组块排放依据左下优先 BL(bottom-left)原则进行放置,如图 2(a)所示的排放图,将二维笛卡尔坐标系的原点设在 C 类型板材 r_0 的左下角,对板材的切割路径定义如下.

(1) 当排放的 *Group* 和 r_0 两条边都不相等时,有两条满足“一刀切”的切割路径,一条是沿着 *Group* 的上边线水平切割(horizontal cut,简称 HC),如图 2(b)所示;另一条是沿着 *Group* 的右边线垂直切割(vertical cut,简称 VC),如图 2(c)所示.任意一种切割方式都会将 r_0 分割成两个面积更小的矩形区域 r_1 和 r_2,两块子区域面积设定为 $r_1<r_2$.

(2) 如果 *Group* 有一边与板材的某一边长度相等,则 r_0 只会切割出一个矩形区 r_1.

(3) 如果 *Group* 与板材任一边都相等,则 r_0 不需要进行切割.

板材排样经过多次组块排放和切割,最终产生如图 3(a)所示的排样图.该排样过程是一个典型的树形递归搜索过程,如图 3(b)所示,采用排样二叉树描述该过程:在递归排样过程中会不断切割出待排矩形区 r,r 对应于排样二叉树一个节点 p,其中,根节点为 C 类型,其他节点为 A 或 B 类型.分支节点中 *HC*、*VC* 分别代表 r 采用的切割方式,叶子节点中数字代表最终排放的组块,W 代表产生的废料.

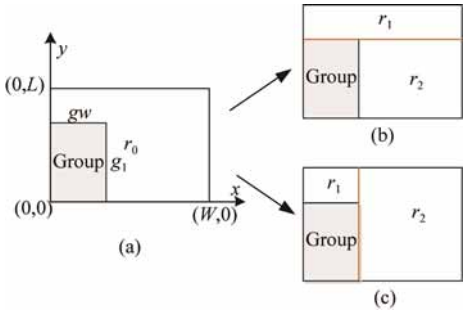

Fig.2 Group location and cutting style

图 2 Group 定位及切割方式

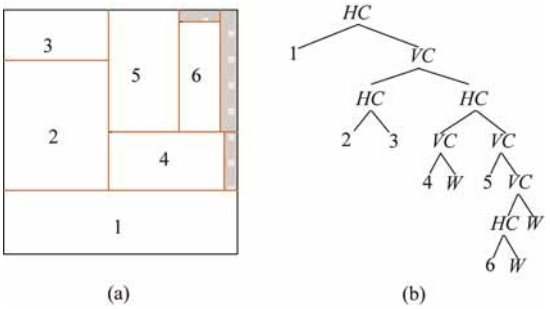

Fig.3 Layout and binary layout tree

图 3 排样图和排样二叉树

2.2 BSDBF后发式算法策略

首先,基于组化规则,BSDBF 启发式算法通过比较组块适应度,快速择优推荐组块进行排放,构造排样区域的局部解.然后,基于废料率进行深度优先递归搜索,寻找单块板材的布局方案.当搜索过程累加的废料面积大于预期废料率所设定的废料面积阈值时,板材当前排样方案寻找失败,根据废料率快速回溯,重新选择组块进行排样,回溯点选择为 C 类型或 $A|B$ 类型待排矩形区.BSDBF 启发式算法描述如下.

2.2.1 基于 *BigItem* 的适应度择优匹配

在组块集合中,*BigItem* 类型组块是仅包含 *BigItem* 类型工件的一元组块.在所有排样方案中,不可能出现两个 *BigItem* 同时占用一块板材的情况.如果 *BigItem* 越靠后使用,就越会出现单个 *BigItem* 占据一块板材,且缺少面积较小组块进行填充的现象,导致后续板材利用率偏低,对整个排样结果有较大影响.因此,针对 C 类型待排矩形区进行排样时,设计了基于 *BigItem* 的最佳适应度择优匹配启发规则如下.

