DOI:10.13196/j.cims.2018.03.005

求解二维矩形装箱问题的启发式算法

尚正阳,顾寄南,丁 卫,Enock A. Duodu
(江苏大学 制造业信息化研究中心,江苏 镇江 212000)

摘 要:为实现二维矩形装箱问题的高效求解,提出一种启发式最优剩余空间算法。该算法以促使小矩形的放置更为紧密和剩余空间更加平滑为基本思想,通过空间分割、放置位置选择和最优解搜索 3 个相来实现对三维矩阵装箱问题的求解。基于两个经典的 C21 和 N13 数据集,最优剩余空间算法与多种算法进行了对比实验。测试结果表明,所提算法能够在最短的时间内得到 C21 全部算例的 100%布置和 N13 数据集的 9 个最优解,在计算效率和计算效果上均优于现阶段的其他算法。

关键词:矩形装箱;布局优化;空间利用率最大;启发式算法
中图分类号:TP391;U673 **文献标识码**:A

Heuristic algorithm for 2D rectangle packing problem

SHANG Zhengyang, GU Jinan, DING Wei, DUODU Enock A
(Mechanical Information Research Center, Jiangsu University, Zhenjiang 212013, China)

Abstract: To solve the two-dimensional Rectangular Packing Problem (RPP) efficiently, a recursive heuristic algorithm named Best Residual Space Algorithm (BRSA) was proposed. Based on the idea to make the items placement more compact and the remaining space smoother, three stages that were space division, placement position selection and optimal solution searching was adopted. Based on two sets of classical benchmark instances C21 and N13, BRSA was compared with many algorithms. The experimental results showed that BRSA was able to obtain all 100% filling rates for C21 and nine optimal solutions for N13 in shortest time, which was superior to the current algorithms in both computational efficiency and effectiveness.

Keywords: rectangular packing; layout optimization; space utilization maximization; heuristic algorithms

0 引言

二维矩形装箱问题 (the 2D Rectangle Packing Problem, RPP)是指将不同尺寸的小矩形放入一个预先设定好尺寸的大矩形中,以使大矩形的空间利用率达到最大。作为一个基础的理论研究,RPP 可以结合不同的约束和目标被应用于不同的工业领域。如造船的空间调度问题,电脑的动态内存分配问题,以及玻璃或者金属的切割问题等[1-3]。

RPP 是一个典型的 NP-hard 问题,为了得到更好的求解效果和效率,学者们设计出大量的基于不同策略的算法。早期作为一种简单的启发式放置规则,学者们提出了 BL(bottom-left)[4] 和 BLF(bottom-left-fill)[5] 的放置方法用于多元求解算法的构建。在此基础上,不同的搜索方法(如退火算法[6-7]、遗传算法[8] 和禁忌搜索算法[9-10] 等)也被引入 RPP 的求解中。进一步,确定性启发式算法因相对于其他传统算法具有较好的求解效果而成为近年来的研究重点。黄文奇等[11-12] 提出一种高效的拟物拟人算法,通过对小矩形放置位置进行定量评价来选择一个最优的布置策略;基于该思想,张德富等提出了递归的启发式(Heuristic Recursivem, HR)算法[13]

收稿日期:2016-11-10;**修订日期**:2017-03-21。Received 10 Nov. 2016; accepted 21 Mar. 2017.
基金项目:国防基础科研资助项目(JCKY2013414C001)。**Foundation item**: Project supported by the National Defense Basic Research Foundation, China(No. JCKY2013414C001).

和基于优先度的启发式(Priority Heuristic, PH)算法[14],并将二维装箱方法有效地扩展到了三维装箱领域[15-16];Wang 等对确定性启发式算法的装箱过程进行了深入分析,并提出了剩余空间最大化算法(Residual-Space-Maximized-Packing, RSMP)[17]。

目前,装箱问题的求解过程通常被分解为基础算法构建和最优解搜索两个阶段。为此,曹大勇等[18]在 HR 算法的基础上,发展了一种两阶段启发式算法;He 等[19]使用树搜索的方式对算法结果进行迭代和优选,并提出迄今为止对 RPP 具有最好求解效果的最优适应度算法(Best Fit Algorithm, BFA);之后,He 等[20]发展了这种算法,并将其应用于不同的装箱问题上;Wei 等[21]提出了最少浪费优先(Least Wasted First, LWF)算法,并为实现大规模条形装箱问题(strip packing problem)的快速求解,发展了一种结构简单的基于顶端参考线的启发式(Improved Skyline based Heuristic, ISH)算法[22]。

虽然现阶段一些确定性的启发式算法能够取得较好的解,但是其无论在求解精度还是在求解效率上都有待提升。因此,本文提出一个更为高效和简单的算法——最优剩余空间算法(Best Residual Space Algorithm, BRSA)。为验证所提出算法的性能,对具有代表性的 C21 数据集(Hopper and Turton[23] 提出)和 N13 数据集(Burke 等[6] 提出)进行了实验测试。结果显示,BRSA 不但能够取得较好的解,而且其运算时间较短,相较于其他算法在计算效果和效率上都具有一定的优势。

