第 31 卷 第 3 期
2008 年 3 月
计 算 机 学 报
CHINESE JOURNAL OF COMPUTERS
Vol. 31 No. 3
Mar. 2008

# 求解矩形 Packing 问题的砌墙式启发式算法

张德富[1) ] 韩水华[2)] 叶卫国[2)]
[1)] (厦门大学信息科学与技术学院 福建 厦门 361005)
[2)] (厦门大学管理科学系 福建 厦门 361005)

**摘 要** 为求解正交矩形 Packing 问题提出了一个新颖而有效的砌墙式启发式算法.该算法主要基于砌墙式启发式策略,其思想主要来源于砖匠在砌墙过程中所积累的经验:基于基准砖的砌墙规则.对国际上公认的大量的 Benchmark 问题例的计算结果表明,该算法的计算速度不仅比著名的现代启发式算法快,而且获得更优的高度.

**关键词** 正交矩形 Packing 问题;启发式;砌墙式规则;局部搜索;基准砖
**中图法分类号** TP18

# A Bricklaying Heuristic Algorithm for the Orthogonal Rectangular Packing Problem

ZHANG De-Fu[1)] HAN Shui-Hua[2)] YE Wei-Guo[2)]
[1)] (*Department of Computer Science and Technology, Xiamen University, Xiamen, Fujian* 361005)
[2)] (*Department of Management Science, Xiamen University, Xiamen, Fujian* 361005)

**Abstract** A novel and effective bricklaying heuristic algorithm for two-dimensional rectangular Packing problem is presented. This algorithm is mainly based on bricklaying heuristic strategies inspired by a large number of experiences accumulated by bricklayers during the process of building the wall, especially, the building wall strategy based on the reference brick is presented. The computational results on large number of Benchmark problems have shown that this algorithm not only runs in shorter time than known meta-heuristic but also finds shorter height.

**Keywords** the orthogonal rectangular Packing problem; heuristic; bricklaying rule; local search; reference brick

## 1 引 言

在工程设计领域中的许多实际问题,例如货物的装运、板材的切割、服装的剪裁下料、集成电路的设计等,这些问题都可以形式化为一个矩形 Packing 问题[1],因此寻找这类问题的快速解法,具有重大的实际意义.同时,这类问题又是一个 NP-难问题[2-3],涉及到计算机科学理论的核心问题,因此具有深刻的理论价值.然而至今的研究实践表明,NP-难问题不存在有效的多项式时间算法.因此,从实际应用的角度出发,人们开始寻找兼顾解的质量以及运行时间的高效启发式算法.

在国内,对 Packing 问题的研究比较少,工作主要集中在圆形 Packing 问题以及矩形排样问题[4-9].在国外,由于矩形 Packing 问题的重要性,对它的研究还是比较多的,其求解算法包括精确算法[10]以及近似算法.精确算法的计算复杂度太高,实用价值不

收稿日期:2005-06-09;最终修改稿收到日期:2007-12-19.本课题得到国家自然科学基金(60773126)、福建省自然科学基金(A0710023)、厦门大学院士启动基金(X01109)、厦门大学"九八五"信息科技基金(0000-X07204)资助.张德富,男,1971 年生,博士,副教授,研究方向为计算智能与数据挖掘.E-mail: dfzhang@xmu.edu.cn.韩水华,男,1970 年生,博士,副教授,研究方向为信息计算以及运筹研究.叶卫国,男,1970 年生,博士,研究方向为算法设计和管理工程.

大.近似算法虽然解的质量不高,但是计算速度比较快,因而受到广泛的关注和研究.特别是启发式算法,由于其能在合理的时间范围内获得不错的解,因此该类方法一直是研究的热点,并且得到了一些有效的非常通用的算法.例如神经网络算法、模拟退火算法、遗传算法等[11-12],特别是文献[13]给出了有关现代启发式算法求解矩形 Packing 问题的研究和评论;Wu 等[14]提出的一个 quasi-human 启发式算法,Hifi 等将构造性方法与遗传算法相结合而得到的 CA GA 算法[15]以及最近提出的混合启发式算法[16],都获得了不错的结果.本文作者通过实际观察工匠们砌墙或铺地砖的过程,从他们在实际 Packing 过程中所积累的经验出发,为矩形 Packing 问题的高效快速求解提出了一个新颖的砌墙式启发式算法.

