Advances in Applied Mathematics 应用数学进展, 2022, 11(5), 2883-2890
Published Online May 2022 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2022.115306

Hans汉斯

带时间效率约束的一维装箱问题

苗睿卿[1]，吴彬彬[1]，张同全[2]

[1]云南民族大学数学与计算机科学学院，云南 昆明
[2]云南民族大学预科教育学院，云南 昆明

收稿日期：2022年4月25日；录用日期：2022年5月19日；发布日期：2022年5月27日

摘 要

一维装箱问题是指把一定数量的物品放入容量相同的一些箱子中，使得每个箱子中的物品大小之和不超过箱子容量并使所用的箱子数目最少。本文研究了带时间效率约束的一维装箱问题，时间效率约束为不同机器的装箱效率不同，目标是装箱数目近似比与装箱时间近似比的乘积最小。本文给出了一种求解带时间效率约束的一维装箱问题的近似算法，分析了问题的NP-困难性，并证明出目标近似比的乘积为 $(2+c)\left(2-\frac{1}{m}\right)$ (其中 $0<c\le 1$)。

关键词

一维装箱，近似算法，NP困难性

One-Dimensional Packing Problem with Time Efficiency Constraint

Ruiqing Miao[1], Binbin Wu[1], Tongquan Zhang[2]

[1]School of Mathematics and Computer Sciences, Yunnan Minzu University, Kunming Yunnan
[2]School of Preparatory Education, Yunnan Minzu University, Kunming Yunnan

Received: Apr. 25th, 2022; accepted: May 19th, 2022; published: May 27th, 2022

Abstract

One dimensional packing problem is to put a certain number of items into some boxes with the

文章引用: 苗睿卿, 吴彬彬, 张同全. 带时间效率约束的一维装箱问题[J]. 应用数学进展, 2022, 11(5): 2883-2890.
DOI: 10.12677/aam.2022.115306

same capacity, so that the sum of the sizes of the items in each box does not exceed the box capacity and minimize the number of boxes used. In this paper, we study one-dimensional packing problem with time efficiency constraint. The time efficiency constraint is that different machines have different packing efficiency. The aim is to minimize the product of the approximate ratio of the number of packing and the approximate ratio of the time of packing. We present an approximation algorithm for the problem and analyze the NP-Hardness. We prove that the algorithm has the approximation ratio is $(2+c)\left(2-\frac{1}{m}\right)$ ($0<c\le 1$).

Keywords

One-Dimensional Packing, Approximation Algorithm, NP-Hardness

1. 引言

一维装箱问题是指把一定数量的物品放入容量相同的一些箱子中，使得每个箱子中的物品大小之和不超过箱子容量并使所用的箱子数目最少。一维装箱问题是组合优化问题中最基本的一类 NP-hard 问题[1]，在 20 世纪 70 年代就引起了广泛的谈论与研究，并有了显著的成果[2] [3] [4]。然而随着社会的发展，研究者们对经典装箱算法的研究已经较为成熟，想要进一步突破经典装箱的求解算法难度很大。因此，研究者们开始研究与现代化实际应用相结合的带约束的装箱问题[5] [6] [7] [8]。

在当今社会上，装箱问题往往与装箱时间相联系，例如：物流运输既需要使用车辆的数目较少，也需要装车时间较短。而装箱时间与装箱机器的速度有关，由于机器的老化会导致装箱的速度不同，从而导致不同机器装同一个物体的时间有差异。因此，本文考虑了一种新的时间约束类型的装箱问题即带时间效率约束的一维装箱问题，分析了问题的 NP 困难性，给出了一种求解带时间效率约束的一维装箱问题的近似算法，并证明出目标近似比的乘积为$(2+c)\left(2-\frac{1}{m}\right)$(其中$0<c\le 1$)。

2. 相关知识

一维装箱问题是给定 n 个物品序列 $L=(a_1,a_2,\cdots,a_n)$，每个物品 a_i 的尺寸为 $s(a_i)\in(0,1]$，$i=1,2,\cdots,n$，要求把这些物品装入若干个单位容量箱子中，目标是使得所使用的箱子数目达到最少。

对于一个装箱问题，L 是问题的任意输入物品序列，设 A 是求解装箱问题的一个近似算法，算法 A 返回的解的目标值为 $A(L)$，序列 L 对应最优解的目标值为 $A^*(L)$，则算法的近似比定义为 $R_A(L)=\frac{A(L)}{A^*(L)}$。