1 问题描述

将小矩形(简称 item)的宽度和长度分别表示为 w_i,h_i,将大矩形(简称 sheet)的宽度和长度分别表示为 W,L。RPP 的目的是将 item 尽可能合理地放置到 sheet 中,在使 sheet 的空间利用率最大的同时使它们之间没有重叠或干涉。实际上,作为一个 NP-hard 问题,RPP 不存在多项式时间复杂度的最优求解算法,即 RPP 的求解就是要在接近"组合爆炸"的可能中寻找一个最优解。图 1 所示为一个简单的例子,将 4 个尺寸为 3×1 和 1 个尺寸为 2×2 的 item 放置到一个尺寸为 4×4 的 sheet 中,且每个 item 可以以任意顺序、任意方式和任意位置放置。不难发现,即使是对于如此简单的例子,也很难从众多的 item 放置组合中找到一个整体最优的布置方案。因此,为能够主动引导 item 的合理布局,往往使用启发式的放置方法,一种最优的 item 放置方案如图 1 中所示。

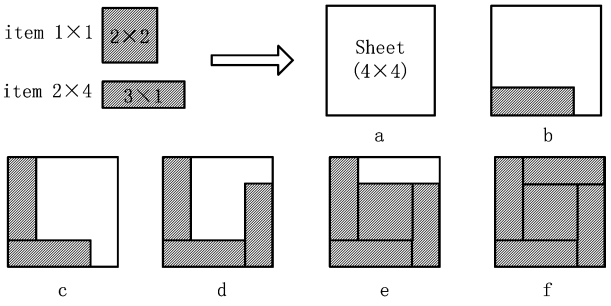


图1 RPP问题示例

为了对 RPP 进行数学描述,将 sheet 看作一个二维坐标系,其左下角为坐标原点。每个 item 在坐标系中的位置用其左下角点(x_{i1},y_{i1})和右上角点(x_{i2},y_{i2})表示,参数 $\delta=1$ 表示该 item 被放入 sheet 中,$\delta=0$ 表示未放入。RPP 的目标和约束可以表示如下:

$$\max \frac{\sum_{i=1}^{n} w_i h_i \delta_i}{W \cdot L}。$$

$$\text{s.t.} \quad (x_{i2}-x_{i1},y_{i2}-y_{i1}) \in \{(w_i,h_i),(h_i,w_i)\}; \tag{1}$$

$$\max(x_{i1}-x_{j2},x_{j1}-x_{i2},y_{i1}-y_{j2},y_{j1}-y_{i2})\delta_i\delta_j \geqslant 0; \tag{2}$$

$$0 \leqslant x_{ik} \leqslant W, 0 \leqslant y_{ik} \leqslant L, k \in \{1,2\}。 \tag{3}$$

其中:式(1)表示每一个 item 的放置必须与 sheet 正交,即 item 被放置后,其边必须与 sheet 的边平行;式(2)表示被放入的 item 之间不能存在干涉;式(3)表示 item 的放置不能超出 sheet 的边界。

2 BRSA 算法的构建

本文 RPP 的求解被分解为空间分割、放置位置选择和最优解的搜索 3 个相。其中,空间分割是一种离散和组合的方法,它将待填充空间划分为多个具有不同尺寸的虚拟矩形空间,并通过不断对单个虚拟矩形空间进行最优放置来实现整体 sheet 的空间利用率最大;放置位置的选择是指在该过程中,定量地评价 item 的放置位置,并选择一个最优的 item 放置策略。这两个相可以结合构建为一个基础算法,实现 RPP 的初始求解。然而,以上基础算法中,其每一次迭代操作都是基于对当下 item 最优放置

策略的求解,缺乏全局性考虑和优化。因此,将最优解搜索引入到算法的求解中。基于这种求解思路,本文提出了 BRSA。

2.1　空间分割

分割和组合的递归求解思想能够很好地求解装箱问题[14,21],因此本算法也引入了这一概念:放置 item 时,将 sheet 中的剩余空间分割成多个矩形的虚拟空间,并递归地对这些虚拟空间进行最优填充。在该过程中,虚拟空间的存在就是要正确地反映出 sheet 中已存在 item 和剩余空间的布局情况,从而引导后续 item 的合理放置。为了更好地实现 RPP 的求解,本文继承和发展了这种求解思想,提出一种新的空间分割方法,并给出一些定义:

(1)*真实边界线*　指 sheet 中的不相交边界线,如图 2a 和图 2b 中的加粗线段。

(2)*虚拟边界线*　指 sheet 中 item 的右上角点,如图 2a 中的 $vc1$ 所示。如果 $vc1$ 点不与其他 item(sheet 可以看做是一个大 item)相交,则通过该点分别向其上方和右方做延伸线,直至与其他 item 或者与已存在的虚边界线相交为止。以此类推,图 2b 中的所有虚实线即为虚拟边界线。

