ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.6, June 2009, pp.1528−1538
doi: 10.3724/SP.J.1001.2009.03395

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

二维矩形条带装箱问题的底部左齐择优匹配算法*

蒋兴波[1,2], 吕肖庆[1,3+], 刘成城[1]

[1](北京大学 计算机科学技术研究所,北京 100871)

[2](第二军医大学 卫生勤务学系,上海 200433)

[3](北京大学 电子出版新技术国家工程研究中心,北京 100871)

Lowest-Level Left Align Best-Fit Algorithm for the 2D Rectangular Strip Packing Problem

JIANG Xing-Bo[1,2], LÜ Xiao-Qing[1,3+], LIU Cheng-Cheng[1]

[1](Institute of Computer Science and Technology, Peking University, Beijing 100871, China)

[2](Faculty of Health Services, Second Military Medical University, Shanghai 200433, China)

[3](National Engineering Research Center of New Technology in Electronic Publishing, Peking University, Beijing 100871, China)

+ Corresponding author: E-mail: lvxiaoqing@icst.pku.edu.cn

Jiang XB, Lü XQ, Liu CC. Lowest-Level left align best-fit algorithm for the 2D rectangular strip packing problem. *Journal of Software*, 2009,20(6):1528−1538. http://www.jos.org.cn/1000-9825/3395.htm

Abstract: In this paper, a heuristic placement algorithm for the two-dimensional rectangular strip packing problem, lowest-level left align best fit (LLABF) algorithm, is presented. The LLABF algorithm is based on the best-fit priority principle with overall consideration of several heuristic rules, such as full-fit first rule, width-fit first rule, height-fit first rule, joint-width-fit first rule and placeable first rule. Unlike the bottom-left (BL), the improved-bottom-left (IBL) and the bottom-left-fill (BLF) heuristic placement algorithms, LLABF algorithm dynamically selects the best-fit rectangle for packing. The computation result shows that 2DR-SPP can be solved more effectively by combining the LLABF algorithm with the genetic algorithm (GA).

Key words: lowest-level left align best fit (LLABF) algorithm; genetic algorithm; 2D rectangular strip packing problem; heuristic placement algorithm

摘 要: 针对二维矩形条带装箱问题提出了一种启发式布局算法,即底部左齐择优匹配算法(lowest-level left align best fit,简称 LLABF). LLABF 算法遵循最佳匹配优先原则,该原则综合考虑完全匹配优先、宽度匹配优先、高度匹配优先、组合宽度匹配优先及可装入优先等启发式规则.与 BL(bottom-left),IBL(improved-bottom-left)与 BLF(bottom-left-fill)等启发算法不同的是,LLABF 能够在矩形装入过程中自动选择与可装区域匹配的下一个待装矩形.计算结果表明,LLABF 结合遗传算法(genetic algorithm,简称 GA)解决二维条带装箱问题更加有效.

关键词: 最低左对齐最佳匹配(LLABF)算法;遗传算法;二维矩形条带装箱问题;启发式布局算法

中图法分类号: TP18 **文献标识码**: A

二维矩形条带装箱问题(2D rectangular strip packing problem,简称 2DR-SPP)是一个典型的组合优化问题,

* Received 2008-02-29; Revised 2008-04-15; Accepted 2008-06-03

其应用相当广泛,如工业领域中的新闻组版、布料切割、金属下料等.2DR-SPP通常是指将若干个不同规格的矩形$\{\pi_1,\pi_2,\ldots,\pi_n\}$装入宽度$W$固定、长度$L$不限的矩形容器$C$中,要求装完所有矩形后占用高度$H_{packing}$最小,从而达到节省材料的目的.在矩形的装入过程中,要求满足:① π_i,π_j互不重叠,$i\neq j,i,j=1,2,\ldots,n$;② π_i必须装入在矩形容器C内,即矩形在装入过程中不能超出容器C的宽度W;③ π_i的边必须与矩形容器C的边平行,π_i可以 90°旋转.

2DR-SPP是一个NP完全问题,其计算复杂度随着问题规模的增大呈指数增长.针对该类问题,国内外相关学者作了大量的研究.Hifi[1]给出了一种基于分支定界方法(branch-and-bound procedure)的精确算法,适用于解决中小规模的 2D装箱问题;Lesh等人[2]针对 2D矩形完美装箱问题,提出了基于分支定界的穷举搜索算法,该算法已证明对于低于 30 个矩形的装箱是有效的.

为了解决大规模的矩形装箱问题,一些启发式算法也相继提出来.Zhang等人[3]提出了一种启发式递归算法HR(heuristic recursive algorithm),该方法基于启发式策略和递归结构,实验结果表明,该算法能够在较短时间内获得较为理想的装箱高度,但其平均运行时间却达到了$O(n^3)$.陈端兵等人[4]根据先占角后占边的原则,提出了一种针对矩形装箱的贪心算法.Chen等人[5]给出了一种two-level搜索算法来求解 2DR-SPP.Cui[6]给出了一种启发式递归分支定界算法HRBB(heuristic recursive branch-and-bound algorithm),该算法将递归结构和分支定界技术结合在一起来求解 2DR-SPP,取得了较好的实验结果.Beltrán等人[7]按照随机搜索原则设计的GRASP(greedy randomized adaptive search procedure)算法与VNS(variable neighbourhood search)结合在一起来求解 2DR-SPP.而Alvarez-Valdes等人[8]提出了Reactive GRASP算法,该算法分为构建和改进两个阶段,其测试结果优于文献[7].张德富等人[9]则提出了一种有效的砌墙式启发式算法PH,实验结果表明,该算法对规模较大的 2DR-SPP能够获得较好的装箱效果.

