软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(10):2600−2611 [doi: 10.3724/SP.J.1001.2012.04187]

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

求解 2D 条带矩形 Packing 问题的迭代启发式算法*

彭碧涛[1,2+], 周永务[2]

[1](广东外语外贸大学 思科信息学院,广东 广州 510006)

[2](华南理工大学 工商管理学院,广东 广州 510641)

Recursive Heuristic Algorithm for the 2D Rectangular Strip Packing Problem

PENG Bi-Tao[1,2+], ZHOU Yong-Wu[2]

[1](Cisco School of Informatics, Guangdong University of Foreign Studies, Guangzhou 510006, China)

[2](School of Business Administration, South China University of Technology, Guangzhou 510641, China)

+ Corresponding author: E-mail: pengbt_job@163.com, http://www.gdufs.edu.cn

Peng BT, Zhou YW. Recursive heuristic algorithm for the 2D rectangular strip packing problem. *Journal of Software*, 2012,23(10):2600−2611 (in Chinese). http://www.jos.org.cn/1000-9825/4187.htm

Abstract: The paper presents a novel and effective heuristic algorithm for the two-dimensional rectangular strip packing problem. This algorithm is mainly based on the best-fit value and tree recursive search rules and selects the maximal fitness rectangle to the packing the space. The computational results on a large number of Benchmark problems have shown that this algorithm is more effective than the existing novel algorithm.

Key words: tree recursive; 2D packing problem; heuristic algorithm; fitness

摘 要: 为求解二维矩形条带装箱问题,提出了一种新颖而有效的启发式算法.算法主要包括矩形装载适应度的计算规则和树型迭代搜索规则,通过选择最高适应度的矩形来装载空间.对大量国际上公认的 Benchmark 问题实例的计算结果表明,相对于当前的很多著名算法,提出的算法更加有效.

关键词: 树型迭代;二维装箱问题;启发式算法;适应度

中图法分类号: TP301 **文献标识码**: A

2D Strip Packing 问题指的是将一组矩形装入一个宽度固定的矩形容器中,使其装载高度最小.该类问题在工业领域中存在大量的应用,如报刊排版、集成电路布局和货物装载等.目前,针对该类问题,国内外学者进行了深入的研究,并提出了各种各样的算法.这些算法可以分为 3 大类:精确算法、启发式构造算法和现代启发式算法.

由于 2D Strip Packing 问题是 NP-难的[1],求解该问题的精确算法并不多,如:Beasley[2]针对二维不要求一刀切的问题提出了一种基于树搜索的精确算法;针对二维完美装箱问题,Lesh 等人[3]基于分支定界,提出了一种穷举搜索算法;Maretl 等人[4]提出了该问题的一种枚举方法;Fekete 等人[5]根据它们的图论模型得到了另一种枚举方法,枚举的一个关键步骤是分支定界,同时,以对偶可行函数为基础,提出了一种为一维乃至高维装箱问题定

* 基金项目: 国家自然科学基金(70771034, 71131003); 国家教育部人文社科基金(12YJC630148); 广东省自然科学基金(S2011010005503); 广州市科技计划基金(7421159402737)

收稿时间: 2011-07-11; 修改时间: 2011-10-08; 定稿时间: 2012-01-20

界的一般性方法.精确算法对于小规模问题可以很好地求解,但是由于其计算复杂度太高,对于大规模问题并不适合.

启发式算法能够在合理的时间内获得较好的解,逐渐成为大规模问题的一种常用求解方法.在国内,如张德富等人[6]提出了一种基于砌墙的拟人 PH(packing heuristic)算法;黄文奇等人基于拟人和拟物提出了两种启发式算法:HA(heuristic algorithm)[7]和 HRP(heuristic rectangular packing)[8],在 HRP 中提出了角度和穴度的概念;在文献[8]中的角度思想的基础上,Chen 等人[9]提出了一种 A1 算法,算法分为两个层次:首先根据普通占角规则选择放置,然后根据全局适应度来评价选择最优放置,计算结果优于前面两种算法的结果.与此不同的是,基于迭代和分支定界方法,Cui 等人[10]提出了一种基于迭代的启发式算法 HRBB(heuristic recursive brand-and-bound).国外比较经典的算法是 Baker 等人[11]提出的经典的 BL(bottom left)算法、Chazelle[12]提出的 BLF(bottom left fill)算法和 Hopper[13]提出的 BLD(bottom-left decreasing)算法.3 种算法都是考虑将待放置的矩形放在最低最左的位置,后两者是对 BL 算法的改进;基于另外的思想,Alvarez-Valdes 等人[14]给出了基于贪婪随机搜索的 GRA SP(greedy randomized adaptive search procedure)算法来求解矩形条带装箱问题,Burke[15]提出一种 BF(best-fit)来求解该问题.两种算法的矩形被放置时取决于当前部分解的状态,选择最佳适应度的矩形来放置;Stephen 等人[16]在 PH 的基础上,结合矩形适应度,提出了一种基于层的 FH(fast heuristic)算法,同时修改 HRP 算法来求解矩形条装箱问题,从而求解条状矩形装箱问题,FH 算法可以快速且高效地得到结果.