(3)*虚拟矩形空间*　虚拟边界线将 sheet 中的剩余空间分割成了多个大小不一的矩形空间,称为虚拟矩形空间。

(4)*角点位置*　以 item 的左下角点为放置角点,空间中所有能够放下最小 item 的角点,或者向左、向下平移后能够放下最小 item 的点,称作角点位置,如图 2b 中的标志"⌞"所示。实际上,角点位置尽量设置在空间左下角,是为了能够使 item 尽量放置在 sheet 中的左下方位置,从而可以最大程度地使零散空间集中在 sheet 中的左上角,便于放置剩余的 item,这一点与 BLF[5] 的放置思想相同。

(5)*放置空间*　在某种放置策略下,能够完全包含被放置 item 的最小虚拟矩形空间,称为该 item 的放置空间。实际上,放置空间是剩余空间状态的一种具象化表示,算法能够通过评价 item 与其放置空间的相交状态,选出一个较好的放置策略。

如以上定义所述,item 只能放置或旋转后放置在 sheet 中的角点位置。当有 item 被放入 sheet 中时,空间中的虚边界线就会更新一次,即空间被重新分割为多个虚拟矩形空间。其中,空间的分割规则

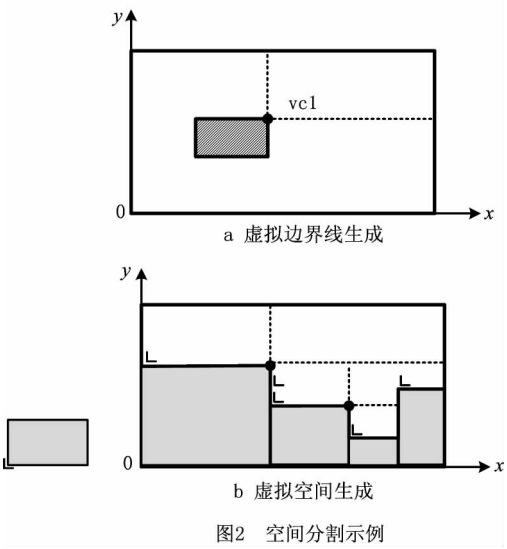


图2　空间分割示例

如算法 1 所示。该规则充分考虑了剩余空间的形状特性,所生成的虚拟矩形空间分布规则且能够准确反映出 sheet 中的格局状态。这是非常有意义的,因为其中一部分虚拟矩形空间将成为 item 的放置空间,直接参与并影响 item 的放置位置评价。最后需要注意的是,虚边界线的构造具有先后顺序,其空间分割的伪代码如下:

算法 1　空间分割。

设置 Q 为 sheet 中虚拟边界线角点的集合,$cv1_i$ 为新加入 item 的虚拟边界线角点。

```
1:while 一个新 item 被放入到 sheet 后
2:Q=[Q;cv1_i]
3:for Q 中的每一个点 do
4:if 这个点与其他 item 相交 then
5:    delete
5:else if
6:    生成虚拟边界线
7:end if
8:end for
9:end while
10:更新集合 Q
```

2.2　放置位置选择

基于由分割规则所产生的虚拟矩形空间,算法需要对 item 的所有可放置位置进行定量评价。而评价的目的就是要为 item 选择一个最优的放置策略(包括放置位置和放置方式)。通常,一个放入 sheet 的 item 与其所在放置空间的关系越紧密越好,且剩余空间的零碎空间越少越好(sheet 中的虚拟矩形空间即为零碎空间)。基于这两点准则,给出如下定义和综合性评价规则:

定义 1　完全重合度。当一个 item 放入 sheet

中，它的边与其所在放置空间的边完全相交的个数，称为该 item 在这种放置状态下的完全重合度。其值表示为 p，$p\in[0,1,2,3,4]$。

在算法的执行过程中，每种 item 的放置策略均为 $d=\frac{a_i,o,p,w_i\times h_i,1}{(x_{i1}\times y_{i1})}$，其中 o 表示 item 是否旋转 90 度。按照 d 中参数出现的顺序逐个比较两种放置策略，数值较大的具有较高优先级。另外，根据 item 在放置过程中的具体状态，给出 3 个评价规则：

(1)对于完全重合度的计算，当存在 item 的相邻两条边与其放置空间的相邻两条边完全重合时，实边界线与虚边界线一同考虑，否则只对实边界线的完全重合数量进行计算(如图 3)。这与 BFA[19] 中的占角思想相同。