## 2 数学模型

假设已知一个宽度为 $W$ 的矩形板,另外又已知 $n$ 个长度为 $l_i$,宽度为 $w_i$ $(1 \le i \le n)$ 的矩形,矩形 Packing 问题的目的是将 $n$ 个矩形互不重叠地放进矩形板,使得所使用地矩形板的高度最小(见图 1).此问题可以更形式化地描述为:将二维笛卡尔坐标的原点取在矩形板的左下角,$(0, H)$ 为矩形板的左上角坐标,$(W, 0)$ 为矩形板的右下角坐标,问题是求 $n$ 个四元组的集合,即

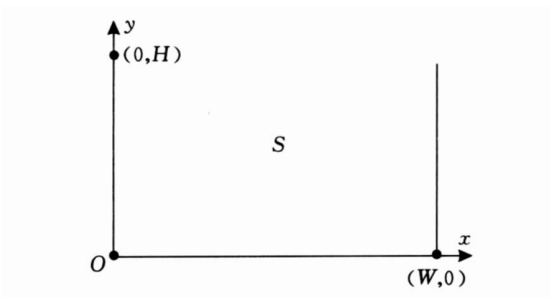


图 1

$$P = \{\langle (x_{li}, y_{li}), (x_{ri}, y_{ri})\rangle | 1 \le i \le n, x_{li} < x_{ri}, y_{li} > y_{ri}\},$$

使得矩形板的高度 $H$ 最小.这里,$(x_{li}, y_{li})$ 表示矩形 $i$ 的左上角坐标,$(x_{ri}, y_{ri})$ 表示矩形 $i$ 的右下角坐标,并且使得对任意的矩形 $i(1 \le i \le n)$,其坐标满足下列 3 个条件:

(1) $x_{ri} - x_{li} = l_i \wedge y_{li} - y_{ri} = w_i$;

或者 $x_{ri} - x_{li} = w_i \wedge y_{li} - y_{ri} = l_i$.

(2) 对任意的矩形 $j(1 \le j \le n, j \ne i)$,矩形 $i$ 和矩形 $j$ 不能互相重叠,即

$x_{ri} \le x_{lj}$, $x_{li} \ge x_{rj}$; 或者 $y_{ri} \ge y_{lj}$, $y_{li} \le y_{rj}$.

(3) $0 \le x_{li} \le W$, $0 \le x_{ri} \le W$; 并且 $0 \le y_{li} \le H$, $0 \le y_{ri} \le H$.

值得注意的是,正交的矩形 Packing 问题指的是在 Packing 的过程中,所有矩形的边都应该平行,即每个矩形的边与坐标轴平行.本文考虑的就是这种正交的矩形 Packing 问题,并为之提出了一个新颖的砌墙式启发式算法.