现代启发式算法利用模拟退火、遗传算法、神经网络等智能算法结合基本启发式算法来求解 2D Strip Packing 问题.在国内,如蒋兴波等人[17]结合遗传算法提出了 GA+LLABF(genetic+lowest-level left align best fit)算法;Liu 等人[18]提出了 GA+IBL(improved bottom left)算法,二者都是在 BL 算法的基础上对其进行改进;基于最小浪费的思想,Zhang 等人[19]提出了 GA+IHR(improved heuristic recursive)算法.在国外,Jakobs[20]结合遗传算法和左底算法提出 GA+BL(bottom left)求解了该问题;Hopper 等人[21]将 BLF(bottom-left first)与 GA,SA 等智能算法相结合,提出了 GA+BLF,SA+BLF 等算法,二者也是基于 BL 算法所进行的优化;在文献[15]的基础上,基于 BF 的思想,Burke 等人[22]分别结合 TS,SA 和 GA 算法,实现并比较了 BF+TS,BF+SA 和 BF+GA 这 3 种算法的性能,取得了比文献[15]更好的结果,同时发现,BF+SA 可以取得最好的效果;Borfeldt[23]结合遗传算法给出了基于层的该问题的 SPGAL(strip packing genetic algorithm, based on layer)算法.

对于上述算法,大都取得了不错的结果.但是研究发现,已有算法在启发式求解时,大部分是基于 BL 算法,并对其进行了相应的改进.然而,BL 算法在选择矩形和放置位置时没有考虑放入矩形与放入后左右邻接空间的关系,只是将矩形按照某种序列排序,如高度递减来进行排列,然后选择待排序列中的能放得下的第 1 个放入最左底的空间.这样,排列很容易造成装载的空洞.还有部分算法虽然通过某种方式考虑了放入矩形与放入后左右邻接矩形的关系,选择最佳适应度的放入矩形,但是适应度计算规则比较简单.同时,选择矩形只是基于当前的最佳适应度来进行的选择,但是当前最佳适应度毕竟不是全局最佳适应度,一旦在某个阶段选择错误,后续的结果将连锁地出现偏差.针对这种问题,本文提出了一种求解 2D Strip Packing 问题的迭代启发式算法(recursive heuristic algorithm,简称 RHA).该算法定义了多种适应度计算规则,同时,基于迭代搜索算法来选择最优放置矩形.一方面,在选择矩形时尽量保持矩形放置后能填充满整个放置空间的宽度,同时与邻接空间齐平;另一方面,通过迭代搜索,获得相对全局最优的放置矩形.

1 RHA 模型

1.1 问题描述

2D Strip Packing 问题可以描述为:给定一个宽度为 W,高度无限制的矩形容器 C,同时给定 n 块宽度为 w_i、高度为 $h_i(i=1,2,...,n)$的矩形块 $R_1,R_2,...,R_n$,要求放入所有的矩形后,占用的容器高度最小.在矩形放置过程中,要求满足以下约束条件:

(1) 对于任意两块放入的矩形 r_i 和 $r_j(i,j=1,2,...,n;i\neq j)$,互相不能够重叠;

(2) 任意一块放入的矩形 r_i,其边不能超出容器的边界;

(3) 任意一块放入的矩形 r_i,其边要与容器的边平行;

(4) 矩形 r_i 可以旋转 90°放置.

1.2 相关定义

- 以容器 C 的左下角为坐标原点(0,0)建立笛卡尔坐标系,X 轴为容器 C 的宽的方向,Y 轴为容器 C 的高的方向;
- 矩形块 R_i 的状态为(x_i,y_i,r_i),其中,(x_i,y_i)表示其左下角的坐标,x_i=−1 表示该矩形块还没有被装载,r_i=0/1 表示矩形块是否旋转;
- 放置空间 S_i 用(x_i,y_i,sw_i)表示,其中,(x_i,y_i)表示该放置空间左下角的坐标,sw_i 表示该放置空间的宽度,同时,放置空间高度为无限制;
- 放置空间列表 $LS=\{S_1,S_2,\ldots,S_{k-1},S_k\}$,$k$ 为当前状态放置空间的个数;
- 系统状态 ST 表示当前所有矩形块的状态和放置空间列表.

2 迭代启发式算法规则

2.1 启发式策略

2.1.1 空间选择规则

假设当前状态为 ST,首先选择待放置的空间 S_i,S_i 的选择规则为:

(1) 选择最低位置(即 y_i 最小)作为下一个矩形块的放置位置;

(2) 如果存在多个最低位置,则选择其中最左边的位置(即 x_i 最小).

放置空间 S_i 确定后,判断该空间是否为可行空间,判断规则为:

遍历当前所有未放置的矩形,如果存在矩形的长或宽不大于 S_i 的宽 sw_i,则该空间为可行空间;否则,按照空间合并规则,将该 S_i 与其相邻空间合并,产生新的 LS,重新按照空间选择规则来选择 S_i.

如果当经过多次空间合并,最后合并成了唯一的放置空间,sw_i 等于容器宽度 W,而此时依然没有可以放置的矩形,则计算结束,同时报告用户剩下的盒子太大,容器根本无法放置.