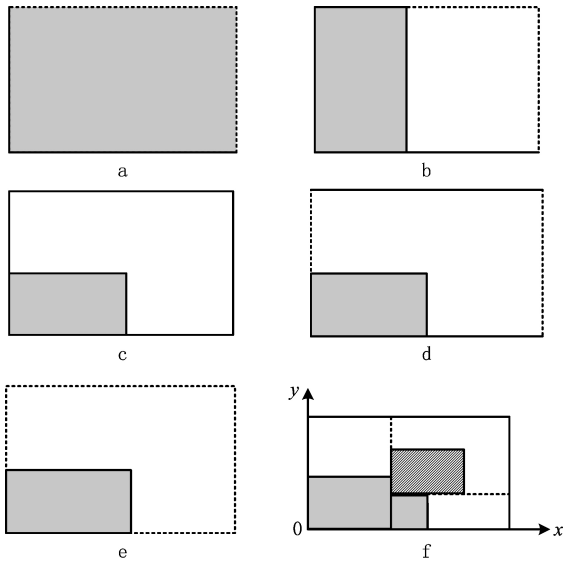


图3 item在放置空间中的不同放置状态

图 3 给出了 item 的几种放置状态。图 3a 表示 item 的 4 条边完全与其所在放置空间的边界线相交，即 $p=4$；图 3b 表示 item 有两条相邻的边与实边界线完全相交，一条边与虚边界线完全相交，其 $p=3$；图 3c 表示 item 有两条边与实边界线相交，故 $p=2$；图 3d 和 3e 表示 item 没有两条相邻的边与实边界线完全重合，在计算其完全重合度时，只考虑实边界线，其值分别为 $p=1$ 和 $p=0$。图 3f 给出了图 3e 在 sheet 中的一种最差的放置状态。

从 sheet 中的虚拟矩形空间数量来看，当 $p=4$ 和 $p=3$ 时，虚拟矩形空间的个数没有增加，新放入的 item 与其放置空间的关系最为紧密，且剩余空间最为平滑，这两种情况被认为是最好的。随着完全重合度值逐渐变小，sheet 中虚拟矩形空间的数量逐渐增多，新放入的 item 对剩余空间的影响逐渐增大。因此，p 值越小，item 的放置效果越差。

(2)在评价过程中，$p=4$ 和 $p=3$ 时的放置效果相同。

(3)如果一个 item 被放入到 sheet 后所产生的最小虚拟矩形空间的宽度小于剩余 item 的最小宽度，即有部分新产生的空间不能被放入任何剩余 item，则 $p=1$。

基于以上空间划分方法和放置位置选择规则，构建基础的 $BRSA_0$ 算法，其伪代码如下：

算法 2 基础算法 $BRSA_0$。

1:初始化

2:while item 没有被全部放入，或者至少有一个矩形可以被放入到 sheet 中时 do

3:for 每一个候选 item

4: for sheet 中的每一个角点位置

5: 按照评价规则，计算 item 在每一种放置位置下的完全重合度 p，并以放置策略 d 的形式进行表示

6: end for

7:end for

8:选择具有最大 p 值的一组放置策略 d，并构建成为集合 D

9:将 D 中的不同放置策略 d 按照 item 面积递减的顺序排列，面积相同的，按照其放置位置坐标的乘积由小到大进行排列。

10:选择 D 中最优(第一个)的放置策略，并执行

11:算法 1

12:end while

13:输出 sheet 的布局及其相应的空间利用率。

基础算法 $BRSA_0$ 对 item 在剩余空间中的每个放置位置进行评价，并组建一个具有最大完全重合度的集合 D。当 D 中存在不止一种放置策略时，按照 item 的面积从大到小和放置位置坐标乘积从小到大的优先级进行排序，最终选出一个最优的策略。依次循环，直到 item 被完全放置，或者 sheet 中不能放入任何剩余 item 为止。

2.3 最优解的搜索

以上基础算法采用了贪婪策略作为对 item 的最优放置策略，即认为在每一次放置中，item 的面积越大、放置坐标越小，越应该被提前放入。实际上具有相同 p 值的不同 item 放置策略，在当前迭代中具有相同效果。任何一种简单的选择规则都不能使解全局最优。因此，本算法采用随机的方法对具有相同 p 值的放置策略进行选取，并通过多次迭代计算求得最优解。其伪代码如下：

算法 3　BRSA 算法

```
1:for i=1 to M do
2:算法 2:1-9
3:设置 N 的值:N=[ceil(P×k%)],P 为 D 中的放置策略个数
4:  if N>bound then
5:  N=bound
6:  end if
7:在 D 中的前 N 个放置策略中,随机选择一种并执行
8:算法 2:11-12
9:If sheet 的空间使用率为 100% then
10:停止算法,并输出此时的 sheet 布局
11:end if
13:end for
14:输出最优的 sheet 的布局及其相应的空间利用率
```

在运算过程中,如果出现多个具有相同 p 值的不同 item 放置策略,则进行随机选取。并通过算法的多次迭代运行获取全局最优解。为缩小每次迭代中的随机搜索范围,只对集合 D 中的前 N 个放置策略进行随机操作,其中参数 k 被设置为 0.25,bound 为 20。

