软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.006161]

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

求解二维装箱问题的强化学习启发式算法*

阳名钢, 陈梦烦, 杨双远, 张德富

(厦门大学 信息学院,福建 厦门 361005)

通讯作者: 张德富, E-mail: dfzhang@xmu.edu.cn

摘 要: 二维带形装箱问题是一个经典的 NP-hard 的组合优化问题,该问题在实际的生活和工业生产中有着广泛的应用,研究该问题对企业节约成本,节约资源以及提高生产效率有着重要的意义.本文提出了一个强化学习求解算法.我们新颖地使用强化学习为启发式算法提供一个初始的装箱序列,来有效的改善启发式冷启动的问题.该强化学习模型能进行自我驱动学习,仅使用启发式计算的解决方案的目标值作为奖励信号来优化网络,使网络能学习到更好的装箱序列.我们使用简化版的指针网络来解码输出装箱序列,该模型由嵌入层、解码器和注意力机制组成.使用了 Actor-Critic 算法对模型进行训练,提高了模型的效率.我们在 714 个标准问题实例和随机生成的 400 个问题实例上测试提出的算法,实验结果显示,提出的算法能有效地改善启发式冷启动的问题,性能超过当前最优秀的启发式求解算法.

关键词: 二维装箱问题;强化学习;指针网络;启发式算法;分层搜索

中图法分类号: TP301

Reinforcement Learning Heuristic Algorithm for Solving the Two-dimensional Strip Packing Problem

YANG Ming-Gang, CHEN Meng-Fan, YANG Shuang-Yuan, ZHANG De-Fu

(School of Informatics, Xiamen University, Xiamen 361005, China)

Abstract: The two-dimensional strip packing problem is a classic NP-hard combinatorial optimization problem, which has been widely used in practical life and industrial production. This paper proposes a reinforcement learning heuristic algorithm for it. The reinforcement learning is used to provide an initial boxing sequence for the heuristic algorithm to effectively improve the heuristic cold start problem. The reinforcement learning model can perform self-driven learning, using only the value of the heuristically calculated solution as a reward signal to optimize the network, so that the network can learn a better packing sequence. We use a simplified version of the pointer network to decode the output boxing sequence. The model consists of an embedding layer, a decoder, and an attention mechanism. Actor-Critic algorithm is used to train the model, which improves the efficiency of the model. We test the reinforcement learning heuristic algorithm on 714 standard problem instances and 400 generated problem instances. Experimental results show that the proposed algorithm can effectively improve the heuristic cold start problem and outperform the state-of-the-art heuristics with much higher solution quality.

Key words: Two-dimensional Strip Packing Problem; Reinforcement Learning, Pointer Network; Heuristics; Hierarchical Search

二维带形装箱问题是指将一组矩形小物品装入一个具有给定宽度,但是高度无限的矩形箱子中,这些矩形只能按照固定方向放置,不能相互重叠,且必须完全装入箱子中.问题的目标是使得使用的高度,尽可能的低.

二维装箱问题是经典的组合优化问题,在各行各业中都有着非常广泛的应用.如在石材切割中,物品是小块

* 基金项目: 国家自然科学基金(61672439)
Foundation item: National Natural Science Foundation of China (61672439)
收稿时间: 2020-07-14; 修改时间: 2020-08-20; 采用时间: 2020-09-17; jos 在线出版时间: 2021-05-20

的石材,箱子则对应待切割的完整石板,目标是使石材的利用率更高.由于装箱问题广泛的存在于各个领域,高效的求解算法有助于公司节省材料和减少资源浪费,提高生产和工作效率,帮助更合理利用有限的资源.多年来,二维带形装箱问题（2DSPP）被众多国内外学者研究以获得更快更优的解决方案.同其他组合优化问题一样,2DSPP 可通过精确算法、近似算法(启发式和元启发式)或者混合算法等来求解[1].

精确算法可以找到问题的最佳解决方案,但在较大的问题实例上通常需要花费大量的时间.Hifi[2]基于分支定界过程提出了一种精确算法用于解决带状切割和包装问题,该算法可以在可接受的执行时间内解决一些中小规模问题实例.Martello 等人[3]提出了一种新的松弛方法,该方法可产生良好的下界,基于此下界构造出较好的启发式算法.Kenmochi[4]等人设计了一个基于分支规则和边界操作的分支定界算法.Côté 等人[5]提出了一种基于 Bender 分解的精确算法,该方法可以在有限的计算时间内解决中小规模的基准实例.大多数精确算法都是基于分支定界策略或者混合整数线性规划,但是因为装箱问题的复杂性,当问题实例增大的时候,不可避免的导致计算量的指数级增长,耗时增加.故大部分精确算法仅在小规模的数据集上表现良好,对于解决大规模的问题实例,更多的研究重点则倾向于使用近似算法来解决.

启发式算法虽然无法保证获得的解是最优的,但是它能在合理的时间范围内给出良好的解,启发式被认为是解决大规模组合优化问题的有效解决方案,因此在装箱领域中十分流行.大多数启发式算法都包含构造性过程,即每次将一块添加到现有的部分解决方案中.1980 年,Baker 等[6]为了构造合理的装填顺序就提出了左下(bottom-left, BL)启发式算法.该算法每次取出列表中的第一个矩形,放入容器最左最低的可用空间.Chazelle 等[7]对 BL 算法进行了改进,提出了左下填充算法(BLF),算法首先确定即将被放入的矩形合适的所有位置,然后选择其中最低位置.Huang 和 Liu[8]提出基于欧氏距离的确定性启发式求解算法.Burke 等[9]提出了最佳填充算法(BF), 该算法会动态地在列表中搜索放入可用空间的最佳候选矩形.Leung 等[10]提出了适合度策略,该策略用于决定哪个矩形要填充到指定位置.

元启发式算法常见的有模拟退火(SA)、遗传算法(GA)、蚁群算法(AC)和禁忌搜索算法(TS)等,大多数研究者会结合构造性启发式和元启发式算法来解决二维带形装箱问题.Jakobs[11]在提出的遗传算法中,使用一个特定的交叉算子和变异算子来交换某些元素的位置,并使用 BL 算法解码解决方案.Hopper 和 Turton[12]混合 BL 方法与三种元启发算法(遗传算法(GA)、模拟退火算法(SA)、朴素演化(NE))和局部搜索启发式算法.其他作者[13,14]也提出多种结合遗传算法的方法来解决 2DSPP.He 等人[15]提出动态归约算法求解面积最小的装箱问题.Wei 等人[16]开发了一种基于配置文件的构造启发式算法的禁忌搜索算法.Alvarez-Valdés[17]提出了一种针对带状包装问题的贪婪随机自适应搜索程序(GRASP).Zhang 等人[18]提出基于随机局部搜索的二分搜索启发式算法.Leung 等人[19] 提出两阶段智能搜索算法(ISA),该算法基于简单的评分规则选择矩形放入,该评分策略中设置了 5 个分值(0~4).在简单随机算法(SRA)[20]中使用了具有 8 个分值的评分规则,并引入了最小浪费优先策略.Chen 等人[21]提出了新的算法框架 CIBA,提出了角增量的概念,只包含了 4 个评分值,用于在选择矩形的时候评价矩形.Chen 等人[22]基于评分规则以及角增量的基础上,再结合分层搜索的思想,提出了一个混合启发式算法.这些不同的评分规则设计的大致思路都是想实现更平坦平滑的布局以此来减少空间的浪费.值得注意的是,Chen 等人[22]研究了解的不同初始化策略,如贪心策略,最后在 6 种贪心策略中选择最好的初始序列,取得了不错的效果.