2.1.2 矩形选择规则与放置规则

可行放置空间 S_i 确定后,接着确定待放置的矩形 R_j.遍历所有的未放置矩形,确定其针对 S_i 的适应度,适应度最大的即为被选择矩形 R_j.如图 1 所示,适应度计算规则为:

(1) 如果矩形 R_j 的长或宽中一边长度等于 sw_i,同时满足另一边的长度加上 y_i 等于其相邻空间齐平,即等于 y_{i-1} 或者 y_{i+1},则该矩形的适应度为 7;

(2) 如果存在两个矩形 R_{j1} 和 R_{j2},二者有一边相等,相等的边放在一起组合以后满足规则(1)的条件,则这两个矩形组合后的矩形 R_j 的适应度为 6.此时又称为组合适应度;

(3) 如果矩形 R_j 的长或宽中一边长度等于 sw_i,但是另一边的长度加上 y_i 并不等于其相邻空间齐平,则该矩形的适应度为 5;

(4) 如果存在两个矩形 R_{j1} 和 R_{j2},二者有一边相等,相等的边放在一起组合后,其另外一边相加等于 sw_i,则这两个矩形组合后的矩形 R_j 适应度为 4.此时又称为组合适应度;

(5) 如果矩形 R_j 的长或宽中一边长度加上 y_i 与其相邻空间齐平,但是另一边的长度并不等于 sw_i,则该矩形的适应度为 3;

(6) 如果存在两个矩形 R_{j1} 和 R_{j2},二者有一边相等且不等于 sw_i,相等的边放在一起组合后,其另外一边相加后再加上 y_i 与其相邻空间齐平,则这两个矩形组合后的矩形 R_j 其适应度为 2.此时又称为组合适应度;

(7) 如果上述条件都不满足,则该矩形的适应度为 1.

因为问题的目标是装载所有的矩形,同时求最小的高度,因此启发式算法考虑在装载的过程中,尽量保持不

要造成空间浪费.同时,装载后与左右相齐平.装载算法就是按照这种思路依次设计了 7 种适应度,然后按照适应度从大到小来选择待装载的矩形.在满足适应度最大的所有矩形中,选择面积最大的矩形作为被选择的矩形 R_j.在初始装载时,考虑矩形的宽度或者高度与放置空间的宽度一致、或者矩形的组合宽度或高度与放置空间的宽度一致、或者没有宽度一致的情况,分别类似于适应度为 5、适应度为 4 和适应度为 1 的情况.在满足适应度最大的所有矩形中,直接选择面积最大的矩形来放置,不需要迭代搜索两层.

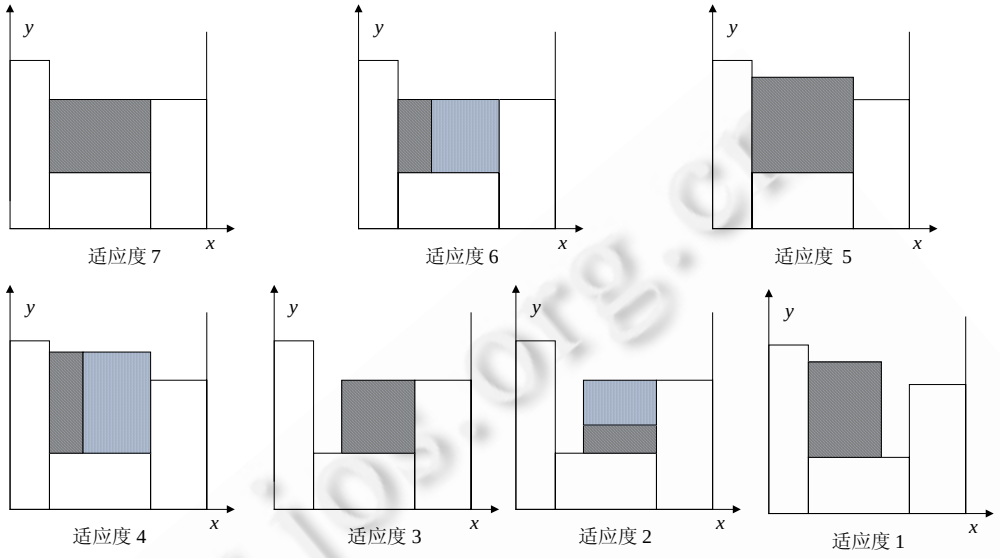


Fig.1 Fitness rules

图 1 适应度规则图

矩形的放置规则与选择规则紧密相关,具体如下:

(1) 当矩形 R_j 的适应度为 7 时,如果矩形 R_j 的宽与放置空间的宽 sw_i 相等,则该矩形直接放置于该放置空间;如果矩形 R_j 的长与放置空间的宽 sw_i 相等,则矩形旋转 90°后放入该放置空间;

(2) 当矩形 R_j 的适应度为 6 时,首先被组合的矩形 R_{j1} 和 R_{j2} 在组合时,已经分别进行了相应的旋转;对于组合后的矩形 R_j,其放置方式与放置规则(1)相同;

(3) 当矩形 R_j 的适应度为 5 时,放置方式与放置规则(1)相同;

(4) 当矩形 R_j 的适应度为 4 时,放置方式与放置规则(2)相同;

(5) 当矩形 R_j 的适应度为 3 时,如果矩形 R_j 的长加上 y_i 与其相邻空间齐平,则该矩形直接放置于该放置空间;如果矩形 R_j 的宽加上 y_i 与其相邻空间齐平,则该矩形旋转 90°后放置于该放置空间;

(6) 当矩形 R_j 的适应度为 2 时,首先被组合的矩形 R_{j1} 和 R_{j2} 在组合时,已经分别进行了相应的旋转;对于组合后的矩形 R_j,其放置方式与放置规则(5)相同;

(7) 当矩形 R_j 的适应度为 1 时,如果矩形的宽等于或者大于长,则将该矩形直接放置于该放置空间;否则,该矩形旋转 90°后放置于该放置空间.

