软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(9):2218−2229 [doi: 10.13328/j.cnki.jos.004848]

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

动作空间带平衡约束圆形 Packing 问题的拟物求解算法*

何 琨, 杨辰凯, 黄梦龙, 黄文奇

(华中科技大学 计算机科学与技术学院,湖北 武汉 430074)

通信作者: 何琨, E-mail: brooklet60@gmail.com

摘 要: 对于一个以卫星舱内设备布局为背景的具有 NP 难度的全局优化问题——带平衡约束的圆形 Packing 问题,提出了基于动作空间的拟物求解算法.在拟物下降遇到局部极小点的陷阱时,如何找到当前格局下的最空闲空间以使搜索过程跳到更有前景的区域去是设计跳坑策略的一个关键难点.借鉴求解矩形 Packing 问题中动作空间的概念,通过化“圆”为“方”,将不规则的空闲空间近似为一系列规则的矩形空间,从而有效地解决了此难点.另外,将拟物法与提前中止、粗精调和自适应步长这 3 个拟人辅助策略相结合,以提高势能下降的效率.对 3 组共 13 个代表性算例的计算结果及与国内外代表性算法的比较表明,所提格局的外包络圆半径多为最小或次小,且在部分算例上找到了有更小外包络圆半径的格局,总体计算结果较好,且静不平衡量的精度较高.

关键词: NP 难度;圆形 Packing;拟物;动作空间;平衡约束

中图法分类号: TP301

中文引用格式: 何琨,杨辰凯,黄梦龙,黄文奇.动作空间带平衡约束圆形 Packing 问题的拟物求解算法.软件学报,2016,27(9):2218−2229. http://www.jos.org.cn/1000-9825/4848.htm

英文引用格式: He K, Yang CK, Huang ML, Huang WQ. Quasi-Physical algorithm based on action space for solving the circles packing problem with equilibrium constraints. Ruan Jian Xue Bao/Journal of Software, 2016,27(9):2218−2229 (in Chinese). http://www.jos.org.cn/1000-9825/4848.htm

Quasi-Physical Algorithm Based on Action Space for Solving the Circles Packing Problem with Equilibrium Constraints

HE Kun, YANG Chen-Kai, HUANG Meng-Long, HUANG Wen-Qi

(School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: This paper proposes a Quasi-physical algorithm based on action space (QPAS) for an NP-hard global optimization problem - the circle packing problem with equilibrium constraints (CPPEC). The algorithm has important applications for the layout design of the satellite modules. A key issue in designing a good basin hopping strategy for CPPEC is how to find the most vacant areas such that the searching procedure can jump from a local minimum basin to a promising area. By borrowing the concept of “action space” proposed for the rectangular packing problem, the new algorithm approximates each circle as a rectangle and the irregular vacant areas are viewed approximately as regular rectangular areas. Consequently the most vacant areas can be found efficiently and accurately. In addition, three quasi-human strategies, namely early termination, coarse-to-fine and adaptive step length, are combined with the quasi-physical approach to speed up the potential energy descending process. Experiments are performed on 13 benchmark instances, and computational results demonstrate the high efficiency of the proposed approach. QPAS achieves the first or the second best results on most instances compared with other algorithms, and in some configurations, it has smaller container radius than the current best results. Meanwhile, QPAS obtains very small equilibrium deviations.

Key words: NP-hard; circle packing; quasi-physics; action space; equilibrium constraints

* 基金项目: 国家自然科学基金(61173180, 61272014)
Foundation item: National Natural Science Foundation of China (61173180, 61272014)
收稿时间: 2014-02-18; 修改时间: 2014-04-17; 采用时间: 2015-01-21

圆形 Packing 问题是高复杂度典型的 NP 难度问题,在无线通信、航空航天及交通运输等领域均有着广泛的应用.随着问题规模的增大,问题的解空间呈指数级膨胀,使得精确算法的求解时间过长而不适于实际应用.国内外研究者多采用启发式方法求解此类问题,如拟物算法[1,2]、禁忌算法[3,4]、模拟退火算法[4−6]、梯度下降算法[7]、遗传算法[8]等.其中,拟物法通过在自然世界中寻找与原始问题等价的物理现象,如模拟弹性物体在弹性力作用下的运动过程,从而设计得到一种高效率的求解算法.通常使用拟物法进行局部搜索,并采取拟人策略跳出局部极小值的陷阱.而如何快速、有效地找到当前格局下的最空闲空间以使搜索过程跳到更有前景的区域去是设计跳坑策略的一个关键难点.

本文研究圆形 Packing 问题的重要扩展问题——带平衡约束的圆形 Packing 问题.该问题在圆形 Packing 问题的基础上引入了静平衡约束,是以卫星舱布局为背景的全局优化问题.相关的求解方法包括模式迭代法和主布模法[9]、遗传算法[10,11]、拟物算法[12]、粒子群优化算法[13,14]、快速局部搜索算法[15]、散射搜索法[16]、吸引盘填充算法[17]、模拟退火算法[18]、基于禁忌搜索的启发式算法[19]和基于粗精调的拟物拟人算法[20]等.这些求解算法大致可以分为两类:一类是具有较强全局搜索能力及较高普适性的算法,如粒子群算法、散射搜索法、遗传算法等.另一类是具有较好局部搜索能力及较强针对性的算法,如拟物算法、快速局部搜索算法、吸引盘填充算法等.由于前者的局部搜索速度较慢,而后者容易落入局部极小点的陷阱,研究者通常将两类算法结合起来以兼顾全局搜索能力和局部搜索速度.对于圆形 Packing 问题,目前已经有比较成熟的拟物模型[1,2].而对于带平衡约束的圆形 Packing 问题,研究者多通过在圆形 Packing 问题的弹力模型的基础上引入静不平衡势能来构造势能函数[15−19],但未给出针对静平衡约束的贴切的拟物模型.2013 年,何琨等人[20]引入了拉力模型以处理静平衡约束并取得了很好的效果.

本文以文献[20]提出的弹力模型和拉力模型为基础,将拟物法与 3 个拟人辅助策略相结合,提高了算法的全局搜索能力和局部搜索速度.当计算落入局部极小点的陷阱时,借鉴求解矩形 Packing 问题的动作空间[21−24]的概念,通过化“圆”为“方”,将外包络圆和各圆饼近似看作矩形,从而使跳坑过程能够较快、较准确地找到当前格局下的最空闲空间,并结合一些拟人跳坑策略[4,20],使搜索过程快速有效地离开局部极小点并跳到更有前景的区域中去.对目前国内外公开的 13 个代表性算例进行了计算.实验结果表明,所提算法是求解带平衡约束的圆形 Packing 问题的一种快速而有效的算法.