(1) 首先遍历 *GroupList*,挑选出 *BigItem* 类型的一元 *Group*,并保存在集合 *BigItemGroupList* 中,按照适应度函数 *fitness*1 的值对 *BigItemGroupList* 进行降序排列.*fitness*1 适应度函数定义如下:

$$finness1(r, BigItem)=\frac{BigItem面积}{板材面积}.$$

(2) 然后遍历 *GroupList*,挑选出非 *BigItem* 类型的 *Group*,并保存在集合 *NoBigItemGroupList* 中,按照适应度函数 *fitness*2 的值对 *NoBigItemGroupList* 进行降序排列.*fitness*2 适应度函数定义如下:

$$finness2(r,Group)=Group与板材相等边的条数+\frac{Group面积}{板材面积}-\frac{Group中切片数量}{1000}.$$

(3) 对 C 类型待排矩形区排样时,优先从 *BigItemGroupList* 中推荐适应度值大的 *Group* 进行排放,*BigItem* 排放完之后再从 *NoBigItemGroupList* 中推荐适应度值大的 *Group* 进行排放.假定对于 C 类型待排矩形区可使用的待排 *Group* 总数量为 k,则搜索区间为[1,k].

2.2.2 双队列推荐适应度择优匹配

由于切割过程会出现数量众多且面积不等的$A|B$类型待排矩形区,因此,在对$A|B$类型待排矩形区进行排样时,如果以最佳适应度值作为标准选择组块进行排放,则需要针对每一个 $A|B$ 类型待排矩形区反复计算组块适应度值,并反复进行适应度值的排序.这种以组块适应度值排序作为递归计算的原子计算单位,会极大地影响整个算法的计算速度.因此,针对 $A|B$ 类型待排矩形区排样时,设计了基于双队列推荐适应度择优匹配启发规则如下.

(1) 将 *GroupList* 中所有待排 *Group* 按照高降序、相同高按宽降序进行排序,以此方法构建基于高和宽双重选择的双队列存储结构,只需执行一次排序,所有 *Group* 始终保持固定顺序不变.

(2) 对任意 $A|B$ 类型待排矩形区进行排样,在双队列中先按高从前向后查找 *Group*,再按宽从前向后查找 *Group*,实现对不同面积待排矩形区都有效的 *Group* 快速排序体系,推荐具有高优先级的 *Group* 进行排放.

(3) *Group* 对 $A|B$ 类型待排矩形区的优先级匹配规则定义如下.

第 1 级 *Group* 两边长与板材两边长都相等;

第 2 级 *Group* 任一边长与板材任一边长相等;

第 3 级满足以上条件,面积大的 *Group* 优先;

第 4 级满足以上条件,包含工件少的 *Group* 优先;

第 5 级满足以上条件,适应度函数 *fitness*3 值大的 *Group* 优先.*fitness*3 适应度函数定义如下:

$$finness3(r,Group)=\frac{gl}{rl}+\frac{gw}{rw}+\frac{gl\times gw}{rl\times rw}.$$

(4) 在对 $A|B$ 类型待排矩形区进行排样时,设置每块排样区域搜索 *Group* 的最大数量为 *limit*,可以计算出 $A|B$ 类型待排矩形区对 *Group* 的整个搜索空间为 $limit^{(m-1)}$,其中,m 代表单块板材切割次数.

2.2.3 基于废料率的树形递归搜索

排样二叉树的每个节点最多可以选择两种切割策略,利用约束剪枝规则,尽量使用一种切割方式,减少切割选择次数,启发式切割规则如下.

(1) 如果 *Group* 任意一边和板材任意一边相等,只选一种切割;

(2) 当板材是 $A|B$ 类型且和 *Group* 都是正方形时,只选垂直切割;

(3) 默认优先选用垂直切割方式,只有垂直切割排样方案失败,才会选择水平切割方式进行尝试;

(4) 分别计算两种切割方式切割后形成的两块子板材的面积差,优先选择面积差小的切割方式.

每一个节点切割后最多出现两个分支,优先选择面积小的分支进行排样.根据不同的待排矩形区类型,选择不同的组块快速推荐策略,优先推荐适应度值高的组块进行排放,构造排样的局部解,并通过废料率评估局部解和全局解的关系.BSDBF 启发式算法采用了基于深度优先的递归搜索,具体步骤如下.

算法 2. BSDBF.

输入:待排矩形区 *r*,组块空间 *GroupList*,工件集合 *itemList*,工件数量集合 *itemAmountMap*,切割方式 *divide*,1 表示垂直切割,2 表示水平切割;

输出:单块板材排样结果 *solution*.