3　结果和分析

3.1　实验结果

使用 C 语言对 BRSA 进行编程,并通过具有 2.6 GHz 运算频率的个人电脑进行计算。测试数据为 Hopper 等[23]提出的 C21 数据集和 Burke 等[6]提出的 N13 数据集,这些数据集在理论上都存在空间利用率为 100% 的填装状态。为了平衡所提算法的计算精度与计算效率,设其迭代次数 M 为中等规模的 5 000 次,计算结果取 100 次运算的平均值。

表 1 给出了 BRSA 和具有代表性的 GRASP[9],TABU[10],LWF[21] 和 BFA[19] 算法的运算结果。可以看出,所提算法能够对所有算例获得 100% 的空间利用率,明显优于传统的 GRASP 和 TABU 算法。相对同样是基于对item的放置位置进行确定性评价和选择的 LWF,BFA 算法,BRSA 则具有最短的 36.78 s 的平均运行时间。实际上,由于算法中加入了选择规则,使 item 在放置过程中有了一定的趋向性,不同放置方式往往能够达到相同的放置效果。正是因为这个原因,基于随机搜索的 LWF 和 BRSA 算法在获得最优解过程中,并不一定需要大量迭代计算,而且算法的平均运行时间也不一定会随算例规模的增加而增加。其中 7 个算例的最优布局如图 4 所示。

表 1　相关算法的空间利用率和运行时间比较

算例	item 的个数	sheet 的尺寸	Grasp		TABU		LWF		BFA		BRSA	
			填充率/%	时间/s	填充率/%	时间/s	填充率/%	时间/s	填充率/%	时间/s	填充率/%	时间/s
*C*11	16	20×20	**100**	94	**100**	42	**100**	0.01	**100**	0.00	**100**	0.01
*C*12	17	20×20	96.5	928	**100**	423	**100**	0.00	**100**	0.11	**100**	0.02
*C*13	16	20×20	**100**	6	**100**	95	**100**	0.02	**100**	0.00	**100**	0.00
*C*21	25	40×15	98.33	1944	**100**	44	**100**	0.00	**100**	0.17	**100**	0.05
*C*22	25	40×15	99.50	1736	**100**	416	**100**	0.01	**100**	0.02	**100**	0.02
*C*23	25	40×15	**100**	71	**100**	0	**100**	0.03	**100**	0.02	**100**	0.03
*C*31	28	60×30	98.06	2680	**100**	491	**100**	0.06	**100**	0.30	**100**	0.08
*C*32	29	60×30	97.50	3735	**100**	1011	**100**	3.38	**100**	0.72	**100**	2.32
*C*33	28	60×30	98.56	3092	**100**	552	**100**	1.49	**100**	1.48	**100**	0.91
*C*41	49	60×60	98.00	10,205	99.44	4527	**100**	1.52	**100**	12.99	**100**	7.53
*C*42	49	60×60	97.89	11,079	99.00	6759	**100**	2.42	**100**	0.09	**100**	5.31
*C*43	49	60×60	98.44	9441	99.44	5111	**100**	0.61	**100**	5.72	**100**	3.58
*C*51	73	60×90	98.30	21,207	98.93	13,597	**100**	3.84	**100**	2.31	**100**	9.69
*C*52	73	60×90	98.39	23,156	99.28	9680	**100**	0.05	**100**	0.09	**100**	2.38
*C*53	73	60×90	98.37	23,124	99.54	8206	**100**	1.71	**100**	6.17	**100**	5.51
*C*61	97	80×120	98.65	48,044	99.46	24,039	**100**	3.10	**100**	40.66	**100**	23.72
*C*62	97	80×120	98.47	46,549	98.42	39,986	**100**	0.12	**100**	1.67	**100**	8.11
*C*63	97	80×120	98.44	47,802	99.64	20,678	**100**	11.55	**100**	8.67	**100**	31.31
*C*71	196	160×240	98.08	376,014	99.03	305,438	**100**	1145.80	**100**	1121.44	**100**	424.96
*C*72	197	160×240	98.80	284,196	99.34	199,070	**100**	346.20	**100**	0.88	**100**	153.24
*C*73	196	160×240	98.29	37,099	98.61	561,575	**100**	7.12	**100**	3721.58	**100**	93.53
平均值			98.50	45,342	99.53	57,225	**100**	72.81	**100**	234.53	**100**	36.78