## 3 新颖的砌墙式启发式算法

### 3.1 砌墙式启发式思想的来源

几千年来,工匠们在砌墙或铺地砖的过程中,积累了丰富的实际操作经验,并代代相传,延续至今.其中,最重要的一个规则是:先角后边.它描述了在砌墙或铺地砖的过程中,砖块放置的优先原则.事实上,据我们观察和了解,还有一个特别有用的规则.在砌墙的过程中,工匠们总是拉一根水平线,然后根据水平线从墙角的一边砌向另一边,例如从左边砌向右边,并且总是从最低的位置砌起,按照左底的策略砌砖.如图 2 所示,空间 $S$ 可以看成一面待砌的墙,现在有编号为 1,2,3,4,5,6 的砖块可以使用.根据这个具体的例子,我们来看看工匠们砌墙的过程.工匠们知道,墙角 $AOB$ 或 $OBC$ 应当先砌,不失一般性,我们先砌 $AOB$.将编号为 1 的砖放入后,工匠们就会在 1 号砖的顶部拉一条线,这里,我们用虚线表示.此时,墙 $S$ 就被划分为两块待砌的空间 $S^1$ 和 $S^2$.按照先砌位置最低空间的原则,工匠们会先砌空间 $S^1$,此时,按照墙角优先的原则,就可以依次把 2,3,4 号砖放入 $S^1$,当把 5 号砖放入的时候,墙角 $OBC$ 都已经放满,如图 3 所示.如果虚线下还有剩余空间,这时,工匠们会考虑选择一个位置最低的空间来放置砖块.例如图 4,空间 $S_3$ 比其他的空间 $S_2$, $S_4$, $S_5$ 的位置都低,因此 $S_3$ 被优先考虑,但此时 6 号砖不能放下去,因此又得寻找新的最低空间,重复这个过程,直到虚线下的空间不能再放置砖块为止,即此时 6 号砖在虚线下的空间内放不下.如果还有砖块没放置,则需要在空间 $S^2$ 的左下角位置放置砖块,此时只有 6 号砖待放,因此在该位置放置 6 号砖,从而导致如图 5 的情形.这时,虚线下的最低空间仍然可以分成 $S_2$, $S_3$, $S_4$, $S_5$,其中 $S_3$ 位置最低,仍然被优先考虑.重复上面的过程,直到所有的砖块都放置完.此时,图 5 就是砌墙的结果,6 号砖的顶高 $OA$ 就是所求的高度.从上面的过程,我们可以看

出,1 号、6 号砖的位置非常重要,别的砖块放置时,都必须向它看齐,因此,我们把这种砖称为基准砖.