具体的选择与放置规则可形式化如下:

HeuristicPacking(ST_i) //在当前状态 ST_i 下,可行放置空间 S_i

Begin

选择可行装载空间 S_i

For (每一个未放置的矩形 R_j)

$Fitness_{j1}=Fit(S_i,R_j)$ //计算矩形的适应度

If (存在适应度为 7 的矩形)

放置该矩形

return

End if

End for

For (每两个未放置的矩形 R_{j1},R_{j2})

$Fitness_{j2}=Fit(S_i,R_{j1},R_{j2})$ //计算组合适应度

End For

按照适应度从 6 到 1 选择最大适应度矩形,并放置该矩形

End

2.1.3 空间合并规则

当前所有未放置的矩形,如果所有矩形的长或宽都大于放置空间 S_i 的宽 sw_i,则 S_i 需要与其相邻的放置空间合并,如图 2 所示,合并规则为:

(1) 如果 S_i 位于容器的最左边或者最右边,则直接与其相邻的放置空间合并;

(2) 如果 S_i 位于容器的中间,且其左右两边放置空间的 y 值相等,则将 S_i 与左右两边放置空间同时合并;否则,选择其相邻放置空间中 y 值较小的合并;

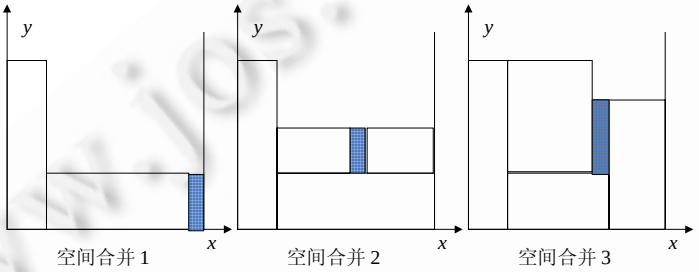


Fig.2 Space combinateion rules

图 2 空间合并规则图

具体的合并规则可形式化如下:

Combinate(S_i)

Begin

If ($S_i.x$==0)

与右边的放置空间合并

Else if ($S_i.x+S_i.sw$==W)

与左边的放置空间合并

Else

与两边相邻放置空间中的 y 值较小空间合并

End if

End if

End

2.2 迭代搜索算法

第 2.1 节中提出了一种基本的启发式算法,该算法针对当前状态可以选择出局部最优放置空间和放置矩形的组合.但是局部方案并不一定是全局最优方案,一种评估局部放置方案好坏的方式是使用某种方法补全它,并

以最终结果来作为局部放置方案的评估值.除了采用补全法以外,另一种方法是进行广度优先搜索当前放置方案,然后在叶子节点使用补全算法来评估叶子节点的好坏,最终选择整个搜索树中最好的叶子节点的评估值作为当前放置方案的评估值.但是,由于在搜索中每个节点都有大量的分支,采用广度优先搜索是不现实的.因此,本文设计了一种带深度限制的迭代搜索算法.该算法如图 3 所示,其中,当前状态 *ST* 指的是在当前所选择的放置空间内装入了被选择的最佳放置矩形后的状态.迭代搜索算法规则如下:

(1) 首先,针对当前状态 *ST*,按照空间选择规则选择可行放置空间 *S*;

(2) 针对 *S* 计算出当前所有未放置矩形块的所有可行放置方式 *P*;

(3) 执行放置 *P* 中的每一个放置方式 P_i,产生第 1 层的状态 ST_i;对每一个状态 ST_i,采用第 2.1 节中的启发式算法补全放置剩下的矩形;计算每一个最终结果的高度,其中,高度计算规则为:计算每一个矩形顶部 *y* 的坐标值,最大值即为其高度;如果只有一个最优高度,则高度所对应的放置方式 P_{opt} 即为当前状态 *ST* 的最佳放置方案,执行放置方式 P_{opt},转至规则(1)继续执行;如果存在多个最优高度,则执行规则(4);

(4) 对多个最优高度对应的放置方式 $P_1,P_2,\ldots,P_m$ 分别执行放置产生第 1 层的状态 $ST_1,ST_2,\ldots,ST_m$,然后分别执行规则(1)、规则(2),产生所有的可行放置方式;执行每一个放置方式,产生第 2 层的状态,采用第 2.1 节中的启发式算法补全放置剩下的矩形,计算各个状态对应的最优高度,但是此次只记录第 1 次得到的最优高度值,选择其对应的初始放置方案 $P_x(x=1,\ldots,m)$ 作为初始当前状态 *ST* 的最优放置方式;

(5) 执行该放置方式,产生新的当前状态,转至规则(1)执行.