求解一维装箱问题的近似算法有很多[9] [10] [11]，其中下次适应算法(Next Fit Algorithm，简称为 NF 算法)是最简单也是最早研究的一个算法，其策略为：按照物品给定的顺序装箱时，始终保持只打开一个箱子，对于当前要装入的物品，检查是否能装入当前打开的箱子，若不能，则关闭该箱子打开一个空箱子装该物品。NF 算法的时间复杂度为 $O(n)$，最坏情况渐近性能比为 2。

3. 问题描述

经典一维装箱问题的目标是使用单位箱子数目最少，但此模型过于单一。该问题的数学模型为

$$\min z(y)=\sum_{j=1}^{m} y_j$$

$$\text{s.t.}\sum_{i=1}^{n} w_i x_{ij} \le W y_j,(j=1,2,\cdots,m) \tag{3.1}$$

$$\sum_{i=1}^{n} x_{ij}=1,(j=1,2,\cdots,m) \tag{3.2}$$

$$x_{ij}=0\text{或}1,(i=1,2,\cdots,n,j=1,2,\cdots,m) \tag{3.3}$$

$$y_j=0\text{或}1,(i=1,2,\cdots,m) \tag{3.4}$$

其中目标函数为使用箱子数目；公式(3.1)表示物品装入箱子的总容量(或总体积)不能超过箱子的容量(或体积)；公式(3.2)表示每个物品只能装入一个箱子；公式(3.3)中，当 $x_{ij}=1$ 时表示物品 i 被装入箱子 j 中，当 $x_{ij}=0$ 时表示物品 i 未被装入箱子 j 中；公式(3.4)中，当 $y_j=1$ 时第 j 个箱子被使用，当 $y_j=0$ 时第 j 个箱子未被使用。

本文考虑的装箱问题加上了时间效率约束，同时考虑装箱数目和时间，在数目上，要求装箱数目最少；在时间上，要求装箱时间最少。为了确保达同时到两个目标，本文采用了一种新的目标设定方法，即目标是装箱数目近似比与装箱时间近似比的乘积最小。与传统的解决多目标问题的方法比较，这种目标设定方法合理地避免了多目标规划问题求解过程中可能出现的达到一级目标但达不到二级目标的结果。

效率约束是指，当使用不同机器装箱时，由于机器的老化程度不同，每个机器的装箱速度不同，不妨设装箱速度最快的机器的效率为 1，且效率低于 0.5 时机器需要维修。装箱时间与机器效率成反比，例如使用效率为 1 的机器将 n 个物体装箱的时间为 $\{t_1,t_2,\cdots,t_n\}$，那么使用效率为 0.6 的机器将 n 个物体装箱的时间为 $\left\{\frac{5t_1}{3},\frac{5t_2}{3},\cdots,\frac{5t_n}{3}\right\}$。

带效率约束的一维装箱问题描述如下：给定 n 个物品序列 $L=(a_1,a_2,\cdots,a_n)$，每个物品 a_i 的尺寸为 $s(a_i)\in(0,1]$，$i=1,2,\cdots,n$，让 m 个正常使用的机器同时装箱，机器效率 $\alpha_j\in[0.5,1]$，$j=1,2,\cdots,m$，使用效率为1的机器将n个物体装箱的时间为$\{t_1,t_2,\cdots,t_n\}$，要求把所有物品装入若干个单位容量的箱子中。目标是装箱数目近似比与装箱时间近似比的乘积最小。

4. 研究内容

4.1. 数学模型

$$\min\ R_{A_T}(L)\times R_A(L)$$

$$\text{s.t.}\quad A(L)=\sum_{r=1}^{s} y_r \tag{4.1}$$

$$A_T(L)=\max_{1\le j\le m}\sum_{r=1}^{s}\sum_{i=1}^{n} x_{ir}\delta_{ij}\frac{t_i}{\alpha_j}y_r \tag{4.2}$$