深度学习算法在分类、回归任务和图像识别等方面取得了非凡的成功,但是将机器学习用于组合优化问题的求解,进展仍然缓慢.1985 年就有学者在组合优化问题中使用神经网络用于优化决策,该论文使用 Hopfield 网络解决小规模的旅行商问题(TSP) [23].组合优化问题大部分都是序列的决策问题,例如旅行商问题就是求解访问城市顺序的问题.Vinyals 等人[24]设计了一种非常适合用于求解组合优化问题的神经网络,指针网络(Pointer Network,Ptr-Net),该模型使用注意力机制[25]作为指针来选择输入序列中的某一个元素输出.作者将该模型应用在解决旅行商问题上.Bello 等人[26]采用了 Actor-Critic 算法来训练 Ptr-Net 并在多达 100 个节点的 TSP 问题上获得最佳结果.受到 Bello 等人 [26]工作的启发,Nazari 等人[27]提出了一个强化学习解决车辆路径问题的端到端框架,该框架中简化了指针网络,采用 Actor-Critic 算法训练模型.Kool 等人[28]提出了一个基于注意力机制的模

型,并用该模型求解了各类型的TSP和VRP,以及其他组合优化问题.Hu等人[29]提出了一个新型的3D装箱问题,受到深度强化学习(DRL)尤其是指针网络在组合优化问题上的应用启发,使用 DRL 方法来优化要装填的物品的顺序,结合启发式算法,将网络的输出序列转换成解决方案,用于计算奖励信号,实验结果优于之前精心设计的启发式算法.

可以看到强化学习在组合优化问题上的研究和应用处于初步的研究阶段,性能还不是很好,但是新的思路不断涌出,研究人员也都在尝试能设计更好的模型来解决组合优化问题.目前强化学习框架的研究大多只在小规模的数据集上获得较为良好的解,如果问题规模增大,信息量爆炸,网络存储能力有限,可能无法获得理想的解.

本文研究了如何利用强化学习来生成一个好的初始解,最终提出了一个强化学习启发式算法,计算结果表明提出的方法超过了当前最优秀的算法.最后,也研究了不同初始化策略的影响.

1 启发式算法

由于强化学习需要奖励机制,本文利用构造性算法获得的高度作为奖励机制.矩形的选择方法是整个构造性启发式算法的关键.本文采用文献[22]提出的矩形选择策略—δ评分规则.算法 1.1 详细地介绍了基于δ评分规则和红黑树的构造性启发式算法的过程.算法中将迭代放置所有的矩形,在每次选择一个放置空间时,会先判断放置空间是否可用,否则会进行$Cave\ smooth$操作,将不可使用的空间合并到 y 坐标差异较小的相邻 $cave$.详细地构造性启发式算法的介绍参考文献[22].

算法 1.1 构造性启发式算法

Algorithm 1.1: ConstructiveHeuristic

Input:
$Layout.cave\_list$ represents all caves in the layout
Seq: unpacked sequence of rectangles
Output:
H: a packing solution, which indicates the current highest y-coordinate of packed rectangles

```
1  w = the current smallest width of Seq
2  Create two red-black trees based on Seq, T_whi and T_hwi
3  while Seq.length > 0 do
4     get the lowest and most left cave C from Layout.cave_list
5     if C.w < w then
6        Cave smooth
7        continue
8     R = RectangleSelection(C,T_whi,T_hwi)
9     pack R into the cave C
10    Cave smooth if necessary
11    update H
12 return H
```

文献[22]提出的混合启发式算法 HHA, 取得了不错的结果, 但是需要一个好的初始解. 本文用强化学习帮助 HHA 找到一个更好的初始解. 在此, 给出 HHA 的算法流程, 如算法 1. 2 所示. 其中涉及的函数如构造性算法 ConstructiveHeuristic、分层搜索 HierachicalSearch 等, 可以参考文献[22].

算法 1.2 HHA 框架

Algorithm 1.2: Overall hybrid heuristic algorithm

Input:

W：the width of the strip

Seq：unpacked sequence of rectangles

Output:

H：a packing solution

1 $EmptyLayout$ only have one cave, which consists of the whole strip, with a width equal to W
2 $bestSep = null, H = \infty$
3 **for** $each\ sortig\ rule$ **do**
4 $sortSeq = using\ the\ sorting\ rule\ sort\ Seq$
5 $tempH =$ ConstructiveHeuristic($EmptyLayout.cave\_list,\ sortSeq$)
6 **if** $tempH < H$ **then**
7 $bestSep = sortSeq$
8 $H = tempH$
9 **while** $the\ stop\ criterion\ is\ not\ satisfied$ **do**
10 $tempH$ = HierachicalSearch($bestSep$)
11 **if** $tempH < H$ **then**
12 $bestSep = sortSeq$
13 $H = SwapRandom(bestSeq)$
14 **return** H

2 强化学习启发式算法

2.1 强化学习求解框架

启发式算法通常会存在冷启动的问题,每次运行算法都需要重新开始缓慢的爬山搜索来寻找最优解,无论算法运行过多少次并不能对后续的运行和搜索产生任何帮助.传统的启发式算法不存在学习或者保存原先经验的能力,而强化学习则可以做到.强化学习是机器学习的一个重要分支,它的算法框架如图 2.1 所示,Agent 是智能体,它会根据所接收的环境状态(state)采取相应的动作(action),环境会根据采取动作给予 Agent 不同的反馈(reward),然后 Agent 根据反馈的情况调整和修正自己的行为.Agent 会反复的与环境互动进行探索和学习,目的是使获得的奖励最大.因此我们考虑使用强化学习的方法通过训练模型来学习装箱经验,为启发式算法寻找一个较优的初始序列,从而优化冷启动的问题.那么这就需要一个网络模型能够学习根据输入的物品序列输出一个初始装箱序列.