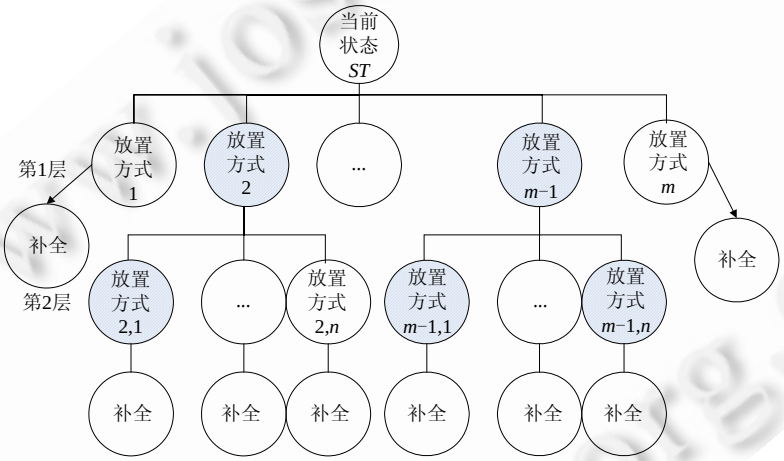


Fig.3 Recursive search rules

图 3 迭代搜索规则图

具体的迭代搜索规则可形式化如下:

```
Recursive_Search(ST)
Begin
  If (存在矩形未被放置)
    选择可行放置空间 S
    For (当前所有未装载矩形)
      生成并放置所有可行放置方式 Pi,产生第 1 层的状态 STi
      For (每一种状态 STi)
        HeuristicPacking(STi)          //启发式填充它
        计算其最终高度
```

```
    End For
  End For
  If (只有一个最优最终高度)
    选择对应的可行放置方式 P_i 为最优放置方式 P_opt
    执行该放置 P_opt
    未放置矩形数减 1,产生新的状态 ST
    Recursive_Search(ST)
  Else
    For (每一个最优最终高度所对应的状态 ST_j)
      选择可行放置空间 S_1
      For (当前所有未装载矩形)
        生成并放置所有可行放置方式 P_ji,产生第 2 层的状态 ST_ji
        For (每一种状态 ST_ji)
          HeuristicPacking(ST_ji)        //启发式填充它
          计算其最终高度
        End For
      End For
    End For
    取第 1 次所得到的最优高度,用其对应的第 1 层的状态 ST_j 代替 ST
    Recursive_Search(ST)
  End If
 End If
End
```

放置方式的评估搜索树的深度最多为 2 层,所以算法一方面考虑了局部评估值与全局评估值的关系,另一方面也兼顾了执行效率.

2.3 时间复杂度

计算单个矩形的适应度,时间复杂度为 $O(n)$;计算矩形的组合适应度,其时间复杂度为 $O(n^2)$.所以,选择启发式算法中,对矩形适应度计算的时间复杂度为 $O(n^2)$.在迭代搜索过程中,当迭代深度为 1 时,第 1 层有 $O(n)$级别个可放置方式节点;然后,每一个放置方式节点再使用启发式算法来填充剩下的所有节点.由于矩形适应度计算的时间复杂度为 $O(n^2)$,所以启发式算法的时间复杂度为 $O(n^3)$,对应第 1 层某一个放置方式所有矩形放置完毕的时间复杂度为 $O(n^3)$.当迭代深度为 2 时,每个第 1 层节点又可扩展出 $O(n)$级别个第 2 层可放置方式节点.同样,某一个第 2 层可放置方式节点所有矩形放置完毕的时间复杂度为 $O(n^3)$.因此,对应的被扩展的第 1 层节点时间复杂度为 $O(n^4)$.考虑所有第 1 层节点都扩展到第 2 层节点,由于第 1 层有 $O(n)$级别个可放置方式节点,则第 1 层所有节点计算完毕的时间复杂度为 $O(n^5)$,即选择一个最优放置矩形的时间复杂度为 $O(n^5)$.因为总共有 n 个矩形需要被放置,所以最终算法的时间复杂度为 $O(n^6)$.因此可以看出,算法还是属于多项式时间的算法,相对于复杂度为指数型的穷举算法而言,时间复杂度还是很低的.

3 实验结果

为了验证迭代启发式算法(RHA)的效率,针对多个标准的测试数据集,对其计算求优解,并与当前多种算法结果进行对比.第 1 个测试数据集 J 来自于文献[20],第 2 个测试数据集 C 来自于文献[21],第 3 个测试数据集 N 来自于文献[15],第 4 个测试集 CX 来自于文献[24].4 个测试数据集都是矩形 Packing 问题的经典测试数据集,4 个测试集的问题规模从小到大,其中,第 4 个测试集的规模达到了 15 000 个.

所有算法用 C 语言编程实现,实验平台:CPU 为 1.6GHz 的 T2050,内存为 1GB.在测试时,由于其他算法的实现都是在不同的测试平台上运行的程序结果,所以本文没有比较其他算法的计算时间,而是给出了 RHA 算法在本文实验平台上的计算时间,以秒为时间单位.4 个测试数据集的最优解都是已知的,在实验中用 H^*表示;装载矩形的个数用 n 表示;装载容器的宽用 W 表示;mean 表示实验结果取的是多次计算的平均值;sol 表示实验结果只有唯一值;best 表示多次实验结果所获得的最优值;time(s)表示计算所花费的时间.由于被比较的算法本身都没有对所有 4 个测试集做实验,有的只对测试集 J 做了实验,有的只对测试集 C 做了实验,有的只对测试集 C 和 N 做了实验,所以实验 1~实验 4 中并没有在所有的实例上对比所有的算法.本算法的终止条件是:(1) 所有矩形装载完毕;(2) 计算时间大于或等于 60s.两个条件只要一个满足即终止计算.

3.1 实验1

测试数据集 J 包括两组测试示例,都是由 40×15 的大矩形切割而成.第 1 组包括 25 个矩形,第 2 组包括 50 个矩形.

本文使用 RHA 算法求解,实验结果对比情况见表 1.由于 RHA 是确定型的启发式算法,对同一个测试数据,每一遍计算的结果都是相同的,所以结果只计算 1 遍;其他算法结果都取多次运行结果的平均值.