1 问题描述

在设计人造卫星舱时,需要将若干圆柱形部件(待布物)互不嵌入地竖直置入舱体内,要求给出待布物的一种紧密布局,使得卫星舱的半径尽可能的小.同时,为了减小卫星舱自旋时产生的震动、噪音、发热等,要求布局后卫星舱的静不平衡量也尽可能的小,即要求所有待布物的重心尽可能接近整个舱体的中心.以卫星舱的一个横截面为研究对象,则上述问题可抽象为一个带平衡约束的二维圆形 Packing 问题:已知 n 个实心圆饼(n 为任意正整数),圆饼 $C_i(i\in\{1,2,\ldots,n\})$的半径为 r_i、质量为 m_i,要求给出这 n 个圆饼的一种无嵌入的紧密布局,使得其外包络圆 C_0 的半径 R_0 尽可能地小,且整个圆饼系统的质心与原点即外包络圆圆心的距离小于一个小的正实数 δ_r.

问题可形式化地定义如下:以外包络圆的圆心为原点建立笛卡尔坐标系,设圆饼 C_i 的圆心坐标为(x_i,y_i),要求给出一个布局 $X=(x_1,y_1,x_2,y_2,\ldots,x_n,y_n)$,使得:

$$\min R_0$$

$$\text{s.t. (1)}\quad d(C_i,C_0)\triangleq\sqrt{x_i^2+y_i^2}\leqslant R_0-r_i,\ i\in\{1,2,\ldots,n\}.$$

$$\text{(2)}\quad d(C_i,C_j)\triangleq\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}\geqslant r_i+r_j,\ i,j\in\{1,2,\ldots,n\},\ i\neq j.$$

$$\text{(3)}\quad R_c\triangleq\frac{\sqrt{\left(\sum_{i=1}^{n}m_ix_i\right)^2+\left(\sum_{i=1}^{n}m_iy_i\right)^2}}{\sum_{i=1}^{n}m_i}<\delta_r.$$

在约束(1)中,$d(C_i,C_0)$表示圆饼 C_i 的圆心到外包络圆 C_0 的圆心的距离,即要求任意圆饼均完全落在外包络圆内.在约束(2)中,$d(C_i,C_j)$表示两圆饼 C_i 与 C_j 的圆心之间的距离,即要求任两圆饼互不嵌入.在约束(3)中,R_c 为圆饼集的质心到原点的距离,当 $R_c<\delta_r$ 时,系统的静不平衡量 $J=\left(\sum_{i=1}^{n}m_i\right)\cdot R_c$ 小于 $\delta_J\triangleq\left(\sum_{i=1}^{n}m_i\right)\cdot\delta_r$.

在上述问题定义中,由于 R_0 不确定,其直接求解的难度较大.本文拟先求解此问题的判定形式,即对给定的 R_0,要求判定是否存在一种布局方案同时满足上述 3 个约束;然后利用二分法不断迭代,就可找到原问题的一个使 R_0 较小的解.因此,只需讨论问题的判定形式下的拟物模型和求解策略.

2 拟物模型

对于带平衡约束的圆形 Packing 问题,文献[20]提出了两个物理模型:弹力模型和拉力模型.

2.1 弹力模型

将 n 个圆饼想象为光滑的弹性实体,将外包络圆想象为一刚性容器,根据弹性力学,这 n 个圆饼在容器中相互挤压时会受到弹性力的作用并产生弹性势能,由此设计得到相应的弹力模型.

将外包络圆编号为物体 0,将圆饼 $C_i(i\in\{1,2,...,n\})$编号为物体 i.在任意时刻,设圆饼 $C_i(i\in\{1,2,...,n\})$的圆心坐标为(x_i,y_i),称 $2n$ 元组 $X=(x_1,y_1,x_2,y_2,...,x_i,y_i)$为一个格局.

对给定格局 X,系统的总弹性势能 $U_e(X)$见式(1).

$$U_e(X)=k_e\cdot\sum_{j=0}^{n-1}\sum_{i=j+1}^{n}D_{ij}^2 \tag{1}$$

其中,k_e 表示圆饼的弹性系数,D_{ij} 表示两物体 i 和 j 之间的嵌入深度,见式(2).

$$D_{ij}=\begin{cases}\max\left(0,r_i+r_j-d\left(C_i,C_j\right)\right), & i,j\in\{1,2,...,n\},i\neq j\\ \max\left(0,r_i-R_0+d\left(C_i,C_0\right)\right), & i\in\{1,2,...,n\},j=0\end{cases} \tag{2}$$

可见,系统的总弹性势能与圆饼的嵌入深度正相关,当且仅当 $U_e(X)=0$ 时,各物体互不嵌入,相应的格局满足约束(1)和约束(2).

2.2 拉力模型

将圆饼集视为一个质点,想象该质点与原点之间有一根很紧的橡皮筋相连,当该质点与原点不重合时,将受到橡皮筋的拉力作用,此时系统存在偏移势能,由此设计得到相应的拉力模型.

对给定格局 X,系统的偏移势能 $U_c(X)$见式(3).

$$U_c(X)=k_c\cdot\sqrt{\bar{X}^2+\bar{Y}^2} \tag{3}$$

其中,k_c 为橡皮筋的拉力系数,$\left(\bar{X},\bar{Y}\right)$ 为圆饼集质心的坐标,见式(4).

$$\left(\bar{X},\bar{Y}\right)=\left(\frac{\sum_{i=1}^{n}m_ix_i}{\sum_{i=1}^{n}m_i},\frac{\sum_{i=1}^{n}m_iy_i}{\sum_{i=1}^{n}m_i}\right) \tag{4}$$

质心 $\left(\bar{X},\bar{Y}\right)$ 到原点的距离即为 R_c.当且仅当 $U_c(x)<k_c\cdot\delta_r$ 时,$R_c<\delta_r$,相应的格局 X 满足约束(3).

综合弹力模型和拉力模型,对给定格局 X,系统的总体势能 $U(X)$见式(5).

$$U(X)=U_e(X)+U_c(X) \tag{5}$$

当 $U(X)>0$ 时,说明各圆饼仍存在嵌入或系统未达到平衡状态.因此,对外包络圆半径给定的判定问题的求解转化为对 $U(X)$最小的格局的求解.

3 拟物算法

本节给出外包络圆半径给定的判定问题的求解算法,包括基于弹力模型和拉力模型的拟物下降算法、基于动作空间的拟人跳坑策略和两者相结合的完整拟物算法.拟物算法的基本思路是:从 t=0 时刻开始,随机生成一个初始格局,然后采用拟物下降法收敛至局部最优格局;若此时系统的总体势能大于 0,即所得格局为非法格局,则执行拟人跳坑策略,从而得到新的格局;如此不断迭代,直至找到合法的解格局成功停机或达到规定时限强制失败停机.

