求解三维装箱问题的混合模拟退火算法

张德富[1),2)] 彭 煜[1)] 朱文兴[3)] 陈火旺[2),4)]

1)(厦门大学计算机科学系 福建 厦门 361005)
2)(东南融通博士后工作站 福建 厦门 361005)
3)(福州大学离散数学与理论计算机科学研究中心 福州 350002)
4)(国防科学技术大学计算机学院 长沙 410073)

摘 要 提出了一个高效求解三维装箱问题(Three Dimensional Container Loading Problem 3D-CLP)的混合模拟退火算法.三维装箱问题要求装载给定箱子集合的一个子集到容器中,使得被装载的箱子总体积最大.文中介绍的混合模拟退火算法基于三个重要算法:(1)复合块生成算法,与传统算法不同的是文中提出的复合块不只包含单一种类的箱子,而是可以在一定的限制条件下包含任意种类的箱子.(2)基础启发式算法,该算法基于块装载,可以按照指定装载序列生成放置方案.(3)模拟退火算法,以复合块生成和基础启发式算法为基础,将装载序列作为可行放置方案的编码,在编码空间中采用模拟退火算法进行搜索以寻找问题的近似最优解.文中采用 1500 个弱异构和强异构的装箱问题数据对算法进行测试.实验结果表明,混合模拟退火算法的填充率超过了目前已知的优秀算法.

关键词 三维装箱;启发式算法;模拟退火
中图法分类号 TP301 **DOI** 号: 10.3724/SP.J.1016.2009.02147

A Hybrid Simulated Annealing Algorithm for the Three-Dimensional Packing Problem

ZHANG De-Fu[1),2)] PENG Yu[1)] ZHU Wen-Xing[3)] CHEN Huo-Wang[2),4)]

1)(*Department of Computer Science, Xiamen University, Xiamen, Fujian* 361005)
2)(*Longtop Group Post-doctoral Research Center, Xiamen, Fujian* 361005)
3)(*Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou* 350002)
4)(*School of Computer, National University of Defense Technology, Changsha* 410073)

Abstract This paper presents an efficient hybrid simulated annealing algorithm for three dimensional container loading problem (3D-CLP). The 3D-CLP is the problem of loading a subset of a given set of rectangular boxes into a rectangular container so that the stowed volume is maximized. The algorithm introduced in this paper is based on three important algorithms. First, complex block generating, complex block can contain any number boxes of different types, which differs from the traditional algorithm. Second, basic heuristic, which is a new construction heuristic algorithm used to generate a feasible packing solution from a packing sequence. Third, simulated annealing algorithm, based on the complex block and basic heuristic, it encodes a feasible packing solution as a packing sequence, and searches in the encoding space to find an approximated optimal solution. 1500 benchmark instances with weakly and strongly heterogeneous boxes are considered in this paper. The computational results show that the volume utilization of hybrid algorithm outperforms current excellent algorithms for the considered problem.

Keywords three-dimensional container loading; heuristic algorithm; simulated annealing algorithm

收稿日期: 2007-12-05; 最终修改稿收到日期: 2009-10-14.张德富, 男, 1971 年生, 博士, 副教授, 主要研究方向为计算智能、金融数据挖掘. E-mail: dfzhang@xmu.edu.cn. 彭 煜, 男, 1984 年生, 博士研究生, 主要研究方向为计算智能. 朱文兴, 男, 1968 年生, 教授, 博士生导师, 主要研究领域为计算智能以及组合优化. 陈火旺, 男, 1936 年生, 博士生导师, 中国工程院院士, 主要研究领域为人工智能.

1 引 言

装箱问题在切割加工业和运输业中有许多应用.高利用率的切割和装载可以节约相当大的成本.一个好的装箱问题求解算法在减少损耗,节约天然资源方面起到重要的作用.实际应用中的装箱问题有不同的优化目标和装载约束,这导致了不同种类的装箱问题.Dyckhoff 和 Finke[1] 概述了不同类型的装箱问题及相关的切割问题.本文所处理的三维装箱问题属于装箱问题中的一类,可以形式化定义如下:

给定一个容器(其体积为 V)和一系列待装载的箱子,容器和箱子的形状都是长方体.问题的目标是要确定一个可行的箱子放置方案使得在满足给定装载约束的情况下,容器中包含的箱子总体积 S 尽可能的大,即填充率尽可能的大,这里填充率指的是 S/V *100%.可行放置方案要求放置满足如下 3 个条件:

(1) 被装载的箱子必须完全被包含在容器中.

(2) 任何两个被装载的箱子不能互相重叠.

(3) 所有被装载的箱子以与容器平行的方式放置.即不能斜放.

在实际应用中,特定的装箱问题有很多约束,本文仅考虑以下两个约束:

(C1)方向约束

在许多应用中,箱子的装载有方向约束.也就是说,每个箱子只有它的 1 条或几条边可以竖直放置作为高度.没有此约束的问题可以被简单地视为所有箱子的 3 条边都可以作为高度.

(C2)稳定性约束

在许多应用中,特别是在交通运输业中,装载必须满足稳定性约束.这意味着每个被装载的箱子必须得到容器底部或者其它箱子的支撑.也就是说,稳定性约束禁止被装载的箱子悬空.

也就是说,本文的放置方案必须满足上述两个约束条件.

所有 3 条边相同并且方向约束也相同的箱子被视为同一类型.只有一种箱子类型的装箱问题被称为同构装箱问题.有少数几种箱子类型且每种类型数量较多的装箱问题被称为弱异构装箱问题.强异构装箱问题则有许多箱子类型且每种类型数量较少.本文主要考虑异构装箱问题.

