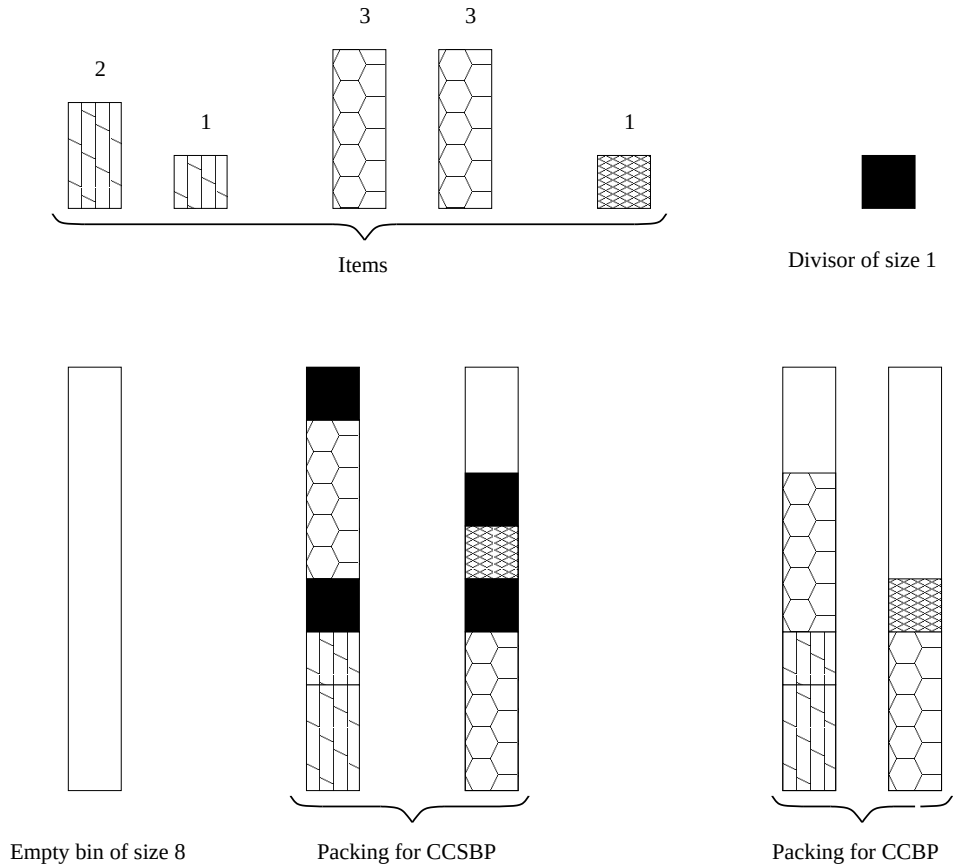

Figure 1: In this example we have three classes of items represented by the different shapes. The first packing is a solution for the CCSBP problem where we consider that $\Delta = 3$. The second packing is a solution for the CCBP problem where we consider that $C = 2$. Notice that with $C = 2$ we can not pack the item of size 1 of the third class in the first bin, although there is room for it.

different classes in the input instance is bounded by a constant. The complexity time of their algorithm is $O(n^{16Q/\varepsilon^2})$. In their paper they presented a dual PTAS using techniques that group small items together. They also said: "We cannot adopt the technique commonly used for packing, where we first consider large items and then add the small items". In this paper we show how to adopt the traditional technique and obtain a dual PTAS with an easier analysis, also considering that Q is a fixed constant. Although the easier analysis, the complexity time of our algorithm is $O(Tn^{O(2^Q Q(\log_{1+\varepsilon} 1/\varepsilon)^{1/\varepsilon})})$, where T is the complexity time to solve a linear program (see Section 3).

In section 2 we present a dual PTAS for the CCSBP problem using traditional techniques, and linear programming to pack small items. In section 3 we use these ideas to obtain a dual PTAS for the CCBP problem, leading to an algorithm with an easier analysis than the one presented by Shachnai and Tamir [12].

2 A dual PTAS for the CCSBP Problem

In this section we present a dual PTAS for the CCSBP problem.