Table 1 Results of RHA and other algorithms for J class test data

表 1 J 类测试集 RHA 与其他算法计算结果比较

				GA+BL[20]	GA+IBL[18]	SPGAL[23]	GA+LLABF[17]	RHA	
数据集	n	W	H^*	Mean	Mean	Mean	Mean	Sol	Time (s)
J1	25	40	15	17.48	16.97	16.00	15.00	15.00	0.07
J2	50	40	15	17.28	17.01	15.00	15.00	15.00	4.36

从表 1 可以看出,RHA 算法比前面 3 种算法都优秀,与第 4 种算法一样都可以得到问题的最优解.因此,实验结果表明,本文的 RHA 算法是一种有效的算法.

3.2 实验2

测试数据集 C 包括 7 类子数据集,每一类中又包括 3 组共 21 组测试示例.矩形个数从第 1 组的 16 个到第 7 组的最大 197 个.该数据集的最优解是已知的,实验对比结果见表 2.

Table 2 Results of RHA and other algorithms for C class test data

表 2 C 类测试集 RHA 与其他算法计算结果比较

				BF[15]	HRBB[10]		A1[9]	HRP[8]	FH[16]	RHA	
数据集	n	W	H^*	Sol	Best	Mean	Sol	Sol	Sol	Sol	Time (s)
C11	16	20	20	21	20	20.00	20	20	20	20	0.06
C12	17	20	20	22	21	21.00	20	20	20	20	0.07
C13	16	20	20	24	20	20.00	20	20	21	20	0.03
C21	25	40	15	16	15	15.00	15	15	16	15	0.21
C22	25	40	15	16	15	15.00	15	15	15	15	0.24
C23	25	40	15	16	15	15.00	15	15	15	15	0.21
C31	28	60	30	32	30	30.00	30	31	31	30	0.32
C32	29	60	30	34	31	31.00	30	31	31	30	0.68
C33	28	60	30	33	30	30.00	31	31	32	30	0.20
C41	49	60	60	63	60	60.50	61	61	61	60	2.13
C42	49	60	60	62	61	61.33	61	60	61	60	1.96
C43	49	60	60	62	61	61.00	61	61	61	60	2.54
C51	73	90	90	93	90	90.67	91	91	91	90	13.73
C52	73	90	90	92	92	92.00	91	90	90	90	9.14
C53	73	90	90	93	91	91.17	91	91	91	90	12.48
C61	97	120	120	123	121	121.00	121	121	121	120	56.90
C62	97	120	120	122	121	121.83	121	121	121	120	33.27
C63	97	120	120	124	121	121.33	121	121	121	120	60.06
C71	196	240	240	247	242	242.17	241	241	241	241	60.03
C72	197	240	240	244	245	245.00	241	241	241	241	60.02
C73	196	240	240	245	241	241.33	241	241	241	241	60.06
平均值				84.95	83	83.16	82.76	82.76	82.95	82.29	–

表 2 给出了 BF,HRBB,A1,HRP 和 FH 以及本文中 RHA 对 21 组测试实例的实际测试结果.基于 BF 算法的计算结果中,一次都不能获得最优结果;HRBB 算法能够获得 7 次最优结果;A1 算法和 HRP 算法同样能够获得 8 次最优结果;FH 算法能够获得 5 次最优结果;而 RHA 能够获得 18 次最优结果.在计算时间上,由于各种算法在计算所使用的机器性能的差异,本文只给出了 RHA 计算的时间,计算时间小于 0.01s 的记为 0.对于大部分测试实例计算时间不到 1s,在最长计算时间算例上,C7 类计算结果比最优结果仅仅多 1.因此,RHA 无论在计算最优高度、最优高度平均值和计算时间上都能取得很好的结果.

表 3 给出了 RHA 与其他算法的平均 *Gap*(*Gap*=(计算最优高度–H^*)/H^*)值.因为有的算法只是给出了 *Gap* 值比较而没有给出具体的计算数值,所以表 3 中比较的算法包含了表 2 中比较的算法.从表 3 可以看出,按照平均 *Gap* 值排在前 5 位的算法依次是 RHA,HRP,FH,SPGAL 和 A1,其中,RHA 的平均值为 0.06,远远低于其他算法的计算结果.因此,实验结果进一步证明,本文提出的 RHA 明显优于其他算法.

Table 3 *Gap* of RHA and other algorithms

表 3 RHA 与其他算法的平均 *Gap* 比较

Gap (%)	C1	C2	C3	C4	C5	C6	C7	Average
GA+BLF[21]	4	7	5	3	4	4	5	4.57
SA+BLF[21]	4	6	5	3	3	3	4	4
BF[15]	11.67	6.7	9.9	3.87	2.93	2.5	2.23	5.69
HRBB[10]	1.70	0	1.10	2.20	1.90	1.40	1.30	1.40
A1[9]	0	0	1.11	1.67	1.11	0.83	0.42	0.73
HRP[8]	0	0	3.33	1.11	0.74	0.83	0.42	0.91
FH[16]	1.67	2.22	4.30	1.67	0.74	0.83	0.42	1.68
GA+IHR[19]	3.33	4.44	2.22	1.67	1.11	0.83	0.83	2.06
SPGAL[23]	1.7	0	2.2	0	0	0.3	0.3	0.6
PH[6]	5	4.44	4.44	3.33	1.11	1.11	1.25	2.95
HR[8]	8.33	4.45	6.67	2.22	1.85	2.5	1.8	3.97
RHA	0	0	0	0	0	0	0.42	0.06

3.3 实验3

测试数据集 N 包括 13 组测试实例,矩形个数从第 1 组的 10 个到第 13 组的 3 152 个.该数据集的最优解也是已知的.实验对比结果见表 4,表 5 给出了相应的 *Gap* 值.表中省略线表示其计算时间过长,无法得到结果.对于 GA+BLF,SA+BLF 和 GA+IHR,计算的是部分平均值.