装箱问题是 NP-Hard 问题[2],采用装箱问题的精确求解算法是不现实的,因此启发式求解方法成为理论研究和实际应用的首选.近年来,涌现出许多解决装箱问题的算法.对于二维装箱问题,已经有一些不错的启发式求解算法[3-6].对于三维装箱问题,Ngoi[7] 以及 Bischoff[8] 等提出了一些启发式算法,并产生了一些三维装箱实例[8-9].而基于图搜索的算法可以参考文献[10-11].Gehring[12] 等报告了遗传算法在装箱问题中的应用.Bortfeldt[13-14] 等分别提出了禁忌搜索以及混合遗传求解算法.并行化的遗传算法则由 Gehring[15] 等提出,Bortfeldt[16] 等提出了并行化的禁忌搜索算法.Lim 等提出了一个新的启发式算法[17].Moura 等提出了一个随机自适应启发式算法 GRASP[18].Bischoff[19] 基于一个寻找最优参数设置的搜索算法,提出了一个新的启发式算法.Bortfeldt[20] 提出了基于分支定界算法的一个启发式算法.国内学者对三维装箱问题的研究,也取得了一些不错的结果[21-25],特别是黄文奇教授等提出了一个有效的求解一类装箱问题的算法[25].对上述文献进行研究分析可知,启发式算法是解决三维装箱问题的有效方法,而且二维装箱问题的有效算法能提高装箱问题解的质量[11, 15, 24].文献[24]采用了一个类似于二维装箱问题的求解思路,即首先提出一个好的启发式算法,得到一个解,然后用现代启发式算法对已有解进行逐步改进,取得了不错的效果.一些研究表明,将构造型启发式算法与模拟退火算法结合,是有效求解装箱问题的思路之一[24, 26-27].本文首先针对三维装箱问题的块装载算法提出了复合块的概念,接着给出一个基于块装载的构造型启发式算法,然后将该启发式算法和模拟退火算法相结合,提出一个有效的三维装箱问题的混合求解算法.对于 1500 个弱异构和强异构的装箱问题测试的计算结果表明,混合算法是相当有效的,对于本文所考虑的问题,混合算法对所有测试样例的填充率都超过了目前已知的优秀算法.

2 基于块装载的基础启发式方法

基础启发式方法基于块装载思想,将作为混合模拟退火算法的映射函数,此方法包括两个最主要的部分:

(1)块生成算法.不同于其它的块生成方法,本文采用的块不仅仅是由同一朝向的同种箱子构成的简单块,还包括由多种箱子构成的复合块.复合块包含一种或多种箱子,箱子的朝向也可以不同.同时,

为了能容纳不同种类的箱子, 复合块可以包含少量不影响装载的空隙. 复合块的引入使得每次装载可选择的块数目增多, 也使每次装载的块的箱子体积和增大, 加快了装载的速度, 也极大提升了算法的效率.

(2) 启发式装载算法. 该算法接受一个装载序列作为输入, 用以指导装载过程中的块选择. 装载序列是一个向量, 它的每一个元素对应装载阶段的一个选择. 具体地说, 基础启发式算法在每个装载阶段按照当前剩余空间计算出按箱子体积降序排列的可行块列表, 然后按照装载序列来选择采用的装载块, 然后将未填充空间重新切割以便下一步装载. 通过这种方式, 我们可以建立装载序列和放置方案之间的映射, 进而使用模拟退火算法进行放置方案的优化.

2.1　数据结构

为方便下文描述, 先介绍一下本文采用的各种数据结构.

2.1.1　基本概念、剩余空间、箱子以及问题实例描述

在装箱问题中, 由于所有的物体都是与坐标轴平行的长方体, 对它们描述都是通过参考点和各个维度上的边长来指定. 所谓参考点就是, 一个物体的左后下角, 也就是其 x, y, z 均最小的点.

本文中的装载是在剩余空间中进行的. 剩余空间是容器中的未填充长方体空间, 其中 x, y, z 描述了它的参考点, lx, ly, lz 描述了它在 3 个维度上的长度.

箱子是问题中被装载的物体, 我们用 lx, ly, lz 三个域来描述它 3 条边的长度. 在这里, 不同朝向的相同箱子被当作不同箱子处理, 箱子有一个域 *type* 指定了箱子的真实类型.

现在我们可以给出问题实例的描述, 三元组 (*container*, *boxList*, *num*) 描述了一个三维装箱问题实例, 其中 *container* 是初始剩余空间, *boxList* 是一个箱子向量, 指定可用于装载的箱子, *num* 则是一个整数向量, 描述了每一种类型箱子的数目.

2.1.2　块和块表

块是基于块装载的启发式算法中装载的最小单位, 它是包含许多箱子的长方体. 块结构中的每个箱子的摆放都满足约束 C1, 而且除了最底部的箱子外都满足 C2. 块结构用 lx, ly, lz 描述三条边的长, *volume* 描述其中箱子的总体积, 整数向量 *require* 描述了块对各种类型箱子的需求数.

由于块中有空隙, 块的顶部有一部分可能由于失去支撑而不能继续放置其它块, 我们通过可行放置矩形来描述块的顶部可以继续放置其它块的矩形区域. 这里, 我们仅考虑包括块顶部左后上角的可行放置矩形, 以块结构的域 ax, ay 表示其长宽.

块表 *blockTable* 是预先生成的按块体积降序排列的所有可能块的列表, 用于迅速生成指定剩余空间的可行块列表. 同时, 块表将块生成算法与装载算法分开, 使得更换块生成算法变得容易.

2.1.3　剩余空间堆栈和剩余箱子

基础启发式中, 剩余空间被组织成堆栈. 算法基本过程可描述为: 从栈顶取一个剩余空间, 若有可行块, 按照装载序列选择一个块放置在该空间, 将未填充空间切割成新的剩余空间加入堆栈, 若无则抛弃此块, 如此反复直至堆栈为空. 在此过程中, *spaceStack* 表示剩余空间堆栈, 整数向量 *avail* 记录各种剩余箱子的数目.