Let $I = (L, s, c, Q, d, \Delta)$ be an instance for the CCSBP problem. We first present a dual PTAS for the case where the maximum size of a shelf plus the shelf divisor satisfy $\Delta + d \leq \varepsilon$.

Hochbaum and Shmoys [5] presented a dual PTAS, which we denote by $\mathcal{A}_{HS}$, for the classical bin packing problem.

Consider an algorithm that constructs a list of shelves S in a straightforward manner: For each class, it packs the items of this class using the algorithm $\mathcal{A}_{HS}$ considering shelves as bins, each one with size Δ. Since the algorithm $\mathcal{A}_{HS}$ is a dual PTAS the number of generated shelves by the algorithm is at most the number of shelves used in any optimal solution, which we denote by $\mathrm{OPT}(I)_s$. Moreover each generated shelf has size at most $(1 + \varepsilon)\Delta$.

Given the list of shelves S consider another algorithm that packs the shelves in the following manner: It packs shelves (including the shelf divisors) in a bin until for the first time the total size of packed shelves becomes greater than 1. Then it proceeds with a new bin. It is easy to prove the following result for this algorithm.

Theorem 2.1 *The presented algorithm is a dual PTAS for the* CCSBP *problem restricted to instances where* $\Delta + d \leq \varepsilon$.

Proof. Notice that the algorithm packs all items in at most $\mathrm{OPT}(I)_s$ shelves and each shelf has its size increased by a factor of at most ε. The total size of items and shelf divisors that the algorithm has to pack into bins is

$$s(L) + d\mathrm{OPT}(I)_s \leq \mathrm{OPT}(I).$$

Since the algorithm generated bins with size greater than 1, the algorithm packs all shelves in at most $\mathrm{OPT}(I)$ bins. Since each shelf has size at most $\varepsilon(1 + \varepsilon) \leq 2\varepsilon$, each generated bin has size at most $(1 + 2\varepsilon)$. □

We now study the case where $\Delta + d \geq \varepsilon$.

Notice that the maximum number of shelves that can be packed in a bin when the shelves are completely filled up to Δ exactly, is at most $\left\lfloor \frac{1}{d+\Delta} \right\rfloor$, which is at most $\frac{1}{\varepsilon}$. Another shelf, not completely filled, can also be packed in the bin. This way, the maximum total size of items that can be packed in a bin is at most $\Delta(1/\varepsilon + 1)$. Then if there is any bin with more than $\frac{2}{\varepsilon} + 2$ shelves of a same class, it has at least two shelves of this class with total items size at most Δ. In this case, these two shelves can be combined into only one shelf. Without loss of generality we assume that each bin in a solution for the CCSBP problem, contains at most $\frac{2}{\varepsilon} + 2$ shelves of a same class.

Throughout the remaining of this section, we assume that s_e for each $e \in L$, d, Δ and the size of the bins are rescaled, such that $\Delta = 1$. We denote by B the new size of the bins.

Now we can apply the standard technique of rounding big items with an enumeration of packings for them. Then we pack the small items using linear programming to get the PTAS. Let L_b be the list of items

with size greater than or equal to ε^2 (big items) and let L_s be the remaining items in L (small items). We round down each item in L_b as follows: each item $e \in L_b$ with size in the interval $[\varepsilon^2(1+\varepsilon^2)^i, \varepsilon^2(1+\varepsilon^2)^{i+1})$ has its size rounded down to $\varepsilon^2(1+\varepsilon^2)^i$, for $i \geq 0$. The rounded items have at most $M = \lceil \log_{(1+\varepsilon^2)} 1/\varepsilon^2 \rceil$ different sizes. The list of rounded items is denoted by L_r.

We can generate a set of packings for the rounded big items in polynomial time as the next lemma guarantees. In the lemma we use the fact that the number of subsets containing at most p items from a set of n items is $\binom{n+p}{p}$.