$$R_A(L)=\frac{A(L)}{A^*(L)} \tag{4.3}$$

$$R_{A_T}(L)=\frac{A_T(L)}{A_T^*(L)} \tag{4.4}$$

$$\sum_{i=1}^{n}s(a_i)x_{ir}\le Wy_r \quad r=1,\cdots,s \tag{4.5}$$

$$\sum_{i=1}^{n}x_{ir}=1 \quad r=1,\cdots,s \tag{4.6}$$

$$\sum_{i=1}^{n}\delta_{ij}=1 \quad j=1,\cdots,m \tag{4.7}$$

$$x_{ir}=0\text{或}1 \quad i=1,\cdots,n;r=1,\cdots,s \tag{4.8}$$

$$\delta_{ij}=0\text{或}1 \quad j=1,\cdots,m;i=1,\cdots,n \tag{4.9}$$

$$y_r=0\text{或}1 \quad r=1,\cdots,s \tag{4.10}$$

$$\gamma_j=1 \quad j=1,\cdots,m \tag{4.11}$$

其中：(4.1)表示用算法 A 装箱使用的箱子数目；(4.2)表示 m 个机器中装箱时间最长的机器所花费的时间；(4.3)表示算法 A 装箱数目近似比；(4.4)表示算法 A 装箱时间近似比；(4.5)表示物体总量不超过箱子容量；$W=1$ 表示箱子容量；(4.6)表示每一个物体只能装入一个箱子；(4.7)表示每一个物体只能分配给一个机器；(4.8)中 $x_{ir}=1$ 表示第 i 个物品装入第 r 个箱子，反之表示第 i 个物品未装入第 r 个箱子；(4.9)中 $\delta_{ij}=1$ 表示第 i 个物品分配给第 j 个机器，反之表示第 i 个物品未分配给第 j 个机器；(4.10)中 $y_r=1$ 表示第 r 个箱子被使用，反之表示第 r 个箱子未被使用；(4.11)表示所有机器都被使用。

4.2. 算法设计

算法 A

Input：物品序列 $L=(a_1,a_2,\cdots,a_n)$ 以及每个物品 a_i 的尺寸为 $s(a_i)\in(0,1]$，$i=1,2,\cdots,n$，机器效率 $\alpha_j\in[0.5,1]$，$j=1,2,\cdots,m$，使用效率为 1 的机器将 n 个物体装箱的时间为 $\{t_1,t_2,\cdots,t_n\}$。

Output：使用的箱子数及装箱方案。

Begin

Step 1：将物体按照装箱时间从大到小排序 $t_1\ge t_2\ge\cdots\ge t_n$，如果装箱时间相同，那么按照物品尺寸排序。

Step 2：将机器效率按照从大到小排序 $\alpha_1\ge\alpha_2\ge\cdots\ge\alpha_m$，其中 $\alpha_1=1$，$\alpha_j\in[0.5,1], j=2,3,\cdots,m$。

Step 3：若 $n=km$，则 n 不变；若 $(k-1)m<n<km$ (k 为任意整数)，则 $n=km$，且 $t_{n+1}=t_{n+2}=\cdots=t_{km}=0$。

Step 4：若 k 为偶数，记物体集合为

$$S_j=\{a_j,a_{2m-j+1},a_{2m+j},a_{4m-j+1},a_{4m+j},\cdots,a_{km-j+1}\},j=1,2,\cdots,m,$$

其对应的装箱时间集合为

$$T_j=\{t_j,t_{2m-j+1},t_{2m+j},t_{4m-j+1},t_{4m+j},\cdots,t_{km-j+1}\},j=1,2,\cdots,m。$$

若 k 为奇数，记物体集合为

$$S_j=\{a_j,a_{2m-j+1},a_{2m+j},a_{4m-j+1},a_{4m+j},\cdots,a_{(k-1)m+j}\},j=1,2,\cdots,m,$$

其对应的装箱时间集合为

$$T_j=\{t_j,t_{2m-j+1},t_{2m+j},t_{4m-j+1},t_{4m+j},\cdots,t_{(k-1)m+j}\},j=1,2,\cdots,m。$$

那么每个机器所需装箱物体的个数为 k 个。

Step 5：令 $l\left(T_j\right)$ 表示 T_j 中元素相加的总和，将 $l\left(T_j\right)$ 从大到小重新排序，设为 $l\left(T_1\right)\geq l\left(T_2\right)\geq\cdots\geq l\left(T_m\right)$，将第 j 个子集 S_j 中所有元素分配给机器 $m_j\left(j=1,2,\cdots,m\right)$，其效率为 α_j。分配后的第 j 台机器所花费的总时间用 $l\left(m_j\right)$ 表示，即 $l\left(m_j\right)=\dfrac{l\left(T_j\right)}{\alpha_j}$。

Step 6：对于第 j 台机器：

令 $B=\varnothing,s_0=0,r_j=1,B_{S_j,r_j}:=B\cup\{a_1\},s\left(B_{S_j,r_j}\right):=s_0+s\left(a_1\right)$