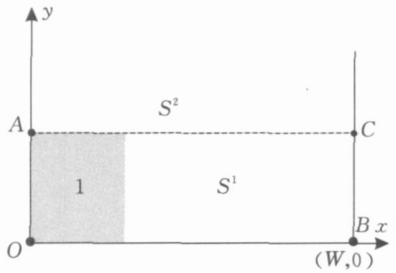

图 2

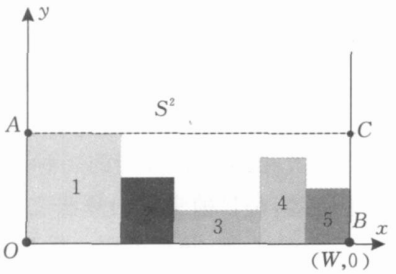

图 3

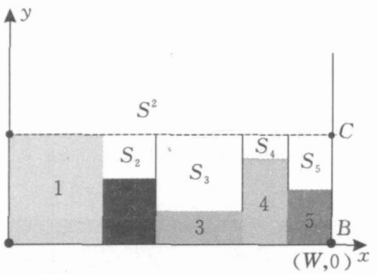

图 4

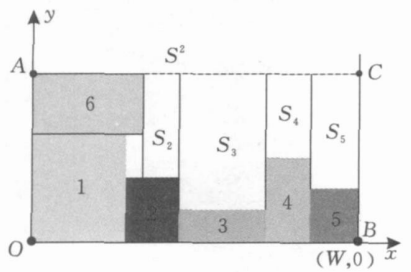

图 5

### 3.2　砌墙式启发式策略

从上节我们对工匠砌墙的过程描述可以看出,工匠主要依据上述两个砌墙的规则:先角后边以及向基准砖看齐的砌墙规则. 由于这两个规则经过了几千年实践的检验,因此非常有效,从而仍然是目前工匠所遵循的砌墙规则之一. 因此,我们从工匠所积累的经验出发,并将这些规则形式化,从而得到矩形 Packing 问题的求解策略. 由于这些策略是受人类几千年所积累的砌墙的经验启发得到的,因此我们称之为砌墙式启发式策略. 具体的,我们可以将先角后边的规则形式化为左底策略,即待填充空间最左下角位置优先. 以基准砖顶部虚线为准,令基准砖虚线下的空间 $S^1$,按照图 1 的坐标体系,现在描述一下填充空间 $S^1$ 的策略:以基准砖顶部虚线为准,确定空间 $S^1$ 中位置最低的最靠左的位置,然后往上碰到顶部虚线以及往右碰到一个放置的砖为止,以便确定一个位置最低的一个矩形空间 $S^d$,然后扫描是否有待填充的矩形能够填入该空间,如有,则按左底策略填入,重复此过程,直到空间 $S^1$ 不能再填充矩形为止. 将向基准砖看齐的填充策略称为基准填充策略,为方便实现,我们令 $(cx, ch)$ 表示待找的 $S^d$ 的左下角坐标,令 $R[j].y$ 表示横坐标为 $j$ 的纵坐标的值,$R[j].s$ 表示横坐标为 $j$ 的位置是否寻找过,设 $n$ 个矩形的标号顺序 $O$ 为 $r[i].order(0 \le i < n)$,$r[r[i].order].isused$ 表示标号为 $r[i].order$ 的矩形是否已经填充,$r[r[i].order].isused = 0$ 表示未填充,$r[r[i].order].isused = 1$ 表示已经填充. 具体的基准填充策略可形式化如下:

```
ReferencePacking(S^1)
  while ch < h do        // h 为基准砖顶部的高度
    cx ← tw - 1;         // tw 为基准砖的最右横坐标
    ch ← R[cx].y;        // 基准砖最顶纵坐标
    for k ← tw to W - 1 do
                         // 下面确定 cx, ch,然后确定 S^d
      if (R[k].y < minh && R[k].s = 0) then
        cx ← k;
        ch ← R[k].y;
    j ← cx;
    while(R[j].y ≤ ch && j < W) do j ← j + 1;
    while (R[cx - 1].y ≤ ch && R[cx - 1].s = 1)
        cx ← cx - 1;
    if(j - cx > 0 && h - ch > 0) then
                         // S^d 的长为 j - cx,宽为 h - minh
      for i ← 0 to n - 1 do
        if(r[r[i].order].isused = 0) then
          矩形 r[i].order 能否填充进 S^d,如果能,则
            更新 cx, R[ ].y, R[cx - 1].s.
```

此外,从工匠砌墙的过程可以看出,基准砖的放置看似简单,其实包含深刻的内涵,因为基准砖直接影响水平线的设置,即"分"的策略,从而直接影响到砌墙的效果,因此,如何选择要放置的矩形以及如何放置这个矩形就显得非常重要. 例如,2,3,4,5 号砖的放置必须向 1 号砖看齐,不能越过虚线的位置. 特别从 6 号砖的放置,可以看出,6 号砖的下面将有一

部分空间不能被利用. 但是,如果 1 号砖比 6 号砖大,则可以避免这种情形. 事实上,从求解的目标来说,问题的目的是为了使高度 $H$ 尽可能低,因此我们应该优先选择大的矩形进行填充. 受这些思想的启发,我们可以得到下面的启发式策略.

启发式排序策略:将所有待填充的矩形按面积由大到小排序.

### 3.3 砌墙式启发式算法描述

给定 $n$ 个矩形,其顺序为 $O$,将前面提出的砌墙式启发式策略进一步形式化,便可以得到求解矩形 Packing 问题的砌墙式启发式填充过程:

```
Pack(S)
  当还有矩形没有填入空间 S 时
    选择一个基准矩形,按照左底策略填入空间 S;
    将剩下的空间划分为两个空间 S^1, S^2;
    ReferencePacking(S^1)
    按照基准填充策略,填充 S^1,直到没有矩形能够
    填入 S^1;
    S ←S^2;
  返回最后一个基准矩形的顶高.
```

其中,基准矩形是当前矩形顺序 $O$ 中第一个未填充的矩形.