针对 2DR-SPP,目前广泛采用的是遗传算法GA(genetic algorithm)、模拟退火算法SAA(simulated annealing algorithm)等搜索算法与BL(bottom-left)、IBL(improved BL)、BLF(bottom-left-fill)等启发式布局算法相结合的方式.Bortfeldt[10]采用了无任何编码的遗传算法来解决矩形装箱问题.Jackobs[11]提出了一种混合算法,通过GA与BL启发式布局算法相结合,从而将矩形的装箱问题转换成相对简单的装入序列问题.由于BL算法容易出现即使穷举所有情况仍不能得到最优解的现象,因此,Liu[12]提出了一种IBL算法,并与GA相结合,取得了优于文献[11]的装箱效果.Yeung等人[13]针对布料切割问题提出了一种布局算法LFLA,该布局算法的计算复杂度为$O(n)$,相对于BL算法(其复杂度为$O(n^2)$),效率上有了较大的提高.Zhang等人[14]针对矩形装箱问题给出了meta-heuristic算法,该算法主要基于启发式递归策略和SAA.Dereli等人[15]则采用SAA与递归布局方法相结合来解决 2DR-SPP.针对不同数量及不同类型的矩形排样,Hopper[16]分别使用了BL,BLF启发式布局算法和GA,NE(naïve evolution),SA,HC(hill-climbing),RS(random search)启发式搜索算法相结合的方式进行求解.

迄今为止,虽然对 2DR-SPP进行了大量的研究,但从相关文献给出的测试结果来看,该问题仍然有进一步研究的必要.例如,BL,IBL,BLF,LFLA以及贾志欣等人[17]提出的最低水平线LHL(lowest-horizontal-line)等常见的启发式布局算法,它们在对矩形装入的过程中严格按照矩形的某个排列序列进行,容易产生较大的空洞,浪费空间,因此装箱效果不够理想.相对于BL而言,BLF算法可以取得较好的装箱效果,但其算法的时间复杂度却达到了$O(n^3)$[18],不适宜解决规模较大的 2DR-SPP.

本文针对上述问题设计出一种启发式布局算法,即底部左齐择优匹配算法(lowest-level left align best fit,简称 LLABF).本文第 1 节重点介绍 LLABF 算法设计.第 2 节介绍 GA+LLABF 算法的实现.第 3 节主要对标准数据集进行模拟测试,并与相关文献结果进行对比分析.第 4 节给出结论.

1 LLABF 算法设计

1.1 定 义

设容器宽度所在方向为横坐标 X,长度方向为纵坐标 Y,容器的左下角为坐标原点(0,0).容器的底部在 $y=0$处,容器可以在 Y 轴正方向无限延伸.

队列$I=\{\pi_1,\pi_2,\ldots,\pi_n\}$表示$n$个矩形的某个装入序列,其中,$\pi_i$是矩形编号,$\pi_i\in[1,n]$,对任意$i,j\in[1,n]$,当$i\neq j$时,

$\pi_i \neq \pi_j$.设$I(i)$表示队列I中第i个位置上对应的矩形,它由五元组构成,即$I(i)=\{x,y,w,h,\theta\}$,其中,$I(i).x$,$I(i).y$分别表示该矩形装入后,其左下角的横、纵坐标;$I(i).w$,$I(i).h$,$I(i).\theta$分别表示该矩形的宽度、高度和旋转角度.

设队列$E=\{e_1,e_2,\ldots,e_m\}$表示装入过程中产生的轮廓线集合,元素e_k为水平线线段(与X坐标轴平行),它由三元组构成,即$e_k=\{x,y,w\}$,其中,$e_k.x$,$e_k.y$表示第k个水平线线段的左端点坐标(起点坐标);$e_k.w$表示第k个水平线线段的宽度;并且对任意$0<k<m$,有$e_k.x<e_{k+1}.x$,即按轮廓线起点的x坐标从小到大排列.队列E具有以下特征:其y坐标具备唯一性,如果相邻线段具有相同的高度y,则进行合并;所有线段在X坐标上的投影不重叠;所有线段的宽度之和刚好等于矩形容器宽度W.

最低水平线定义为队列 E 中其 y 坐标最小的水平线.

最优高度H_{opt}表示穷举所有可能情况得到的最小装箱高度,也称为最优解.

装箱高度$H_{packing}$表示根据当前装入序列,按照布局算法将所有矩形装入矩形容器C内所得到的最小高度.

空洞是指在装箱过程中,由矩形或者矩形与容器边界(如$x=0$,$y=0$,$x=W$或者$y=H_{packing}$)围成的空白区.空洞越多,材料浪费的程度就越严重.

1.2 LLABF启发规则设计

1.2.1 基本思想

一般情况下,对 NP 难问题很难直接构造出一个最优解或满意解,所以只能通过搜索的方法在整个解空间中寻找最优解或者满意解.当问题规模较大时,这种盲目搜索或者遍历就变得十分困难,甚至根本不可行.因此,相关学者常利用问题解本身的某些结构特征来构造出一些启发式规则,并按照该规则设计出启发式算法,由该算法求得问题的一个满意解.

对于矩形条带装箱问题,它有以下几个可以利用的结构特征:

(1) 面积较大的矩形装入后产生的空洞较大,面积较小的矩形装入后产生的空洞较小;

(2) 面积较大的矩形装入后产生的空洞常常可以装入面积较小的矩形;

(3) 装入过程中产生的轮廓线越规整,即组成轮廓线的水平线数量越少,就越有利于后期矩形的装入.

本文中,LLABF 启发式算法利用了矩形条带装箱问题的 3 个结构特征,采用了动态选择方法.该方法遵照最佳匹配优先原则来动态选择下一个待装矩形,使得在矩形装入的每一步都尽可能地获得一个较优的装箱结果.重复执行,直至最终获得整体较优解.

最佳匹配优先原则(best-fit priority,简称 BFP)即在矩形的装入过程中,优先考虑与当前可装入区域宽度或高度相匹配以及可装入的未排矩形,它包括完全匹配优先、宽度匹配优先、高度匹配优先、组合宽度匹配优先以及可装入优先这 5 种启发式规则.

5 种启发式规则中完全匹配优先、宽度匹配优先以及组合宽度匹配优先可以减少空洞的产生,特别是在装入初期,由于用来与可装区域进行比较的矩形较多,因此,其匹配的概率越高,就越不容易产生空洞;完全匹配优先以及高度匹配优先使得装箱轮廓相对规整;高度匹配优先以及可装入优先可以将较小矩形延后装入.5 种启发式规则结合起来不但可以使矩形装入后产生的空洞数量较少,而且所有的空洞总面积相对也较小.