3.1 拟物下降算法

拟物下降法是一种高效的连续优化方法,其基本思路是:在 t 时刻对给定的格局 X,若各圆饼所受的弹性合力大于 0 或圆饼集质心所受的拉力大于 0,则各圆饼在弹性力和拉力作用下进行一次移动,从而得到 t+1 时刻的新格局 X';如此不断迭代,直至找到局部最优格局停止移动.在整个拟物过程中,系统的总体势能呈下降的趋势.

在 t 时刻对给定格局 X,让各圆饼在弹性力、拉力或两者合力的作用下进行一次移动如式(6)、式(7)或式(8)所示,分别称为弹力动作、拉力动作和综合动作.

$$\vec{r}_i^{(t+1)}=\vec{r}_i^{(t)}+h_e\cdot\vec{F}_i^{(t)},\ \ i\in\{1,2,...,n\} \tag{6}$$

其中,$\vec{F}_i^{(t)}$ 为 t 时刻圆饼 C_i 所受的弹性合力,$\vec{r}_i^{(t)}$ 为 t 时刻圆饼 C_i 的位置;h_e 为弹性力作用下的移动步长.

$$\vec{r}_i^{(t+1)}=\vec{r}_i^{(t)}+h_c\cdot\vec{F}_c^{(t)},\ \ i\in\{1,2,...,n\} \tag{7}$$

其中,$\vec{F}_c^{(t)}$ 为 t 时刻圆饼集质心所受的拉力,h_c 为拉力作用下的移动步长.

$$\vec{r}_i^{(t+1)}=\vec{r}_i^{(t)}+h_e\cdot\vec{F}_i^{(t)}+h_c\cdot\vec{F}_c^{(t)},\ \ i\in\{1,2,...,n\} \tag{8}$$

定义 1(卡壳格局). 若圆饼作一次移动后对任意 $C_i(i\in\{1,2,...,n\})$均有 $\vec{r}_i^{(t+1)}=\vec{r}_i^{(t)}$, 则称相应的格局 X 为卡壳格局.

带平衡约束的圆形 Packing 问题需综合考虑连续优化和组合优化两个因素.问题的解空间是连续的,核心困难源自其复杂的组合特征,即局部极小点的数量会随着圆饼数的增加而迅速膨胀.对于解空间中大量的局部极小点,只有少量甚至唯一的全局最小点.如果对每个搜索区域都严格执行拟物下降计算直至该区域的最小点,势必要消耗大量的计算时间.本文将拟物法与以下 3 个拟人辅助策略相结合以提高拟物下降的效率.

(1) 提前中止策略

在问题解空间大范围的搜索区域中,有些区域的局部极小点附近总体势能较大,称为无前景区域.在拟物过程中,由于早期势能的下降梯度较大,通常经过较少次迭代后各圆饼的相对位置就基本确定,如果此时总体势能仍然较大,则认为该搜索区域找不到问题的解,应尽早离开.同时,若当前格局的总弹性势能较小但偏移势能较大,则认为该格局无法收敛至解格局,应停止拟物计算.

具体做法是:(a) 在整个拟物计算过程中,记录下总弹性势能最小的格局 X_{best}.从当前格局 X 出发迭代 15 次,得到新格局 X',若 $U_e(X')>U_e(X_{best})$且 $U_e(X')>1$,则提前中止对该区域的搜索.(b) 在拟物过程中对当前格局 X,若 $U_e(X)\leqslant10^{-10}$ 但 $U_c(X)>10^{-6}$,则停止计算.

(2) 粗精调策略

在拟物过程中,若当前格局的总体势能较大,则各圆饼的相对位置较不稳定,弹性力起主导作用.当计算接近局部极小点时,总体势能的收敛速度较慢,为了不在无前景区域的局部极小点附近花费过多时间,同时又不错过有前景的搜索区域,文献[20]将拟物算法分成粗调和精调两个阶段:对给定格局 X,若 $U_e(X)>10^{-10}$,则进行弹力粗调,各圆饼执行弹力动作,直至 $U_e(X)\leqslant10^{-10}$.若此时 $U(X)>10^{-10}$,则进行拉力粗调,各圆饼执行一次拉力动作.每次进行弹力粗调后就重新进行拉力粗调,直至总体势能足够小才进入精调阶段.

通过实验发现,若在某区域的搜索过程连续多次进入拉力粗调阶段,则可以预测其最终将进入精调阶段.为减少计算量,若 $U(X)\leqslant10^{-10}$ 或在该区域的搜索过程连续 3 次进入拉力粗调阶段,则进行精调,各圆饼执行综合动作.在整个粗精调过程中,若当前格局为卡壳格局或满足提前中止条件,则停止拟物下降计算并跳坑.

(3) 自适应步长策略

在拟物过程中,移动步长的大小对计算的收敛速度影响甚大.由于初始阶段圆饼的嵌入深度相对较大,将 k_e 和 h_e 的初始值设置为稍大于 1 的值,可以加快计算的收敛速度.因此,设置 k_e=1.2,h_e=1.2.同时,将 h_c 的初始值设置为 1,使得经过一次拉力调整后,圆饼集的质心刚好与外包络圆的圆心重合以尽快满足静平衡约束.当计算接近局部极小点时,若移动步长太大,将导致移动过量,使搜索在两个或多个格局之间来回振荡.因此,在圆饼的每次移动后,根据总体势能的变化情况动态调整下一轮迭代的移动步长[20].对当前格局 X 进行一次迭代后得到新格局 X',若 $U_e(X')>U_e(X)$,则将 h_e 减小 15%;若 $U_c(X')>U_c(X)$,则将 h_c 减小 5%.

使用上述方法进行多次调整后,可能使移动步长变得太小而导致收敛速度变得极慢.因此,若连续 50 次迭代中,$U_e(X')$都小于 $U_e(X)$,则将 h_e 和 h_c 恢复到初始值.

在实际计算中,由于受到计算机的精度限制,无法使总体势能严格地降至 0,因此,约定总体势能小于 10^{-20} 时即可成功停机.设置 $k_c=10^{-8}$,$\delta_r=10^{-12}$,则停机时 $U_c(x)<k_c\cdot\delta_r=10^{-20}$.基于上述辅助策略,拟物下降算法(quasi-physical descent algorithm,简称 QPD)的描述见算法 1.

算法 1.

