基于最小松弛量的启发式一维装箱算法

罗 飞 任 强 丁炜超 卢海峰
(华东理工大学信息科学与工程学院 上海 200237)

摘 要 一维装箱问题是组合优化中的NP难问题,在有限的时间内获得问题的精确解非常困难。启发式算法和遗传算法是解决装箱问题的两类主要方法,但是,采用经典启发式装箱算法得到的结果在极端情况下非常差,而遗传算法在解决装箱问题的过程中容易出现无效解,致使需要处理的数据量十分巨大。为了获得装箱问题的近似最优解,文中针对目前的装箱问题算法展开分析,提出了一种新型的启发式装箱算法。提出的IAMBS算法允许装箱有一定的松弛量,使用随机思想搜索局部最优,进而获得装箱问题的全局最优解。随机松弛量使该算法不易陷入局部最优,具有较强的发现全局最优解的能力。采用来自两个数据集的1410个基准测试实例进行实验。最终,IAMBS算法获得了1152个实例的最优解。实验数据表明,IAMBS算法可以有效地获得近似最优解,比经典装箱算法更有优势。

关键词 装箱问题,启发式算法,随机算法,蒙特卡洛
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2019.09.048

Heuristic One-dimensional Bin Packing Algorithm Based on Minimum Slack

LUO Fei REN Qiang DING Wei-chao LU Hai-feng
(Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China)

Abstract The one-dimensional bin packing problem is a NP-hard problem in the combinatorial optimization, and it is extremely difficult to obtain an accurate solution of the problem in a limited time. Heuristic algorithms and genetic algorithms are the two main methods to solve the bin packing problem. However, the results obtained by the classical heuristic packing algorithm are very poor in extreme cases. The genetic algorithm is prone to generate invalid solutions in the process of solving the packing problem, thus resulting in large amount of data to be processed. In order to obtain the approximate optimal solution of the packing problem, this paper analyzed the current packing problem algorithm and proposed a new heuristic packing algorithm. The proposed IAMBS algorithm uses the idea of random to search for local optimum by allowing a certain amount of slack in the bin-packing, and then obtains the global optimal solution of the packing problem. The allowable slack can prevent this algorithm from falling into local optimum, and has strong ability to discover global optimal solutions. 1410 benchmark test instances from two sources were utilized for the experiment, and the optimal solution of 1152 instances were implemented by the IAMBS algorithm. Experimental data demonstrate that the IAMBS algorithm can effectively obtain the approximate optimal solution, and it is more advantageous than the traditional classical packing algorithm.

Keywords Bin packing problem, Heuristic algorithm, Random algorithm, Monte Carlo

1 引言

一维装箱问题是许多优化问题的基础,因此一直是众多领域的研究热点,在电子、运输、建筑、云计算等领域中有着非常广泛的应用。例如,在微电子行业,需要把不同长度的代码片段存储到有限容量的存储器中;在物流运输行业,需要把不同重量的包裹装入一定载重的卡车中;在建筑行业,需要把定长的金属材料切割成所需的长度并尽可能使材料总用量最少。由此可见,高效的一维装箱问题的解决方案对处理这些实际问题至关重要。

装箱问题是复杂的离散组合最优化问题。组合优化是指在离散的、有限的数学结构上,寻找能够满足约束条件,并且使目标函数达到最大值或最小值的解。组合优化问题往往具有很多局部极值点。一维装箱问题是寻求将大小不同的 n 个

到稿日期:2018-08-16 返修日期:2018-12-02 本文受国家自然科学基金(61472139),华东理工大学2017年教育教学规律与方法研究项目(ZH1726107)资助。

罗 飞(1978-),男,博士,副教授,主要研究方向为分布式计算,E-mail:luof@ecust.edu.cn;任 强(1993-),男,硕士生,主要研究方向为云计算,E-mail:renqiqiang@outlook.com(通信作者);丁炜超(1989-),男,博士,讲师,主要研究方向为云资源调度、分布式计算、多目标优化等;卢海峰(1993-),男,博士生,主要研究方向为边缘计算和强化学习。

物体分配到固定容量的容器中并使所用容器最少的解决策略。Garey 和 Johnson 证明一维装箱问题是一个强 NP 难的问题[1],即使所有的输入参数的长度都被多项式限定,也不存在能在有效的时间内求得精确解的算法。

从 20 世纪 70 年代开始,装箱问题引起了广泛的探讨和研究[2]。近 40 年来,人们逐渐建立了较为完善的理论体系,同时研发出了大量的算法。尽管多年来许多学者的研究极大地推动了装箱问题的研究进展,但是关于装箱问题仍有继续深入探索的空间。