1.2.2 启发规则设计

完全匹配优先(full-fit first,简称FFF). 在可装入的轮廓线中选取最低的水平线e_k,如果有多个线段,则优先选取最左边的一段.从待装矩形中按照装入序列依次将矩形与e_k进行比较,如果存在宽度或者高度与该线段宽度$e_k.w$相等且装入后刚好左填平或者右填平的矩形则优先装入.完全匹配优先能够减少装入后产生的轮廓线数量,使得装入轮廓朝着顶部平齐的方向发展.

宽度匹配优先(width-fit first,简称WFF). 在装入过程中,优先装入宽度或者高度与最低水平线e_k等宽的矩形,如果存在多个匹配矩形,则优先装入面积最大的.与完全匹配优先规则不同的是,宽度匹配优先并不要求装入后能够实现左填平或者右填平;同时,该规则使得较小矩形有推迟装入的趋势.另外,WFF不会增加装入轮廓线数量.

高度匹配优先(height-fit first,简称HFF). 在待装矩形中,按照装入序列查询宽度或高度不大于最低水平线

e_k宽度且装入后能够实现左填平的矩形,若存在则装入查询到的首个矩形.与FFF和WFF不同,HFF可能会在最低水平线上产生新的、更小的可装入区域,但却增加了轮廓线e_{k-1}的宽度.

组合宽度匹配优先(joint-width-fit first,简称JWFF). 按装入序列对两个矩形进行组合,如果组合后的宽度与最低水平线宽度e_k相等,则优先装入组合序列中的首个矩形.例如,存在两种组合$I(i1).w+I(j1).w=e_k.w$, $I(i2).w+I(j2).w=e_k.w$,如果$I(i1)$的面积大于$I(i2)$,则首先装入$I(i1)$,否则装入$I(i2)$.

JWFF,FFF与WFF规则避免了在最低水平线e_k上产生新的、更小的可装入区域,从而减少了整个装箱过程产生空洞的可能性.为了保证时效性,算法中设置了查询范围*searchNum*,即从当前位置开始,最多可以进行*searchNum*×*searchNum*次连续矩形的两两组合查询.

可装入优先(placeable first,简称PF). 在一定范围内,从待装矩形件中按照装入序列依次查找宽度或高度不大于最低水平线e_k宽度的矩形,若存在,则将其装入;若存在多个,则装入面积最大的矩形.PF可能在最低水平线上产生新的、更小的可装入区域,同时使得较小矩形延迟装入.

在矩形的装入过程中,满足 FFF,WFF,HFF 以及 JWFF 规则的矩形可能并不存在,此时就必须考虑 PF.如果满足 PF 规则的矩形仍不存在,那么必然会在最低水平线上产生空洞.与 BL,IBL,LFLA 以及 LHL 算法不同,该空洞是在一定范围内由于不存在可装入的矩形而产生的,因此其面积更小.与 BLF 算法不同,LLABF 算法并不需要保存新产生的空洞.

图 1 给出了按照最佳匹配优先原则的部分启发规则实现矩形装入的示意图,其中,矩形的装入序列是 4 2 7 6 5 1 8 3.图 1(a)表示已经装入了 4 个矩形,产生了由 4 个线段e_1,e_2,e_3,e_4组成的装入轮廓,其中,e_2为最低水平线,其对应的区域是接下来首先要考虑的区域.图 1(b)采用完全匹配优先:与最低水平线e_2等宽的矩形有 1 号和 3 号,但由于只有 3 号矩形装入后能够实现左填平,因此首先选择该矩形装入.图 1(c)采用宽度匹配优先:满足该条件的矩形有 1 号和 3 号,但由于 1 号矩形面积较大,所以优先装入.图 1(d)采用高度优先:1,8,3 这 3 个矩形都满足其宽度不大于$e_2.w$,但只有 8 号和 3 号矩形装入后能够实现左填平,且 8 号矩形位置在 3 号之前,所以优先装入.

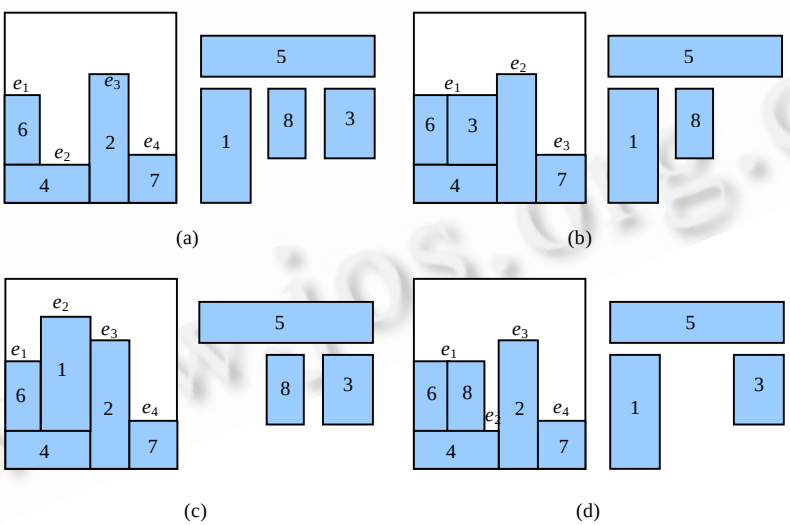


Fig.1 Packing result according to BFP principle

图 1 按 BFP 原则的装箱结果图

1.3 LLABF算法的实现

LLABF 算法是按照 BFP 原则进行装箱的启发式布局算法,其算法实现如下所示.