输入:当前格局 X;最好格局 X_{best};

输出:拟物中止格局 X.

```
1    弹力粗调阶段连续迭代次数 l←0;
2    k_e=1.2;k_c=10^-8;δ_r=10^-12;h_e=1.2;h_c=1.0;
3    while true do
4        计算总势能 U(X);
5        if U(X)<10^-20 then
6            return X; //返回格局 X,成功退出;
7        end if
8        while U(X)>10^-10 do  //弹力粗调
9            计算每个圆饼的弹性合力,执行弹力动作,得到新格局 X';
10           l++;
11           自适应调整步长;
12           X=X';
13           if (l>15 and U_e(X)>1 and U_e(X)>U_e(X_best)) or 卡壳 then
14               return X;  // 提前中止策略(a)及卡壳中止
15           end if
16       end while
17       if U_e(X)<10^-10 and U(X)>10^-10 and l<3 then   //拉力粗调
18           if U_c(X)>10^-6 then
19               return X;  //提前中止策略(b)
20           else
21               计算系统的拉力,执行拉力动作,得到新格局 X';
22               X=X';
23           end if
24           l←0;
25       else if U_e(X)<10^-10   //精调
26           while U(X)>10^-20 do
27               if U_c(X)<k_c·δ_r then
28                   计算每个圆饼的弹性合力,执行弹力动作,得到新格局 X';
29               else
30                   计算每个圆饼的弹性合力及系统的拉力,执行综合动作,得到新格局 X';
31               end if
32               自适应调整步长;
33               X=X';
34               if 卡壳 then
35                   return X;   //卡壳中止
```

```
                end if
            end while
            l←0;
        end if
    end while
    return X
```

3.2 拟人跳坑策略

当拟物计算落入局部极小点的陷阱时,研究者多使用拟人策略进行跳坑.常用的方法有两种:一是按一定的规则选择两个圆饼互换位置.这种方法对不等圆 Packing 问题效果较好,但在求解等圆 Packing 问题时由于互换的两圆饼等大,格局未发生任何变化,没有达到跳坑的目的.另一种方法是挑出当前格局中受挤压较严重的一至多个圆,并重新随机放入空腔内以打破僵局.此策略不仅能使计算跳出局部极小点的陷阱,且能保留计算过程中积累的有用信息,但随机性较大,对于一些难解的问题,由于搜索空间非常大,让圆饼跳到合适位置上的概率较低.因此,如何找到更贴切的拟人策略以提高跳坑的效率,宣泄大量的选择过程中搜索空间的膨胀,是提高算法的全局搜索能力的一个关键难点.

观察日常生活中的一些现象,如在拥挤的公共汽车上,受挤压最严重的人通常会设法往车上最宽松的地方移动.受此乘车现象的启发设计得到以下跳坑策略:当搜索落入局部极小点的陷阱时,挑出受挤压最严重的若干圆饼,将其重新放置到当前格局下最空闲的若干区域.通过观察发现,半径较大的圆饼的位置相对比较固定,而半径较小的圆饼的位置相对比较灵活,因此,让小圆饼有更多的跳坑机会.用相对痛苦度来评价系统中圆饼的受挤压程度.

定义 2(相对痛苦度). 圆饼 $C_i(i\in\{1,2,...,n\})$的相对痛苦度见式(9).

$$P_i=\frac{\sum_{j=0,j\neq i}^{n}D_{ij}^2}{r_i^2} \tag{9}$$

当总体势能较大时,说明有多个圆饼受到较严重的挤压,选出相对痛苦度最大的几个圆饼进行跳坑;当总体势能接近 0 时,各圆饼嵌入程度已经很小,故只选择出相对痛苦度最大的一个圆饼进行跳坑.

由于外包络圆中的空闲空间为不规则区域,直接计算其面积或周长都很困难.本文借鉴求解矩形 Packing 问题的拟人算法中动作空间[21−24]的概念,通过化“圆”为“方”,将不规则的空闲空间近似为一系列规则的、容易计算的矩形空间,从而以较快的速度、较准确地找出给定格局下外包络圆中的最空闲区域.

定义 3(动作容器 G_0). 如图 1 所示,将外包络圆近似为两个矩形空间 *abcd* 和 *efgh* 的并集,称为动作容器 G_0. 其中,$ab=ef=\sqrt{2}R_0$,$ad=eh=2R_0$.G_0 关于原点对称.

定义 4(动作矩形块 G_i). 将圆饼 $C_i(i\in\{1,2,...,n\})$近似转换为动作矩形块 G_i,如图 1 中阴影区域所示.为了更好地近似圆饼,取 G_i 的边长为 C_i 的外接矩形和内接矩形边长的平均值,即 $\left(1+\frac{\sqrt{2}}{2}\right)\cdot r_i$.

定义 5(动作空间). 在当前格局下,往容器的剩余空间中放入一个尽可能大的虚拟矩形块,使其上、下、左、右 4 条边均与动作矩形块 G_i 或动作容器 G_0 的边相贴(即重合的长度大于 0),将该虚拟矩形块所占的空间称为一个动作空间.

图 1 给出了一个动作空间的示例.

往一个动作空间中放入一个动作矩形块后,若该矩形块所占的区域与原动作空间的 k 条边相交($k\in\{0,1,...,4\}$),则原动作空间被破坏,生成 $4-k$ 个新的矩形空间,如图 2 所示.

计算给定格局下的所有动作空间的具体过程可描述如下:

步骤 1. 初始化.

将外包络圆近似为两个矩形空闲空间,作为初始动作空间集合 *S*.

将待置入的圆饼集合近似为动作矩形块集合 *G*.

步骤 2. 对每个动作矩形块 G_i\{

对每个动作空间 S_j\{

若 G_i 与 S_j 重叠,则把 S_j 从 S 中删除,并把更新后的动作空间添加到 S 中.

\}\}

最终得到的 S 集即所有动作空间.

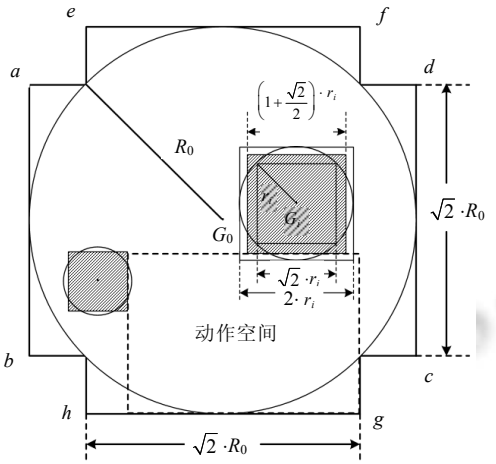

Fig.1 An example of squaring the circles

图 1 化圆为方示意图

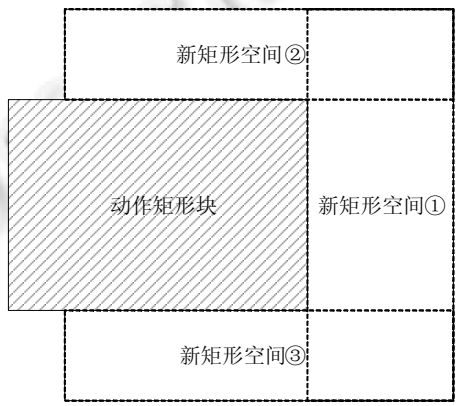

Fig.2 An example of updating action spaces

图 2 动作空间的更新(k=1)

定义 6(空闲度). 对给定的一个动作空间,其空闲度见式(10).

$$\text{空闲度}=\langle\text{短边长,长边长,中点横坐标,中点纵坐标,宽长}\rangle \tag{10}$$

两个动作空间空闲度的比较即按字典序比较各项的值.例如,若动作空间 S_a 的空闲度为⟨3,4,0,1,3⟩,动作空间 S_b 的空闲度为⟨3,4,1,0,4⟩,则 S_a 比 S_b 空闲.