2 相关工作

为了获得装箱问题的近似最优解,最早提出的是在线启发式装箱算法。First Fit(FF),Best Fit(BF)和 Next Fit(NF)等在线启发式方法是按照物品到达的顺序对其进行装箱。Heydrich 等提出了目前已知的最好的在线装箱算法[3],渐进竞争比为 1.5816。他们提出的算法是对 Seiden 提出的算法的改进,原算法的渐进竞争比为 1.588 89[4]。在线算法的最坏情况性能比的已知最好下界为 1.540 14[5]。离线启发式算法首先按照物体大小的非递增顺序对物体进行排序,然后应用 FF 或 BF。First Fit Decreasing(FFD)和 Best Fit Decreasing(BFD)是常见的离线启发式算法。Coffman 等对经典启发式算法的性能进行了详细的分析与研究[6]。此外,Gupta 等提出了一种新型的启发式算法 MBS[7]。Fleszar 等提出了基于 MBS 算法的启发式算法和可变邻域的搜索方法[8]。

Martello 等提出了基于分支定界的 MTP[9]精确装箱算法,该算法被之后的大多数研究当作参考算法。Scholl 等在此基础上结合启发式算法提出了 BISON[10]。Carvalho 提出的精确算法基于列生成和分支定界[11]。

在元启发式算法中,Falkenauer 提出了一种混合分组遗传算法(HGGA)[12]。Bhatia 等提出了一种多染色体分组遗传算法(MGGA)和一种更好的启发式适应算法[13]。Tansel 等提出了一套稳健且可扩展的混合并行算法[14],利用并行计算技术、进化分组遗传元启发式法以及面向箱的启发式法来获得大规模一维 BPP 实例的解。

分支定界方法、启发式算法和元启发式算法都是解决离线装箱问题的常用方法。分支定界方法利用递归方法寻找问题的可行解,采取一定的限制条件排除可行域中大量非最优区域。启发式算法多数以贪心方法为特点,采用固定的规则对数据进行处理。该类算法求解的结果与物体的体积、数据分布等有较大的关系,并且在极端情况下求解结果非常差。另外,简单遗传算法求解装箱问题的过程中容易产生无效染色体[15],使得某些箱子的体积之和超过箱子规定的容量,从而使结果的收敛速度变慢,运算效率变低。

本文针对 MBS 启发式算法进行了深入研究,并基于 MBS 算法提出了一种新型的启发式装箱算法。实验表明,本文提出的新型启发式装箱算法与经典装箱算法相比更容易发现近似最优解,在解决一维装箱问题上具有重要意义。本文第 3 节介绍了装箱问题的数学描述以及经典的启发式装箱算法和 MBS 系列的装箱算法;第 4 节详细介绍了本文提出的基于随机松弛量的改进算法;第 5 节通过实验对比了算法改进前后的性能差异,以及本文算法与经典算法相比所具有的优势;最后总结全文。

3 解决装箱问题的启发式算法

3.1 一维装箱问题的数学描述

一维装箱问题是将 n 个物品的序列 σ 装入等容量的箱子中,每个物品有一定的重量 w,每个箱子都有相同的重量限制 c,目的是寻找将物品分配到箱子的最优方案,即装完所有物品所使用的箱子数量最少。

装箱问题的数学表示为:

$$\min z(y)=\sum_{i=1}^{n} y_i \tag{1}$$

$$\text{s.t. } \sum_{j=1}^{n} w_j x_{ij} \leqslant c y_i, i\in N=\{1,2,\cdots,n\} \tag{2}$$

$$\sum_{i=1}^{n} x_{ij}=1, j\in N \tag{3}$$

$$y_i=0 \text{ or } 1, i\in N \tag{4}$$

$$x_{ij}=0 \text{ or } 1, j\in N \tag{5}$$

其中,变量的含义是:

$$y_i=\begin{cases}1, & \text{第 } i \text{ 个箱子被使用}\\ 0, & \text{else}\end{cases}, i=1,2,\cdots,n$$

$$x_{ij}=\begin{cases}1, & \text{第 } j \text{ 个物体放入第 } i \text{ 个箱子}\\ 0, & \text{else}\end{cases}, i,j=1,2,\cdots,n$$

3.2 经典的启发式算法

自装箱问题被提出以后,陆续提出的装箱算法都是启发式算法。对于一维装箱问题,目前已经有很多经典的启发式算法,如下次适应算法(Next Fit Algorithm,NF)、首次适应算法(First Fit Algorithm,FF)、最差适应算法(Worst Fit Algorithm,WF)、渐近最差适应算法(Almost Worst Fit Algorithm,AWF)、最佳适应算法(Best Fit Algorithm,BF)、降序下次适应算法(Next Fit Decreasing Algorithm,NDF)、降序首次适应算法(First Fit Decreasing Algorithm,FFD)、降序最差适应算法(Worst Fit Decreasing Algorithm,WFD)、降序渐近最差适应算法(Almost Worst Fit Decreasing Algorithm,AWFD)、降序最佳适应算法(Best Fit Decreasing Algorithm,BFD)等。表 1 对比分析了这几种启发式算法的特性。