此外,如果严格按照矩形的大小顺序进行填充,并不一定能够得到好结果. 实际上,在日常生活中,我们将东西放在柜子里或者箱子里的时候,我们一般考虑交叉放置、大小相间的放置策略. 这启发我们可以改变待放置矩形的次序,多尝试几种填充顺序,进行局部搜索,然后从中选择最好的结果. 有了填充过程 Pack$(S)$ 后,结合提出的启发式策略,便可以得到如下砌墙式启发式算法:

```
砌墙式启发式法()
  按启发式排序策略对面积进行排序;
  besth ←∞;
  for i ←1 to n - 1 do
    for j ←i to n do
      在当前填充序中交换矩形 i 和矩形 j 的顺序;
      currenth ←Pack(S);
      if currenth < besth then
        besth ←currenth;
        ci ←i;
        cj ←j;
      交换矩形 i 和矩形 j 的顺序;
    交换矩形 ci 和矩形 cj 的顺序.
```

这里,$currenth$ 表示当前矩形序列所获得高度,$besth$ 表示目前为止所获得的最优高度.

### 3.4 算法复杂性

按照填充过程 Pack$(S)$,我们可以看出,该过程的时间主要花费在找位置最低的空间上,而找位置最坏需要扫描 $W \times h$ 次. 设 $T(n)$ 表示规模为 $n$ 时,对矩形 $S$ 进行填充所花的时间,则可以得到 $T(n) = O(nWh)$. 这里 $h$ 表示当前基准砖的顶高. 因此,砌墙式启发式算法的复杂度为 $T(n) = O(n^3 Wh)$.

## 4 计算结果

为了验证砌墙式启发式算法(PH)的效率,我们将 PH 与 SA + BLF、GA + BLF[13]以及 CAGA[15]进行了对比. 其中,SA + BLF 和 GA + BLF 是分别将模拟退火算法和遗传算法与 BLF 启发式相结合而得到的,而 CAGA 是将遗传算法与构造性启发式相结合而得到的,这些算法都是将现代启发式算法与具体的布局启发式相结合而得到的一种混合算法,因此特别快速有效,是目前最好的算法之一. 我们的算法用 C 语言编程实现,并在 CPU 为 1.6GHz,内存 256MB 的 PC 机上进行了大量的实例计算,总共测试了 40 个问题例,其中 C1～C7 来自于文献[13],包括 21 个例子,SCPL1～SCPL9 来自于文献[15]. 另外,还包括随机生成的 10 个规模比较大的例子. 此外,我们给出了 PH 计算随机生成例子 RPP3 的一个 Packing 结果,如图 6 所示.