Table 4 Results of RHA and other algorithms for N class test data

表 4 N 类测试集 RHA 与其他算法计算结果比较

				BF[15]	GA+BLF[21]	SA+BLF[21]	HA[7]	HRP[8]	FH[16]	GA+IHR[19]	RHA	
数据集	*n*	*W*	H^*	Sol	Mean	Mean	Sol	Sol	Sol	Mean	Sol	Time (s)
N1	10	40	40	45	40	40	40	40	40	45	40	0
N2	20	30	50	53	51	52	50	51	52	54	50	0.07
N3	30	30	50	52	52	52	50	51	51	51	50	0.17
N4	40	80	80	83	83	83	81	81	83	83	81	1.58
N5	50	100	100	105	106	106	102	102	102	103	101	6.00
N6	60	50	100	103	103	103	101	102	101	102	100	7.11
N7	70	80	100	107	106	106	101	102	102	104	100	36.72
N8	80	100	80	84	85	85	81	81	81	82	80	60.03
N9	100	50	150	152	155	155	151	152	151	152	150	60.02
N10	200	70	150	152	154	154	151	151	151	151	150	60.07
N11	300	70	150	152	155	155	151	151	151	151	150	60.01
N12	500	100	300	306	313	312	303	305	301	304	301	60.07
N13	3 152	640	900	964	…	…	962	972	960	…	962	>60
平均值				181.38	116.92	116.92	178.77	180.08	178.92	115.17	178.08	–

Table 5 *Gap* of RHA and other algorithms

表 5 N 类测试集 RHA 与其他算法的 *Gap* 比较

Gap (%)	N1	N2	N3	N4	N5	N6	N7	N8	N9	N10	N11	N12	N13	平均值
BF[15]	12.5	6	4	3.75	5	3	7	5	1.33	1.33	1.33	2	0.42	4.05
GA+BLF[21]	0	2	4	3.75	6	3	6	6.25	3.33	2.67	3.33	4.33	…	3.72
SA+BLF[21]	0	4	4	3.75	6	3	6	6.25	3.33	2.67	3.33	4	…	3.86
HA[7]	0	0	0	3.75	2	1	1	1.25	0.67	0.67	0.67	1	0.21	0.75
HRP[8]	0	2	2	1.25	2	2	2	1.25	1.33	0.67	0.67	1.67	1.25	1.39
FH[16]	0	4	2	3.75	2	1	2	1.25	0.67	0.67	0.67	0.33	0	1.41
GA+IHR[19]	12.5	8	2	3.75	3	2	4	2.5	1.33	0.67	0.67	0.13	…	3.48
RHA	0	0	0	1.25	1	0	0	0	0	0	0	0.33	0.21	0.21

从表 4 和表 5 可以看出:BF,GA+BLF,SA+BLF 和 GA+IHR 算法没有一次得到最优解,HA 算法得到 3 次最优解,HRP 和 HF 算法分别得到 2 次最优解,而 RHA 算法在 9 组测试实例中均获得最优解;在 RHA 算法未取得最优值的 3 组测试实例中,其计算结果与最优解的差距为 1,也小于其他大部分算法的计算结果;在计算时间上,RHA 有 3 组花费时间小于 1s,最长时间花费为 60.07s;在 *Gap* 值中,RHA 所得到的结果在所有算法中都是最小值.对于 N 类测试集的计算,与其他算法比较,RHA 的优势非常明显.因此,该测试也验证了 RHA 算法是一种高效的算法.

3.4 实验4

测试数据集 CX 包括 7 组测试实例,矩形的个数从第 1 组的 50 个到第 7 组的 15 000 个,该数据集的最优解也是已知的.实验对比结果见表 6,表中省略号表示其计算时间过长,无法得到结果.

Table 6 Results of RHA and other algorithms for CX class test data

表 6 CX 类测试集 RHA 与其他算法计算结果比较

				HRP[8]	FH[16]	RHA	
数据集	*n*	*W*	H^*	Sol	Sol	Sol	Time (s)
50CX	50	400	600	615	624	603	5.96
100CX	100	400	600	615	619	615	60.02
500CX	500	400	600	611	600	600	60.08
1000CX	1000	400	600	607	600	608	60.03
5000CX	5000	400	600	607	600	600	>60
10000CX	10000	400	600	…	600	600	>60
15000CX	15 000	400	600	…	600	612	>60
平均值				611.00	606.14	605.43	–

从表 6 可以看出:HRP 算法没有一次得到最优解,FH 算法得到 5 次最优解,而 RHA 算法在 7 组测试实例中 3 次获得最优解;在计算时间上,RHA 只有第 1 组花费时间小于 60s,其他组的花费都超过了 60s;在计算平均值方面,RHA 取得了最小的平均值.因此,对于超大规模的 CX 类测试集,与其他算法比较,RHA 算法也存在一定的优势.

为了在更大规模的问题上进一步验证算法的性能,基于测试数据集 15000CX,将前 1 000 个矩形按照长度等分为两个相同的矩形后得到 16 000 个矩形.对得到的数据进行测试,程序在 60s 内无法得到结果;对第 2 个 1 000 个矩形和第 3 个 1 000 个矩形做同样的等分后,对得到的数据进行测试,程序在 60s 内也无法得到结果.因此,尽管本文提出的算法是多项式时间算法,但 *n* 的方次比较高,当数据规模很大时,该算法无法进行有效处理.