2.1.4　装载序列

装载序列是一个向量, 它的每一个元素对应装载阶段的一个选择. 具体地说, 假定有装载序列 *ps*, *ps*[*k*] 表示在第 *k* 装载阶段应该选择的块号. 有时迭代次数可能会超过装载序列的长度. 在这种情况下, 我们可以延长装载序列, 即通过在装载序列的最后添零的方法来解决.

2.1.5　放　置

一个放置是一个剩余空间和块的二元组. 例如 (*space*, *block*) 表示将块 *block* 放置在剩余空间 *space* 上. 放置是通过将块的参考点和剩余空间的参考点重合得到的. 由于我们的算法保证剩余空间总是被支撑的, 而块中除了底部箱子, 所有的箱子都满足约束 C1 和 C2, 所以每个放置都是满足约束 C1 和 C2 的.

2.1.6　放置方案

放置方案是满足约束的放置的序列. 它包括一个域 *volume*, 表示被装载箱子的总体积. 而所谓满足约束是指放置方案中的放置不会互相冲突, 也即是说放置方案是问题的一个可行解. 基础启发式表达的就是一个从装载序列到放置方案的一个映射.

2.2　块生成算法

2.2.1　简单块生成

简单块是由同一朝向的同种类型的箱子堆叠而成的, 箱子和箱子之间没有空隙, 堆叠的结果必须正好形成一个长方体. 图 1 展示了两个简单块的例子, 其中 nx, ny, nz 表示在每个维度上的箱子数. $nx \times ny \times nz$ 则是简单块所需要的总箱子数.

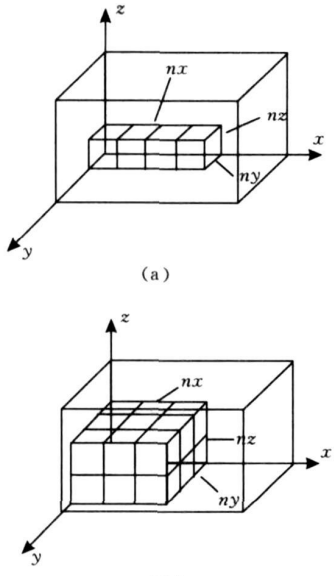

图 1 简单块

简单块的生成算法枚举所有合法的组合(nx, ny, nz), 并将其对应的简单块加入块表, 图 2 描述了此算法.

```
algorithm GenSimpleBlock
input (container, boxList, num)
blockTable := {}
for each box in boxList do
  for nx := 1 to num[box.type] do
    for ny := 1 to num[box.type]/nx do
      for nz := 1 to num[box.type]/nx/ny do
        if (box.lx *nx ≤container.lx
          and box.ly *ny ≤container.ly
          and box.lz *nz ≤container.lz) then
          add block which have nx, ny, nz box on three dimension
          respectively to blockTable.
        endif
      endfor
    endfor
  endfor
endfor
sort blockTable into nonincreasing order by volume of blocks.
```

图 2 简单块生成算法

2 2 2 复合块生成

复合块是通过不断复合简单块而得到的, 我们定义复合块如下:

(1)简单块是最基本的复合块.

(2)有两个复合块 a, b. 可以按 3 种方式进行复合得到复合块 c: 按 x 轴方向复合, 按 y 轴方向复合, 按 z 轴方向复合. c 的大小是包含 a, b 的最小长方体.

图 3 展示了 3 种复合方式, 其中的虚线描绘的是新复合块的范围.

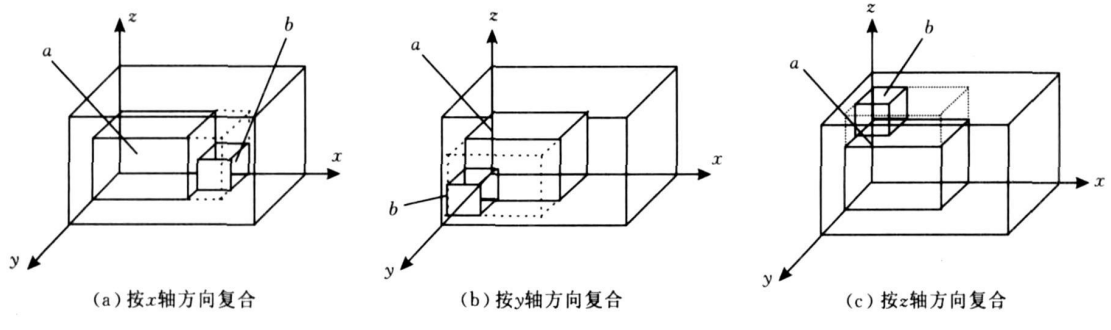

图 3 复合块

显然, 按照前面的定义, 复合块的数量将是箱子数目的指数级, 而且生成块中可能有很多未利用空间, 非常不利于装载. 所以我们对复合块施加一定的限制是必要的, 生成的复合块必须满足下面 7 个条件:

(1) 复合块的大小不大于容器的大小.

(2) 复合块中可以有空隙, 但它的填充率至少要达到 *MinFillRate*.

(3) 受复合块中空隙的影响, 复合块顶部有支撑的可行放置矩形可能很小, 为了进一步的装载, 我们限定可行放置矩形与相应的复合块顶部面积的比至少要达到 *MinAreaRate*.

(4) 为控制复合块的复杂程度, 定义复合块的复杂度如下: 简单块的复杂度为 0, 其它复合块的复杂度为其子块的复杂度的最大值加 1. 块的 *level* 域描述了复合块的复杂度, 我们限制生成块的最大复杂度为 *MaxLevel*.

(5)按 x 轴方向、按 y 轴方向复合的时候, 子块要保证顶部可行放置矩形也能进行复合. 在按 z 轴方向复合时, 子块要保证复合满足约束 C2. 具体的复合要求和结果如表 1 所示.

(6)拥有相同 3 边长度、箱子需求和顶部可行放置矩形的复合块被视为等价块, 重复生成的等价块将被忽略.

(7) 在满足以上约束的情况下, 块数目仍然可能会很大, 我们的生成算法将在块数目达到 *MaxBlocks* 时停止生成.