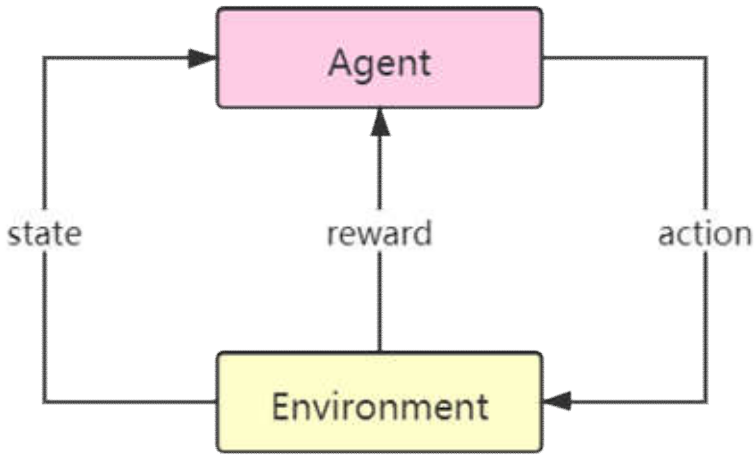

Fig.2.1 Reinforcement Learning graph

图 2.1 强化学习示意图

指针网络非常适合解决输出序列强依赖输入序列的问题.而二维装箱问题也可以看成是序列到序列的问题,它的装箱序列完全来自输入的物品,因此可以利用指针网络来学习输出一个合适的装箱初始序列.对于训练好的网络模型其优势在于,当输入全新的数据时,模型可以根据学习到的经验给出一个较理想的初始序列,好的初始序列不仅可以加快启发式搜索算法的响应速度,还可以节省爬山消耗的时间提高算法性能.

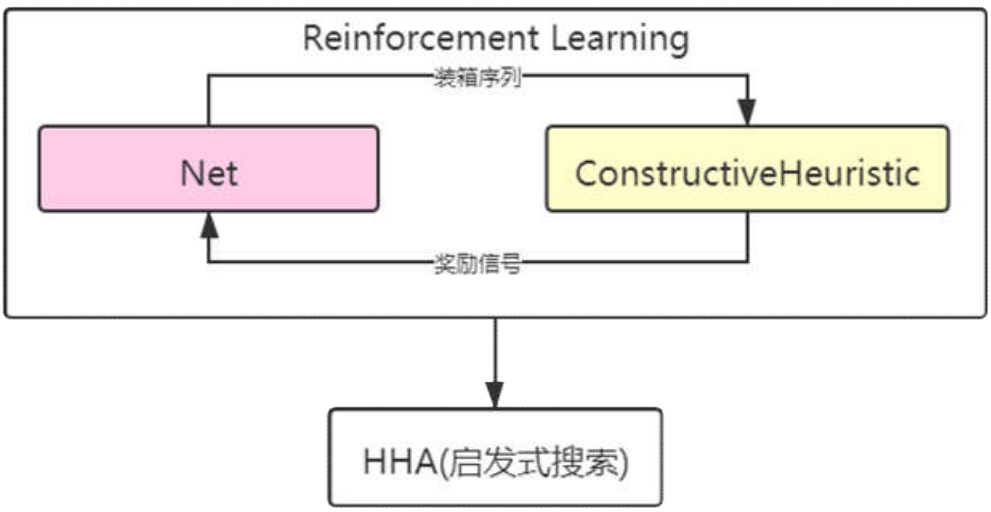

Fig.2.2 Reinforcement Learning heuristic algorithm framework

图 2.2 强化学习启发式算法框架

2DSPP 主要有两个决策变量：(1) 矩形物品放入箱子的顺序;(2) 物品摆放的位置.算法 1.1 可以用于解决

输入物品的序列如何放置物品的问题.而提出的强化学习方法用于优化物品的放入顺序.

本文提出的框架是使用强化学习训练好的模型为启发式方法提供初始的装箱序列,改善冷启动问题.具体的框架图 2.2 所示,强化学习方法是一种自我学习驱动的过程,该过程仅依赖输出序列所计算获得的奖励值来训练网络.网络模型(即图 2.2 中的 Net)采用简化版的指针网络输出一个初始的装箱序列,然后调用 ConstructiveHeuristic（即算法 1.1)计算该装箱序列所获得的解,将该解获得的高度值作为奖励信号来验证网络的可行性,根据奖励值不断的调整网络,使网络能以最大的概率输出较好的装箱序列.训练好的模型将为 HHA 提供初始序列,整体的算法本文称之为强化学习启发式算法(RLHA),详细的算法流程如算法 2.1.

算法 2.1 强化学习启发式算法(RLHA)

Algorithm 2.1: RLHA

Input:

W：the width of the strip

Seq：unpacked sequence of rectangles

Output:

H：a packing solution

1 Load the trained reinforcement learning model
2 $EmptyLayout$ only have one cave, which consists of the whole strip, with a width equal to W
3 $H = \infty$
4 Obtaining initial solution *bestSeq* from reinforcement learning
5 **while** $the\ stop\ criterion\ is\ not\ satisfied$ **do**
6 $tempH$ = HierachicalSearch($bestSep$)
7 **if** $tempH < H$ **then**
8 $bestSep = sortSeq$
9 $H = SwapRandom(bestSeq)$
10 **return** H

2.2 网络模型

网络模型使用的是简化版的指针网络,该模型结构与 Nazari[27]等人使用的结构一致,他们的框架考虑了静态和动态的元素.针对二维装箱问题我们仅考虑了静态元素,因为装箱问题与 VRP 不同,2DSPP 不存在动态的元素,比如 VRP 的配送点除了坐标,还有配送点的货物量,这个值是会动态改变的.模型由循环神经网络(GRU)和注意力机制组成,每一步把输入元素的嵌入(Embedding)直接作为解码器的输入,解码器的输出会传递给注意力机制,获得输出的概率分布.Nazari[27]等人认为循环神经网络编码器增加了额外的复杂性,因为只有当输入需要传递顺序信息的时候才需要循环神经网络,比如文本翻译,需要知道词的先后顺序才能准确翻译.但是在组合优化中输入集并没有顺序之说,任何随机排列都包含与原始输入相同的信息.因此在简化版指针网络中,忽略了编码器,直接使用了嵌入层(Embedding layer)代替编码器.本文同样使用了简化版指针网络,因为通过这种修改减少了计算的复杂性,提高了性能.

本文的网络模型如图 2.3 所示,该模型由两个部分组成,首先嵌入层(Embedding)对输入的序列嵌入映射成 D 维的向量空间,使用的是一维卷积层对输入进行嵌入.右边是解码器保存序列的信息,它将利用注意力机制将解码器的隐藏状态和嵌入的输入指向一个输入元素.