Step 1. 初始化:

板材预期废料率为λ,计算单块板材允许的废料面积 $S_\lambda=WL\lambda$,S 为单块板材累计总废料面积.初始化 $\lambda=0$,先进行无废料排样,如果失败,则设定$\lambda=0.5$,再尝试有废料排样.

Step 2. 执行 *group* 的排放,检查废料率的合法性,返回布尔值 *isRectLayout*

```
if (r 不能放得下最小工件){
    if (r 是废料){
        if (S>S_λ)
            return false
```

else
return true
}
}
if (*r* 不是废料){
if (*r* 是 *C* 类型){
基于 *BigItem* 适应度择优匹配规则,将 *GroupList* 中 *Group* 按适应度值降序保存到集合 *fitnessList* 中
}else if (*r* 是 *A*|*B* 类型){
基于双队列推荐适应度择优匹配规则,将 *GroupList* 中 *Group* 按适应度值降序保存到集合 *fitnessList* 中
}
for (遍历所有 *fitnessList*){
针对不同类型的 *r*,从相应 *fitnessList* 中选择适应度高的 *Group* 进行排放,依据启发式切割规则进行 *r* 的切割,设定切割方式为 *divide*.
依据 *divide* 值,执行 **Step 3**,将 *r* 进行分割,将结果赋值给 *isRectLayout*
if (*isRectLayout*)
将排样结果添加到 *solution*
if (!*isRectLayout*)
使用另一种切割方式,执行 **Step 3**,将 *r* 进行分割,将结果赋值给 *isRectLayout*,
if (*isRectLayout*)
将排样结果添加到 *solution*
}
}
返回 *isRectLayout*

Step 3. 依据 *divide* 值,对 *r* 进行分割操作,检查是否排样成功,返回布尔值 *isRectDivide*
if (*Group* 刚好填充满 *r*)
将 *Group* 所包含的工件添加到排样结果集 *newSolution* 中
if (*Group* 未填充满 *r*){
依据 *divide* 值选择切割方式,分割出两个待排矩形区 *r*1、*r*2,面积 $S_{r1}<S_{r2}$
if ($S_{r1}>0$)
优先 *r*1 排样,执行 **Step 2**,将结果赋值给 *isRectDivide*
if (*isRectDivide*)
继续 *r*2 排样,执行 **Step 2**,将结果赋值给 *isRectDivide*
if (*isRectDivide*)
Group 排放在 *r* 的左下角,并将 *Group* 中包含的工件添加到 *newSolution* 中
}
返回 *isRectDivide*

2.3 排样方案的贪心搜索

第 2.2 节提出了 BSDBF 启发式搜索算法,可以获得单块板材的排样方案,但单块板材的初次排样方案不一定是最优的局部方案,也不一定能构成最优的整体排样方案,因此要对单块板材的排样方案进行评估和进一步选择.本文设置了如下两种排样方案的择优规则.

(1) 最佳利用率优先:在单块板材所有排样方案中,板材利用率最大的排样方案优先选择.

(2) 最少工件个数优先:在满足(1)的条件下,当利用率相同时,板材使用工件数量最少的排样方案优先选择.

在 BSDBF 启发式算法外围,采用了贪婪选择算法,以获得单块板材的优解,并构造整体排样方案的初始解.贪心选择排样方案具体步骤如下.

算法 3. GreedySearch.

输入:板材集合 *materialList*,组块空间 *GroupList*,工件集合 *itemList*,工件数量集合 *itemAmountMap*;

输出:初始整体排样结果 *solutionList*.

Step 1. 根据单块板材初次排样方案,计算单块板材实际废料率.

Step 2. 如果废料率为 0,则是无废料排样,扩大 *k* 值和 *limit* 值以扩大 *Group* 搜索范围,继续进行无废料方案搜索,依据最少工件个数优先原则,记录工件个数最少的无废料排样方案作为最优解 *bestSolution*.

Step 3. 如果废料率不为 0,则是有废料排样,采用了改进的动态二分法,逐步降低预期废料率λ,从而获得高利用率的排样方案,具体步骤如下.

(1) 设定预期废料率上限为λ_{high}=0.5,下限为λ_{low}=0,二分法上下限精度为 e,初始值 e_1=0.12,开始二分法排样.

(2) 利用 BSDBF 启发式算法获得单块板材的一次排样结果,将当前废料率最小的排样方案保存到 *bestSolution* 中,若废料率相同,则保存工件数量最少的排样方案到 *bestSolution* 中.