For $c=2$ to k do

If $s\left(a_c\right)+s\left(B_{S_j,r_j}\right)\leq 1$ then

置 $B_{S_j,r_j}:=B_{S_j,r_j}\cup\{a_c\},s\left(B_{S_j,r_j}\right):=s\left(B_{S_j,r_j}\right)+s\left(a_c\right)$

Else

$r_j:=r_j+1,B_{S_j,r_j}:=B\cup\{a_c\},s\left(B_{S_j,r_j}\right):=s_0+s\left(a_c\right)$

令装箱总数 $h=r_1+r_2+\cdots+r_j$。

Step 7：输出箱子数目 h 和装箱方案 $B_{S_1,1},B_{S_1,2},\cdots,B_{S_1,r_1},\cdots,B_{S_j,1},B_{S_j,2},\cdots,B_{S_j,r_j}$。

End

4.3. 算法分析

引理 1：对任意的 $k\geq 2$，对任意实例 I，有 $R_{A_T}\left(I\right)=2-\dfrac{1}{m}$。

证明：设

$$l\left(m_a\right)=\max\left(l\left(m_j\right)\right),j=1,2,\cdots,m;\ l\left(m_b\right)=\min\left(l\left(m_j\right)\right),j=1,2,\cdots,m,$$

则有 $A_T\left(I\right)=l\left(m_a\right)$。设机器 m_a 所需装箱的元素集合为 S_a，对应装箱时间为 T_a，机器 m_b 的元素集合为 S_b，对应装箱时间为 T_b。因此 $l\left(m_a\right)-l\left(m_b\right)=\dfrac{l\left(T_a\right)}{\alpha_a}-\dfrac{l\left(T_b\right)}{\alpha_b}$。

下面讨论 $l\left(m_a\right)-l\left(m_b\right)$ 的大小。

当 $A_T^*\left(I\right)\geq 2t_1$ 时，令

$$\alpha_c=\max\{\alpha_a,\alpha_b\},l\left(T_c\right)=\max\left\{l\left(T_a\right),l\left(T_b\right)\right\},$$

$$\alpha_d=\min\{\alpha_a,\alpha_b\},l\left(T_d\right)=\min\left\{l\left(T_a\right),l\left(T_b\right)\right\},$$

那么

$$l\left(m_a\right)-l\left(m_b\right)=\frac{l\left(T_a\right)}{\alpha_a}-\frac{l\left(T_b\right)}{\alpha_b}=\frac{\alpha_b l\left(T_a\right)-\alpha_a l\left(T_b\right)}{\alpha_a\alpha_b}$$

$$\leq\frac{\alpha_c\left|l\left(T_a\right)-l\left(T_b\right)\right|}{\alpha_a\alpha_b}=\frac{\alpha_c\left(l\left(T_c\right)-l\left(T_d\right)\right)}{\alpha_a\alpha_b}$$

为了方便计算，不妨设 $T=\left(t_{i1},t_{i2},t_{i3},\cdots,t_{ik}\right),i=1,2,\cdots,m$。

那么

$$l(m_a)-l(m_b)\le\frac{\alpha_c\left(l(T_c)-l(T_d)\right)}{\alpha_a\alpha_b}$$
$$=\frac{\alpha_c}{\alpha_a\alpha_b}\left[\left(t_{c1}+t_{c2}+t_{c3}+\cdots+t_{ck}\right)-\left(t_{d1}+t_{d2}+t_{d3}+\cdots+t_{dk}\right)\right]$$
$$=\frac{\alpha_c}{\alpha_a\alpha_b}\left[t_{c1}-\left(t_{d1}-t_{c2}\right)-\left(t_{d2}-t_{c3}\right)-\cdots-\left(t_{d(k-1)}-t_{ck}\right)-t_{dk}\right]$$
$$\le\frac{\alpha_c}{\alpha_a\alpha_b}\left(t_{c1}-t_{dk}\right)\le\frac{\alpha_c}{\alpha_a\alpha_b}t_{c1}\le 2t_1$$