设输入集为$X = \{x_i, i = 1, \dots, N\}$,其中$x_i = (w_i, h_i)$是个元组,表示每个物品的宽度和高度.$X_t$表示在$t$时刻所有输入的状态集合.从任意的输入$X_0$开始,使用指针$y_0$指向该输入,每一次解码时刻$t$,$y_{t+1}$将指

向当前可用的输入X_t中的一个,并将其作为下一个解码器的输入,如此重复,直到所有的序列都被输出.整个过程将产生一个装箱序列,高度同输入相等为 N,$Y = \{y_t, t = 1, \dots, N\}$.模型是要找到随机策略$\pi$,使得在满足问题约束的同时最小化损失目标的方式生成序列Y.我们要做的就是使得π尽量的接近最优解π^r,尽量以百分百的概率生成最优解.使用概率链规则分解生成序列 Y 的概率$P(Y|X_0)$表示如下：

$$P(Y|X_0) = \prod_{t=0}^{T} P(y_{t+1}|Y_t, X_t) \tag{2.1}$$

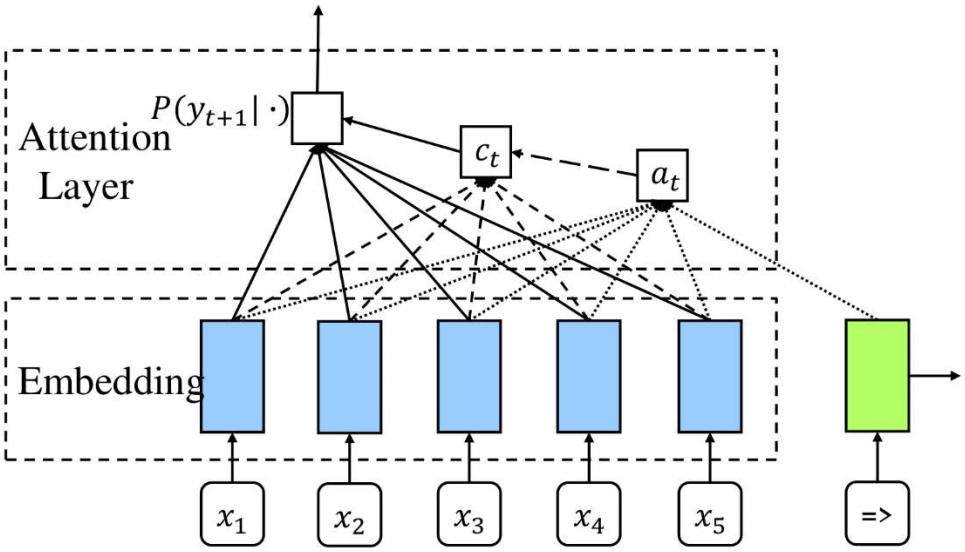

Fig.2.3 Network model

图 2.3 网络模型示意图

在解码器的每一步使用注意力机制生成下一个输入的概率分布.设$\bar{x}_{it}$是嵌入式输入i,h_t是解码器在t时刻隐藏层的状态,对齐向量a_{it}表示输入的序列在下一个解码时刻t中的相关程度,公式如下：

$$a_{it} = softmax(e_{it}) \tag{2.2}$$

$$e_{it} = v_a^T tanh(W_a[\bar{x}_{it}; h_t]) \tag{2.3}$$

这里的对齐模型采用 concat 映射, “;” 表示拼接两个向量.上下文向量c_t计算公式如下：

$$c_t = \sum_{i=1}^{N} a_{it}\bar{x}_{it} \tag{2.4}$$

结合c_t和嵌入式输入,然后使用$softmax$函数归一化结果可得：

$$P(y_{t+1}|Y_t,X_t) = softmax(\bar{e}_{it}) \tag{2.5}$$

$$\bar{e}_{it} = v_c^T tanh(W_c[\bar{x}_{it}; c_t]) \tag{2.6}$$

其中v_a^T, v_c^T, W_a, W_c是训练参数.

2.3模型训练

2.3.1Actor-Critic

为了训练网络模型,本文采用强化学习的方法.我们使用基于策略梯度的强化学习来优化网络参数θ.策略梯度的思路是使奖励越大的动作出现的概率变大,通过反复的训练,不断地调整网络参数.本文选择的Actor-Critic 的方法来训练网络,相比传统的策略梯度下降方法,它有以下优点：(1)Actor-Critic 使用了策略梯度的方法,可以在连续动作或者高维的动作空间中选择适合的动作,采用值函数的方法是做不到的.(2) Actor-Critic 相比单纯的策略梯度的方法,它结合 Q-learning 的做法,使得 Actor-Critic 可以做到单步更新,比起单纯的策略梯度的回合更新效率更高.

Actor-Critic 是由两个网络 Actor(执行者网络)和 Critic(评论者网络) 组成,如图 2.4 所示.Actor 网络是基于概率选择动作,Critic 网络基于 Actor 网络选择的动作评价动作的优劣,Actor 网络根据 Critic 网络的评价修改行为的概率.它将值函数和策略梯度两种强化学习算法优点结合,使得算法可以处理连续和离散的问题,还可以进行单步更新.

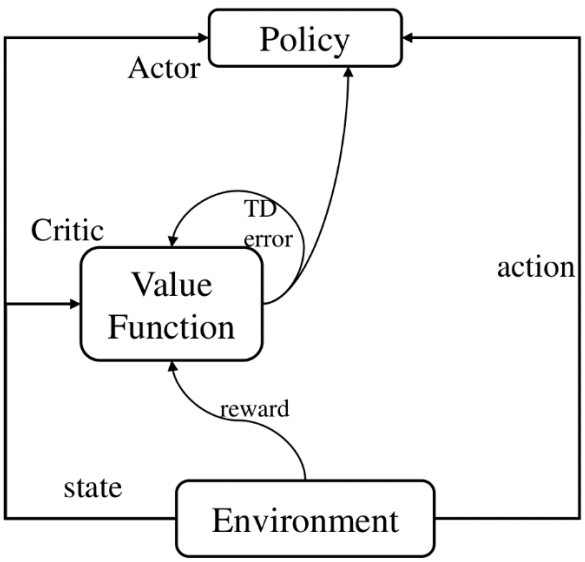

Fig.2.4 Actor-Critic network structure

图 2.4 Actor-Critic 网络结构

设 Actor 网络为$\pi(s)$,则其网络参数θ的梯度公式表示如下：

$$d\theta = \frac{1}{N}\sum_{n=1}^{N}(R_n - V(X_0^n;\omega))\nabla_\theta logP(Y^n|X_0^n) \tag{2.7}$$

其中ω是 Critic 网络$V(s)$的网络参数,它的网络参数梯度公式如下：

$$d\omega = \frac{1}{N}\sum_{n=1}^{N}\nabla_\omega(R_n - V(X_0^n;\omega))^2 \tag{2.8}$$

在公式 2.7 和公式 2.8 中,N是样本实例,R_n表示第 n 个样本实例获得的奖励值,X_0^n表示第 n 个样本实例所有输入的状态集合,$V(X_0^n;\omega)$是 Critic 网络的输出,R_n是当前获得的奖励值,由启发式算法根据网络输出序列计算获得.$R_n - V(X_0^n;\omega)$是优势函数,它表示在X_0^n的状态下采取动作的优劣.如果优势函数的结果是正的,则会增加该动作出现的概率,否则降低概率.