表 1　复合块的顶部可行放置矩形条件和结果

复合的方向	子块 a, b 满足的条件	复合块 c
按 x 轴方向	$a.ax=a.lx$ $b.ax=b.lx$ $a.lz=b.lz$	$c.ax=a.ax+b.ax$ $c.ay=\min(a.ay, b.ay)$
按 y 轴方向	$a.ay=a.ly$ $b.ay=b.ly$ $a.lz=b.lz$	$c.ax=\min(a.ax, b.ax)$ $c.ay=a.ay+b.ay$
按 z 轴方向	$a.ax\geq b.lx$ $a.ay\geq b.ly$	$c.ax=b.ax$ $c.ay=b.ay$

根据复合块的定义, 可以得到图 4 所示的复合块生成算法.

```
algorithm GenComplexBlock
input (container, boxList, num)
GenSimpleBlock(container, boxList, num)
for level := 1 to MaxLevel do
  newBlockTable := {}
  for each a, b in blockTable do
    if a.level = level or b.level = level then
      if a and b satisfied the constrain in x-direction then
        add the block combined by a and b in x-direction to
            newBlockTable
      endif
      if a and b satisfied the constrain in y-direction then
        add the block combined by a and b in y-direction to
            newBlockTable
      endif
      if a and b satisfied the constrain in z-direction then
        add the block combined by a and b in z-direction to
            newBlockTable
      endif
    endif
  endfor
  blockTable := blockTable+newBlockTable
  reduce duplicated block in blockTable
endfor
sort blockTable into nonincreasing order by volume of blocks.
```

图 4　复合块生成算法

2.3　启发式装载

2.3.1　启发式算法

本小节详细描述了基于块装载的构造型启发式算法. 该算法的思想类似于二维装箱问题的递归启发式算法[3], 是对 Bortfeld 和 Gehring[16] 中提到的基础启发式算法的改进. 基于块装载的构造型启发式算法维护一个剩余空间堆栈, 自顶向下地装载剩余空间. 在算法开始时, 容器是列表中唯一的剩余空间. 在装载过程中, 栈顶剩余空间被取出, 在有可行块时选择装载序列指定的块与剩余空间结合生成一个放置, 并将未填充空间切割成新的剩余空间加入堆栈, 在无可行块的情况下则抛弃该空间并试图将此空间中的可转移空间并入新的栈顶元素. 不断重复上述过程, 直到堆栈为空. 完整的算法如图 5 所述.

```
algorithm BasicHeuristic
input (ps, container, num)
avail := num
plan := {}
plan.volume = 0
spaceStack := {}
spaceStack.push(container)
index := 0
while spaceStack ≠ {} do
  space := spaceStack.pop()
  blockList := genBlockList(space, avail)
  if blockList ≠ {} then
    block := blockList[ps[index]]
    index := index+1
    avail := avail− block.require
    plan := plan+(space, block)
    plan.volume := plan.volume+block.volume
    spaceStack.push(genResidulSpace(space, block))
  else then
    transferSpace(space, spaceStack)
  endif
endwhile
return plan
```

图 5　基础启发式算法

2.3.2　可行块生成

可行块生成算法 $genBlockList(space, avail)$ 用于从 $blockTable$ 中获取适合当前剩余空间的可行块列表. 该算法扫描 $blockTable$, 返回所有能放入剩余空间 $space$ 并且 $avail$ 有足够剩余箱子满足 $require$ 的块. 由于 $blockTable$ 是按块中箱子总体积降序排列的, 返回的 $blockList$ 也是按箱子总体积降序排列的.

2.3.3　剩余空间生成和剩余空间转移

在每个装载阶段一个剩余空间被装载, 装载分为两种情况: 有可行块, 无可行块. 在有可行块时, 算法按照装载序列 ps 选择可行块, 然后将未填充空间切割成新的剩余空间. 此时如果 ps 指定的块号大于可行块表的长度, 则指定块号对块表长度取模. 在无可行块时, 当前剩余空间被抛弃, 若其中的一部分空间还可以利用则进行空间转移.

剩余空间装载一个块以后的状态如图 6 所示. 未填充空间将被分成 3 个剩余空间. 块顶部的可放置矩形确定了一个新的剩余空间 $spaceZ$, 其余的两个剩余空间为 x 轴方向上的 $spaceX$ 和 y 轴方向上的 $spaceY$. 图 6(a)和图 6(b)展示了两种分割方式, 它们的区别在于图 6(c)所示的可转移空间的归属. 我们希望切割出的剩余空间尽可能的大, 而衡量空间大小的方法有很多, 这里使用剩余空间在放置块以后在 x 轴和 y 轴上的剩余长度 mx、my 作为度量, 将可转移空间分给剩余量较大的方向上的新剩

余空间. *genResidualSpace*(*space*, *block*)执行剩余空间的切割, 其返回的剩余空间顺序为(*spaceZ*, *spaceX*, *spaceY*)或(*spaceZ*, *spaceY*, *spaceX*), 保证最后入栈的是包含可转移空间的剩余空间. 这样, 由于包含可转移空间的剩余空间后入栈, 其必然被先装载. 在无可行块的情况下, 可转移空间可以被转移给新的栈顶剩余空间以重新利用. *transferSpace*(*space*, *spaceStack*)判定当前剩余空间是否有可转移空间, 如果有则转移给 *spaceStack* 的新栈顶剩余空间.