LLABF (input: I; output: $H_{packing}$)

```
begin
  InitContours(E); InitCoordinate(I); H_packing=0;        // 初始化;H_packing:装箱高度
  ①  for (ii=1; ii≤n; ii++)
  begin
    ②  if (I(ii).x>−1) goto ①;                           // 当前矩形件已被装入
    在E中取最低水平线e_k;
    j=−1;                                                 // 记录用于装入的矩形件所在序列号
    FullFitFirst(ii,I,e_k,j); if (j>−1) goto ④;           // 完全匹配FFF
    WidthFitFirst(ii,I,e_k,j); if (j>−1) goto ④;          // 宽度匹配WFF
    HeightFitFirst(ii,I,e_k,j); if (j>−1) goto ④;         // 高度匹配HFF
    JointWidthFitFirst(ii,I,e_k,7,j); if (j>−1) goto ④;   // 组合宽度匹配JWFF
    PlaceableFirst(ii,I,e_k,n/6,j); if (j>−1) goto ④;     // 可装入优先PF
    ③  合并 E 中的边,转②;
    ④  将I集合中第j位元素对应的矩形件I(j)装入在E中的最低水平线e_k上;如果未旋转,则
       H_packing=max(H_packing,I(j).x+I(j).h),否则,H_packing=max(H_packing,I(j).x+I(j).w);更新集合I和E;如果j>ii,
       则转②;
  end
end
```

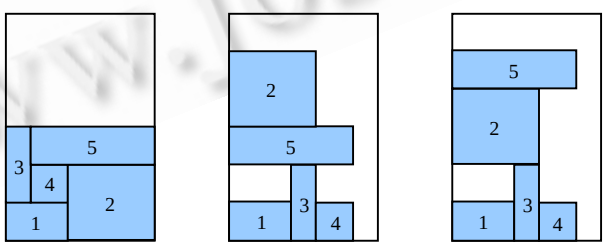
其中,*InitCoordinate*(*I*)是对所有矩形块的坐标 *x*,*y* 进行初始化.在初始阶段,由于所有的矩形都未进行装入,*InitCoordinate*(*I*)将其坐标 *x*,*y* 全部初始化为−1,即对任意 $i\in[1,n]$,$I(i).x=I(i).y=-1$.

InitContours(*E*)对装箱轮廓线进行初始化.在初始阶段,由于容器*C*未被装入,因此,其轮廓线仅由一个水平线线段构成,即$E=\{e_1\}$,其中,$e_1=\{0,0,W\}$,*W*为容器*C*的宽度.

FullFitFirst,*WidthFitFirst*,*HeightFitFirst*,*JointWidthFitFirst*,*PlaceableFirst* 分别表示按照 FFF,WFF,HFF,JWFF以及PF等启发式规则在队列*I*中第*ii*位开始向后依次查找与最低水平线e_k相匹配矩形的查询过程,其返回值为*j*.其中,*JointWidthFitFirst*,*PlaceableFirst*中的查询范围分别设置为 7,*n*/6,为多次实验后取得的经验值.

1.4 LLABF算法举例

图 2(a)是 5 个矩形构成的一个完美装箱图,其装箱高度$H_{packing}$等于最优高度H_{opt},空间的利用率为 100%.对于BL,IBL,LFLA以及LHL算法,要实现图 2(a)的完美装箱(不考虑旋转),其装入序列就必须为 1 2 3 4 5.对于序列为 1 3 4 5 2 和 1 3 2 4 5,其装箱结果分别如图 2 (b)和图 2(c)所示,这两种装入序列都会产生较大的空洞.



(a) (b) (c)

Fig.2 Packing result according to BL, IBL, LFLA or LHL algorithm

图 2 BL,IBL,LFLA 或 LHL 算法的装箱结果图

对于同样的装入序列,采用 LLABF 算法则能得到如图 2(a)所示的完美装箱.图 3 显示了装入序列为 1 3 4 5 2、采用 LLABF 算法得到的装箱结果(其中,*表示该位置上的矩形已经装入容器中).

初始化:$E=\{e_1\}$,$e_1=\{0,0,W\}$;对任意$i\in[1,n]$,$I(i).x=I(i).y=-1$;

开始阶段虽然所有矩形都不满足完全匹配FFF、宽度匹配WFF以及高度匹配,但$I(1).w+I(5).w=e_1.w$,满足两两组合宽度匹配JWFF,所以将编号为 1 的矩形装入在矩形容器C的左下角;装入结果如图 3(c)所示,此时,$E=\{e_1,e_2\}$;

取最低水平线e_2,按照LLABF启发式规则依次查询匹配矩形,其中$I(5)$,即编号为 2 的矩形满足WFF,故将其装入在e_2上;装入结果如图 3(d)所示,此时,$E=\{e_1,e_2\}$;

以此类推,其装入过程如图 3(e)~图 3(g)所示.

根据 LLABF 算法,如果装入序列符合 1 2 3 * *或者 1 3 * * * 模式(其中,*代表未列出的矩形编号),其结果都与 1 2 3 4 5 序列的装箱结果相同,即都能得到如图 3(a)所示的完美装箱.在图 3 中,满足第 1 种模式的装入序列包括 1 2 3 4 5,1 2 3 5 4 两种,满足第 2 种模式的装入序列包括 1 3 2 4 5,1 3 4 2 5,1 3 2 5 4,1 3 5 2 4,1 3 5 4 2,1 3 4 5 2 这 6 种.针对图 2 中的 5 个矩形的装箱问题,LLABF 找到最优解的概率是 BL,IBL,LFLA 以及 LHL 的 8 倍之多.因此,相对于传统布局算法,LLABF 算法更容易找到最优解.