为了避免重复搜索,在设计跳坑策略时结合了禁忌方法.具体做法为:设定一个禁忌表,若某个圆饼连续 3 次被选中,则将其加入禁忌表中并设置其禁忌长度为 3,同时将表中其他圆饼的禁忌长度减 1.若表中某个圆饼的禁忌长度减至 0 就将其解禁并从表中删除.为了提高算法的全局搜索能力,在每次跳坑时,依次按如下策略生成多个新的格局作为下一轮拟物搜索的起点.

(1) 挑出相对痛苦度最大且未被禁忌的圆饼,分别将其圆心置于空闲度最大和次大的动作空间的中心.

(2) 若 $U(X)>1$,则挑出相对痛苦度最大且未被禁忌的两个圆饼,分别将其圆心置于空闲度最大和次大的动作空间的中心.

(3) 挑出相对痛苦度最大且未被禁忌的圆饼,将其圆心置于随机选取的一个动作空间的中心.

其中,前两个策略的针对性较强,可能使算法迂回于某些区域,因此加入了随机性较强的第 3 个策略.

3.3 QPAS算法

综合上述拟物下降算法和基于动作空间的拟人跳坑策略,完整的基于动作空间的拟物算法(quasi-physical algorithm based on action space,简称 QPAS)可描述见算法 2.

算法 2.

输入:运行时间上限 L;

输出:最好格局 X_{best}.

```
1    随机生成 m 个格局;
2    运行时间 t←0;
3    while t<L do
4        当前最好格局 X_best←∅;
```

```
    for i←0 to m do
        对第 i 个格局 X_i 执行 QPD 算法,得到格局 X_i';
        if U(X_i')<10^-20 then
            return X_i';    //成功退出
        end if
        if U_e(X_i')<U_e(X_best) or X_best==∅ then
            X_best ← X_i';
        end if
    end for
    对 X_best 进行拟人跳坑,得到 m 个新格局;
end while
return X_best
```

4 计算结果与分析

本文将 QPAS 算法用 C++语言编程实现,并在 CPU 主频为 2.80 GHz、内存为 4 GB 的 PC 机上对 3 组共 13 个国际上公开的代表性算例进行了计算.第 1 组算例共 5 个,分别于 1994 年、1999 年和 2001 年由滕弘飞等人[9−11]提出,并于 2010 年由刘景发和李刚[17]进行了汇总.第 2 组算例共 6 个,于 2006 年由黄文奇等人[12]提出.这两组算例的具体数据见文献[17].

假设各圆饼均为单位厚度且密度均匀,物体的质量正比于其半径的平方.据此对文献[4]中的规模较大的两个算例进行转换,形成第 3 组算例,以进一步检验 QPAS 的性能.第 3 组算例见表 1.

Table 1 The third set of instances

表 1 第 3 组算例

算例	n	R_i	$m_i(i=1,2,\ldots,n)$
算例 3.1[4]	61	$R_i=20,i=1,2,\ldots,61$	$m_i = R_i^2$
算例 3.2[4]	91	$R_i=20,i=1,2,\ldots,91$	

对每个算例均独立计算 5 次,取其中最小的外包络圆半径 R_0 和相应的静不平衡量 J,并计算 5 次的平均耗时 $T(s)$.主要用外包络圆的半径来衡量计算结果的优劣.对于第 1 组算例,表 2 给出了 QPAS 与国内外文献中的代表性算法 MPSO[13]、APSO[14]、IQP[12]、ISS[16]、BF[17]、HSA[18]、TSH[19]和 QPCFA[20]的计算结果比较.在 9 种代表性算法中,QPAS 有一个算例找到了更小的外包络圆半径,其布局坐标见表 3,布局图如图 3 所示;有两个与当前的最好记录持平,另有两个排名第 2.在 9 种算法中,IQP、ISS、HSA、TSH、QPCFA 和 QPAS 都基于黄文奇等人[2]提出的能量函数进行优化,并在能量函数足够小时停机.其中,IQP 和 ISS 的停机条件分别是 $U(X)<10^{-6}$ 和 $U(X)<10^{-8}$,而 HSA、TSH、QPCFA 和 QPAS 的停机条件是 $U(X)<10^{-20}$,停机精度远高于前者,因此前者有可能得到更小的外包络圆半径.

对于第 2 组算例,由于提出的时间晚于第 1 组,目前只见 IQP[12]、HSA[18]、TSH[19]和 QPCFA[20]这 4 种算法对其进行了计算,表 4 给出了计算结果的比较.在这 5 种算法中,QPAS 有 2 个算例找到了更小的外包络圆半径,其布局坐标见表 5 和表 6,布局图如图 4 和图 5 所示;有 3 个持平于当前的最好记录,另有 1 个略弱于何琨等人[20]于 2013 年最新报道的结果.

由前两组算例的计算结果对比可见,对于静不平衡量,总的来说,QPAS 与 QPCFA 的精度基本相当,比其他算法的精度普遍要高出 1~7 个数量级.对于计算的时间,MPSO 和 APSO 取 40 次计算的平均时间,ISS 取 50 次计算的平均时间,QPCFA 取 1 次计算的时间,IQP、BF、HSA、TSH 和 QPAS 取 5 次计算的平均时间.各算法的编程语言和运行的硬件环境如下:MASP 在 CPU 主频为 166 MHz 的 PC 机上实验,编程语言和机器内存未见报道;APSO 使用 MATLAB 语言编程实现,在 CPU 主频为 2.0 GHz、内存为 256 MB 的 PC 机上进行计算;IQP 和 ISS 使用 C 语言编程实现,在 CPU 主频为 2.4 GHz、内存为 512 MB 的 PC 机上进行计算;HSA 使用 Java 语言编程实现,在 CPU 主频为 1.6 GHz、内存为 512 MB 的 PC 机上进行计算;BF 和 TSH 使用 Java 语言编程实现,在

CPU 主频为 1.6 GHz、内存为 1 GB 的笔记本上进行计算;QPCFA 使用 C++语言编程实现,在 CPU 主频为 1.9 GHz、内存为 1 GB 的 PC 机上进行计算.虽然计算环境有所差异,但基本处于同一数量级.从两个表中可见,QPAS 与其他几种算法的计算时间相当甚至略快.