表 1 经典启发式算法的特性

Table 1 Features of traditional heuristics algorithms

算法	是否具有在线特性	时间复杂度	最坏情况渐进性能比
下次适应算法(NF)	在线	$O(n)$	2.0[16]
降序下次适应算法(NFD)	离线	$O(n\log n)$	2.0[16]
首次适应算法(FF)	在线	$O(n\log n)$	1.7[16]
降序首次适应算法(FFD)	离线	$O(n\log n)$	$\frac{11}{9}$[12]
最差适应算法(WF)	在线	$O(n\log n)$	2.0[16]
降序最差适应算法(WFD)	离线	$O(n\log n)$	1.25[16-17]
渐近最差适应算法(AWF)	在线	$O(n\log n)$	1.7[15-16]
降序渐近最差适应算法(AWFD)	离线	$O(n\log n)$	$\frac{11}{9}$[16-17]
最佳适应算法(BF)	在线	$O(n\log n)$	$\frac{11}{9}$
降序最佳适应算法(BFD)	离线	$O(n\log n)$	$\frac{11}{9}$[17]

表 1 中的算法有一个共同的特点,即把所有的箱子分成已装物品的箱子和空箱子两大类。当物品装入时,优先在已装入物品的箱子中装入,如果找不到合适的箱子,则启用空箱子。这样可以充分利用已经打开的箱子,尽量避免使用新的箱子。所有采用这种方法的装箱算法称为适应算法(Any Fit Algorithm)。MBS 算法[7]并未采用这种方式,在装箱过程中始终只对当前的箱子进行操作,寻找适合当前箱子的最优解。

3.3 MBS 算法

Gupta 等提出了基于最小松弛量的启发式装箱算法(MBS)[7]。MBS 装箱算法以箱子为中心,在装箱的每一个步骤中都尽可能找出最适合当前箱子容量的物体集合。也就是说每一次装箱都遍历所有的剩余未装箱物体的数据,尽可能找到把当前箱子装得最满的最优子集。在每一次寻找最优子集的过程中采用字典搜索优化程序(Lexicographic Search Optimization Procedure,本文称之为 L 算法)。在这一点上,MBS 算法和霍夫曼算法(一种用于解决装配线平衡问题的算法)[18]类似。

MBS 算法的描述如算法 1 所示。

算法 1　MBS 算法

Input:t_i for i=1,…,n;C;k=1,s=n

Step 1　使用 L 算法寻找应该分配给箱子 k 的物体集合 S_k。让 $\sigma=(\sigma(1),\sigma(2),\cdots,\sigma(s))$为未分配箱子的物体编号。

Step 2　如果 $\sigma=\varnothing$,进入 Step 3;否则 k=k+1,进入 Step 1。

Step 3　把 $S=(S_1,S_2,\cdots,S_k)$逐个放入到箱子(1,2,…,k)中,最小的松弛量为 $kC-\sum_{i=1}^{n}t_i$。

L 算法的描述如算法 2 所示。

算法 2　L 算法

Input:t_i for i=1,…,s;C;k=1,$\sigma=(\sigma(1),\sigma(2),\cdots,\sigma(s))$,其中 $t_{\sigma(1)}\geqslant t_{\sigma(2)}\geqslant\cdots\geqslant t_{\sigma(s)}$;$\alpha=0$;j=1;$\pi=(\pi(1),\cdots,\pi(j))=(\sigma(1),\sigma(2),\cdots,\sigma(j))$

Step 1　如果 $P_\pi=C$,令 $S_k=\pi$,进入 Step 6;否则进入 Step 2。

Step 2　找到 q 使 $\sigma(q)=\pi(j)$。如果 $P_\pi<C$,令 j=j+1,进入 Step 3;否则进入 Step 4。

Step 3　如果 $P_\pi>\alpha$,令 $\alpha=P_\pi$,$S_k=\pi$,进入 Step 4。

Step 4　如果 q<s,令 $\pi(j)=\sigma(q+1)$进入 Step 1,否则进入 Step 5。

Step 5　如果 j=1,进入 step 6;否则,令 j=j+1,找到 q 使 $\sigma(q)=\pi(j)$,进入 Step 4。

Step 6　把 S_k 中的物体放置到第 k 个箱子中,对应的最小松弛量为 $C-\alpha$。

Gupta 等指出,MBS 算法在条件下一定能够获得最优解,而且对于可完全精确装箱问题也一定能够获得最优解[4],如表 2 所列。

表 2　MBS 算法的装箱结果

Table 2　MBS' solution of bin-packing problem

物体	箱子容量	装箱结果	竞争比
60,50,30,20,20,20	100	[60,20,20],[50,30,20]	1
3,3,2,2,2,2	7	[3,2,2],[3,2,2]	1
7,5,4,4,4,3,3,3,3,3	13	[7,3,3],[5,4,4],[4,3,3,3]	1