(3) 计算本次排样结果的真实废料率,重新设定废料率上下限,继续执行(2),直到满足条件$|\lambda_{high}-\lambda_{low}|<e$,则二分法终止.

(4) 此后,e 按照公式 $e_{i+1}=e_i+0.01\times(n-4)$产生,$n$ 等于 0,1,2,3,逐步降低 e,并遍历所有 e 值,继续执行(1).遍历完毕后,将 *bestSolution* 作为本次有废料排样的最好排样结果.

Step 4. 对 *materialList* 中所有板材依次进行排样,通过组合所有单块板材排样方案结果 *bestSolution*,获得整体排样结果的初始解 *solutionList*.

2.4 排样方案后处理

为了获得整个排样方案的优解,需要对不同板材的排样结果进一步协调,以提升整体利用率.因此,本文基于大小工件分治策略和利用率优劣集划分策略,设计了排样方案的后处理算法,详细步骤如下.

算法 4. PostProcess.

输入:初始整体排样结果;

输出:后处理最终排样结果.

Step 1. 判断整个排样方案总废料面积,当总废料面积大于一块板材面积时,即启动排样后处理的检查机制.

Step 2. 检查初始排样结果是否存在 *BigItem*,如果存在,选择 **Step 3** 进行求解,如果不存在,选择 **Step 4** 进行求解.

Step 3. 包含 *BigItem* 时,重排策略如下.

(1) 找出单片利用率等于 100%且包含 *BigItem* 的排样结果,对该块板材排样结果进行锁定.当其余未锁定板材的数量大于 1 时,启动局部重排.

(2) 将未锁定的板材排样结果组成重排序列,拆分以上排样结果,还原工件到待排样 *Group* 中,将重排时的组化粒度设定为 1 和 2,分别进行排样尝试,保存利用率最好的后处理结果.

(3) 当上一次后处理排样结果利用率提高小于 0.005 时,则停止后处理.

Step 4. 不包含 *BigItem* 时,根据总利用率选择优集和劣集进行后处理,重排策略如下.

(1) 对板材排样结果以当前总利用率为界限,大于为优解,小于为劣解,将优解集合按利用率降序排序,将劣解集合按利用率升序排序.

(2) 选择优集利用率最高的前 1/3 方案,选择劣集利用率最低的前 1/3 方案,组成重排序列,拆分以上排样结果,还原工件到待排样 *Group* 中,将重排时的组化粒度设定为 1 和 2,分别进行排样尝试,保存利用率最好的后处理结果.

(3) 当上一次后处理排样结果利用率提高小于 0.005 时,则停止后处理.

Step 5. 比较后处理前后的排样结果,选择利用率高的作为最终排样结果.

3 计算实验

基于 Java 语言编程,我们开发了单规格一刀切矩形件排样软件,且未使用多线程技术进行算法实现.算法测试平台为笔记本电脑,操作系统为 32 位 Windows 7 SP1,CPU 为 Core i3 2.13 GHz,内存为 3GB.为了检验算法的有效性,我们测试了国际上流行的单规格矩形件排样通用测试案例.

3.1 实验1及分析

表 1 选择了 cgcut、gcut 和 ngcut 这 3 个系列的通用测试案例,其中,案例 cgcut 来源于 Christofides 和 Whitlock(1977)[28],包含 3 个案例,案例 gcut 来源于 Beasley(1985a)[29],包含 13 个案例,案例 ngcut 来源于 Beasley (1985b)[30],包含 12 个案例.与其他文献中的算法进行了结果对比,见表 1.

Table 1 Comparisons with other algorithms for cgcut, gcut and ngcut

表 1 cgcut、gcut 和 ngcut 系列案例结果与其他算法对比情况

算例集	算例数量	计算板材数				
		VND-3NE-H[21]	VND-3NE-V[21]	SNS-3NE-H[21]	SNS-3NE-V[21]	BSDBF
cgcut	3	23	24	23	24	23
gcut	13	103	103	101	101	98
ngcut	12	31	31	32	32	30

在表 1 的实验结果中,cgcut 系列使用板材数量等于文献报道的测试结果,gcut 和 ngcut 系列使用板材数量均低于上述文献报道的测试结果.所有案例能在平均 1s 时间内完成,说明本算法对小规模排样具有较高的求解质量.