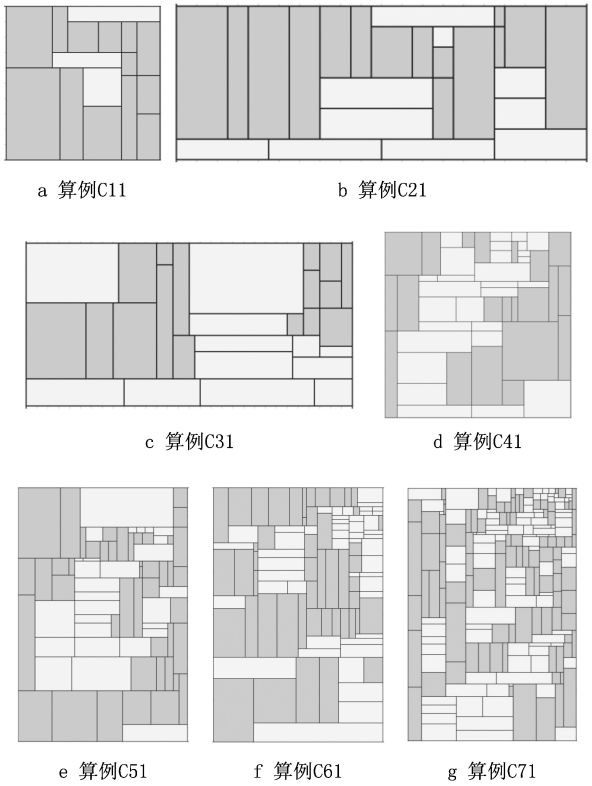


图4 部分算例C21的装箱结果

另外，一组包含 item 数量在 10～3152 的 N13 数据集也进行了测试，这是一组较难求解的 RPP 算例。如表 2 所示[19]，大量已存在算法只能实现 N13 中 3 个较为简单算例的最优布置，而即使是 BFA 也只取得了 7 个子算例的 100%填充率。因此，基于 N13 数据集，对 BFA 与 BRSA 算法的计算效果进行了详细对比。实验结果如表 3 所示，BRSA取得了 9 个算例的最优解，明显高于 BFA 和其他算法，且相对于 BFA，99.92%的平均空间填充率和 54.14 的平均运行时间均具优势。其中，两个算例 N6 和 N8 的最优空间布局如图 5 所示。

表 2 针对 N13 算例的最优解个数对比

算法	最优解算例	最优解算例个数	平均运行时间/s
BF	Null	0	0.12
BF+SA	*N*1，*N*2	2	＜60
GRASP	*N*1，*N*2	2	＜60
HA	*N*1，*N*2，*N*3	3	45 371.32
SWP	*N*1，*N*2，*N*3	3	8.37
BFA	*N*1，*N*2，*N*3，*N*9，*N*10，*N*11，*N*13	7	417.25

表 3 BFA 与 BRSA 算法针对 N13 算例的计算结果对比

算例	Item 的个数	Sheet 的尺寸	BFA		BRSA	
			填充率/%	时间/s	填充率/%	时间/s
*N*1	10	40×40	**100**	0.00	**100**	0.00
*N*2	20	30×50	**100**	0.44	**100**	0.41
*N*3	30	30×50	**100**	1.06	**100**	0.76
*N*4	40	80×80	99.75	9.06	99.75	27.12
*N*5	50	100×100	98.68	44.55	99.40	50.46
*N*6	60	50×100	99.82	19.91	100	4.62
*N*7	70	80×100	99.95	57.86	99.90	65.94
*N*8	80	100×80	99.89	139.56	**100**	17.33
*N*9	100	50×150	**100**	109.36	**100**	5.06
*N*10	200	70×150	**100**	337.20	**100**	23.29
*N*11	300	70×150	**100**	233.38	**100**	0.67
*N*12	500	100×300	99.85	4 453.11	99.97	483.22
*N*13	3152	640×960	**100**	18.74	**100**	25.00
平均值			99.84	417.25	99.92	54.14

3.2 BRSA 的计算复杂度

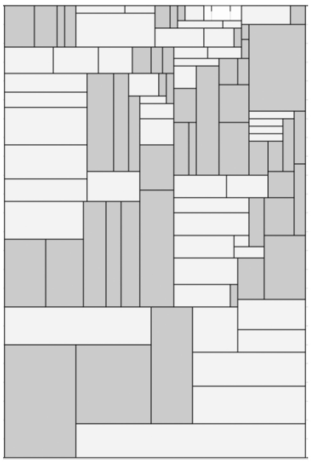
BRSA 的运行实际上是一个不断迭代基础算法 $BRSA_0$，并对其运算结果进行选优的过程。因此，用 $BRSA_0$ 代替 BRSA，分析其计算复杂度。正如上文所述，$BRSA_0$ 假设将每一个剩余 item 放入 sheet 中所有可放置位置，进而对其放置效果进行定量评价，以选择一个最优的 item 放置策略。因此，$BRSA_0$ 的计算复杂度可以分为迭代复杂度和评价复杂度两部分。