Table 2 Computing results on the first set of instances

表 2 第 1 组算例计算结果比较

算例	n	MPSO			APSO			IQP		
		R_0	J	$T(s)$	R_0	J	$T(s)$	R_0	J	$T(s)$
算例 1.1	5	120.711	2.712×10^{-4}	287	120.710 7	7.03×10^{-4}	263	120.710	7.03×10^{-4}	3.82
算例 1.2	9	–	–	–	–	–	–	–	–	–
算例 1.3	5	–	–	–	–	–	–	22.327	3.9×10^{-4}	2.91
算例 1.4	7	31.985	1.82×10^{-6}	1 002	31.885	2×10^{-6}	982	31.981	4.11×10^{-3}	76.33
算例 1.5	40	843.94	3.9×10^{-4}	2 523	820.987	7.26×10^{-4}	2 318	742.75	5.4×10^{-4}	12.27

算例	n	ISS			BF			HSA		
		R_0	J	$T(s)$	R_0	J	$T(s)$	R_0	J	$T(s)$
算例 1.1	5	**120.710 2**	8.45×10^{-8}	1.275	120.710 678 2	3.89×10^{-9}	0.594	120.710 678 2	4.30×10^{-9}	0.969
算例 1.2	9	1.000 005	1.78×10^{-4}	4.63	1.000 000	9.70×10^{-9}	2.891	–	–	–
算例 1.3	5	–	–	–	22.249 0	3.12×10^{-9}	1.313	22.246 3	3.52×10^{-9}	0.469
算例 1.4	7	31.954	5.68×10^{-14}	20.18	31.854 6	1.50×10^{-9}	54.828	31.841 2	2.03×10^{-9}	58.797
算例 1.5	40	740.58	4.01×10^{-5}	20.21	725.043 5	2.47×10^{-9}	7.187	716.678 2	2.86×10^{-9}	154.594

算例	n	TSH			QPCFA			QPAS			
		R_0	J	$T(s)$	R_0	J	$T(s)$	R_0	J	$T(s)$	Rank
算例 1.1	5	120.710 678 2	3.25×10^{-9}	0.350	120.710 678 2	2.11×10^{-10}	0.046	120.710 678 1	3.63×10^{-12}	0.243	2
算例 1.2	9	**1.000 000 0**	2.38×10^{-9}	0.225	**1.000 000 0**	2.00×10^{-12}	0.172	**1.000 000 0**	6.88×10^{-13}	0.405	1
算例 1.3	5	**22.246 203 3**	2.63×10^{-9}	0.506	**22.246 203 3**	3.44×10^{-10}	0.015	**22.246 203 3**	1.03×10^{-10}	1.014	1
算例 1.4	7	31.841 133 6	5.89×10^{-9}	27.766	31.842 500 0	4.11×10^{-10}	1.406	**31.841 131 1**	7.69×10^{-10}	23.889	1*
算例 1.5	40	716.610 931 9	1.77×10^{-9}	63.263	**712.406 250 0**	1.60×10^{-10}	171.719	713.000 735 0	8.58×10^{-11}	13.230	2

注:最小的 R_0 加粗显示,QPAS 的 Rank 列加'*'表示在该算例上找到了更小的 R_0

Table 3 The detailed coordinates of the disks for instance 1.4

表 3 QPAS 在算例 1.4 上的布局坐标

序号	x	y	r	序号	x	y	r
1	19.3112968661142940	–11.4635142724843050	8.5	5	5.1822580360157149	20.1865536743619420	11.0
2	21.1287846797804700	6.6410930363183311	9.5	6	–16.4613174967613850	–11.9493364971069820	11.5
3	2.7148316941352619	–21.6717488627866220	10.0	7	–16.1339803280241370	11.5483836131007860	12.0
4	2.7179319590434856	–1.1717490902839900	10.5				

Table 4 Computing results on the second set of instances

表 4 第 2 组算例计算结果比较

算例	n	IQP			HAS			TSH		
		R_0	J	$T(s)$	R_0	J	$T(s)$	R_0	J	$T(s)$
算例 2.1	7	60.00	3.6×10^{-3}	7.78	60.000	1.59×10^{-9}	1.031	**60.000 000 0**	7.38×10^{-9}	0.206
算例 2.2	12	215.47	9.5×10^{-3}	444.78	215.470 054	2.74×10^{-7}	11.562	215.470 053 9	3.77×10^{-7}	3.042
算例 2.3	15	39.78	7.6×10^{-3}	91.92	39.123 4	2.97×10^{-8}	130.362	39.065 104 8	4.13×10^{-8}	365.797
算例 2.4	17	49.72	5.1×10^{-3}	157.92	49.507 8	6.83×10^{-8}	123.407	49.368 194 2	2.50×10^{-8}	597.719
算例 2.5	37	135.176	6.7×10^{-3}	18.29	135.175 410	1.23×10^{-8}	0.203	**135.175 409 7**	4.91×10^{-8}	0.425
算例 2.6	50	159.57	8.0×10^{-3}	348.97	158.967 8	4.81×10^{-9}	453.625	158.967 698 1	3.49×10^{-9}	1 078.386

算例	n	QPCFA			QPAS			
		R_0	J	$T(s)$	R_0	J	$T(s)$	Rank
算例 2.1	7	**60.000 000 0**	2.66×10^{-9}	0.015	**60.000 000 0**	3.54×10^{-10}	0.108	1
算例 2.2	12	215.470 053 9	2.92×10^{-8}	0.391	**215.470 053 8**	2.91×10^{-11}	1.279	1*
算例 2.3	15	**39.050 390 7**	8.90×10^{-10}	36.343	**38.998 235 1**	1.19×10^{-9}	16.310	1*
算例 2.4	17	**49.359 012 5**	8.51×10^{-10}	10.562	49.367 764 9	2.00×10^{-9}	22.348 8	2
算例 2.5	37	**135.175 409 7**	6.95×10^{-9}	0.234	**135.175 409 7**	3.66×10^{-11}	1.467 8	1
算例 2.6	50	**158.964 400 0**	5.96×10^{-9}	842.906	**158.964 400 0**	8.00×10^{-9}	312.131	1