3.2 实验2及分析

表 2 选择了 Class 系列 10 组通用测试案例,每组 Class 分别有 50 个测试案例,10 组总计 500 个测试案例.其中,前 6 组案例来源于 Berkey 和 Wang(1987)[16],后 4 组案例来源于 Martello 和 Vigo(1998)[5].另外,所有 Class 系列测试案例板材全部是正方形,板材尺寸在 10×10~300×300 之间取值.每个 Class 系列分别包含 10 个 20 个工件、40 个工件、60 个工件、80 个工件和 100 个工件的案例.Class 系列测试案例包含的工件种类多,相同种类的工件数量较少,并且包含了二维排样的各种极端情况,能够很好地反映算法的求解质量.Class 系列测试案例与其他文献中的算法进行结果对比情况见表 2,与其他较好结果的算法详细数据对比情况见表 3.

在表 3 所示的实验结果中,500 个测试案例使用板材总数量为 7 051,均低于各类文献报道的测试结果.其中,Class 1、Class 3、Class 4、Class 5、Class 6、Class 10 系列案例使用板材数量低于文献记录,Class 2、Class 9 系列案例使用板材数量等于文献记录,Class 7、Class 8 系列案例使用板材数量稍高于文献记录,进一步说明本算法对多类型工件的小规模排样具有很高的求解质量.

Table 2 Comparisons with other algorithms for Class

表 2 Class 系列案例结果与其他算法对比情况

算法	算法来源	计算板材数	平均偏差(2007)[31]	处理器
FC	Lodi, *et al*.(1999)[2]		1.089	SG 195MHz
KP	Lodi, *et al*.(1999)[2]	7 297	1.085	SG 195MHz
TSKP	Lodi, *et al*.(1999)[2]	7 101	1.055	SG 195MHz
VND-3NE-V	Chan, *et al*.(2010)[21]	7 124		Core2 2 GHz
SNS-3NE-V	Chan, *et al*. (2010) [21]	7 110		Core2 2 GHz
GBL	Polyakovsky, *et al*.(2009)[14,18,19]	7 367	1.094	Athlon XP 2800
A-B	Polyakovsky, *et al*.(2009)[18]	7 063	1.036	Athlon XP 2800
CH	Charalambous (2011)[23]	7 191	1.059	Core i3 2.13 GHz
CHB	Charalambous (2011)[23]	7 117	1.050	Core i3 2.13 GHz
CHBP	Charalambous (2011)[23]	7 064	1.037	Core i3 2.13 GHz
CFIH	Krzysztof (2013)[24]	7 100	1.044	Core2 2.33 GHz
CFIH+J4	Krzysztof (2013)[24]	7 080	1.040	Core2 2.33 GHz
BSDBF		7 051	1.026	Core i3 2.13 GHz

Table 3 Comparison of detailed results for Class

表 3 Class 系列案例详细数据对比情况

算例集	排样工件数量	算例数量	计算板材数		
			CHBP	CFIH+J4	BSDBF
Class 1	20	10	66	66	66
	40	10	129	129	129
	60	10	195	195	195
	80	10	271	271	270
	100	10	314	314	313
		总数	975	975	973
Class 2	20	10	10	10	10
	40	10	19	19	19
	60	10	25	25	25
	80	10	31	32	31
	100	10	39	39	39
		总数	124	125	124
Class 3	20	10	48	48	49
	40	10	94	94	93
	60	10	136	136	137
	80	10	186	185	186
	100	10	223	223	221
		总数	687	686	686
Class 4	20	10	10	10	10
	40	10	19	19	19
	60	10	25	25	23
	80	10	33	33	30
	100	10	38	38	37
		总数	125	125	119
Class 5	20	10	59	59	59
	40	10	115	117	115
	60	10	176	175	175
	80	10	240	241	240
	100	10	282	281	282
		总数	872	873	871
Class 6	20	10	10	10	10
	40	10	18	18	17
	60	10	21	21	21
	80	10	30	30	30
	100	10	34	34	32
		总数	113	113	110
Class 7	20	10	52	52	52
	40	10	104	106	105
	60	10	148	151	147
	80	10	211	214	213
	100	10	255	258	254
		总数	770	781	771
Class 8	20	10	53	53	53
	40	10	105	105	105
	60	10	150	152	150
	80	10	210	211	211
	100	10	258	258	259
		总数	776	779	778
Class 9	20	10	143	143	143
	40	10	275	275	275
	60	10	435	435	435
	80	10	573	573	573
	100	10	693	693	693
		总数	2 119	2 119	2 119
Class 10	20	10	41	42	41
	40	10	73	73	73
	60	10	100	101	100
	80	10	130	129	128
	100	10	159	159	158
		总数	503	504	500
总计			7 064	7 080	7 051