迭代复杂度指 $BRSA_0$ 在迭代选择过程中的计算花费，表示为

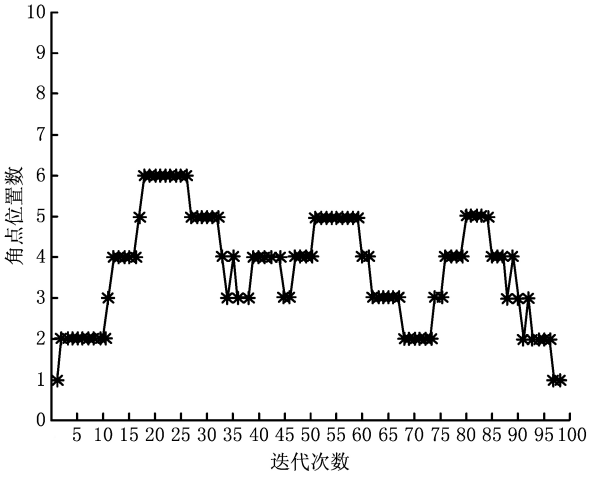
$$O(n)\times g_1+O(n-1)\times g_2+O(n-2)\times g_3+\cdots+O(1)\times g_n=O(n^2)\times\bar{g}。\quad(4)$$

式中：$O(n)$表示需要对剩余 item 进行计算的复杂度；g_i 表示每次迭代需要进行搜索的角点位置数。为便于表达，将其迭代复杂度近似表示为 $O(n^2)\times\bar{g}$，其中 $\bar{g}$ 是整个迭代过程中角点位置数的平均值。实际上，因为本算法的核心思想是在每次对 item 进行操作时，都选择一个能使 sheet 布局最紧密和剩余零碎空间最少的放置策略，所以空间中的角点位置数也就随之尽可能地减少了，相应算法中的 $\bar{g}$ 值始终维持在一个较低的水平。如图 6 所示为一个随机的求解 C61 的例子，其中：图 6a 为其最终布置效

果,图 6b 所示为其在每次迭代中的角点位置数。可以看出,在 item 的整个装箱过程中,其角点位置数的平均值为 3.71,计算复杂度相对较小。



a 放置结果



b 迭代过程中的角点位置数

图6　C61的一种装箱结果及其在迭代过程中的角点位置数

在评价复杂度上,当 item 被假设放入 sheet 后,$BRSA_0$ 只需要简单计算该 item 与其所在放置空间的完全重合度,其评价复杂度为 $O(1)$,因此 $BRSA_0$ 的总复杂度为 $O(n^2)\times\bar{g}$,与当前大多数基于搜索和迭代的求解算法类似。相对于现阶段性能最好的 BFA[19],它们在迭代复杂度上相似,但是在评价复杂度上,$BRSA_0$ 并不需要对剩余空间的状态进行额外操作,明显优于基于对活动空间进行搜索和判定的 BFA(BFA 的评价复杂度为 $O(1)\sim O(n^3)$)[20],而且本文对 *C*21 和 *N*13 数据集的对比实验也证明了这一点。

4　结束语

为高效获得 RPP 的最优解,本文将其求解过程分解为空间分割,放置位置选择和最优解搜索 3 部分,并据此提出了 BRSA。BRSA 通过在每一次迭代中使 item 的布局更为紧密,并使剩余空间更为平滑,来获得当前最优解,并使用随机的本地搜索方法实现装箱效果的全局最优化。然后,将所提出算法与其他具有代表性的典型算法进行实验对比,基于 C21 数据集的测试结果显示,BRSA 不但能够为其所有算例求得 100%的空间利用率,而且还具有最短的运算时间。为了进一步对比所提算法的计算效果,运行了一个较难求解的 N13 数据集。实验结果显示,BRSA 取得了 9 个算例的最优解,且相对于其他算法,其在计算效果和平均运算时间上都具有明显优势,而且给出了所提算法的计算复杂度。可以看出,所提算法在执行过程中不需要对剩余空间进行额外操作,相对于同样具有优秀性能的 BFA,大大减少了计算时间。总之,BRSA 算法不但性能优秀,而且具有构造简单和计算复杂度小的特点,能够实现 RPP 的高效求解。今后的研究的重点是进一步提高 BRSA 算法的运算精度,并在此基础上实现对其他相关装箱问题的求解。

参考文献:

[1] ANDREA L, SILVANO M, MICHELE M. Two-dimensional packing problems: a survey[J]. European Journal of Operational Research, 2002, 141(2): 241-52.

[2] DOWSLAND K A, DOWSLAND W B. Packing problems[J]. European Journal of Operational Research, 1992, 56(1): 2-14.

[3] DAVID P. Heuristics for the container loading problem[J]. European Journal of Operational Research, 2002, 141(2): 382-392.

[4] BAKER B S, COFFMAN JR E G, RIVEST R L. Orthogonal packing in two dimensions[J]. SIAM Journal on Computing, 1980, 9(4): 846-55.