MBS 算法对于解决上述问题有着显著的优势,但是大部分装箱问题并不满足这两个条件。实际应用中的装箱场景都是接近满的装箱问题,并非对个别箱子进行精确装箱就可以提高整个装箱问题的解的优越性。

3.4 MBS 衍生算法

Fleszar 等在 MBS 算法的基础上提出了 MBS′算法[8]。由于所有的物体都要装入箱子,MBS′算法先把当前最大的物体装入箱子,将它作为种子,并在此基础上对剩下的物体使用 MBS 算法进行装箱。这种算法减少了 MBS 算法的时间复杂度,但是对装箱效果的改善不够显著。Fleszar 等进而提出了 Relaxed MBS′算法,允许每次装箱有一定的松弛量。Relaxed MBS′算法将 L 算法每次搜索的终止条件从 $S=0$ 变为 $S\leqslant$ allowable Slack。每次允许的松弛量变化的增量为 $v=[0.5\%\times c]$,增量变化区间限制为 $\min\{40,c/v\}$[8]。如果提前达到装箱问题的下限则停止增量的变化。Fleszar 等在 Relaxed MBS′算法中提出了允许合理的松弛量这一思想,但是松弛量的选择却采用了一种经验值的设定,在后续的实验中并未有效证明选取此松弛量的依据。这种允许一定松弛量的思想,对于解决非精确装箱问题提供了一个新的研究思路。本文将基于允许一定松弛量的思想对 MBS 算法进行改进。

3.5 蒙特卡洛算法和拉斯维加斯算法

蒙特卡洛(Monte Carlo)[19]算法和拉斯维加斯(Las Vegas)算法[20]是常用的概率随机算法。蒙特卡洛算法保证了对问题所有的实例都以高概率求出正确解,但是求出的解未必都是正确的。拉斯维加斯算法总能得到正确的结果,也就是说一旦通过拉斯维加斯算法得到一个解,这个解一定是正确解。拉斯维加斯算法找到正确解的概率会随着它所用的计算成本的增加而提高。蒙特卡洛算法的关键是提高解的正确率,拉斯维加斯算法的关键是提高解的成功率。根据问题的实际情况,可以选择不同的算法。拉斯维加斯算法和蒙特卡洛算法也可以结合使用。本文将综合这两种算法的思想进行算法设计。

4 AMBS 算法

无论是经典启发式算法还是 MBS 算法,都是在固定的规则上寻找一个局部最优解,进而获得一个近似的全局最优解。MBS 算法以箱子为中心,每一次装箱过程中都有且仅有一个箱子被打开。对打开的箱子,使用 L 算法进行搜索,尽可能找到当前的最优解,即可以完全装满箱子的解,称之为局部最优解。对于精确装箱的问题,找到所有的局部最优就可以获得该问题的最优解。然而,对于大多数装箱问题,找到了局部最优解并不意味着可以获得整体问题的最优解。例如,箱子的容量是 9,物体的重量分别是 3,3,3,5,5,5。此时,如果使用 MBS 算法求得的结果是{3,3,3},{5},{5},{5},共 4 个箱子,而最优解却是{3,5},{3,5},{3,5},只需 3 个箱子。所以,在 MBS 算法的基础上允许每个箱子具有一定的松弛量不失为一种好的改进方法。因此,针对大多数装箱问题,可以使用一定的随机算法尝试允许一定的松弛量,然后寻找允许一定松弛量的局部最优。

本文基于允许合理的松弛量这一思想以及概率随机算法的思想,对 MBS 算法进行了改进,并将改进后的算法称为 AMBS(Allowable Min Bin Slack)算法。AMBS 算法仅对搜索方法 L 进行了改进,改进后的搜索算法称为 L′。

搜索算法 L′描述如算法 3 所示。

算法 3 L′算法

Input: t_i for i=1,…,s; C; k=1, σ=(σ(1),σ(2),…,σ(s)),其中 $t_{\sigma(1)} \geqslant t_{\sigma(2)} \geqslant \cdots \geqslant t_{\sigma(s)}$; α=0; j=1; π=(π(1),…,π(j))=(σ(1),σ(2),…,σ(j))