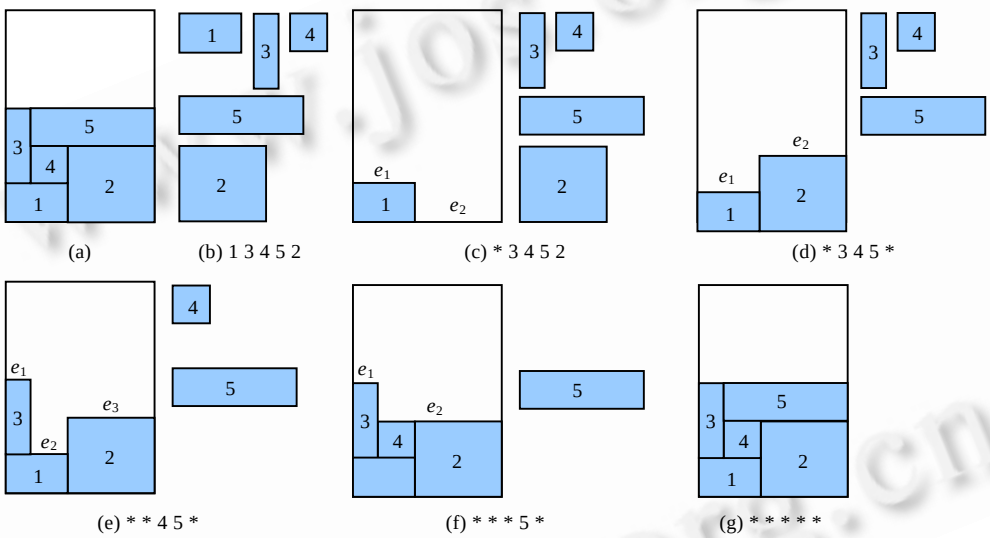


Fig.3 Packing result according to LLABF algorithm

图 3 LLABF 算法的装箱结果图

2 GA+LLABF 算法实现

2.1 算法的总体流程

本文采用遗传算法 GA 与 LLABF 相结合的方式来解决二维矩形条带装箱问题.其中,GA 用来实现矩形装入序列的初步确定,LLABF 则根据自身的启发式规则将矩形装入到矩形容器中.

GA+LLABF 的算法流程如下所示.

(1) 个体染色体编码.

(2) k=0;随机产生m个个体组成初始群体$pop(k)=\{I_1,I_2,\ldots,I_m\}$.

(3) 对单个个体I_i执行LLABF(Ii,$H_{packing}$);记录个体适应度$f(I_i)$($f(I_i)=1/H_{packing}$).

(4) 判断是否满足停止条件.若是,则停止计算,输出最佳结果;否则,继续.

(5) 利用排序选择操作选择 m 个个体形成新的群体 $selpop(k+1)$.

(6) 根据交叉概率P_c进行交叉操作,产生群体$crosspop(k+1)$.

(7) 根据变异概率P_m进行变异操作,产生群体$mutpop(k+1)$.

(8) $pop(k+1)=mutpop(k+1)$;$k=k+1$;转(3),循环.

2.2 编码及适应度函数

矩形装箱采用整数编码(encoding),即n个矩形分别用整数 1,2,…,n编号,矩形的一个装箱方案对应于一个染色体编码,$I_j=(\pi_1,\pi_2,\ldots,\pi_n)$,其中,$|\pi_i|$表示矩形编号($1\le|\pi_i|\le n$),当$\pi_i<0$ 时,矩形旋转 90°,否则不旋转;I_j表示第j个个体.

例如:个体I_j的染色体编码为(–5 3 –7 6 1 10 8 –2 4 –9),表示装入序列为 5 3 7 6 1 10 8 2 4 9.在装入过程中,编号为 5,7,2,9 的矩形进行 90°旋转.

适应度函数(fitness function)可以由装箱高度的倒数表示,即

$$f(I_j)=1/H_{packing}(I_j).$$

2.3 选 择

为了避免丢失上一代产生的最佳个体,本文在采用排序选择方法(rank-based select model)的同时,还混合使用了最优保存策略(elitist model).

排序选择方法主要依据个体适应度值之间的大小关系,对个体适应度是否取正值或负值以及个体适应度之间的数值差异程度并无特别要求.其步骤如下:

(1) 对群体中所有个体按照适应度从大到小排序.

(2) 根据下列公式计算每个个体的选择概率:

$$P_s^{(i)}=\alpha\times(1-\alpha/m)^I,$$

其中,$P_s^{(i)}$表示排列在第 i 个位置的个体的选择概率,m 表示群体大小,α为调节参数.

(3) 根据步骤(2)计算得出的个体概率值作为其能够被遗传到下一代的概率,基于该值应用比例选择(赌盘选择)的方法产生下一代群体.

2.4 交 叉(crossover)

交叉是产生新个体的主要方法.本文主要采用环形部分交叉(circular-based part crossover):① 随机生成起始交叉点$P_{cr}\in[1,n]$;② 随机生成交叉长度$L_{cr}\in[1,n]$;③ 将P_{cr}后的L_{cr}位基因进行交换,如果$P_{cr}+L_{cr}>n$,则将染色体前$P_{cr}+L_{cr}-n$位基因进行互换;④ 然后依次将未出现的基因填入空白基因座.

2.5 变 异(mutation)

变异是产生新个体的辅助方法,它决定了遗传算法的局部搜索能力,保持群体的多样性.本文中的变异包括两部分:一是旋转变异,二是装入序列变异.

(1) 旋转变异

旋转变异采用了单点取反和环形部分取反两种方式.

单点取反(single point reversion)变异:① 随机生成变异点$P_{mu}\in[1,n]$;② 取反.例如,对于个体$I_j=(\pi_1,\pi_2,\ldots,\pi_i,\ldots,\pi_n)$,$P_{mu}=i$,则变异后产生的新个体为 $I'_j=(\pi_1,\pi_2,\ldots,-\pi_i,\ldots,\pi_n)$.

环形部分取反(circular-based part reversion)变异:① 随机生成变异点$P_{mu}\in[1,n]$;② 随机生成变异长度$L_{mu}\in[1,n]$;③ 将P_{mu}后的L_{mu}位取反,如果$P_{mu}+L_{mu}>n$,则将染色体首部的$P_{mu}+L_{mu}-n$位基因取反.

(2) 装入序列变异

装入序列变异采用了两点位置互换与环形部分逆转两种方式.

两点位置互换(two point exchange)变异:① 随机生成两个变异点$i,j\in[1,n]$;② 将两个基因互换.例如:$I_j=(\pi_1,\pi_2,\ldots,\pi_{i1},\ldots,\pi_{i2},\ldots,\pi_n)$,$i1,i2\in[1,n]$,则变异后产生的新个体为 $I'_j=(\pi_1,\pi_2,\ldots,\pi_{i2},\ldots,\pi_{i1},\ldots,\pi_n)$.