[5] CHAZELLE B. The bottom-left bin packing heuristic: an efficient implementation[J]. IEEE Transactions on Computers, 1983, 32(8): 697-707.

[6] BURKE E K, KENDALL G, WHITWELL G. A new placement heuristic for the orthogonal stock-cutting problem[J]. Operations Research, 2004, 52(4): 655-71.

[7] BURKE E K, KENDALL G, WHITWELL G. A simulated annealing enhancement of the best-fit heuristic for the orthogonal stock-cutting problem[J]. INFORMS Journal on Computing, 2009, 21(3): 505-16.

[8] BORTFELDT A. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces[J]. European Journal of Operational Research, 2006, 172(3): 814-37.

[9] ALVAREZ V R, PARREÑO F, TAMARIT J M. A GRASP

algorithm for constrained two-dimensional non-guillotine cutting problems[J]. Journal of Operational Research Society, 2005,56(4):414-25.

[10] ALVAREZ V R, PARREÑO F, TAMARIT J M. A tabu search algorithm for two-dimensional non-guillotine cutting problems[J]. European Journal of Operational Research, 2007,183(3):1167-82.

[11] HUANG Wenqi, CHEN Duanbing. An efficient quasi-physical and quasi-human block-packing algorithm[J]. Computer Science,2005,32(11):182-186(in Chinese).[黄文奇,陈端兵.一种求解矩形块布局问题的拟物拟人算法[J].计算机科学,2005,32(11):182-186.]

[12] HUANG W Q, CHEN D B, XU R C. A new heuristic algorithm for rectangle packing[J]. Computers and Operations Research,2007,34(11):3270-80.

[13] ZHANG D F, KANG Y, DENG A S. A new heuristic recursive algorithm for the strip rectangular packing problem[J]. Computers and Operations Research, 2006, 33 (8): 2209-2217.

[14] ZHANG D F, SHI LY, STEPHEN C H, et al. A priority heuristic for the guillotine rectangular packing[J]. Information Processing Letters,2016,116(1):15-21.

[15] ZHANG Defu, PENG Yu, ZHU Wenxing, et al. A hybrid simulated annealing algorithm for the three-dimension packing problem[J]. Chinese Journal of Computers, 2009, 32 (11):2147-2156(in Chinese).[张德福,彭 煜,朱文兴,等.求解三维装箱问题的混合模拟退火算法[J].计算机学报,2009,32(11):2147-2156.]

[16] ZHANG Defu, PENG Yu, ZHANG Lili. A multi-layer heuristic search algorithm for three dimensional container loading problem[J]. Chinese Journal of Computers, 2012, 35(12): 2553-2561(in Chinese).[张德福,彭 煜,张丽丽.求解三维装箱问题的多层启发式搜索算法[J].计算机学报,2012,35(12):2553-2561.]

[17] WANG Y C, CHEN L J. Two-dimension residual-space-maximized packing[J]. Expert Systems with Applications, 2015,42(7):3297-3305.

[18] CAO Dayong, YANG Mei, KOTOV V M, et al. Two-stage heuristic algorithm for two-dimensional guillotine bin packing problem[J]. Computer Integrated Manufacturing Systems, 2012,18(9):1954-1963(in Chinese).[曹大勇,杨 梅,科托夫·弗拉基米尔·米哈伊拉维奇,等.二维一刀切装箱问题的两阶段启发式算法[J].计算机集成制造系统,2012,18(9):1954-1963.]

[19] HE K, HUANG W Q, JIN Y. An efficient deterministic heuristic for two-dimensional rectangular packing[J]. Computers and Operations Research,2012,39(7):1355-1363.

[20] HE K, JI P L, LI C M. Dynamic reduction heuristics for the rectangle packing area minimization problem[J]. European Journal of Operational Research,2015,241(3):674-685.

[21] WEI L J, ZHANG D F, CHEN Q S. A least wasted first heuristic algorithm for the rectangular packing problem[J]. Computers and Operations Research,2009,36(5):1608-14.

[22] WEI L J, HU Q, STEPHEN CH, et al. An improved skyline based heuristic for the 2D strip packing problem and its efficient implementation[J]. Computers and Operations Research,2017,80(4):113-127.

[23] HOPPER E, TURTON B. An empirical investigation of meta-heuristic and heuristic algorithm for a 2D packing problem [J]. European Journal of Operational Research, 2001, 128 (1):34-57.

作者简介:

尚正阳(1987—),男,河南义马人,博士研究生,研究方向:机械设计、仿真分析、智能算法,E-mail:shzy0205@163.com;

顾寄南(1964—),男,江苏镇江人,教授,博士,研究方向:基于网络的设计制造及远程监控、智能移动机器人的分析、仿真与控制技术;

丁 卫(1989—),女,江苏苏州人,博士研究生,研究方向:机械设计、仿真分析、机器视觉;

Enock A. Duodu(1975—),男,加纳人,博士研究生,研究方向:流体工程、仿真分析。