算法 2.2 Actor -Critic 算法流程

Algorithm 2.2: Actor-Critic

1 initializes the network parameters θ, ω for actor and critic
2 **while** *the stop criterion is not satisfied* **do**
3 reset gradients: $d\theta \leftarrow 0$ and $d\omega \leftarrow 0$
4 divide the sample into N groups
5 **for** $n = 0 \rightarrow N-1$ **do**
6 $t \leftarrow 0$
7 **while** *the stop condition is not satisfied* **do**
8 choose y_{t+1}^n according to the distribution $P(y_{t+1}^n|Y_t^n, X_t^n)$
9 observe new state X_{t+1}^n
10 $t \leftarrow t+1$
11 calculate reward $R_n = R(Y^n, X_0^n)$
// calculate the actor network gradient
12 $d\theta = \frac{1}{N}\sum_{n=1}^{N}(R_n - V(X_0^n;\omega))\nabla_\theta logP(Y^n|X_0^n)$
// calculate the critic network gradient
13 $d\omega = \frac{1}{N}\sum_{n=1}^{N}\nabla_\omega(R_n - V(X_0^n;\omega))^2$
// update the network parameters
14 update θ using $d\theta$ and ω using $d\omega$

Actor -Critic 的算法流程如算法 2.2 所示,我们使用θ和ω分别表示是 Actor 和 Critic 的网络参数.在第 4 行,我们将样本分成N输入网络,第 7-10 行是使用简化的指针网络以及注意力机制获得下一个输出的概率分布,每

次选择一个物品,停止的条件是当所有物品都被输出.第 11 行计算该输出的奖励值,$R(\cdot)$是奖励计算函数,也就是算法 1.1 计算装箱的高度将作为奖励信号R_n.第 12-13 行将计算相应的奖励以及策略梯度.最后根据计算的梯度更新两个网络.Critic 网络将使用 Actor 网络输出的概率分布来计算嵌入式输入的加权和.该网络具有两个隐藏层,第一层是使用 relu 激活函数的稠密层,另一个是具有单个输出的线性层.最后对两个网络进行更新.

2.3.2 探索机制

强化学习的探索机制会直接影响采样的性能,本文在模型的训练阶段会通过根据概率分布按概率进行随机采样下一步作为输出.在测试期间采用的是贪婪的方法,选择概率分布中概率最高的输出.

3 计算结果

为了验证强化学习启发式算法(RLHA)并评估其有效性,本小节将进行一系列实验分析.首先阐述了实验数据的生成方法,然后对实验环境和参数设置进行说明,最后对实验结果进行分析.

3.1实验数据

由于现有的标准数据集都是较为零散且实例的数量较少.目前缺少较大量的装箱问题的数据集来用于强化学习模型的训练,因此本文采用了 Bortfeldt 和 Gehring[30]提出的装箱数据生成算法和 Wang 和 Valenzela[31]提出的装箱数据生成算法.

算法 3.1 数据生成算法 1

Algorithm 3.1: GenerateDataset_1

Input:
wC: the width of the strip
n: the number of rectangles
nt: the number of rectangle types
Δ_g,ρ_g, $seed$
Output:
$rect\_list$

1 $dav_g = wC/\Delta_g$
2 $dmin_g = 2 * dav_g/(1+\rho_g)$
3 $dmax_g = dmin_g * \rho_g$
4 initialize random number generation by $seed$
5 $rects, rect\_list$
6 **for** $0 \longrightarrow nt$ **do**
7 randomly generated $w, w \in [dmin_g, dmax_g]$
8 randomly generated $h, h \in [dmin_g, dmax_g]$
9 $rects.add([w,h])$
10 **for** $0 \longrightarrow n$ **do**
11 $rect\_list[i] = rects[i\%nt]$
12 **return** $rect\_list$

Bortfeldt 和 Gehring[30]生成数据的思路是,通过一些参数控制生成的矩形的数量、大小、宽高比以及面

积比等.该算法生成的数据是非零浪费率的,因此我们无法知道生成的数据的最优解.算法 3.1 是该算法的流程,该算法需要输入 6 个参数,wC表示矩形条的宽度,n表示矩形总个数,nt表示要生成的矩形的种类,Δ_g和ρ_g这两个参数是用于计算矩形的最大边长和最小边长,最后$seed$是随机种子.γ是异质性比等于nt/n.首先算法根据输入的参数,计算生成矩形的边长范围$[dmin_g, dmax_g]$,然后根据随机种子初始化随机数生成器.再循环nt次生成nt个类型的矩形$rects$,矩形长宽的范围介于$[dmin_g, dmax_g]$.最后循环n次生成n个矩形$rect_list$,$rect_list$依照$rect_list[i] = rects[i\%nt], 0 \le i < n$规律生成.

算法 3.2 数据生成算法 2

Algorithm 3.2: GenerateDataset 2

Input:

H: The height of the rectangle to be divided

n: the number of rectangles

γ: the area ratio, $\gamma \ge 2$

ρ: the aspect ratio, $\rho \ge 2$

Output:

$rect\_list$

randomly generated W, $W \in [2H/\rho, \rho H/2]$

$rect\_list = [[W, H]]$

for $0 \longrightarrow n-1$ **do**

  $m =$ the area of the largest rectangle in $rect\_list$

  randomly selected rectangle R from $rect\_list$,and the area of R satisfies $S_R > 2m/\gamma$

  delete R from $rect\_list$

  // randomly select the slicing direction

  if $2R.h/\rho \le R.w \le \rho R.h/2$ **then**

    randomly select horizontal or vertical slicing

  else if $2R.w/\rho \le R.h \le 2\rho R.w$ **then**

    horizontal slicing

  else if $2R.h/\rho \le R.w \le 2\rho R.h$ **then**

    vertical slicing

  // randomly select the slicing position

  if horizontal slicing **then**

    randomly selected $y, y \in [min\left(\rho R.w, \frac{R.h - R.w}{\rho}, \frac{R.h}{2}\right), max\left(\frac{R.w}{\rho}, R.h - \rho R.w, \frac{m}{\gamma R.h}\right)]$

    $rect\_list.add([R.w, y])$, $rect\_list.add([R.w, R.h - y])$

  if vertical slicing **then**

    randomly selected $x, x \in [min\left(\rho R.h, \frac{R.h - R.w}{\rho}, \frac{R.w}{2}\right), max\left(\frac{R.h}{\rho}, R.w - \rho R.h, \frac{m}{\gamma R.h}\right)]$

    $rect\_list.add([x, R.h])$, $rect\_list.add([R.w - x, R.h])$

return $rect\_list$

Wang 和 Valenzela[31]生成数据的算法思路是通过将一个大的矩形不断的进行分割,分割成满足一定面积比(γ)和宽高比(ρ)的小矩形.这种思路生成的数据集是属于零浪费类型的,装箱的最优解由输入参数控制.具体的算法流程如算法 3.2 所示.该算法输入的参数有 4 个,n表示生成小矩形个数,H表示待分割矩形的高度.γ和ρ控制生成矩形的面积比和宽高比.算法一开始根据输入的参数随机选择宽度W.然后在 3-18 行进行$n-1$次切割,生成 n 个矩形.首先随机选择满足面积小于$2m/\gamma$的矩形进行切割,m是矩形序列中最大矩形的面积.在 7-13 行,根据不同的条件,选择切割方向.第 13-18 行根据不同的切割方向,在可切割的范围内随机选择切割点,并把生成的两个新矩形加入列表中.