Step 1 产生随机松弛量 Slack∈[0,⌈min(σ)×0.368⌉],进入 Step 2。

Step 2 如果 $0 \leqslant P_\pi - C \leqslant$ Slack,令 $S_k=\pi$,进入 Step 7;否则进入 Step 3。

Step 3 找到 q 使 σ(q)=π(j)。如果 $P_\pi < C$,令 j=j+1,进入 Step 4;否则进入 step 6。

Step 4 如果 $P_\pi > \alpha$,令 $\alpha=P_\pi$, $S_k=\pi$,进入 Step 5。

Step 5 如果 q<s,令 π(j)=σ(q+1)进入 Step 1,否则进入 Step 6。

Step 6 如果 j=1,进入 Step 7;否则,令 j=j+1,找到 q 使 σ(q)=π(j),进入 Step 5。

Step 7 把 S_k 中的物体放置到第 k 个箱子中。

AMBS 算法解决装箱问题的过程也是以当前箱子为中心,寻找适合当前箱子的最优解。但是,每次寻求当前箱子的最优解时,并不要求完全装满当前箱子,而是允许一定的松弛量。因为在实际的装箱问题中,最优解并不一定是所有的箱子都装到最满。允许的松弛量可看作随机变量,我们可以确定这个随机变量的最大范围 $Slack \in [0, \min(\sigma))$。对每一个箱子进行装箱时,搜索算法在每一次判断搜索结果是否满足条件时,当前的最好情况就是可以刚好装满当前箱子,即 $Slack=0$。当前最差的情况就是当前箱子剩余的空间刚刚小于物体集合中最小的物体。但是,此时局部最优解肯定不是当前的搜索结果,在剩余物体的随机组合中可以得到比当前结果更好的解。因此允许松弛量的最大值不能是当前剩余物体的最小值,而应是小于当前剩余物体的最小值。

本文提出的算法中允许的松弛量最大为当前剩余物体的质量最小值的 0.368 倍。在蒙特卡洛模拟过程中,在 36.8% 以后的区间中做决策获得最优解的概率最大[21]。因此,每一次搜索过程的松弛量设定为 $Slack \in [0, \lceil \min(\sigma) \times 0.368 \rceil]$, *Slack* 为随机产生。

对每一个箱子的装箱过程均使用改进后的搜索算法 L′获得。图 1 为 L′算法的流程图。一次完整的装箱过程要一直执行 L′算法,直至所有的物体都被装入箱子中。

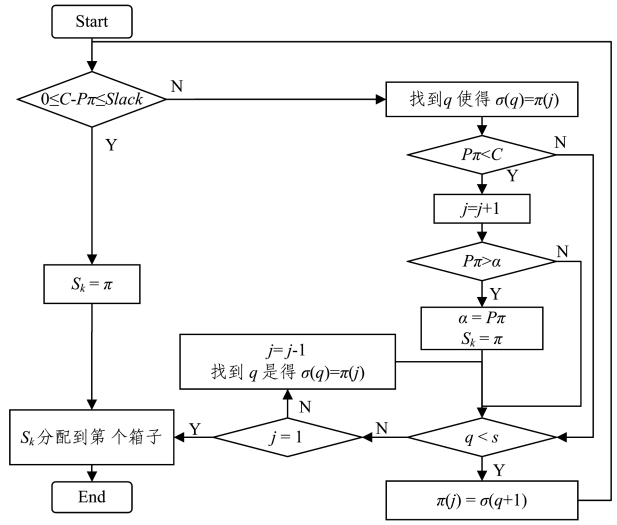

图 1 AMBS 搜索算法 L′的流程图

Fig. 1 Flow chart of search algorithm L′ for AMBS

5 IAMBS

使用 AMBS 算法求解装箱问题时,每一次的装箱结果都是在允许的随机松弛量的条件下搜索到的局部最优解。因此一次完整的装箱结果几乎不会与下次的装箱结果相同。为了得到近似最优解,可设定装箱次数的阈值为 N,进行 N 次装箱以后,获得 N 个装箱策略,其中的最优策略即为当前问题的近似最优解。随着装箱次数的增加,获取全局最优解的概率也随之增大。