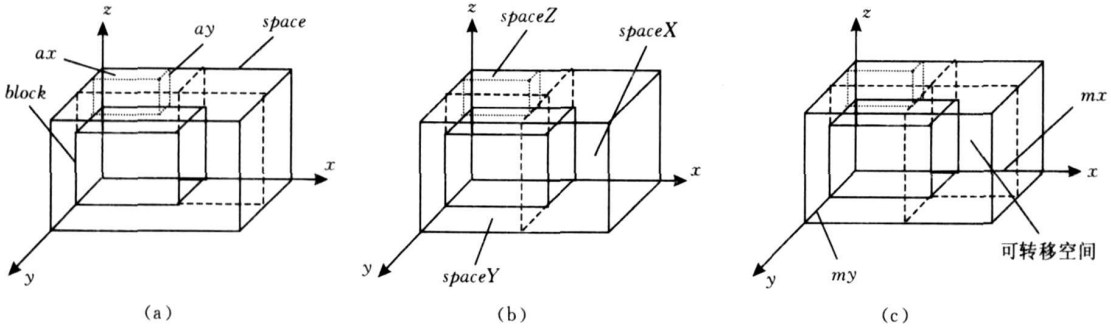

图 6 剩余空间切割和转移

3 混合模拟退火算法

在本节中, 我们介绍如何将可行放置方案编码为一个装载序列; 接着以这种编码方式为基础, 提出三维装箱问题的混合模拟退火算法; 在本节的最后, 我们介绍如何利用并行化技术使得搜索更多样化.

3.1 模拟退火算法的邻域选择

在第 2 节, 我们提出了一个基于块装载的基础启发式算法用以从一个装载序列产生一个可行的放置方案, 装载序列的一个元素对应启发式算法的一次选择. 因此, 使用基础启发式算法作为映射函数, 装载序列可以当作是放置方案的一种编码, 对放置方案空间的搜索可以用对装载序列空间的搜索来近似.

模拟退火算法是一种全局优化算法, 在局部搜索的过程中引进了随机扰动的机制. 模拟退火算法利用一个概率机制来控制解的接受(或更新)过程. 对于好的解, 无条件接受, 对于差的解以一定的概率接受. 这就使得模拟退火算法有更多的机会逃离局部最优的"陷阱", 在一定程度上避免了其它局部搜索算法或梯度算法存在的缺陷, 真正显示了随机算法的优势. 影响模拟退火算法性能的因素主要包括邻域构造以及参数设置, 关于模拟退火算法及其参数设置可参考文献[28-29].

我们在装载序列的空间上应用模拟退火算法进行搜索. 模拟退火算法的邻域设计如下: 给定一个装载序列 ps, 邻域 $N(ps)$包括所有满足下列条件的装载序列 ps': ps'与 ps 有相同的长度, 它们除了第 j 元素不同外, 其它所有的元素都相同. 实验表明在大多数的实例中选择排序靠前的块将取得较好的结果, 算法限制每个阶段的块选择数最大为 *MaxSelect*.

3.2 混合模拟退火算法

完整的装箱问题的混合模拟退火求解算法如图 7 所示.

```
algorithm SA
input(ts, tf, dt, length, isLinear, isComplex)
if isComplex then
  genSimpleBlock()
else then
  genComplexBlock()
endif
ps := zero vector with size MaxSeq
plan := basicHeuristic(ps)
best := plan
t := ts
while t > tf do
  for i := 1 to length do
    k := random[0, ps.size-1]
    ps' := ps
    ps'[k] := random[0, MaxSelect-1]
    plan' := basicHeuristic(ps')
    if plan'.volume > plan.volume then
      ps := ps', plan := plan'
    else if random(0, 1) <
          exp((plan'.volume-plan.volume)/t) then
      ps := ps', plan := plan'
    endif
    if plan.volume > best.volume then
      best = plan
    endif
  endfor
  if isLinear then
    t := dt * t
  else then
    t := (1-t*dt) * t
  endif
endwhile
return best
```

图 7 混合模拟退火算法

在搜索开始时, 算法使用一个长度为 *MaxSeq* 的零向量作为初始装载序列, 也就是说在初始状态下每次选择体积最大的块. *MaxSeq* 是一个经验性最大装载序列长度. 算法中的随机函数有两种用法, *random*[*a*, *b*] 返回[*a*, *b*] 中的随机整数, *random*(*a*, *b*) 则返回(*a*, *b*)中的随机实数. 控制变量 *ts*, *tf*, *dt*, *length*, *isLinear* 和 *isComplex* 作为算法的输入以增加算法的灵活性. 其中 *ts*, *tf*, *dt* 和 *length* 用于控制模拟退火过程: *ts* 是开始温度, *tf* 是终止温度, *dt* 是退火系数, *length* 是马尔可夫链长. 混合模拟退火算法采用两种降温方式, 输入变量 *isLinear* 控制是采用线性降温还是非线性降温. 虽然算法采用复合块在大多数情况下优于简单块, 但在一些实例上采用简单块能取得更好的结果. 为了保证算法的多样性, 我们采用变量 *isComplex* 控制采用简单块或复合块.

3 3　并行化和多样化

尽管采用模拟退火算法增强了全局搜索能力, 但是在运行过程中算法仍然可能陷入局部最优解, 控制变量 *ts*, *tf*, *dt*, *length*, *isLinear* 和 *isComplex* 取不同的值将会极大地影响计算结果的质量. 实验结果表明, 不同的参数设置能使混合模拟退火算法在不同的问题实例上表现优秀. 因此, 我们采用并行化技术以不同参数设置同时运行算法使得搜索更多样化. 在本文的实验中, 我们采用 20 组参数进行并行计算, 具体的参数如表 2 所示.

表 2　参数表

ts	*tf*	*dt*	*length*	*isLinear*	*isComplex*
1	0 01	0 95	200	True	False
1	0 01	0 96	200	True	False
1	0 01	0 97	200	True	False
1	0 01	0 98	200	True	False
1	0 01	0 99	200	True	False
1	0 01	0 40	200	False	False
1	0 02	0 40	200	False	False
1	0 01	0 20	200	False	False
1	0 02	0 20	200	False	False
1	0 03	0 20	200	False	False
1	0 01	0 95	200	True	True
1	0 01	0 96	200	True	True
1	0 01	0 97	200	True	True
1	0 01	0 98	200	True	True
1	0 01	0 99	200	True	True
1	0 01	0 40	200	False	True
1	0 02	0 40	200	False	True
1	0 01	0 20	200	False	True
1	0 02	0 20	200	False	True
1	0 03	0 20	200	False	True

4　计算结果

混合模拟退火算法以 C＋＋实现, 实验程序运行在 Intel Core2 Duo 2 0 GHz 的处理器上. 实验中采用的各种参数设置如表 3 所示.