Lemma 2.2 *Let $I' = (L_r, s, c, Q, d, \Delta = 1, B)$ be an instance of the CCSBP problem considering the list of rounded big items where the number of distinct items sizes in L_r is at most a constant M, the number of different classes is bounded by a constant Q, and each item $e \in L_r$ has size $s_e \geq \varepsilon^2$. Then there exists a polynomial time algorithm that generates all possible shelf packings of L_r, with at most $\frac{2}{\varepsilon} + 2$ shelves of a same class in each bin. Moreover, each shelf in each generated packing has a sign that indicates if small items can or cannot be packed in it.*

Proof. The maximum number of big items that can be packed in a shelf is bounded by $p = 1/\varepsilon^2$. Given a class, the number of different shelves configurations is bounded by $r' = \binom{p+M}{p}$, including the empty shelf that can be used latter to pack only small items. For each shelf we have the possibility of packing or not small items, then the number of shelves configurations is bounded by $2r'$. The number of different shelves configurations can then be bounded by $r = Q2r'$. Since the number of shelves packed in a bin is bounded by $q = Q(\frac{2}{\varepsilon} + 2)$, the number of different bins configurations is bounded by $u = \binom{q+r}{q}$. Notice that u is a (large) constant since all the values p, q, r and u depend only on ε, Q and M which are constants.

Therefore, the number of all feasible packings is bounded by $\binom{n+u}{n}$, which is bounded by $(n+u)^u$, which in turn is polynomial in n where $u = O((Q/\varepsilon)^{(Q/\varepsilon)^{1/\varepsilon}})$. □

We will denote by **ALL** the algorithm that generates all packings of the list L_r according to Lemma 2.2. The algorithm **ALL** generates a set, which we denote by $\mathbb{P}$, of all possible packings of the rounded big items.

Consider an optimal packing O for the original instance I. Remove the small items from the packing O and round down the big items according to the rounding procedure. Notice that the obtained packing denoted by O' belongs to the set $\mathbb{P}$.

For each packing in $\mathbb{P}$, it is then generated new packings where the small items are packed. Denote by **Small** the algorithm that packs small items in each of the packings in $\mathbb{P}$.

Let $\mathcal{P} = \{P_1, \ldots, P_k\} \in \mathbb{P}$ be a shelf packing of a list of items L_r and suppose we have to pack a list L_s of small items into $\mathcal{P}$. The packing of the small items is obtained from a solution of a linear program. Let $N_i \subseteq \{1, \ldots, Q\}$ be the set of possible classes that are packed in the bin P_i and let $S_1^{ic}, \ldots, S_{n_{ic}}^{ic}$ be the shelves of class $c \in N_i$ in the bin P_i of the packing $\mathcal{P}$. For each shelf S_j^{ic}, define a non-negative variable x_j^{ic}. The variable x_j^{ic} indicates the total size of small items of class c that is to be packed in the shelf S_j^{ic}. Note that here we only consider the shelves that can be used to pack small items. Denote by $s(S_j^{ic})$ the total size of big items already packed in the shelf S_j^{ic}. Consider the following linear program denoted by LPS1:

$$\max \sum_{i=1}^{k} \sum_{c \in N_i} \sum_{j=1}^{n_{ic}} x_j^{ic}$$

$$\begin{aligned}
s(S_j^{ic}) + x_j^{ic} &\leq \Delta && \forall\, i \in [k],\ c \in N_i,\ j \in [n_{ic}], && (1)\\
\sum_{c \in N_i} \sum_{j=1}^{n_{ic}} (s(S_j^{ic}) + x_j^{ic} + d) &\leq B && \forall\, i \in [k], && (2)\\
\sum_{i=1}^{k} \sum_{j=1}^{n_{ic}} x_j^{ic} &\leq s(L_s^c) && \forall\, c \in [Q], && (3)\\
x_j^{ic} &\geq 0 && \forall\, i \in [k],\ c \in [N_i],\ j \in [n_{ic}] && (4)
\end{aligned}$$

(LPS1)

where L_s^c is the set of small items of class c in L_s.

The constraint (1) guarantees that the amount of space used in each shelf is at most 1 and constraint (2) guarantees that the amount of space used in each bin is at most B. The constraint (3) guarantees that variables x_j^{ic} are not greater than the total size of small items. The number of variables in LPS1 is bounded by $O(nQ2/\varepsilon)$ and the number of constraints is bounded by $O(nQ2/\varepsilon + n + Q)$.