注:最小的 R_0 加粗显示,QPAS 的 Rank 列加'*'表示在该算例上找到了更小的 R_0

Table 5 The detailed coordinates of the disks for instance 2.2

表 5 QPAS 在算例 2.2 上的布局坐标

序号	x	y	r	序号	x	y	r
1	80.1351323284289380	−174.2023068567893600	23.72	7	8.4862674099740953	−166.9895225393457300	48.26
2	−190.9252746029229700	17.7190075114753010	23.72	8	−148.8655241280515600	76.1366994473319070	48.26
3	−126.1192497662789800	144.4281879554832200	23.72	9	140.3744397256365100	90.8527312565984600	48.26
4	188.1446426120991200	37.0091421663224690	23.72	10	−5.8696703055848909	115.3207713477796400	100.00
5	−62.0317319779710490	−181.4379769711466100	23.72	11	102.8055527267827500	−52.5771020719231150	100.00
6	110.8164212362197800	156.4843261553089900	23.72	12	−96.9358824217438840	−62.7436692651842520	100.00

Table 6 The detailed coordinates of the disks for instance 2.3

表 6 QPAS 在算例 2.3 上的布局坐标

序号	x	y	r	序号	x	y	r
1	−34.4285028640997980	15.3599224650793880	1	9	29.5978182608661910	4.8850041441504137	9
2	33.7982848157157110	15.0514234115894800	2	10	19.8145470847353020	21.1726560649652740	10
3	−21.6524602536760750	28.4376011278335700	3	11	24.0525861758465070	−14.3308851436723210	11
4	12.0850842044562640	32.8455050566270970	4	12	−3.0581017131112964	26.7483741738768860	12
5	−14.3018406622605100	−30.8437570028816100	5	13	2.9847288377083285	−25.8263358602666740	13
6	−25.3871342260474290	20.2490597241375770	6	14	5.2082358848268644	2.0974581114548387	14
7	−30.5954319015216850	8.3379877604766062	7	15	−21.0677897267708370	−11.4918895611060560	15
8	−15.0382783779291960	10.7043541958554160	8				

对于第 3 组算例,表 7 给出了 QPAS 与 PA[5]和 SATS[4]的计算结果的比较.在这 3 种算法中,QPAS 的计算结果很接近于典型的圆形 Packing 问题的最好结果.由于带平衡约束的圆形 Packing 问题比典型的圆形 Packing 问题多了一项平衡约束,后者的最优解一定好于等于前者的最优解.因此 QPAS 计算结果略弱于典型圆形 Packing 问题的解是可以理解的.另外,QPAS 在算例 3.1 找到了更小的外包络圆半径.QPAS 在第 3 组算例取得的布局如图 6 和图 7 所示.

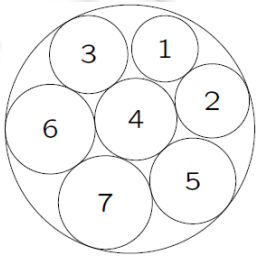

Fig.3 Packing layout of instance 1.4

图 3 算例 1.4 的布局图

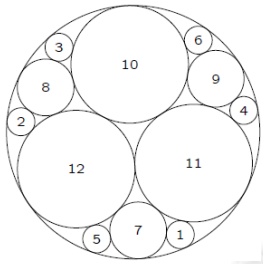

Fig.4 Packing layout of instance 2.2

图 4 算例 2.2 的布局图

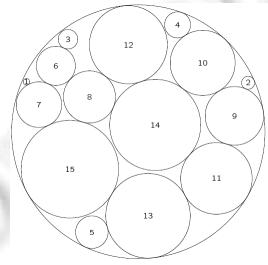

Fig.5 Packing layout of instance 2.3

图 5 算例 2.3 的布局图

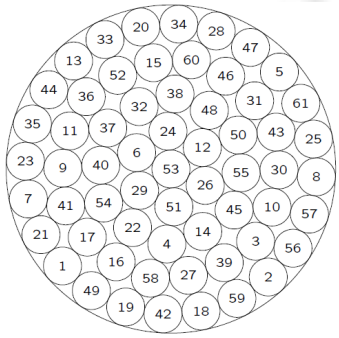

Fig.6 Packing layout of instance 3.1

图 6 算例 3.1 的布局图

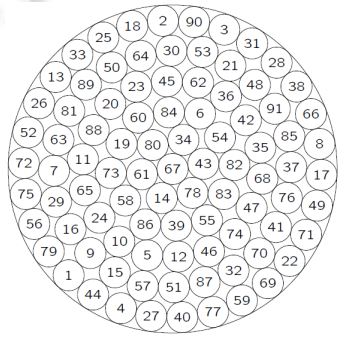

Fig.7 Packing layout of instance 3.2

图 7 算例 3.2 的布局图

Table 7 Computing results on the third set of instances

表 7 第 3 组算例计算结果比较

算例	n	PA		SATS		QPAS		
		R_0	$T(s)$	R_0	$T(s)$	R_0	J	$T(s)$
算例 3.1	61	173,226	0.32	173.226	0.52	**173.225 951 5**	2.40×10^{-8}	345.423
算例 3.2	91	–	–	211.334	4.85	211.335 449 2	3.53×10^{-8}	453.229

5 结 论

对于带平衡约束的圆形 Packing 问题,基于弹力模型和拉力模型,把拟物下降方法与若干拟人策略相结合以提高连续优化的效率.当拟物过程遇到局部陷阱需要跳坑时,由于外包络圆中的剩余空间都是不规则的,很难通过精确计算从中选出最空闲的空间以作为跳坑的位置.本文通过化圆为方,将外包络圆和待置入圆饼近似看做矩形,并借鉴求解矩形 Packing 问题的拟人算法中动作空间的概念,把不规则的剩余空间转化为易计算和比较的矩形空间,从而较快且较准确地找到最空闲空间.对国际上公开的 13 个代表性算例的计算结果表明,所提的拟物求解算法是快速且有效的.在今后的工作中,拟进一步提高基于动作空间的跳坑方法的有效性,并将其应用到一般的圆形 Packing 问题的求解中去.

References:

[1] Kang Y, Huang WQ. A fast quasi-physical algorithm for the disks packing problem. Computer Engineering and Applications, 2003, 39(35):30−32 (in Chinese with English abstract).

[2] Wang HQ, Huang WQ, Zhang Q, Xu DM. An improved algorithm for the packing of unequal circles within a larger containing circle. European Journal of Operational Research, 2002,141(2):440−453.

[3] Liu JF, Zhou GC, Pan JJ. Heuristic algorithm based on Taboo search for sphere packing problem. Application Research of Computers, 2011,28(3):892−894 (in Chinese with English abstract).