表 3　实验参数

MinFillRate	*MinAreaRate*	*MaxLevel*	*MaxBlocks*	*MaxSeq*	*MaxSelect*
0 98	0 96	5	10000	200	47

4 1　测试数据

本文实验采用的测试数据来自文献[8-9], 包括 1500 个三维装箱问题, 这些数据可以从 OR-Library[30] 或者 http://59. 77. 16 229/Download.aspx #p4 网站上下载. 这些问题共分为 15 类, 每类 100 个问题, 这些问题有相同数目的箱子类型. BR1 ~ BR7 为弱异构问题, BR8 ~ BR15 为强异构问题. 15 类问题中箱子的类型数从 3 ~ 100, 异构性由弱到强, 能够很好地反映算法在不同异构性装箱问题中的表现.

4 2　比较实验

对于来自文献[8-9] 的 1500 个测试问题, 许多研究者对它们做过测试. 作为比较, 本文包括了 Bischoff 等在文献[8] 提出的启发式算法 H_BR, Gehring 等在文献[12] 提出的 GA_GB 算法和 Bortfeldt 等[13] 提出的禁忌搜索算法 TS_BG, Bortfeldt 等[14] 提出的混合遗传算法 HGA_BG 以及并行遗传算法 PGA_GB[15], Bortfeldt 等[16] 提出的并行禁忌搜索算法 PTSA, Lim[17] 提出的 MFB 算法, Moura 等提出的 GRASP 算法[18], Bischoff[19] 提出的新启发式算法 H_B 以及 Bortfeldt 等[20] 提出的启发式算法 SPBBL-CC4, 张德富等[24] 提出的组合启发式算法 CH 以及黄文奇等提出的 A_2 算法[25]. 这些算法有的只计算弱异构问题, 有的只计算强异构问题, 它们的计算结果直接来自于文献. 表 4 给出了这些算法与混合模拟退火算法(HSA)对弱异构问题的计算结果, 表 5 给出了这些算法与 HSA 对强异构问题的计算结果. 表中所有数据表示填充率(%), 它表示不同算法针对一类问题所得到的平均填充率, 而 Mean 表示算法对每类问题所得到的平均填充率.

表 4 和表 5 中所有算法的装箱结果都满足 C1 约束, 此外 H_BR、GA_GB、TS_BG 和 HSA 算法的装箱结果都是基于完全支撑的, 也就是满足 C2 约束. 问题满足的约束条件越多, 求解起来更困难. 但

是,对于弱异构问题,HSA 在每个样例的填充率都超过了目前已知的算法 H_BR、GA_GB、TS_BG、PTSA、MFB、H_B、SPBBL-CC4 和 CH,HSA 的平均填充率更是达到 93.30%.对于强异构问题,HSA 在每个样例的填充率都超过了目前已知的算法 H_BR、GA_GB、TS_BG、HGA_BG、PGA_GB 和 GRASP.对于最新发布的算法 A_2,HSA 算法在 BR8~BR11 类问题上,填充率超过 A_2,在 BR12~BR15 类问题上落后 A_2,但是,A_2 只计算了强异构问题,而且 HSA 算法的平均填充率超过 A_2,特别注意到 HSA 算法的装箱结果还满足约束 C2,就足以看出 HSA 算法的优越性.

从表 4 和表 5 可以看出,HSA 算法超过了目前已知的各种算法,由此可见,本文提出的 HSA 算法是非常有效的.

表 4 各种算法对弱异构问题的填充率比较

测试例	约束	填充率/%							
		BR1	BR2	BR3	BR4	BR5	BR6	BR7	平均
箱子的类型数		3	5	8	10	12	15	20	
H_BR[8]	C1 和 C2	83.79	84.44	83.94	83.71	83.80	82.44	82.01	83.50
GA_GB[12]	C1 和 C2	85.80	87.26	88.10	88.04	87.86	87.85	87.68	87.50
TS_BG[13]	C1 和 C2	87.81	89.40	90.48	90.63	90.73	92.72	90.65	90.10
PTSA[16]	C1	93.52	93.77	93.58	93.05	92.34	91.72	90.55	92.70
MFB[17]	C1	87.40	88.70	89.30	89.70	89.70	89.70	89.40	89.10
H_B[19]	C1	89.39	90.26	91.08	90.90	91.05	90.70	90.44	90.55
SPBBL-CC4[20]	C1	87.30	88.60	89.40	90.10	89.30	89.70	89.20	89.09
CH[24]	C1	89.94	91.13	92.09	91.94	91.72	91.45	90.94	91.32
HSA	C1 和 C2	**93.81**	**93.94**	**93.86**	**93.57**	**93.22**	**92.72**	**91.99**	**93.30**
拟人启发式[26]	C1	82.89	83.64	83.46	83.40	83.57	83.33	82.74	83.29
基础启发式	C1 和 C2	**85.58**	**84.60**	**85.14**	**84.78**	**85.15**	**84.20**	**83.86**	**84.76**