Notice that since $O' \in \mathbb{P}$ at least this packing has a solution in LPS1 where all small items can be packed.

Now we have a description of the algorithm **Small**: Given a packing $\mathcal{P} \in \mathbb{P}$, and a list L_s of small items, the algorithm first solves the linear program LPS1, and then packs small items in the following way: For each variable x_j^{ic} the algorithm packs, while possible, small items of class c into shelf S_j^{ic} of the bin P_i, so that the total size of the packed small items is at most $x_j^{ic} + \varepsilon^2$.

A complete description of the dual-PTAS algorithm that generates the complete packing is given in Figure 2. The algorithm returns a packing that uses the minimum number of bins and that packs all items in bins.

Since Q and ε are constants, the size of $\mathbb{P}$ is bounded by a polynomial in n. Since the complexity time to solve LPS1 is polynomial, the presented algorithm has a polynomial time complexity. Now we conclude with the following theorem.

Theorem 2.3 *The presented algorithm is a dual PTAS for the* CCSBP *problem when* $\Delta + d \geq \varepsilon$.

Proof. Let $O = \{P_1^*, \ldots, P_k^*\}$ be an optimal packing for an instance I of the CCSBP problem (notice that $\mathrm{OPT}(I) = k$). Round down the big items according to the rounding we have presented and remove the small items from O obtaining another packing O'. Clearly $O' \in \mathbb{P}$ and has an indication of the shelves where small items were packed.

Notice that there is enough room to pack all small items in O'. The algorithm **Small** packs all small items in O' in such a way that each shelf has its size increased by at most ε^2.

When the algorithm considers the big items with their original sizes, the size in each shelf of O' increases by at most ε^2 again.

Since the maximum number of shelves in a bin is $(\frac{2}{\varepsilon} + 2)Q$, then the total size of each bin is increased to at most $B + (\frac{2}{\varepsilon} + 2)Q2\varepsilon^2 \leq (1 + (\frac{2}{\varepsilon} + 2)Q2\varepsilon^2)B$ □

ALGORITHM Dual-Pack(I)

Input: Instance $I = (L, s, c, Q, d, \Delta, B)$ where the maximum capacity of a shelf is $\Delta = 1$;

Output: A shelf packing $\mathcal{P}$.

Subroutines: Algorithms **ALL** and **Small**.

1. Partition L into a list L_b containing items with size ε^2 (big items) and L_s with the remaining items (small items).
2. For each item e in L_b with size in $[\varepsilon^2(1+\varepsilon^2)^i, \varepsilon^2(1+\varepsilon^2)^{i+1})$, round down its size to $\varepsilon^2(1+\varepsilon^2)^i$.
3. Let L_r be the list of the rounded big items.
4. Let $\mathbb{P}$ be the set of all possible packings obtained with the algorithm ALL over the instance $I = (L_r, s, c, Q, d, \Delta, B)$.
5. For each packing $\mathcal{P} \in \mathbb{P}$ do
6. Find a solution x^* for LPS1 considering the packing $\mathcal{P}$.
7. Pack the items in L_s into $\mathcal{P}$ using algorithm **Small**.
8. Round up the big items in $\mathcal{P}$ to their original sizes.
9. Return the packing $\mathcal{P} \in \mathbb{P}$ with the minimum number of bins and where all items are packed.

Figure 2: The dual-PTAS algorithm.

3 A dual PTAS for the CCBP Problem

In this section we present a dual PTAS for the CCBP problem using the same ideas of the previous section leading to an algorithm with an easier analysis than the one presented by Shachnai and Tamir [12].

Let L_b be the set of items in L with size greater than or equal to ε (big items) and let L_s be the remaining items in L (small items). We round down each item in L_b as follows: each item $e \in L_b$ with size in the interval $[\varepsilon(1+\varepsilon)^i, \varepsilon(1+\varepsilon)^{i+1})$ has its size rounded down to $\varepsilon(1+\varepsilon)^i$, for $i \geq 0$. The rounded items have at most $M = \lceil \log_{(1+\varepsilon)} 1/\varepsilon \rceil$ different sizes. The list of rounded items is denoted by L_r.

It is not hard to prove the following lemma that is similar to Lemma 2.2.