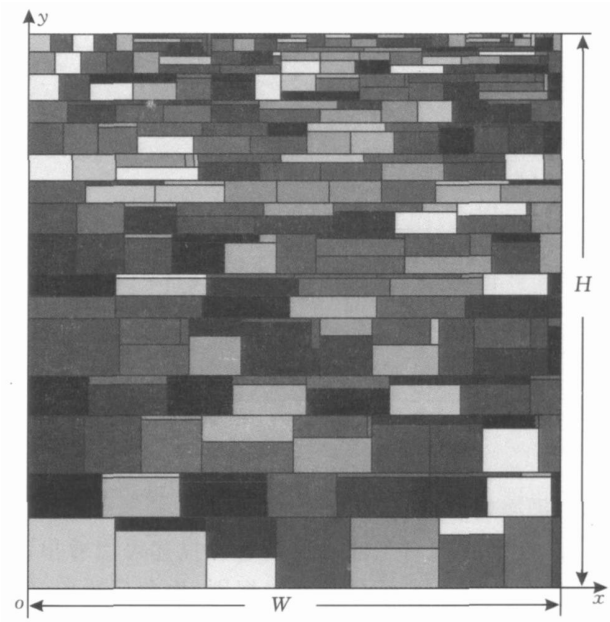


图 6 PH 计算 RPP3 例子的一个填充结果:耗时 417s,获得最优高度

### 4.1 与 SA + BLF 和 GA + BLF 的比较

虽然,我们没有实现文献[13]中的 SA + BLF 和 GA + BLF 算法,但是他们对 C1～C7 例子进行

了计算,并报告了时间和相对高度百分比 *Gap*,因此,我们直接取他们的结果. 由于我们计算机运行速度比他们的快大约 8 倍,因此,我们对文献[13]中的计算时间做了一些近似处理,将他们的计算时间除以 8, 3 种算法的计算结果见表 1 和表 2. 其中, $Gap = 100 \times (H - H^*)/H^*$, $H$ 表示近似解的高度, $H^*$ 表示最优高度或者目前已知的最佳高度. 这里 PH 对每个算例只计算一次. 表 1 中的值表示一个算法计算例子 C*i*($1 \leq i \leq 7$)所获得的平均 *Gap* 以及所有例子平均 *Gap* 的平均值. 从表 1 可以看出,PH 只有两类例子 C1 和 C3 的平均 *Gap* 比 GA + BLF 和 SA + BLF 的高,其它 5 类例子的平均 *Gap* 都比它们的低(见表中粗体数字). PH 的平均值只有 2.95,而 GA + BLF 和 SA + BLF 的平均值分别为 4.57 和 4. 从表 1 还可以看出,随着问题规模的增大,PH 所得到的平均 *Gap* 有减少的趋势. 这表明,PH 对大的例子表现得更好. 表 2 报告了 3 种算法的计算时间,*RT* 表示运行时间,单位为 s,以下同. 从表 2 可以看出,PH 的计算速度比 GA + BLF 和 SA + BLF 的速度快很多.

**表 1　PH 与 GA + BLF 和 SA + BLF 的平均 Gap 比较**

| | *Gap*/ % | | | | | | | 平均值 |
|---|---|---|---|---|---|---|---|---|
| | C1 | C2 | C3 | C4 | C5 | C6 | C7 | |
| GA + BLF | 4 | 7 | 5 | 3 | 4 | 4 | 5 | 4.57 |
| SA + BLF | 4 | 6 | 5 | 3 | 3 | 3 | 4 | 4 |
| PH | 5 | **4.44** | **4.44** | 3.33 | **1.11** | **1.11** | **1.25** | **2.95** |

**表 2　PH 与 GA + BLF 和 SA + BLF 的计算时间 RT 比较**

| | *RT*/ s | | | | | | | 平均值 |
|---|---|---|---|---|---|---|---|---|
| | C1 | C2 | C3 | C4 | C5 | C6 | C7 | |
| GA + BLF | 7.5 | 15 | 22.5 | 97.5 | 270 | 645 | 5827.5 | 983.57 |
| SA + BLF | 5.25 | 18 | 30 | 247.5 | 862.5 | 2865 | 31357.5 | 5055.11 |
| PH | 0 | 0 | 0 | 1.51 | 5.69 | 23.74 | 707.12 | **105.45** |

### 4.2　与 CAGA 的比较

CAGA 算法是将构造性方法与遗传算法相结合而得到的,因此计算效率比较高. 该算法所使用的是 CPU 为 733MHz,内存为 128MB 的 PC 机[15],虽然,我们没有实现 CAGA,但是文献[15]对 SCPL1～SCPL9 例子进行了计算,并报告了 *Gap* 和运行时间,因此,我们直接取文献[15]的结果. 由于我们计算机运行速度比他们的快大约 2 倍,因此,我们对文献[15]中的计算时间做了一些处理,将他们的计算时间除以 2, 3 种算法的计算结果见表 3. 从表 3 可以看出,PH 的平均高度比 CAGA 的低,PH 只有两类例子 SCPL6 和 SCPL8 的高度比他们的高,其他的都比他们的低. 特别地,对于 SCPL7,PH 算法给出了一个比其目前最佳高度还要低的高度.