允许一定随机松弛量的算法会给精确装箱问题带来巨大的计算量,有可能在一定的迭代次数范围内得不到最优解。因此,为了保证能够得到一个实际问题的最优解,有必要对装箱的数据进行预先判断,以避免计算资源的浪费。改进后的算法称为 IAMBS(Improved IAMBS)算法,其流程如图 2 所示。首先,计算理论装箱的最下界 $L=\lceil \sum_{j=1}^{n} w_j / C \rceil$,用 MBS 算法得到一个解 S,如果 $S=L$,则该装箱问题是一个精确装箱问题,MBS 算法获得的解即为最优解。如果 MBS 算法的解大于理论装箱下界,则使用 AMBS 算法进行装箱,N 次装箱后,选取 N 次装箱过程的最优值 D。在 S 与 D 中选取的最优值即为该装箱问题的近似最优解。如果在 N 次装箱的过程中提前获得与 L 相等的解,则算法结束,L 即为当前最优解。

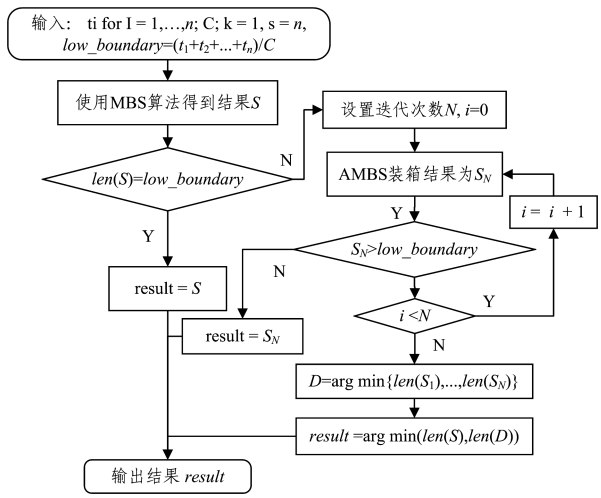

图 2 IAMBS 算法的流程

Fig. 2 Process of IAMBS

6 实验分析

6.1 实验环境

为了对比算法改进前后的性能差异,本文使用 Python3 实现了 NFD,FFD,WFD,AWFD,BFD,MBS,MBS′算法,以及本文提出的改进算法 AMBS 和 IAMBS。由于降序启发式算法被证明优于非降序启发式算法[16],此处对非降序启发式算法不做讨论。实验平台的硬件配置为 CPU 24 Intel(R) Xeon(R) CPU E5-2620 0 @ 2.00 GHz,内存 128 GB;软件配置为 Ubuntu16.04;开发环境为 Pycharm(2018.1)。

6.2 算法性能评价指标

衡量装箱算法性能的标准有很多种,本文关注的是算法的竞争比。对于一个给定的极端情况的实例来说,一个算法

的耗费可能很高,因此需要将其与其他算法进行比较才更有意义。对于任意一个给定的问题实例,一个算法的解与最优解的差距的评价指标被称为竞争比,其最先由 Sleator 和 Tarjan[22] 提出。许多问题都是使用竞争比分析来展开研究的,除了装箱问题,还有诸如调度问题、页面置换问题、网络上的路由及任务控制问题。

令作用于一个包含 n 个物体的输入序列 σ 的算法 A 的耗费为 $A(\sigma)$,最优的耗费为 $OPT(\sigma)$。装箱问题是要求最小化代价,用竞争比分析对比装箱算法 A 的输出 $A(\sigma)$ 和最优解 $OPT(\sigma)$,则竞争比的定义为:

$$R(A)=\sup\frac{A(\sigma)}{OPT(\sigma)} \tag{6}$$

竞争比越小,代表装箱算法的性能越优异,对于任何装箱算法来说,竞争比的值都不小于 1。

在实际解决装箱问题的过程中,最优耗费 $OPT(\sigma)$ 一般是未知的,所以通常采用理论最优解代替 $OPT(\sigma)$。

$$OPT(\sigma)=\left\lceil\frac{\sum_{i=1}^{n}w_i}{C}\right\rceil \tag{7}$$

由于本文采用了已知目前最优解的数据集,因此使用两种评价标准对算法的性能进行评价,第一个是实现目前已知最优解的数量,第二个是对应算法的竞争比。

6.3　实验数据集

本文采用了 BINDATA[10] 和 SCH_WAE[23] 两个数据集。Scholl 等提出的 BINDATA 基准数据集用于装箱问题算法的测试[10]。该数据集分为 3 部分:第 1 组数据(Bin1data)有 720 个实例,构建方法与 Martello 等提出的用于测试程序 MTP[9] 的方法类似;第 2 组数据(Bin2data)有 480 个实例,包含第 1 组没有包含的参数;第 3 组数据(Bin3data)只有 10 实例,是非常困难的实例。该数据集给出了目前已知的最优解。SCH_WAE 数据集是从 Schwerin 和 Waescher 的论文中收集得到的。该数据集包含 200 个问题实例,给出了目前已知的最优解。