Lemma 3.1 *Let $I = (L_r, s, c, C, Q)$ be an instance of the CCBP problem after the rounding step, where the number of distinct items sizes in L_r is at most a constant M, the number of different classes is bounded by a constant Q, and each item $e \in L_r$ has size $s_e \geq \varepsilon$. Then there exists a polynomial time algorithm that generates all possible packings of L_r. Moreover, each bin of each generated packing has an indication of the possible classes that may be used to pack the small items.*

Proof. The number of big items that can be packed in a bin is bounded by $p = 1/\varepsilon$. The number of distinct types of big items is bounded by MQ. The number of different configurations of bins is bounded by $r' = \binom{p+MQ+1}{p}$, including the empty bin. If we also consider additional classes to pack small items in each configuration, the number of different configurations is bounded by $r = r'2^Q$, which is a constant. Notice that we only consider configurations that satisfy the class constraints.

The number of all feasible packings is bounded by $\binom{n+r}{n}$, which is bounded by $(n+r)^r$, which in turn is polynomial in n where $r = O(2^Q Q(\log_{1+\varepsilon} 1/\varepsilon)^{1/\varepsilon})$. □

The algorithm generates a set, which we denote by $\mathbb{P}$, of all possible packings of the rounded big items. For each one of these packings the algorithm packs the small items in the following way: Let $\mathcal{P} = \{P_1, \ldots, P_k\}$ be a packing of the list of items L_r and suppose we have to pack a list L_s of small items, with size at most ε, into $\mathcal{P}$. The packing of the small items is obtained from a solution of a linear program. Let $N_i \subseteq \{1, \ldots, Q\}$ be the set of possible classes that may be used to pack the small items in the bin P_i of the packing $\mathcal{P}$. For each class $c \in N_i$, define a non-negative variable x_c^i. The variable x_c^i indicates the total size of small items of class c to be packed in the bin P_i. Denote by $s(P_i)$ the total size of big items already packed in the bin P_i. Consider the following linear program denoted by LPS2:

$$\max \sum_{i=1}^{k} \sum_{c \in N_i} x_c^i$$

$$\begin{aligned}
s(P_i) + \sum_{c \in N_i} x_c^i &\leq 1 && \forall\, i \in [k] && (1)\\
\sum_{i=1}^{k} x_c^i &\leq s(L_s^c) && \forall\, c \in [Q], && (2)\\
x_c^i &\geq 0 && \forall\, i \in [k], c \in [N_i], && (3)
\end{aligned} \qquad \text{(LPS2)}$$

where L_s^c is the set of small items of class c in L_s.

The constraint (1) guarantees that the items packed in each bin satisfy its capacities and constraint (2) guarantees that the total use of variables x_c^i is not greater than the total size of small items for each class c. In this linear program, the number of variables is bounded by nQ and the number of constraints is bounded by $n + Q$.

Given a packing $\mathcal{P}$, and a list L_s of small items, the algorithm first solves the linear program LPS2, and then packs small items in the following way: For each variable x_c^i, it packs, while possible, the small items of class c into the bin P_i, so that the total size of the packed small items is at most $x_c^i + \varepsilon$.

We then consider the original size of the big items in each of the generated packings. In this case, the size of each bin increases by at most a factor of ε.

The algorithm returns a packing that uses the minimum number of bins and that packs all items in bins of size at most $(1 + (C + 1)\varepsilon)$. The number of packings in the set $\mathbb{P}$ can be bounded by $T_1 = O(n^{2^Q Q(\log_{1+\varepsilon} 1/\varepsilon)^{1/\varepsilon}})$. Let T_2 be the worst complexity time to solve a linear program LPS2. The complexity time of the entire algorithm can be bounded by $O(T_1 T_2)$, which is polynomial since Q and ε are constants and the complexity time T_2 is polynomial.

We conclude with the following theorem.