环形部分逆转(circular-based part inversion)变异:① 随机生成变异点$P_{mu}\in[1,n]$;② 随机生成变异长度$L_{mu}\in[1,n]$;③ 将P_{mu}后的L_{mu}位逆转,即将$(P_{mu}+i)\bmod n$(即$P_{mu}+i$对n取模)与$(P_{mu}+L_{mu}-i)\bmod n$互换,$i\in[0,L_{mu}/2]$.

3 实验结果

实验平台:Pentium 4 3.0GHz,512M,Windows Server 2003.

运行参数:交叉概率P_c=0.95,变异概率P_m=0.85.

本文的实验针对两组标准测试问题来进行,第 1 组来自文献[11],第 2 组来自文献[16].

3.1 实验1

文献[11]给出了两组标准测试实例,均由 40×15 的大矩形切割而成.其中,第 1 组实例由 25 个矩形组成,第 2 组实例由 50 个矩形组成.本文实验中,GA 的群体大小和运行代数参照文献[12],分别设为 20 和 100.对每组实例皆重复运行 100 次.实验结果对比见表 1.

Table 1 Computational results of four algorithms

表 1 4 种算法的计算结果

$W\times H$	n	Jakobs S[11] GA+BL		Liu DQ[12] GA+IBL		Bortfeldt A[10] SPGAL		GA+LLABF		
		$H_{\min}$	H_{avg}	$H_{\min}$	H_{avg}	$H_{\min}$	H_{avg}	$H_{\min}$	H_{avg}	$H_{\max}$
40×15	25	17	17.48	16	16.97	16	16.00	15	15.00	15
40×15	50	17	17.28	16	17.01	15	15.00	15	15.00	15

表 1 中,$H_{\min}$,$H_{\max}$,H_{avg}分别指多次运行中得到的最小$H_{packing}$、最大$H_{packing}$及平均$H_{packing}$.

由表 1 可以看出,GA+LLABF算法在对 2 组标准实例进行的 100 次测试中均找到了最优解 15,而其他 3 种算法中,除了SPGAL找到了第 2 组数据集的最优解之外,其余算法求得的$H_{packing}$均大于H_{opt}.因此,实验结果表明,本文的GA+LLABF算法更加有效.

3.2 实验2

在文献[16]中,Hopper等人给出了 21 组不同尺寸的测试实例,这些实例分成 7 个大类,每个大类由 3 组实例组成,其矩形个数从 16 个~197 个不等.每个实例的最优高度H_{opt}已经给出,其相关数据见文献[16].

在本文的算法中,初始种群大小参照文献[16],设为 50,终止条件为进化代数等于 2 500 代,或者装箱高度H_{packin}等于最优解H_{opt}.对每组测试实例重复运行 10 次,记录每个大类 30 次(每个大类 3 个实例,每个实例进行 10 次测试)测试的平均装箱高度到最优解的相对距离$RDBSOH$,$RDBSOH(\%)=100\times(30\text{ 次平均}H_{packing}-H_{opt})/H_{opt}$.实验结果对比见表 2.

Table 2 Computational results of 12 algorithms (relative distance of best solution to optimum height (%))

表 2 12 种算法的计算结果(RDBSOH(%))

Algorithm	$C1$	$C2$	$C3$	$C4$	$C5$	$C6$	$C7$	Average
GA+BLF[16]	4	7	5	3	4	4	5	4.57
SA+BLF[16]	4	6	5	3	3	3	4	4.00
GRASP+VNS[7]	14.40	17.33	12.93	6.80	4.51	3.55	2.89	8.92
HR[3]	8.33	4.45	6.67	2.22	1.85	2.5	1.8	3.97
Hybrid SA[15]	1.66	4.88	4.00	3.94	3.18	3.30	3.38	3.48
SA+HR[14]	5.00	4.47	2.23	2.22	1.86	2.5	3.24	3.07
PH[9]	5.00	4.44	4.44	3.33	1.11	1.11	1.25	2.95
HRBB[6]	1.70	0.00	1.10	2.20	1.90	1.40	1.30	1.40
GRASP[8]	0.00	0.00	1.08	1.64	1.10	1.56	1.36	1.33
SPGAL[10]	1.70	0.90	2.20	1.40	0.00	0.70	0.50	1.00
A_1[5]	0.00	0.00	1.11	1.67	1.11	0.83	0.42	0.73
GA+LLABF	0.00	0.00	0.00	0.00	0.00	0.00	0.40	0.06

从表 2 可以看出,在现有的算法中,按照Average即平均RDBSOH,位居前 4 位的较优算法依次为A_1,SPGAL,GRASP与HRBB,它们对 7 大类实例计算得出的Average分别为 0.73,1.0,1.33 和 1.4.而本文给出的GA+LLABF算法计算得出的Average为 0.06,其中 6 大类实例的RDBSOH为 0,即对应的所有实例其 10 次实验结果皆取得了最优解.因此,实验结果进一步表明,针对 2DR-SPP,本文提出的GA+LLABF算法优于其他算法.

表 3 给出了GRASP+VNS,GRASP,HRBB以及本文的GA+LLABF对 21 组实例求得的实际测试结果$H_{packing}$.在GA+LLABF算法对 21 组实例的测试中,10 次实验皆能获得最优解H_{opt}的有 18 组(21 组中,除*C*71,*C*72,*C*73 外),占全部实例的 86%(18/21),而GRASP为 38%(8/21)、HRBB为 33%(7/21);10 次实验中至少获得 1 次最优解的实例有 19 组(即最小高度$H_{\min}$=最优高度H_{opt}),占全部实例的 91%(19/21),而GRASP为 38%(8/21)、HRBB为 43%(9/21);10 次实验获得的最大高度$H_{\max}$与最优解H_{opt}之间的距离最多为 1 个基本单位.因此,无论是最优解所占比例还是最小高度$H_{\min}$、平均高度H_{avg}以及最大高度$H_{\max}$,GA+LLABF求得的结果均具有明显的优势.