6.4　实验结果

本文实验分别使用 BINDATA 以及 SCH_WAE 数据集,对经典算法以及改进后的算法进行实验。

MBS 算法的时间复杂度为 $O(2^n)$,而采取 AMBS 算法通过允许随机松弛量可以减少很多不必要的搜索,降低算法的时间复杂度。使用 Bin1data 数据集进行实验,结果如表 3 所列。

表 3　MBS 和 AMBS 在 Bin1data 上的实验结果

Table 3　Results of MBS and AMBS on Bin1data

算法	实现已知最优解	未实现已知最优解	测试数据总数	平均竞争比	平均计算时间
MBS	252	468	720	1.0645	46.1694
AMBS	614	106	720	1.0459	6.0781

AMBS 算法可以获得比 MBS 算法更优秀的解集,同时 AMBS 算法的平均计算时间远远低于 MBS 算法。

将 IAMBS 算法的迭代次数都设置为 10。使用两个数据集对经典启发式算法和本文改进的算法进行实验,结果如表 4－表 8 所列。

表 4　Bin1data 数据集上的实验结果

Table 4　Experimental results on Bin1data

算法	实现已知最优解	未实现已知最优解	测试数据总数	平均竞争比
NFD	0	720	720	1.3502
FFD	546	174	720	1.0497
WFD	442	278	720	1.0537
AWFD	163	557	720	1.0663
BFD	547	173	720	1.0497
MBS[7]	252	468	720	1.0645
MBS[8]	633	87	720	1.0471
AMBS	614	106	720	1.0472
IAMBS	669	51	720	1.0459

表 5　Bin2data 数据集上的实验结果

Table 5　Experimental results on Bin2data

算法	实现已知最优解	未实现已知最优解	测试数据总数	平均竞争比
NFD	59	421	480	1.1271
FFD	236	244	480	1.0315
WFD	213	267	480	1.0336
AWFD	1	479	480	1.0806
BFD	236	244	480	1.0315
MBS[7]	125	355	480	1.0829
MBS′[8]	247	233	480	1.0264
AMBS	289	191	480	1.0118
IAMBS	331	149	480	1.0109

表 6　Bin3data 数据集上的实验结果

Table 6　Experimental results on Bin3data

算法	实现已知最优解	未实现已知最优解	测试数据总数	平均竞争比
NFD	0	10	10	1.1711
FFD	0	10	10	1.0739
WFD	0	10	10	1.0739
AWFD	0	10	10	1.0865
BFD	0	10	10	1.0739
MBS[7]	0	10	10	1.0594
MBS′[8]	0	10	10	1.0721
AMBS	4	6	10	1.0252
IAMBS	6	4	10	1.0198

表 7　SCH_WAE 数据集实验结果

Table 7　Experimental results of SCH_WAE

算法	实现已知最优解	未实现已知最优解	测试数据总数	平均竞争比
NFD	1	199	200	1.0622
FFD	1	199	200	1.0614
WFD	1	199	200	1.0614
AWFD	0	200	200	1.1069
BFD	1	199	200	1.0614
MBS[7]	25	175	200	1.0574
MBS′[8]	32	168	200	1.0513
AMBS	98	102	200	1.0356
IAMBS	146	54	200	1.0289

表 8　各算法累计实现已知最优解的数量以及平均竞争比

Table 8　Number of known optimal solutions and average competition ratio of each algorithm

算法	累计实现已知最优解的数量	累计平均竞争比
NFD	60	1.1776
FFD	783	1.0541
WFD	656	1.0556
AWFD	164	1.0850
BFD	784	1.0541
MBS[7]	402	1.0660
MBS′[8]	912	1.0492
AMBS	1005	1.0299
IAMBS	1152	1.0263

使用标准数据集进行实验，对 1410 个实例进行求解，对比经典启发式算法以及 MBS 算法，本文提出的 IAMBS 算法可以获得 1152 个目前已知最优解，具有较强的获得全局最优解的能力。

IAMBS 算法属于随机搜索算法，算法实现最优解的概率与算法的迭代次数是相关的。算法迭代的次数越多，获得最优解的概率就越大，因此本文设置迭代次数分别为 10，20，30，实验结果如表 9 所列。实验中采用 Bin1data 数据集，当迭代次数越大时，获得近似最优解的概率越大。

表 9 迭代次数对 IAMBS 实验结果的影响

Table 9 Influence of iteration times on experiment results of IAMBS algorithm

算法	实现已知最优解	未实现已知最优解	测试数据总数	平均竞争比
10	669	51	720	1.0459
20	683	37	720	1.0432
30	690	30	720	1.0389