本文使用算法 3.1 和 3.2 共生成 1000000 组的数据用于训练模型,共三种矩形块数：(1)矩形块数为 200 生成 400000 组数据;(2)矩形块数为 100 生成 300000 组数据;(3)矩形块数为 50 的块数共生成 300000 组数据.同时生成 100000 组数据集作为测试样本.通过随机种子生成较大的训练数据集是为了能尽可能的涵盖不同的样本,使得模型能学习到丰富的样本.

3.2 实验环境和参数设置

强化学习模型主要通过 pytorch 实现,实验运行环境如下：Ubuntu 16.04 LST,处理器为 Intel Core E5-2620 v4,处理器频率 2.10GHz,内存 62G,显卡为 NVIDIA GeForce RTX 2080 Ti,显存为 11G.

在实验中我们使用了 1000000 组的训练数据和 100000 组的测试数据.在实验过程中,每批训练的样本量是 128,解码器中 GRU 的隐层单元数量为 128,输入数据也将被嵌入到 128 的向量中.本文使用 Xavier[39]初始化方法对 Actor 和 Critic 网络进行初始化,使用 Adam 优化器,学习速率为 0.0001,同时为了防止过拟合的问题,使用了 L2 范数对梯度进行裁剪.在解码器中使用了 0.1 的 dropout,dropout 可以比较有效的缓解过拟合的产生.模型在单个 GPU 上训练 1000 个时期.

训练好的模型将为 HHA 提供初始解,接下来我们将把 RLHA 和目前优秀的启发式算法 GRASP[17]、SVC[32]、ISA[19]、SRA[20]、CIBA[21]和 HHA[22]算法进行实验对比,验证算法的效果.每个算法将在所有的数据集上运行 10 次,每个实例的运行时间限制为 60 秒,部分大规模数据集运行时间会超过 60 秒.对比算法的实验结果直接取自原论文(原论文中的参数设置,都是原作者通过实验,选择的最优参数),以保证算法对比的公平性.

3.3 实验结果

为了验证 RLHA 的有效性,我们将训练好的模型框架同当前著名的启发式算法进行对比分析.本小节进行对比分析的实验数据一部分来自标准的数据集,为了更合理的说明 RLHA 的性能,另一部分是由算法 3.1 和 3.2 生成的数据,由于训练的数据集与强化学习的算法性能还是存在一定关联的,因此另一部分实验分析将在数据生成算法生成的数据上进行,可以更客观的展现算法的性能.

(1) 标准数据集的对比分析

我们将在 C[12],N[9],NT[33],2SP[17],BWMV[34,35]和 Nice&Path[36] 这 6 个标准数据集上进行实验,对比 7 个算法的实验效果,用粗体标出最佳解的数据.之所以没有考虑标准数据集 ZDF[37]的原因在于这个数据集存在大规模的实例,目前强化学习模型在处理大规模的实例时,运行时间会过长.由于计算能力和运行时间的限制,这两个包含大规模问题实例的数据集就不做实验分析.

Table 3.1 Benchmark data

表 3.1 标准数据集

Dataset	*Instances*	*n*	*zero-waste*
C	21	16-197	Yes
N	13	10-3152	Yes
NT	70	17-199	Yes
2SP	38	7-200	No
BWMV	500	20-100	No
Nice&Path	72	25-5000	No

从表 3.2 的实验结果可以看出,RLHA 在 N 数据集上的表现同 HHA 表现一样,N 数据集是一个零浪费率的数据集,我们的算法在大部分实例上都已经取得最优解,因此两个算法在这个数据集上表现并无差别.RLHA 在 2SP 数据集上的表现与 HHA 获得的结果持平.出现这个情况的原因可能与该数据的特点有关系,该数据矩形个数较少且存在一些比较特殊的实例,需要更精细的策略来改善.我们的启发式设计的规则在这个数据集上的表现可能陷入了瓶颈,因此并没有取得更优的结果.对其余数据集,RLHA 均超过了 HHA,在所有数据集的表现,RLHA 均显著超过了其他 5 个算法.

Table 3.2 Computational results on benchmark data

表 3.2 标准数据集的实验结果

Dataset	*average_gap_rate(%)*						
	GRASP	SVC	ISA	SRA	CIBA	HHA	RLHA
C	0.95	1.03	0.76	0.69	0.58	0.39	**0.32**
N	0.96	0.63	0.41	0.23	0.13	**0.12**	**0.12**
NT	2.32	2.27	2.24	1.60	1.37	0.97	**0.96**
2SP	2.68	2.80	3.02	3.07	2.87	**2.7**	**2.7**
BWMV	1.77	1.80	1.66	1.63	1.55	1.45	**1.42**
Nice	1.84	1.14	1.00	0.54	0.56	0.56	**0.54**
Path	1.68	1.00	0.77	0.40	0.35	0.35	**0.34**
Average	1.74	1.43	1.35	1.13	1.04	0.93	**0.91**

RLHA 在 C,NT,BWMV,Nice 和 Path 上的表现都比 HHA 略提升了一些,虽然提升的幅度不大,但是我们可以看到强化学习模型的加入的确提升了实验结果.这也可以说明强化学习模型能有效的改进启发式获得解的质量.

值得注意的是由于目前现有的针对组合优化问题提出的神经网络模型[26-28]在解决 TSP、VRP、装箱等问题时,其数据集规模大部分都是针对少于 200 个节点的问题实例.这是由于处理大规模实例的数据集时会导致模型训练的时间过长.由于实验条件等限制,本文训练数据集的矩形块数分别是 50,100,200.本文的强化学习模型与解决问题的规模是无关的,也就是说可以把训练好的模型用于不同规模的数据集上.我们将训练的模型用于处理矩形块数为 1000-5000 的数据集 Nice 和 Path,从表 3.2 中 Nice 和 Path 的实验结果可以看出,虽然模型并没有训练过矩形块数大于 200 块的数据集,但是训练之后的模型在中小规模的数据集上的表现也有一定的提升,这也说明模型可以根据保存的装箱经验,为启发式算法提供一个有效的初始解序列,帮助算法更好的运行,提高算法性能.