表 5 各种算法对强异构问题的填充率比较

测试例	约束	填充率/%								
		BR8	BR9	BR10	BR11	BR12	BR13	BR14	BR15	平均
箱子的类型数		30	40	50	60	70	80	90	100	
H_BR[8]	C1 和 C2	80.10	78.03	76.53	75.08	74.37	73.56	73.37	73.38	75.55
GA_GB[12]	C1 和 C2	87.52	86.46	85.53	84.82	84.25	83.67	82.99	82.47	84.71
TS_BG[13]	C1 和 C2	87.11	85.76	84.73	83.55	82.79	82.29	81.33	80.85	83.55
HGA_BG[14]	C1	89.73	89.06	88.40	87.53	86.94	86.25	85.55	85.23	87.34
PGA_GB[15]	C1	90.26	89.5	88.73	87.87	87.18	86.7	85.81	85.48	87.69
GRASP[18]	C1	86.13	85.08	84.21	83.98	83.64	83.54	83.25	83.21	84.13
A_2[25]	C1	88.41	88.14	87.9	87.88	**87.92**	**87.92**	**87.82**	**87.73**	87.97
HSA	C1 和 C2	**90.56**	**89.70**	**89.06**	**88.18**	87.73	86.97	86.16	85.44	**87.98**

此外,表 4 的最后两行还给出了在不采用模拟退火进行调整的情况下,组合启发式算法中的拟人启发式[26]与混合模拟退火算法中的基础启发式的填充率.它们均采用默认输入进行计算,其中拟人启发式的输入为体积从大到小排序的箱子序列,基础启发式的输入为长度为 *MaxSeq* 的零向量.值得注意的是,拟人启发式只考虑满足约束 C1,而基础启发式却要考虑满足约束 C1 和 C2,即使如此,由表 4 可以看出,基础启发式在每个实例上的计算结果都超过了拟人启发式,平均填充率高出了约 1.5%.而混合模拟退火在每个实例上的计算结果也都超过了组合启发式,平均填充率更是高出了约 2.0%,这充分显示了基础启发式算法的优势.由此可见,一个好的启发式算法对于提升基于邻域搜索的算法在三维装箱问题上的应用效果是很有帮助的.

表 6 给出了混合模拟退火算法运行的具体细节.其中 Minimum、Maximum 和 Average 分别表示模拟退火算法在同一类型的 100 个问题的运行中,所记录下的最短运行时间或最低填充率、最长运行时间或最高填充率以及平均时间或者平均填充率.表 6 中的时间是程序的实际运行时间,包括所有并行线程的运行时间.每个数据保留小数点后两位,运行时间用秒(s)表示.

从表 6 可知,随着箱子类型数的增加,算法的运行时间近似单调增加.其原因是当箱子种类增加,每个装载阶段可能地块增多,导致基础启发式算法的运行时间增多.而且随着箱子类型数的增加,算法的填充率近似地单调下降.这是因为随着箱子类型数增多,搜索空间迅速扩大,找到好放置方案变得更加困难.

表 6　混合模拟退火算法的一些测试细节

测试例	箱子类型	运行时间/s			填充率/%		
		Minimum	Maximum	Average	Minimum	Maximum	Average
BR1	3	6.02	74.39	20.33	86.62	97.57	93.80
BR2	5	12.13	99.97	35.68	88.03	97.28	93.94
BR3	8	17.67	209.63	59.00	90.45	96.52	93.86
BR4	10	20.64	224.39	75.05	90.74	95.72	93.57
BR5	12	30.28	216.25	80.63	90.95	95.02	93.22
BR6	15	26.45	244.53	88.89	90.86	95.19	92.72
BR7	20	42.56	267.58	101.52	89.93	93.78	91.99
BR8	30	144.51	448.41	261.87	88.61	92.34	90.56
BR9	40	142.50	608.34	276.12	86.62	91.51	89.70
BR10	50	160.65	640.53	288.90	85.50	90.68	89.06
BR11	60	171.81	536.10	287.25	85.08	90.54	88.18
BR12	70	184.23	596.34	306.36	85.06	90.17	87.73
BR13	80	206.16	549.75	307.68	84.04	88.72	86.97
BR14	90	206.28	471.36	305.82	83.78	88.43	86.16
BR15	100	202.08	486.84	301.26	82.71	87.25	85.44
Mean	**39.53**	**104.93**	**378.29**	**186.42**	**87.27**	**92.71**	**90.46**

5　结　论

本文提出了一个三维装箱问题的混合模拟退火算法. 算法先提出复合块的概念及其生成算法, 再引入一个基于块装载的基础启发式算法将可行放置方案编码成装载序列, 然后使用模拟退火算法在编码空间中搜索近似解, 接着通过并行化技术对算法进行进一步的优化. 对 1500 个测试问题的计算结果表明混合模拟退火算法的填充率超过了目前已知的优秀算法. 这表明允许少量空隙的复合块的应用对基于块装载的启发式算法的效果有很大改善, 而邻域搜索技术结合高效的启发式算法是一个很好的求解三维装箱问题的解决方案. 继续优化混合模拟退火算法, 测试更多不同类型的实例, 进一步提高计算的效率是将来的研究工作.

参　考　文　献

[1] Dyckhoff H, Finke U. Cutting and packing in production and distribution, physica, heidelberg, 1992

[2] Scheithauer G. Algorithms for the container loading problem//Operations Research Proceedings, Springer, Berlin, 1992: 445-452

[3] Zhang De-Fu, Kang Yan, Deng An-Sheng. A new heuristic recursive algorithm for the strip rectangular packing problem. Computers & Operations Research, 2006, 33(8): 2209-2217

[4] Cui Yao-Dong, He Dong-Li, Song Xiao-Xia. Generating optimal two-section cutting patterns for rectangular blanks. Computers & Operations Research, 2006, 33(6): 1505-1520

[5] Zhang De-Fu, Han Shui-Hua, Ye Wei-Guo. A bricklaying heuristic algorithm for the orthogonal rectangular packing problem. Chinese Journal of Computers, 2008, 23(3): 509-515(in Chinese)
(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, 23(3): 509-515)

[6] Huang Wen-Qi, Chen Duan-Bing, Xu Ru-Chu. A new heuristic algorithm for rectangle packing. Computers & Operations Research, 2007, 34(11): 3270-3280

[7] Ngoi B K A, Tay M L, Chua E S. Applying spatial representation techniques to the container packing problem. International Journal of Production Research, 1994, 32(1): 111-123