Table 3 Computational results ($H_{packing}$) of 21 instances

表 3 21 组实例计算结果

Category	H_{opt}	n	Beltrán DJ GRASP+VNS[7] (20 runs)			Alvarez-Valdes R GRASP[8] (10 runs)		Cui YD HRBB[6] (6 runs)			GA+LLABF (20 runs)		
			$H_{\min}$	H_{avg}	$H_{\max}$	$H_{\min}$	H_{avg}	$H_{\min}$	H_{avg}	$H_{\max}$	$H_{\min}$	H_{avg}	$H_{\max}$
*C*11	20	16	21	22.30	23	20	20.00	20	20.00	20	20	20.00	20
*C*12	20	17	21	23.20	25	20	20.00	21	21.00	21	20	20.00	20
*C*13	20	16	22	23.15	24	20	20.00	20	20.00	20	20	20.00	20
*C*21	15	25	16	17.20	19	15	15.00	15	15.00	15	15	15.00	15
*C*22	15	25	16	18.00	20	15	15.00	15	15.00	15	15	15.00	15
*C*23	15	25	16	17.60	20	15	15.00	15	15.00	15	15	15.00	15
*C*31	30	28	32	32.55	34	30	30.00	30	30.00	30	30	30.00	30
*C*32	30	29	32	34.25	37	31	31.00	31	31.00	31	30	30.00	30
*C*33	30	28	33	34.85	37	30	30.00	30	30.00	30	30	30.00	30
*C*41	60	49	63	63.55	66	61	61.00	60	60.50	61	60	60.00	60
*C*42	60	49	61	64.60	68	61	61.00	61	61.33	62	60	60.00	60
*C*43	60	49	62	64.10	66	61	61.00	61	61.00	61	60	60.00	60
*C*51	90	73	92	93.45	95	91	91.00	90	90.67	91	90	90.00	90
*C*52	90	73	93	94.75	97	91	91.00	92	92.00	92	90	90.00	90
*C*53	90	73	92	94.00	98	91	91.00	91	91.17	92	90	90.00	90
*C*61	120	97	123	124.40	130	121	121.90	121	121.00	121	120	120.00	120
*C*62	120	97	123	124.65	128	121	121.90	121	121.83	122	120	120.00	120
*C*63	120	97	122	123.75	126	121	121.90	121	121.33	122	120	120.00	120
*C*71	240	196	244	246.45	248	244	244.00	242	242.17	243	241	241.00	241
*C*72	240	197	244	247.05	255	242	242.90	245	245.00	245	241	241.00	241
*C*73	240	196	245	247.30	258	243	243.00	241	241.33	242	240	240.90	241

图 4($H_{packing}$=H_{opt}=120)、图 5($H_{packing}$=H_{opt}=240)显示了由GA+LLABF算法求得的部分实例装箱图,其装箱高度皆为最优高度.

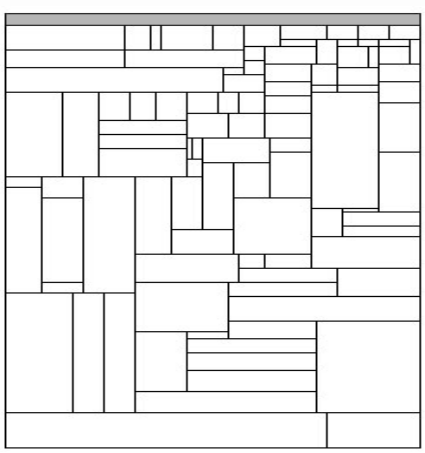


Fig.4 Packing result of *C*63

图 4 *C*63 的装箱结果图

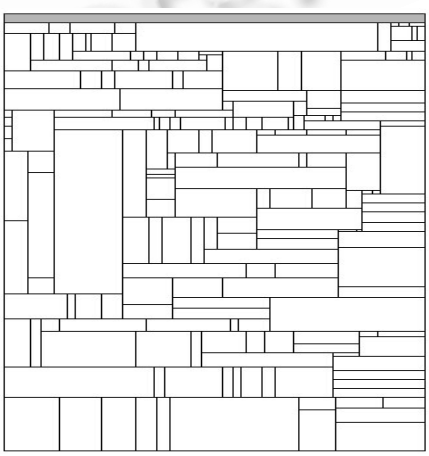


Fig.5 Packing result of *C*73

图 5 *C*73 的装箱结果图

LLABF算法在对矩形的装入过程中,每一步均按照FFF,WFF,HFF,JWFF以及PF规则依次查询与最低水平

线相匹配的矩形.对于FFF,如果某个阶段存在i个待装矩形,那么最坏情况下将比较i次;其余4个启发式规则除了JWFF的比较次数为常数K以外,皆与完全匹配优先类似,因此,单步比较次数最坏情况下为 $4i+K$.将所有n个矩形全部装入容器中,总的比较次数最坏情况下为 $4\times(n+n-1+n-2+\ldots+2+1)+n\times K$,共 $2n\times(n+1)+n\times K$,因此其复杂度为$O(n^2)$.对于矩形在容器中的装入,主要的比较次数在于查找最低水平线,而最低水平线的数量不会超过n,所以,其复杂度最坏情况下为$O(n^2)$.整体上,LLABF的时间复杂度为$O(n^2)$.同时,对于GA,由于群体规模和进化代数皆为常数,其本身的比较次数也为常数,所以GA+LLABF算法的复杂度主要由LLABF决定,即最坏情况下为$O(n^2)$.

表 4 给出了表 2 中所有算法的计算时间.由于算法的实现方式、测试平台与环境的不同,要对算法的快慢进行精确比较是不科学也是不可行的.但是从表 4 中,我们能够大致看出各种算法的运行快慢程度.由于本文算法采用了装箱高度等于最优解作为终止运行条件之一,因此,虽然 $C5$ 的规模大于 $C4$,但其获得最优解的速度明显快于 $C4$,所以运行时间相对较小.