Theorem 3.2 *The presented algorithm is a dual PTAS for the* CCBP *problem.*

Proof. Let $O = \{P_1^*, \ldots, P_k^*\}$ be an optimal packing for an instance I of the CCBP problem. Round down the big items according to the rounding we have presented and remove the small items of O obtaining

another packing O'. Clearly $O' \in \mathbb{P}$ and has an indication of the classes of small items that were packed on it. There is enough room to pack all small items in O'. So the variables x sums to the total size of small items. During the packing of the small items we increase the size of each bin by at most ε for each class in the bin. When the algorithm packs the big items with their original size, the size of each bin in O' increases by at most ε again. So the total size of each bin is increased to at most $(1 + (C + 1)\varepsilon)$. □

References

[1] M. Dawande, J. Kalagnanam, and J. Sethuranam. Variable sized bin packing with color constraints. In *Proceedings of the 1th Brazilian Symposium on Graph Algorithms and Combinatorics*, volume 7 of *Electronic Notes in Discrete Mathematics*, 2001.

[2] J. S. Ferreira, M. A. Neves, and P. Fonseca e Castro. A two-phase roll cutting problem. *European J. Operational Research*, 44:185–196, 1990.

[3] S. Ghandeharizadeh and R. R. Muntz. Design and implementation of scalable continous media servers. *Parallel Computing Journal*, 24(1):91–122, 1998.

[4] L. Golubchik, S. Khanna, S. Khuller, R. Thurimella, and A. Zhu. Approximation algorithms for data placement on parallel disks. In *Proceedings of SODA*, pages 223–232, 2000.

[5] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems: practical and theoretical results. *journal of the ACM*, 34(1):144–162, 1987.

[6] R. Hoto, M. Arenales, and N. Maculan. The one dimensional compartmentalized cutting stock problem: a case study. *European Journal of Operational Research*, 183(3):1183–1195, 2007.

[7] J. R. Kalagnanam, M. W. Dawande, M. Trumbo, and H. S. Lee. The surplus inventory matching problem in the process industry. *Operations Research*, 48(4):505–516, 2000.

[8] S. R. Kashyap and S. Khuller. Algorithms for non-uniform size data placement on parallel disks. *J. Algorithms*, 60(2):144–167, 2006.

[9] F. P. Marques and M. Arenales. The constrained compartmentalized knapsack problem. *Computer & Operations Research*, 34(7):2109–2129, 2007.

[10] M. Peeters and Z. Degraeve. The co-printing problem: A packing problem with a color constraint. *Operations Research*, 52(4):623–638, 2004.

[11] H. Shachnai and T. Tamir. On two class-constrained versions of the multiple knapsack problem. *Algorithmica*, 29:442–467, 2001.

[12] H. Shachnai and T. Tamir. Polynomial time approximation schemes for class-constrained packing problems. *Journal of Scheduling*, 4(6):313–338, 2001.

[13] H. Shachnai and T. Tamir. Multiprocessor scheduling with machine allotment and parallelism constraints. *Algorithmica*, 32(4):651–678, 2002.

[14] H. Shachnai and T. Tamir. Approximation schemes for generalized 2-dimensional vector packing with application to data placement. In *Proceedings of 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, RANDOM-APPROX*, volume 2764 of *Lecture Notes in Computer Science*, pages 165–177, 2003.

[15] H. Shachnai and T. Tamir. Tight bounds for online class-constrained packing. *Theoretical Computer Science*, 321(1):103–123, 2004.

[16] G. J. Woeginger. When does a dynamic programming formulation guarantee the existence of a fully polynomial time approximation scheme (fptas)? *INFORMS Journal on Computing*, 12(1):57–74, 2000.

[17] J. L. Wolf, P. S. Wu, and H. Shachnai. Disk load balancing for video-on-demand-systems. *ACM Multimedia Systems Journal*, 5:358–370, 1997.

[18] E. C. Xavier and F. K. Miyazawa. Approximation schemes for knapsack problems with shelf divisions. *Theoretical Computer Science*, 352(1-3):71–84, 2006.

[19] E. C. Xavier and Flávio Keidi Miyazawa. The class constrained bin packing problem with applications to video-on-demand. *Theoretical Computer Science*, 393(1-3):240–259, 2008.

[20] E. C. Xavier and Flávio Keidi Miyazawa. A one-dimensional bin packing problem with shelf divisions. *Discrete Applied Mathematics*, 156(7):1083–1096, 2008.