[8] Bischoff E E, Ratcliff B S W. Issues in the development of approaches to container loading. Omega, 1995, 23(4): 377-390

[9] Davies A P, Bischoff E E. Weight distribution considerations in container loading. European Journal of Operational Research, 1999, 114(3): 509-527

[10] Morabito R, Arenales M. An AND/OR-graph approach to the container loading problem. International Transactions in Operational Research, 1994, 1(1): 59-73

[11] Pisinger D. Heuristics for the container loading problem. European Journal of Operational Research, 2002, 141(2): 382-392

[12] Gehring H, Bortfeldt A. A genetic algorithm for solving the container loading problem. International Transactions in Operational Research, 1997, 4(5-6): 401-418

[13] Bortfeldt A, Gehring H. A tabu search algorithm for weakly heterogeneous container loading problems. OR Spectrum, 1998, 20(4): 237-250

[14] Bortfeldt A, Gehring H. A hybrid genetic algorithm for the container loading problem. European Journal of Operational Research, 2001, 131(1): 143-161

[15] Gehring H, Bortfeldt A. A parallel genetic algorithm for solving the container loading problem. International Transactions in Operational Research, 2002, 9(4): 497-511

[16] Bortfeldt A, Gehring H, Mack D. A parallel tabu search algorithm for solving the container loading problem. Parallel Computing, 2003, 29(5): 641-662

[17] Lim A, Rodrigues B, Yang Y. 3-D Container Packing Heuristics. Applied Intelligence, 2005, 22(2): 125-134

[18] Moura A, Oliveira J F. A GRASP approach to the container-loading problem. IEEE Intelligent Systems, 2005, 20(4): 50-57

[19] Bischoff E E. Three-dimensional packing of items with limited load bearing strength. European Journal of Operational Research, 2006, 168(3): 952-966

[20] Bortfeldt A, Mack D. A heuristic for the three-dimensional strip packing problem. European Journal of Operational Research, 2007, 183(3): 1267-1279

[21] He Da-Yong, Zha Jian-Zhong, Jiang Yi-Dong. Research on solution to complex container-loading problem based on genetic algorithm. Journal of Software, 2001, 12(9): 1380-1385(in Chinese)
(何大勇, 查建中, 姜义东. 遗传算法求解复杂集装箱装载问题方法研究. 软件学报, 2001, 12(9): 1380-1385)

[22] Lu Yi-Ping, Zha Jian-Zhong. Sequence triplet method for 3D rectangle packing problem. Journal of Software, 2002, 13(11): 2183-2187(in Chinese)
(陆一平, 查建中. 三维矩形块布局的序列三元组编码方法. 软件学报, 2002, 13(11): 2183-2187)

[23] Li Guang-Qiang, Teng Hong-Fei. Isomorphic and non-isomorphic layout patterns of packing problems. Chinese Journal of Computers, 2003, 26(10): 1248-1254(in Chinese)
(李广强, 滕弘飞. 装填布局的同构和非同构模式. 计算机学报, 2003, 10: 1248-1254)

[24] Zhang De-Fu, Wei Li-Jun, Chen Qing-Shan, Chen Huo-Wang. A combinational heuristic algorithm for the three-dimensional packing problem. Journal of Software, 2007, 18(9): 2083-2089(in Chinese)
(张德富, 魏丽军, 陈青山, 陈火旺. 三维装箱问题的组合启发式算法. 软件学报, 2007, 18(9): 2083-2089)

[25] Huang Wen-Qi, He Kun. A caving degree approach for the single container loading problem. European Journal of Operational Research, 2009, 196(1): 93-101

[26] Zhang De-Fu, Huang Wen-Qi, Wang Hou-Xiang. Personification annealing algorithm for solving SAT problem. Chinese Journal of Computers, 2002, 25(2): 148-152(in Chinese)
(张德富, 黄文奇, 汪厚祥. 求解SAT 问题的拟人退火算法. 计算机学报, 2002, 25(2): 148-152)

[27] Zhang De-Fu, Li Xin. A personified annealing algorithm for circles packing problem, Acta Automatica Sinica, 2005, 31(4): 590-595

[28] Kang Li-Shan, Xie Yun, You Shi-Yong, Lou Zu-Hua. Non-Numerical Parallel Algorithms, No. 1, Simulation Annealing Algorithm. Beijing: Science Press, 1998(in Chinese)
(康立山, 谢云, 尤矢勇, 罗祖华. 非数值并行算法(第一册) 模拟退火算法. 北京: 科学出版社, 1998)

[29] Zhang De-Fu. Design and Anlysis of Algorithms. Beijing: National Defense Industry Press, 2007(in Chinese)
(张德富. 算法设计与分析(高级教程). 北京: 国防工业出版社, 2007)

[30] OR-Library. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/thpackinfo.html

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

PENG Yu, born in 1984, Ph. D. candidate. His research interests include computational intelligence.

ZHU Wen-Xing, born in 1968, professor, Ph.D. supervisor. His research interests include computational intelligence and optimization.

CHEN Huo-Wang, born in 1936, Ph. D. supervisor, member of Chinese Academy of Engineering. His research interests include artificial intelligence.

Background

Three-dimensional packing problem belongs to NP hard problem. At present, some excellent algorithms have been presented to solve the problem. The hybrid simulated annealing algorithm based on a constructive heuristic algorithm is presented, the computational results have shown that this algorithm outperforms the current excellent algorithms. The work is supported by the National Nature Science Foundation of China (Grant no. 60773126) and the Province Nature Science Foundation of Fujian (Grant No 2007J0037). 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 algorithms for SAT problem, the circle packing problem, the rectangle packing problem, three-dimensional packing problem and Timtabling problem, some results have been published. The fundamental aim of this paper was to investigate three-dimensional packing problem and to develop state-of-the-art algorithms that could be used in an industrial field. The paper belongs to one of the key parts of the project.