4 结束语

基于国际上公认的 Benchmark 实例的实验结果表明,我们提出的 RHA 是一种快速且高效的算法.该算法在组合规则上进行了更细致的考虑,近似解的性能会更好,实验也证明了这个结果.本文的算法可以进一步推广矩形条装箱的其他类型、背包问题、矩形装箱问题等其他矩形 Packing 问题.

下一步的工作将从两个方向开展:1) 如何进一步提高 RHA 的效率和效果;2) 将其扩展到其他 Packing 问题的求解.

致谢 在此,我们向对本文的工作给予支持和建议的同行表示感谢.

References:

[1] Martello S, Monaci M, Vigo D. An exact approach to the strip-packing problem. Informs Journal on Computing, 2003,15(3): 310−319. [doi: 10.1287/ijoc.15.3.310.16082]

[2] Beasley JE. An exact two-dimensional non-guillotine cutting tree search procedure. Operations Research, 1985,33(1):49−64. [doi: 10.1287/opre.33.1.49]

[3] Lesh N, Marks J, McMahon A, Mitzenmacher M. Exhaustive approaches to 2D rectangular perfect packing. Information Processing Letters, 2004,90(1):7−14. [doi: 10.1016/j.ipl.2004.01.006]

[4] Martello S, Vigo D. Exact solution of the two-dimensional finite bin packing problem. Management Science, 1998,44(3):388−399. [doi: 10.1287/mnsc.44.3.388]

[5] Fekete SP, Schepers J. On more-dimensional packing III: Exact algorithms. Technical Report, ZPR97-290, Koln: Mathematicsches Institut. Yniversitat zu Koln, 1997.

[6] 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).

[7] Huang WQ, Chen DB. An efficient heuristic algorithm for rectangle-packing problem. Simulation Modelling Practice and Theory, 2007,15(10):1356−1365. [doi: 10.1016/j.simpat.2007.09.004]

[8] Huang WQ, Chen DB, Xu RC. A new heuristic algorithm for rectangle packing. Computer & Operations Research, 2007,34(11): 3270−3280. [doi: 10.1016/j.cor.2005.12.005]

[9] Chen M, Huang WQ. A two-level search algorithm for 2D rectangular packing problem. Computer & Industrial Engineering, 2007, 53(1):123−136. [doi: 10.1016/j.cie.2007.04.007]

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

[11] Baker BS, Jr Coffman EG, Rivest RL. Orthogonal packings in two dimensions. SIAM Journal on Computing, 1980,9(4):808−826. [doi: 10.1137%2f0209064]

[12] Chazelle B. The bottom-left bin packing heuristic: An efficient implementation. IEEE Trans. on Computers, 1983,32(8):697−707. [doi: 10.1109/TC.1983.1676307]

[13] Hopper E. Two-Dimensional packing utilising evolutionary algorithm and other meta-heuristic methods [Ph.D. Thesis]. Cardiff University, 2000.

[14] Alvarez-Valdes R, Parreno F, Tamarit JM. Reactive GRASP for the strip-packing problem. Computers & Operations Research, 2008,35(4):1065−1083. [doi: 10.1016/j.cor.2006.07.004]

[15] Burke EK, Kendall G, Whitwell G. A new placement heuristic for the orthogonal stock-cutting problem. Operations Research, 2004, 52(4):655−671. [doi: 10.1287/opre.1040.0109]

[16] Leung SCH, Zhang Df. A fast layer-based heuristic for non-guillotine strip packing. Expert Systems with Applications, 2011,38(10): 13032−13042. [doi: 10.1016/j.eswa.2011.04.105]

[17] Jiang XB, Lu 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 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3395.htm [doi: 10.3724/SP.J. 1001.2009.03395]

[18] Liu DQ, Teng HF. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. European Journal of Operation Research, 2006,172(3):814−837. [doi: 10.1016/S0377-2217(97)00437-2]

[19] Zhang DF, Chen SD, Liu YJ. An improved heuristic recursive strategy based on genetic algorithm for the strip rectangular packing problem. Acta Automatica Sinica, 2007,33(9):911−916. [doi: 10.1360/aas-007-0911]

[20] Jakobs S. On genetic algorithms for the packing of polygons. European Journal of Operational Research, 1996,88(1):165−181. [doi: 10.1016/0377-2217(94)00166-9]

[21] 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. [doi: 10.1016/S0377-2217(99)00357-4]

[22] Burke EK, Kendall G, Whitwell G. Metaheuristic enhancements of the best-fit heuristic for the orthogonal stocking cutting problem. Technical Report, NOTTCS-TR-2006-3, Nottingham: University of Nottingham, 2006.

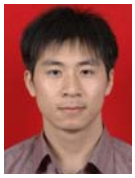
[23] Bortfeldt A. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. European Journal of Operational Research, 2006,172(3):814−837 [doi: 10.1016/j.ejor.2004.11.016]

[24] http://59.77.16.8/download.aspx#p4

附中文参考文献:

[6] 张德富,韩水化,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] 蒋兴波,吕肖庆,刘成城.二维矩形条带装箱问题的底部左齐择优匹配算法.软件学报,2009,20(6):1528−1538. http://www.jos.org.cn/1000-9825/3395.htm [doi: 10.3724/SP.J.1001.2009.03395]



彭碧涛(1978－),男,湖北天门人,博士,讲师,主要研究领域为算法优化,人工智能.

周永务(1964－),男,博士,教授,博士生导师,主要研究领域为运筹与管理,算法优化.