(2) 生成数据集的对比分析

表 3.3 是生成数据集的明细,分别生成矩形块数为 50,100,200 以及 1000 的数据集各 100 组,用于对比实验的数据集.由于算法 3.1 生成的数据集的最优解不确定,因此这里分别将算法在这 4 组数据上运行,将获得的最优解的装箱高度的平均值作为比较指标.表 3.4 为实验的结果,*avg_height* 表示装箱高度的平均值,更优的解将用粗体标出.

Table 3.3 Details on generated data set

表 3.3 生成数据集明细

Dataset	*Instances*	*n*
Test50	100	50
Test100	100	100
Test200	100	200
Test1000	100	1000

上节实验结果表明,HHA 和 RLHA 显著的超过了其他 7 个算法,这节只将两个最好的 HHA 和 RLHA 算法进行对比.表 3.4 是 HHA 和 RLHA 在 4 组数据集上获得的平均装箱高度的数据.从中可以看出在数据集 Test50 上 HHA 和 RLHA 获得平均装箱高是一样的,这是因为在较小的数据集上,两个算法框架的搜索能力都足够,因此实验表现一样.在数据集 Test100,Test200 和 Test1000 上 RLHA 的表现均优于 HHA 的表现,说明了训练的强化学习模型能使 HHA 获得更好的性能.

Table 3.4 Computational results on generated data set

表 3.4 生成数据集实验结果

Dataset	*average_height*	
	HHA	RLHA
Test50	1263	1263
Test100	4965.60	**4959.79**
Test200	1275.05	**1239.8**
Test1000	1246.29	**1242.98**
Average	2187.48	**2176.39**

图 3.1 是 Test200 某个实例上分别运行 RLHA 和 HHA 的结果,两个算法分别运行 240 秒,每秒钟输出当前获得最优解的值,图中横坐标是时间(s),纵坐标是最优解的值,也就是装箱的高度.图中蓝色的曲线是 RLHA,橙色的曲线是 HHA.从曲线趋势我们可以观察到,RLHA 获得的初始解比 HHA 获得的初始解更好,虽然开始优势并不是特别明显.但经过一段时间的运行,RLHA 能较快的朝更好的解方向发展,这也说明输入序列对启发式算法的影响.中间部分 RLHA 同 HHA 一样都处在一个较为缓慢的搜索状态.在运行后期,RLHA 获得的装箱高度不断地下降,虽然 HHA 也取得了一些不错的进步,但是 RLHA 的效果优于 HHA.因此也可以看出 RLHA 能较有效的改善启发式冷启动的问题,并能提高算法的搜索性能.

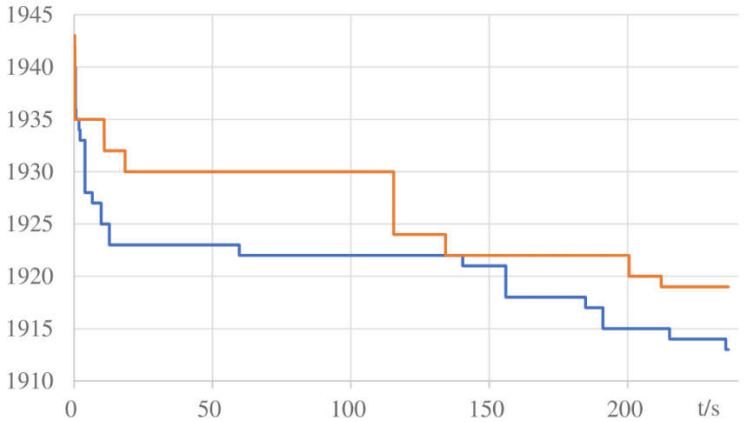

Fig.3.1 Comparison of running time

图 3.1 同时间运行对比

3.4 不同初始化方法的影响

RLHA 是使用强化学习获得初始序列,而 HHA 是使用六种排序方法[38]（见表 3.5 中方法 1~6）,并选择最好的序列,表 3.2 和表 3.4 的计算结果表明 RLHA 的性能超过了 HHA.为进一步研究初始化方法的影响,我们再增加了一种随机初始序列方法.为测试不同初始化方法的影响效果,我们采用典型的 BWMV 数据（该数据包含 500 个实例）.表 3.5 给出了 8 种产生初始序列方法的计算结果,从中可以看出,随机初始序列方法效果最差,使用强化学习获得初始序列,效果明显超过其他 7 种方法.

Table 3.5 Computational results on different initialize methods for BWMV

表 3.5 不同初始化方法计算 BWMV 的结果

获得初始解的方法	Average_gap_rate(%)
1. 面积大的优先, 如果面积相等, 则宽度大的优先	1.63
2. 面积大的优先, 如果面积相等, 则高度大的优先	1.62
3. 周长大的优先, 如果周长相等, 则宽度大的优先	1.61
4. 周长大的优先, 如果周长相等, 则高度大的优先	1.63
5. 宽度大的优先, 如果宽度相等, 则高度大的优先	1.61
6. 高度大的优先, 如果高度相等, 则宽度大的优先	1.75
7. 随机初始序列方法	1.85
8. RLHA	**1.42**

4 结论

强化学习启发式算法(RLHA)框架中,我们创新性的使用强化学习的方法来改善启发式算法冷启动问题.强化学习模型仅使用了启发式算法计算的装箱值作为奖励信号,来训练网络,最后输出给定分布中最优的序列.使用简化版的指针网络来输出装箱序列,该网络简化编码器部分的网络,节省了一大部分的计算能力,提高效率.采用 Actor-Critic 算法来训练网络由于,Actor-Critic 可以单步更新,比起单纯的策略梯度的回合更新效率更高.实验部分,本文通过数据生成算法生成了训练模型的数据集.在 714 个标准问题实例和 400 个生成的问题实例上测试 RLHA,实验结果显示 RLHA 获得了比当前著名启发式算法更好的解,这也表明了训练的强化学习模型能为 HHA 提供更好的初始的装箱序列,有效的改善冷启动的问题.将来的工作就是在改进框架的基础上,对启发式算法进行进一步的改进.同时,将强化学习应用于别的 NP 难问题的求解,总结出一般性的求解框架,并进行更多的应用.

References:

[1] Andrea Lodi, Martello Silvano, Monaci Michele. Two-dimensional packing problems: A survey. European journal of operational research, 2002, 141(2): 241-252.

[2] Mhand Hifi. Exact algorithms for the guillotine strip cutting/packing problem. Computers & Operations Research, 1998, 25(11): 925-940.

[3] Silvano Martello, Monaci Michele, Vigo Daniele. An exact approach to the strip-packing problem. INFORMS Journal on Computing, 2003, 15(3): 310-319.