**表 3　PH 与 CAGA 的比较**

| 例子 | $H^*$ | CAGA 计算结果 | | | PH 计算结果 | | |
|---|---|---|---|---|---|---|---|
| | | *H* | *Gap*/ % | *RT*/ s | *H* | *Gap*/ % | *RT*/ s |
| SCPL1 | 48.08 | 52 | 8.153 | 183.5 | 52 | 8.153 | 213 |
| SCPL2 | 150.157 | 155 | 3.225 | 463.5 | **153** | 1.893 | 108 |
| SCPL3 | 155.836 | 166 | 6.522 | 321 | **164** | 5.239 | 158 |
| SCPL4 | 58.263 | 61 | 4.698 | 684.5 | **60** | 2.981 | 136 |
| SCPL5 | 128.885 | 133 | 3.193 | 480 | **132** | 2.417 | 117 |
| SCPL6 | 18.728 | 21 | 12.132 | 234 | 23 | 22.811 | 72 |
| SCPL7 | 121 | 121 | 0 | 0 | **103** | - 14.876 | 606 |
| SCPL8 | 63.419 | 66 | 4.07 | 900 | 68 | 7.223 | 955 |
| SCPL9 | 89.96 | 95 | 5.602 | 507 | **93** | 3.379 | 185 |
| 平均 | 92.703 | 96.67 | 5.28 | 419.28 | **94.22** | **4.358** | **283.33** |

### 4.3　对随机生成例子的计算

为了测试 PH 计算问题规模比较大例子的计算能力,我们还随机生成了 10 个例子,这些例子的规模从 313～418,测试数据可以从 ftp://cai:cai@210.34.19.198/张德富/packing.txt 上查阅,计算结果见表 4. 从表 4 可以看出,PH 的平均 *Gap* 为 0.654,特别是找到了例子 RPP3 的最优高度. 平均运行时间 *RT* 为 951.9. 计算结果表明,PH 具有解

决规模比较大问题的能力.

表 4 PH对随机生成例子的计算结果

| | n | W | H* | H | Gap/ % | RT/ s |
|---|---|---|---|---|---|---|
| RPP1 | 346 | 124 | 139 | 140 | 0.719 | 714 |
| RPP2 | 406 | 148 | 131 | 132 | 0.763 | 1297 |
| RPP3 | 313 | 122 | 125 | 125 | 0 | 417 |
| RPP4 | 403 | 143 | 139 | 140 | 0.719 | 1536 |
| RPP5 | 327 | 126 | 136 | 137 | 0.735 | 651 |
| RPP6 | 418 | 139 | 146 | 147 | 0.685 | 1288 |
| RPP7 | 381 | 125 | 138 | 139 | 0.725 | 795 |
| RPP8 | 390 | 140 | 126 | 127 | 0.794 | 1053 |
| RPP9 | 350 | 129 | 144 | 145 | 0.694 | 907 |
| RPP10 | 375 | 123 | 141 | 142 | 0.709 | 861 |

## 5 结 论

基于国际上公认的 Benchmark 例的实验结果表明,我们提出的砌墙式启发式算法不仅简单,而且是快速和高效的,因此本文算法可以用来解决大量的实际问题,特别是规模比较大的问题,具有重大的实际应用价值.

本文算法也很容易推广应用到其他的填充布局问题,例如三维的矩形 Packing 问题等,还可以在许多工程实际问题中得到广泛地应用,例如服装的剪裁,板材的切割以及集装箱的装运等等. 在将来的研究中,我们还希望通过结合现代启发式方法,进一步改进砌墙式启发式算法以及开发出能够实际应用的软件.

**致 谢** 在此,我们向对本文的工作给予支持和建议的匿名评审老师以及国防科技大学陈火旺院士、华中科技大学的黄文奇教授表示感谢!

## 参 考 文 献

[1] Lodi A, Martello S, Monaci M. Two-dimensional Packing problems: A survey. European Journal of Operational Research, 2002, 141(2): 241-252

[2] Hochbaum D S, Wolfgang M. Approximation schemes for covering and Packing problems in image processing and VLSI. Journal of the Association for Computing Machinery, 1985, 32(1): 130-136

[3] Leung J, Tam T, Wong C S, Gilbert Young, Francis Chin. Packing squares into square. Journal of Parallel and Distributed Computing, 1990, 10(3): 271-275