[4] Zhang D, Deng A. An effective hybrid algorithm for the problem of packing circles into a larger containing circle. Computers & Operations Research, 2005,32(8):1941−1951.

[5] Zhang DF, Li X. A personified annealing algorithm for circles packing problem. Acta Automatica Sinica, 2005,31(4):590.

[6] Theodoracatos VE, Grimsley JL. The optimal packing of arbitrarily-shaped polygons sing simulated annealing and polynomial-time cooling schedules. Computer & Methods in Applied Mechanics and Engineering, 1995,125(1):53−70.

[7] Graham RL, Lubachevsky BD, Nurmela KJ, Östergård PRJ. Dense packing of congruent circles in a circle. Discrete Mathematics, 1998, 181(1):139−154.

[8] Yu Y, Cha JZ, Tang XJ. Learning based GA and application in packing. Chinese Journal of Computers, 2001,24(12):1242−1249 (in Chinese with English abstract).

[9] Teng HF, Sun SL, Ge WH, Zhong WX. Layout optimization for the dishes installed on a rotating table-the packing problem with equilibrium behavioural constraints. SCIECE IN CHINA (Series A), 1994,37(10):1272−1280.

[10] Tang F, Teng HF. A modified genetic algorithm and its application to layout optimization. Ruan Jian Xue Bao/Journal of Software, 1999,10(10):1096−1102 (in Chinese with English abstract). [doi: 10.13328/j.cnki.jos.1999.10.014]

[11] Qian ZQ, Teng HF, Sun ZG. Human-Computer interactive genetic algorithm and its application to constrained layout optimization. Chinese Chinese Journal of Computers, 2001,24(5):553−559 (in Chinese with English abstract).

[12] Huang WQ, Chen M. Note on: An improved algorithm for the packing of unequal circles within a larger containing circle. Computers & Industrial Engineering, 2006,50(3):338−344.

[13] Li N, Liu F, Sun DB. A study on the particle swarm optimization with mutation operator constrained layout optimization. Chinese Journal of Computers, 2004,27(7):897−903 (in Chinese with English abstract).

[14] Lei KY, Qiu YH. A study of constrained layout optimization using adaptive particle swarm optimizer. Journal of Computer Research and Development, 2006,43(10):1724−1731 (in Chinese with English abstract).

[15] Liu J, Huang WQ. A fast local search algorithm for solving circles packing problem with constraints of equilibrium. Journal of Image and Graphics, 2008,13(5):991−997 (in Chinese with English abstract).

[16] Wang YS, Shi YJ, Teng HF. An improved scatter search for circles packing problem with the equilibrium constraint. Chinese Journal of Computers, 2009,32(6):1214−1221 (in Chinese with English abstract).

[17] 刘景发,李刚.求解带平衡性能约束的圆形装填问题的吸引盘填充算法.中国科学:信息科学,2010,40(3):423−432.

[18] Liu JF, Li G, Chen DB, Liu WJ, Wang YL. Two-Dimensional equilibrium constraint layout ssing simulated annealing. Computers & Industrial Engineering, 2010,59(4):530−536.

[19] 李刚,刘景发.基于禁忌搜索的启发式算法求解带平衡约束的圆形装填问题.中国科学:信息科学,2011,41(9):1076−1088.

[20] He K, Mo DZ, Xu RC, Huang WQ. A quasi-physical algorithm based on coarse and fine adjustment for solving circles packing problem with constraints of equilibrium. Chinese Journal of Computers, 2013,26(6):1224−1234 (in Chinese with English abstract).

[21] He K, Huang WQ. An efficient placement heuristic for three-dimensional rectangular packing. Computer & Operations Research, 2011,38(1):227−233.

[22] 何琨,黄文奇.三维矩形 Packing 问题的拟人求解算法.中国科学:信息科学,2010,40(12):1586−1595.

[23] He K, Huang WQ, Jin Y. An efficient deterministic heuristic for two-dimensional rectangular packing. Computers & Operations Research, 2012,39(7):1355−1363.

[24] He K, Huang WQ, Jin Y. Efficient algorithm based on action space for solving the 2D rectangular packing problem. Ruan Jian Xue Bao/Journal of Software, 2012,23(5):1037−1044 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3986.htm [doi: 10.3724/SP.J.1001.2012.03986]

附中文参考文献:

[1] 康雁,黄文奇.求解圆形 Packing 问题的一个快速拟物算法.计算机工程与应用,2003,39(35):30−32.

[3] 刘景发,周国城,潘锦基.基于禁忌搜索的启发式算法求解球体 Packing 问题.计算机应用研究,2011,28(3):892−894.

[8] 于洋,查建中,唐晓君.基于学习的遗传算法及其在布局中的应用.计算机学报,2001,24(12):1242−1249.

[10] 唐飞,滕弘飞.一种改进的遗传算法及其在布局优化中的应用.软件学报,1999,10(10):1096−1102. [doi: 10.13328/j.cnki.jos.1999.10.014]

[11] 钱志勤,滕弘飞,孙治国.人机交互的遗传算法及其在约束布局优化中的应用.计算机学报,2001,24(5):553−559.

[13] 李宁,刘飞,孙德宝.基于带变异算子粒子群优化算法的约束布局优化研究.计算机学报,2004,27(7):897−903.

[14] 雷开友,邱玉辉.基于自适应粒子群优化算法的约束布局优化研究.计算机研究与发展,2006,43(10):1724−1731.

[15] 刘建,黄文奇.求解带平衡约束圆形 Packing 问题的快速局部搜索算法.中国图像图形学报,2008,13(5):991−997.

[16] 王奕首,史彦军,滕弘飞.用改进的散射搜索法求解带平衡约束的圆 Packing 问题.计算机学报,2009,32(6):1214−1221.

[20] 何琨,莫旦增,许如初,黄文奇.基于粗精调技术的求解带平衡约束圆形 Packing 问题的拟物算法.计算机学报,2013,26(6):1224−1234.

[24] 何琨,黄文奇,金燕.基于动作空间的求解二维矩形 Packing 问题的高效启发式算法.软件学报,2012,23(5):1037−1044. http://www.jos.org.cn/1000-9825/3986.htm [doi: 10.3724/SP.J.1001.2012.03986]

何琨(1972－),女,北京人,博士,教授,CCF 高级会员,主要研究领域为 NP 难度问题现实求解,算法设计与分析.

杨辰凯(1988－),男,硕士生,主要研究领域为 NP 难度问题现实求解,算法设计与分析.

黄梦龙(1988－),男,硕士生,主要研究领域为 NP 难度问题现实求解,算法设计与分析.

黄文奇(1938－2013),男,教授,博士生导师,主要研究领域为 NP 难度问题现实求解,算法设计与分析.