[4] Mitsutoshi Kenmochi, Imamichi Takashi, Nonobe Koji, et al. Exact algorithms for the two-dimensional strip packing problem with and without rotations. European Journal of Operational Research, 2009, 198(1): 73-83.

[5] Jean-François Côté, Dell'Amico Mauro, Iori Manuel. Combinatorial Benders' cuts for the strip packing problem. Operations Research, 2014, 62(3): 643-661.

[6] Brenda-S Baker, Coffman Jr-Edward-G, Rivest Ronald-L. Orthogonal packings in two dimensions. SIAM Journal on computing, 1980, 9(4): 846-855.

[7] Bernard Chazelle. The bottom-left bin-packing heuristic: An efficient implementation. IEEE Transactions on Computers, 1983, (8): 697-707.

[8] Wenqi Huang, jingfa Liu. A Deterministic Heuristic Algorithm Based on Euclidian Distance for Solving the Rectangles Packing. Chinese Journal of Computers, 2006,5:734-739.

[9] Edmund-K Burke, Kendall Graham, Whitwell Glenn. A new placement heuristic for the orthogonal stock-cutting problem. Operations Research, 2004, 52(4): 655-671.

[10] Stephen-CH Leung, Defu Zhang, Changle Zhou, et al. A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem. Computers & Operations Research, 2012, 39(1): 64-73.

[11] Jakobs S. On genetic algorithms for the packing of polygons. European journal of operational research, 1996, 88(1): 165-181.

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

[13] Andreas Bortfeldt. A genetic algorithm for the two-dimensional strip packing problem with rectangular pieces. European Journal of Operational Research, 2006, 172(3): 814-837.

[14] Dequan Liu, Hongfei Teng. An improved BL-algorithm for genetic algorithm of the orthogonal packing of rectangles. European Journal of Operational Research, 1999, 112(2): 413-420.

[15] Kun He, Peng-Li Ji, Chu-Min Li. Heuristics for Solving the 2D Rectangle Packing Area Minimization Problem Basing on a Dynamic Reduction Method. Journal of Software,2013,9:2078-2088.

[16] Lijun Wei, Wee-Chong Oon, Wenbin Zhu, et al. A skyline heuristic for the 2D rectangular packing and strip packing problems. European Journal of Operational Research, 2011, 215(2): 337-346.

[17] Ramón Alvarez-Valdés, Parreño Francisco, Tamarit José-Manuel. Reactive GRASP for the strip-packing problem. Computers & Operations Research, 2008, 35(4): 1065-1083.

[18] Defu Zhang, Lijun Wei, Stephen-CH Leung, et al. A binary search heuristic algorithm based on randomized local search for the rectangular strip-packing problem. INFORMS Journal on Computing, 2013, 25(2): 332-345.

[19] Stephen-CH Leung, Defu Zhang, Kwang-Mong Sim. A two-stage intelligent search algorithm for the two-dimensional strip packing problem. European Journal of Operational Research, 2011, 215(1): 57-69.

[20] Shuangyuan Yang, Shuihua Han, Weiguo Ye. A simple randomized algorithm for two-dimensional strip packing. Computers & Operations Research, 2013, 40(1): 1-8.

[21] Zhen Chen, Jianli Chen. An Effective Corner Increment-Based Algorithm for the Two-Dimensional Strip Packing Problem. IEEE Access, 2018, 6: 72906-72924.

[22] Mengfan Chen, Kai Li, Defu Zhang, et al. Hierarchical Search-Embedded Hybrid Heuristic Algorithm for Two-Dimensional Strip Packing Problem, IEEE Access 2019,7, 179086-179103

[23] John-J Hopfield, Tank David-W.“Neural”computation of decisions in optimization problems. Biological cybernetics, 1985, 52(3): 141-152.

[24] Oriol Vinyals, Fortunato Meire, Jaitly Navdeep. Pointer networks, Advances in Neural Information Processing Systems (NIPS), 2015: 2692-2700.

[25] Volodymyr Mnih, Heess Nicolas, Graves Alex. Recurrent models of visual attention. arXiv:1406.6247 2014: 2204-2212.

[26] Irwan Bello, Pham Hieu, Le Quoc-V, et al. Neural combinatorial optimization with reinforcement learning. arXiv preprint arXiv:1611.09940, 2016.

[27] Mohammadreza Nazari, Oroojlooy Afshin, Snyder Lawrence, et al. Reinforcement learning for solving the vehicle routing problem, Advances in Neural Information Processing Systems (NIPS) 31 2018: 9839-9849.

[28] Wouter Kool, Van Hoof Herke, Welling Max. Attention, learn to solve routing problems! arXiv preprint arXiv:1803.08475, 2018.

[29] Haoyuan Hu, Xiaodong Zhang, Xiaowei Yan, et al. Solving a new 3d bin packing problem with deep reinforcement learning method. arXiv preprint arXiv:1708.05930, 2017.

[30] Andreas Bortfeldt, Gehring Hermann. New Large benchmark instances for the two-dimensional strip packing problem with rectangular pieces. Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06) IEEE, 2006: 30b.

[31] Pearl-Y Wang, Valenzela Christine-L. Data set generation for rectangular placement problems. European Journal of Operational Research, 2001, 134(2): 378-391.

[32] Gleb Belov, Scheithauer Guntram, Mukhacheva E-A, et al. One-dimensional heuristics adapted for two-dimensional rectangular strip packing. Journal of the Operational Research Society, 2008, 59: 823-832.

[33] Eva Hopper. Two-dimensional packing utilising evolutionary algorithms and other meta-heuristic methods. University of Wales. Cardiff, 2000.

[34] Judith-O Berkey, Wang Pearl-Y. Two-dimensional finite bin-packing algorithms. Journal of the operational research society, 1987, 38(5): 423-429.

[35] Silvano Martello, Vigo Daniele. Exact solution of the two-dimensional finite bin packing problem. Management science, 1998, 44(3): 388-399.

[36] Christine-L Mumford-Valenzuela, Vick Janis, Wang Pearl-Y. Heuristics for large strip packing problems with guillotine patterns: An empirical study. Springer, 2004: 501-522.

[37] Stephen-CH Leung, Defu Zhang. A fast layer-based heuristic for non-guillotine strip packing. Expert Systems with Applications, 2011, 38(10): 13032-13042.

[38] Mengfan Chen. Research on Hybrid Heuristic Algorithm for Two-dimensional Strip Packing Problem (Master thesis). Xiamen: Xiamen university, 2020.

附中文参考文献:

[8] 黄文奇, 刘景发. 基于欧氏距离的矩形 Packing 问题的确定性启发式求解算法. 计算机学报,2006,5:734-739.

[15] 何琨, 姬朋立, 李初民. 求解二维矩形 Packing 面积最小化问题的动态归约算法. 软件学报,2013,9:2078-2088.

[38] 陈梦烦. 二维条形装箱问题的混合启发式算法研究[硕士学位论文]. 厦门:厦门大学, 2020.