Table 4 Run times of 12 algorithms (s)

表 4 12 种算法的运行时间(s)

Algorithm	*C*1	*C*2	*C*3	*C*4	*C*5	*C*6	*C*7	Average
[a]GA+BLF[16]	60.00	120.00	180.00	780.00	2 160.00	5 160.00	46 620.00	7 869.57
[a]SA+BLF[16]	42.00	144.00	240.00	1 980.00	6 900.00	22 920.00	250 860.00	40 441.86
[b]GRASP+VNS[7]	0.00	0.00	0.00	0.00	0.02	0.07	1.37	0.21
[c]HR[3]	0.00	0.00	0.03	0.14	0.69	2.21	36.07	5.59
dHybrid SA[15]	3.64	6.36	18.78	101.20	287.27	757.20	1650.6	403.57
[e]SA+HR[14]	23.20	42.80	59.20	170.00	343.60	444.90	1328.80	344.64
[f]PH[9]	0.00	0.00	0.00	1.51	5.69	23.74	707.12	105.45
[g]HRBB[6]	0.27	0.33	0.88	2.19	2.83	2.24	4.26	1.86
[h]GRASP[8]	–	–	–	–	–	–	–	60.00
[i]SPGAL[10]	–	–	–	–	–	–	–	139.00
[j]A_1[5]	0.37	0.61	1.71	0.15	0.40	3.98	45.02	7.46
GA+ LLABF	0.00	0.03	2.23	3.6	1.3	33.2	328.3	52.67

[a]—Pentium Pro 200MHz and 65MB memory [b]—Pentium 166MHz and 96MB RAM
[c]—Dell GX260 with a 2.4GHz CPU [d]—Pentium Ⅲ 797MHZ
[e]—Dell GX270 with a 3.0GHz CPU [f]—Pentium 4 1.6GHZ, 256MB memory
[g]—Pentium 4 2.8GHZ, 512MB memory [h]—Pentium 4 mobile 2.0GHZ, 512MB memory
[i]—Pentium PC with 2GHz [j]—2.4GHz PC with 512MB memory

4 结论与展望

本文分析了传统布局算法的不足,提出了一种启发式布局算法 LLABF,并将其与 GA 相结合.实验结果表明,GA+LLABF 算法解决 2DR-SPP 问题更加有效.但是要评价一种算法的优劣,除了要考虑算法的求解结果以外,还应兼顾算法的运行效率.因此,改进算法以进一步提高运算速度,将是我们下一步的研究重点.

References:

[1] Hifi M. Exact algorithms for the guillotine strip cutting/packing problem. Computers & Operations Research, 1998,25(11): 925−940.

[2] Lesh N, Marks J, McMahon A, Mitzenmacher M. Exhaustive approaches to 2D rectanglar perfect packings. Information Processing Letters, 2004,90:7−14.

[3] Zhang DF, Kang Y, Deng AS. A new heuristic recursive algorithm for the strip rectangular packing problem. Computers & Operations Research, 2006,33(8):2209−2217.

[4] Chen DB, Huang WQ. Greedy algorithm for recatngle-packing problem. Computer Engineering, 2007,33(4):160−162 (in Chinese with English abstract).

[5] Chen M, Huang WQ. A two-level search algorithm for 2D rectangular packing problem. Computers & Industrial Engineering, 2007, 53(1):123−136.

[6] Cui YD, Yang YL, Cheng X, Song PH. A recursive branch-and-bound algorithm for the rectangular guillotine strip packing problem. Computers & Operations Research, 2008,35(4):1281−1291.

[7] Beltrán JD, Calderón JE, Cabrera RJ, Cabrera RJ. GRASP-VNS hybrid for the strip packing problem. In: Proc. of the 1st in Workshop on Hybrid Metaheuristics. 2004. 79−90.

[8] Alvarez-Valdes R, Parreno F, Tamarit JM. Reactive GRASP for the strip-packing problem. Computers & Operations Research, 2008,35(4):1065−1083.

[9] Zhang DF, Han SH, Ye WG. A bricklaying heuristic algorithm for the orthogonal rectangular packing problem. Chinese Journal of Computers, 2008,31(3):509−515 (in Chinese with English abstract).

[10] Bortfeldt A. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. European Journal of Operational Research, 2006,172(3):814−837.

[11] Jakobs S. On the genetic algorithms for the packing of polygons. European Journal of Operational Research, 1996,88(1):165−181.

[12] Liu DQ, Teng HF. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. European Journal of Operational Research, 1999,112(2):413−420.

[13] Yeung LHW, Tang WKS. A hybrid genetic approach for garment cutting in the clothing industry. IEEE Trans. on Industrial Electronics, 2003,50(3):449−455.

[14] Zhang DF, Liu YJ, Chen SD, Xie XG. A meta-heuristic algorithm for the strip rectangular packing problem. Lecture Notes In Computer Science, 2005:1235−1241.

[15] Dereli T, Das GS. A hybrid simulated-annealing algorithm for two-dimensional strip packing problem. In: Proc. of the ICANNGA. Berlin: Springer-Verlag, 2007. 508−516.

[16] Hopper E, Turton BCH. An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem. European Journal of Operational Research, 2001,128(1):34−57.

[17] Jia ZX, Yin GF, Luo Y. Two-Dimensional irregular parts packing with genetic algorithm. Journal of Computer-Aided Design & Computer Graphics, 2002,14(5):467−470 (in Chinese with English abstract).

[18] Chazelle B. The bottom-left bin packing heuristic: An efficient implementation. IEEE Trans. on Computers, 1983,C32(8):697−707.

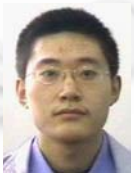
附中文参考文献:

[4] 陈端兵,黄文奇.求解矩形 Packing 问题的贪心算法.计算机工程,2007,33(4):160−162.

[9] 张德富,韩水华,叶卫国.求解矩形 Packing 问题的砌墙式启发式算法.计算机学报,2008,31(3):509−515.

[17] 贾志欣,殷国富,罗阳.二维不规则零件排样问题的遗传算法求解.计算机辅助设计与图形学学报,2002,14(5):467−470.

蒋兴波(1971－),男,四川广安人,讲师,主要研究领域为文字与图形信息处理,智能算法.



刘成城(1983－),男,硕士,主要研究领域为文字与图形信息处理,智能算法.

吕肖庆(1967－),男,副研究员,CCF 高级会员,主要研究领域为出版自动化,人工智能与模式识别.