[4] Huang Wen-Qi, Xu Ru-Chu. Two personification strategies for solving circles Packing problem. Science in China (Series E), 1999, 29(4): 347-353(in Chinese)
(黄文奇,许如初. 支持求解圆形 Packing 问题的两个拟人策略. 中国科学(E辑), 1999, 29(4): 347-353)

[5] Wang Jin-Min, Chen Dong-Xiang, Ma Feng-Ning, Cha Jian-Zhong. A simulated annealing Packing algorithm. Journal of Computer-Aided Design & Computer Graphics, 1998, 10(3): 253-259(in Chinese)
(王金敏,陈东祥,马丰宁,查建中. 布局问题的模拟退火算法. 计算机辅助设计与图形学学报, 1998, 10(3): 253-259)

[6] Liu D, Teng H. An improved BL-algorithm for genetic algorithm of the orthogonal Packing of rectangles. European Journal of Operational Research, 1999, 112(2): 413-419

[7] Dong She-Qin, Hong Xian-Long, Wu Yu-Liang et al. Deterministic VLSI block placement algorithm using less flexibility first principle. Journal of Computer Science and Technology, 2003, 18(6): 739-746

[8] Zhang De-Fu, Huang Wen-Qi. A simulated annealing algorithm for the circles Packing problem// Proceedings of the 4th International Conference on Computational Science. Krakow, Poland, 2004: 206-214

[9] Zhang De-fu, Deng An-Sheng. An effective hybrid algorithm for the problem of Packing circles into a larger containing circle. Computers & Operations Research, 2005, 32(8): 1941-1951

[10] Beasley J E. An exact two-dimensional non-guillotine cutting tree search procedure. Operations Research, 1985, 33(1): 49-64

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

[12] Dagli C H, Poshyanonda P. New approaches to nesting rectangular patterns. Journal of Intelligent Manufacturing, 1997, 8(3): 177-190

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

[14] Wu Yu-Liang, Huang Wen-Qi, Lau Siu-Chung, Wong C K, Young G H. An effective quasi-human based heuristic for solving the rectangle Packing problem. European Journal of Operational Research, 2002, 141(2): 341-358

[15] Hifi M, Hallah R. A hybrid algorithm for the two-dimensional layout problem: The cases of regular and irregular shapes. International Transactions in Operational Research, 2003, 10(3): 195-216

[16] Zhang De-Fu, Deng An-Sheng, Kang Yan. A hybrid heuristic algorithm for the rectangular Packing problem// Proceedings of the International Conference on Computational Science. Atlanta, GA, USA, 2005: 783-791

**ZHANG De-Fu**, born in 1971, Ph.D., associate professor. His research interests include computational intelligence and data mining.

**HAN Shui-Hua**, born in 1970, Ph.D., associate professor. His research interests include information computing and operations research.

**YE Wei-Guo**, born in 1970, Ph.D.. His research interests include algorithm design and management engineering.

**Background**

The strip Packing problem with rotation constraint belongs to NP hard problem. At present, some excellent algorithms have been presented to solve the problem. The bricklaying heuristic algorithm based on the reference line is novel and efficient. The computational results have shown that this algorithm not only runs in shorter time than known metaheuristic but also finds shorter height. The work is supported by the National Natural Science Foundation of China (grant No. 60773126) and the Province Nature Science Foundation of Fujian (grant No. A0710023) and academician startup fund (grant No. X01109) and 985 information technology fund (grant No. 0000-X07204) in Xiamen University. The projects aim at designing highly efficient algorithms for NP hard problems, such as the Packing problem and timetabling problem, which are the most important part of automated Packing and timetabling software. The project team has designed some efficient algorithm for SAT problem, the circle Packing problem, the rectangle Packing problem and three dimensional Packing problem, some results has been published. The fundamental aim of this paper is to investigate two-dimensional Packing problems and to develop state-of-the-art algorithms that could be used in an industrial setting. The paper belongs to one of the key parts of the project.