这时，我们有 $A_T^*(I)\ge 2t_1\ge l(m_a)-l(m_b)$。

当 $A_T^*(I)<2t_1$ 时，若 $\alpha_a>\alpha_b$，由算法可知，$l(T_a)>l(T_b)$。就有

$$l(m_a)-l(m_b)-\frac{t_{a1}}{\alpha_a}=\frac{l(T_a)}{\alpha_a}-\frac{l(T_b)}{\alpha_b}-\frac{t_{a1}}{\alpha_a}$$
$$=\frac{\alpha_b\left(t_{a2}+t_{a3}+t_{a4}+\cdots+t_{ak}\right)-\alpha_a\left(t_{b1}+t_{b2}+t_{b3}+\cdots+t_{bk}\right)}{\alpha_a\alpha_b}$$
$$\le\frac{\alpha_a}{\alpha_a\alpha_b}\left[\left(t_{a2}+t_{a3}+t_{a4}+\cdots+t_{ak}\right)-\left(t_{b1}+t_{b2}+t_{b3}+\cdots+t_{bk}\right)\right]$$
$$=\frac{1}{\alpha_b}\left[\left(t_{a2}-t_{b1}\right)+\left(t_{a3}-t_{b2}\right)+\cdots+\left(t_{ak}-t_{b(k-1)}\right)-t_{bk}\right]\le 0$$

即 $l(m_a)-l(m_b)\le\frac{t_{a1}}{\alpha_a}$。而 $A_T^*(I)\ge\frac{t_{a1}}{\alpha_a}$，故有 $A_T^*(I)\ge\frac{t_{a1}}{\alpha_a}\ge l(m_a)-l(m_b)$。

若 $\alpha_a\le\alpha_b$，由算法可知 $l(T_a)\le l(T_b)$，且有 $l(m_a)\ge l(m_b)$，即 $\frac{l(T_a)}{\alpha_a}\ge\frac{l(T_b)}{\alpha_b}$。此时有 $l(T_a)\le l(T_b)\le\frac{\alpha_b}{\alpha_a}l(T_a)$。当 t_1 确定时，这种情况是可数的，并且在这种情况下，$l(m_a)-l(m_b)$ 的极限情况是当 $\alpha_a=0.5,\alpha_b=1,l(T_a)=l(T_b)$ 时。

那么此时有

$$l(m_a)-l(m_b)=2l(T_a)-l(T_b)=2l(T_a)-l(T_a)=l(T_a)$$

而

$$A_T^*\ge l(m_b)+\frac{l(m_a)-l(m_b)}{m}=l(T_b)+\frac{2l(T_a)-l(T_b)}{m}\ge l(T_a)=l(m_a)-l(m_b)。$$

故当 $\alpha_a\le\alpha_b$ 时，$A_T^*(I)\ge l(m_a)-l(m_b)$。

综上我们有 $A_T^*(I)\ge l(m_a)-l(m_b)$，那么就有

$$A_T(I)-A_T^*(I)=l(m_a)-A_T^*(I)\le l(m_a)-\left(l(m_b)+\frac{l(m_a)-l(m_b)}{m}\right)$$
$$=\left(1-\frac{1}{m}\right)\left(l(m_a)-l(m_b)\right)\le\left(1-\frac{1}{m}\right)A_T^*(I)$$

从而有 $A_T(I)\le\left(2-\frac{1}{m}\right)A_T^*(I)$。得证。

引理 2：对任意实例 I，有 $R_A(I)=2+c$ （其中 $0<c\le 1$）。

证明：由算法可知，第 j 台机器所需装箱的元素集合为 S_j，假设使用算法将 S_j 装完需要 β_j 个箱子，那么将 S_j 中的所需箱子两两结合相加得到 S_j 中所有元素大小之和 $s\left(S_j\right)\geq\frac{\left(\beta_j-1\right)}{2}$，对 $s\left(S_j\right)$ 求和，就可以得到

$$s(I)=\sum_{j=1}^{m}s\left(S_j\right)\geq\frac{\sum_{j=1}^{m}\beta_j-m}{2}$$

又因为 $A(I)=\sum_{j=1}^{m}\beta_j$，$A^*(I)\geq s(I)$，那么有 $\frac{A(I)}{A^*(I)}\leq 2+\frac{m}{A^*(I)}$，而 $A^*(I)$ 表示使用 m 个机器同时装箱对应的最优解，那么就有 $A^*(I)\geq m$。令 $c=\frac{m}{A^*(I)}$，那么 $0<c\leq 1$，则有 $\frac{A(I)}{A^*(I)}\leq 2+c\left(0<c\leq 1\right)$。得证。

定理 1：装箱时间近似比与装箱数目近似比的乘积是 $\left(2+c\right)\left(2-\frac{1}{m}\right)$ (其中 $0<c\leq 1$)。