结束语 装箱问题是一个庞大而且复杂的问题，在理论和实际应用上都有一定的难度。本文提出了一种基于随机松弛量的启发式装箱算法，并通过实验验证本文提出的 IAMBS 算法对解决装箱问题优于传统的经典算法以及 MBS 算法，对于实际生产应用有积极的意义。改进后的算法可以有效地找到当前装箱问题的最优解。然而获得最优解的概率与迭代次数不是正相关，较少的迭代次数也有可能产生最优解。智能优化技术的出现为解决装箱问题提供了很多新的方法。本文还需要结合这些新的算法进一步改进。在下一步的工作中将结合遗传算法，确定一定的搜索空间，使求解结果在一定的迭代次数内收敛。另外，我们将挖掘带有约束的装箱问题与经典装箱问题的联系，应用经典装箱问题的分析方法来处理带有约束的装箱问题。

参 考 文 献

[1] DELORMEM，IORIM，MARTELLO S. Bin packing and cutting stock problems：Mathematical models and exact algorithms[J]. European Journal of Operational Research，2016，255(1)：1-20.

[2] JOHNSON D S. Near-Optimal Bin Packing Algorithms [D]. Cambridge，MA，USA：Massachusetts Institute of Technology，1973.

[3] HEYDRICH S，STEE R V. Beating the Harmonic lower bound for online bin packing[J]. arXiv：1511.00876.

[4] SEIDEN S S. On the online bin packing problem[J]. Journal of the Acm，2002，49(5)：640-671.

[5] BALOGH J，BÉKÉSI J，GALAMBOS G. New Lower Bounds for Certain Classes of Bin Packing Algorithms[J]. Theoretical Computer Science，2012，440-441(8)：1-13.

[6] COFFMAN E G J，GAREY M R，JOHNSON D S. Approximation algorithms for bin packing：a survey[C]// Approximation algorithms for NP-hard problems. PWS Publishing Co.，1997：46-93.

[7] GUPTA J D，HO J. A new heuristic algorithm for the one-dimensional bin-packing problem[J]. Production Planning & Control，1999，10(6)：598-603.

[8] FLESZARK，KHALIL S. New heuristics for one-dimensional bin-packing [J]. Computers & Operations Research，2002，29(7)：821-839.

[9] MARTELLO S，TOTH P. Knapsack problems：algorithms and computer implementations[J]. Journal of the Operational Research Society，1991，42(6)：513-513.

[10] SCHOLL A，KLEIN R，JÜRGENS C. Bison：A fast hybrid procedure for exactly solving the one-dimensional bin packing problem[J]. Publications of Darmstadt Technical University Institute for Business Studies，1997，24(7)：627-645.

[11] CARVALHO J M V D. Exact solution of cutting stock problems using column generation and branch-and-Bound 1[J]. Annals of Operations Research，1999，86(1)：629-659.

[12] FALKENAUER E. A hybrid grouping genetic algorithm for bin packing[J]. Journal of Heuristics，1996，2(1)：5-30.

[13] BHATIA A K，BASU S K. Packing Bins Using Multi-chromosomal Genetic Representation and Better-Fit Heuristic[C]// Neural Information Processing，International Conference，ICONIP 2004. Calcutta，India，2004：181-186.

[14] TANSEL D，AHMET C. Optimization of one-dimensional Bin Packing Problem with island parallel grouping genetic algorithms[J]. Computers & Industrial Engineering，2014，75：176-186.

[15] 玄光男，程润伟. 遗传算法与工程优化[M]. 北京：清华大学出版社，2004：26-27.

[16] JOHNSON D S. Fast algorithms for bin packing [J]. Journal of Computer & System Sciences，1974，8(3)：272-314.

[17] HOCHBAUM D. Approximation algorithms for NP-hard problems[J]. ACM Sigact News，1997，28(2)：40-52.

[18] HOFFMANN T R. Assembly line balancing with a precedence matrix[J]. Management Science，1963，9(4)：551-562.

[19] KROESE D P，BRERETON T，TAIMRE T，et al. Why the Monte Carlo method is so important today[J]. Wires Computational Statistics，2014，6(6)：386-392.

[20] BRATTKA V，GHERARDI G，HÖLZL R. Las Vegas computability and algorithmic randomness[J]. International Symposium on Theoretical Aspects of Computer Science，2015，30：130-142.

[21] ROBERT P C. Monte Carlo statistical methods [M]. World Book Publishing Company，2009.

[22] SLEATOR D D，TARJAN R E. Amortized efficiency of list update and paging rules[J]. Communications of the Acm，1985，28(2)：202-208.

[23] SCHWERIN P，WÄESCHER G. The Bin-Packing Problem：A Problem Generator and Some Numerical Experiments with FFD Packing and MTP [J]. International Transactions in Operational Research，1997，4(5/6)：377-389.