证明：由定理 1 和定理 2 可知装箱时间近似比为 $2-\frac{1}{m}$，箱子数目的近似比为 $(2+c)\,(0<c\leq 1)$，故装箱时间近似比与装箱数目近似比的乘积是 $\left(2+c\right)\left(2-\frac{1}{m}\right)$ (其中 $0<c\leq 1$)。

定理 2：算法 A 的时间复杂度为 $O(n\log n)$。

证明：算法中的 step 1 的时间复杂度为 $O(n\log n)$，step 2 和 step 5 的时间复杂度为 $O(m\log m)$，step 6 的时间复杂度为 $O(n)$，所以算法 A 的时间按复杂度为 $O(n\log n)$。

4.4. 算法应用

下面给一个算法的应用实例。给定物品序列 $L=\left(a_1,a_2,\cdots,a_{10}\right)$，其尺寸为 $s\left(a_i\right)=\{0.4,0.2,0.7,0.3,0.5,0.4,0.2,0.3,0.6,0.2\}$，$i=1,2,\cdots,n$，两台机器同时装箱，装箱效率为 $\alpha_j=\{1,0.8\}$，$j=1,2$，使用效率为 1 的机器将 n 个物体装箱的时间为 $T=\{2,1,3,2,2,1,1,3,2,1\}$。

那么使用算法 A 装箱的结果为 $A_T(L)=\frac{45}{4}$，$A(L)=5$，而 $A_T^*(L)=10$，$A^*(L)=4$，故满足算法证明结果。

5. 小结

本文提出了一种带时间效率约束的一类一维装箱问题，在传统一维装箱的基础上加了时间效率约束，同时考虑装箱数目和装箱时间，设计算法求出近似比，分析其时间复杂度，并取得了较好的结果。

此外，当物品数目较少且机器数目较多时，该算法的性能较差，本文进一步的工作是根据物品数目的多少来选择不同数量的机器装箱，以达到省时省力的最佳效果。我们未来的工作将继续研究此类问题，以期能对实际生活提供帮助。

致　谢

非常感谢对本稿提出意见的审稿人及导师。

基金项目

云南民族大学数学与计算机科学学院研究生项目(SJXY-2021-025)。

参考文献

[1] Garey, M.R. and Johnson, D.S. (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, San Francisco.

[2] Coffman, E.G., Garey, M.R. and Johnson, D.S. (1996) Approximation Algorithms for Bin Packing: A Survey. Approximation Algorithms for NP-Hard Problems. PWS Publishing Co., 46-93.

[3] Johnson, D.S. (1974) Fast Algorithms for Bin Packing. *Journal of Computer and System Sciences*, **8**, 272-314. https://doi.org/10.1016/S0022-0000(74)80026-7

[4] Lee, C.C. and Lee, D.T. (1985) A Simple On-Line Packing Algorithm. *Journal of ACM*, **32**, 562-572. https://doi.org/10.1145/3828.3833

[5] Júnior, B.A. and Pinheiro, P.R. (2014) Dealing with Nonregular Shapes Packing. *Mathematical Problems in Engineering*, **2014**, Article ID: 548957. https://doi.org/10.1155/2014/548957

[6] Wu, H.T., Leung, S.C.H., Si, Y.-W., Zhang, D.F. and Lin, A. (2017) Three-Stage Heuristic Algorithm for Three-Dimensional Irregular Packing Problem. *Applied Mathematical Modelling*, **41**, 431-444. https://doi.org/10.1016/j.apm.2016.09.018

[7] Hu, Q., Wei, L. and Lim, A. (2017) The Two-Dimensional Vector Packing Problem with General Costs. *Omega*, **74**, 59-69. https://doi.org/10.1016/j.omega.2017.01.006

[8] Gzara, F., Elhedhli, S. and Yildiz, B.C. (2020) The Pallet Loading Problem: Three-Dimensional Bin Packing with Practical Constraints. *European Journal of Operational Research*, **287**, 1062-1074. https://doi.org/10.1016/j.ejor.2020.04.053

[9] Johnson, D.S., Demers, A., Ullman, J.D., *et al.* (1974) Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms. *SIAM Journal of Computing*, **3**, 299-325. https://doi.org/10.1137/0203025

[10] Yao, A.C.-C. (1980) New Algorithms for Bin Packing. *Journal of the ACM*, **27**, 207-227. https://doi.org/10.1145/322186.322187

[11] Brown, D.J. (1979) A Lower Bound for On-Line One-Dimensional Bin Packing Algorithms. Technical Rept., R-864 (ACT-19), UILU-78-2257.