BSDBF 算法求解 2BP|R|G 问题的平均偏差值为 1.026,见表 2,相较于几种文献报道的算法,均取得了微弱领先的优势.文献[9]针对 2BP|O|G 问题,获得更低的平均偏差值 1.024.常规而言,2BP|O|G 的解构造比 2BP|R|G 问题更加困难,但 2BP|R|G 问题比 2BP|O|G 问题的搜索空间更大.目前尚不清楚,针对两类问题的算法之间是否存在可折算的比较关系.然而,文献[9]提出的混合式元启发算法和可改变的分数选择策略,对 2BP|×|G 问题的求解极具借鉴意义.Class 系列部分测试案例的排样图如图 4 所示,其中阴影部分为未排样区域.

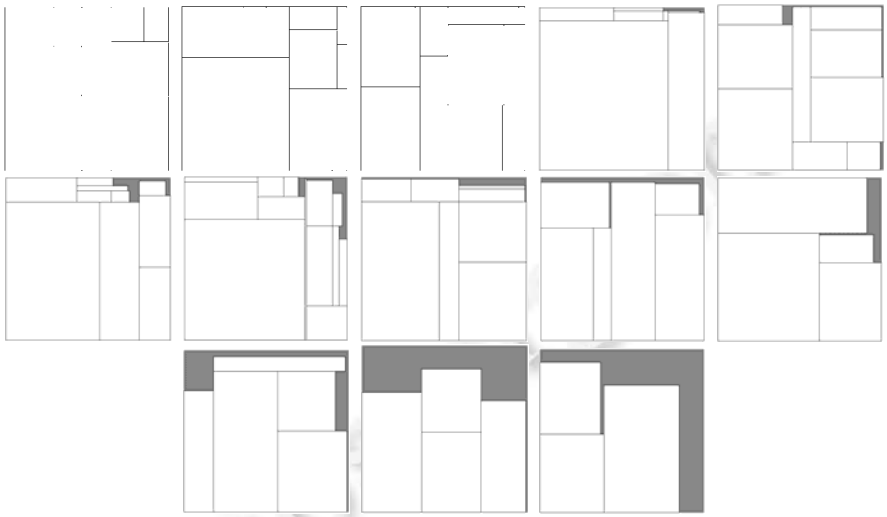

Fig.4 Packing result for Class 496

图 4 Class 496 的排样结果

3.3 实验3及分析

表 4 的测试数据参考了来源于 Hopper 和 Turton[32]的 21 个 C 系列案例,该 21 个案例利用率均为 100%.我们将 21 个 C 系列案例的板材数量和工件数量分别扩大 10 倍,构造了 21 个 10C 系列案例 10C1P1~10C7P3,这些案例仍可满足最优解的利用率为 100%.10C 系列案例的测试结果见表 4.

Table 4 Computational results (BSDBF) of 21 instances for 10C

表 4 10C 系列案例 21 组算例计算结果(BSDBF)

算例	排样工件数量	板材高度	板材宽度	计算结果		
				板材数	利用率	计算时间(s)
10C1P1	160	20	20	10	100	<0.1
10C1P2	170	20	20	10	100	<0.1
10C1P3	160	20	20	10	100	<0.1
10C2P1	250	40	15	10	100	<0.1
10C2P2	250	40	15	10	100	<0.1
10C2P3	250	40	15	10	100	<0.1
10C3P1	280	60	30	10	100	<0.1
10C3P2	290	60	30	10	100	<0.1
10C3P3	280	60	30	10	100	<0.1
10C4P1	490	60	60	10	100	<0.3
10C4P2	490	60	60	10	100	<0.1
10C4P3	490	60	60	10	100	<0.1
10C5P1	730	90	60	10	100	<0.1
10C5P2	730	90	60	10	100	<0.1
10C5P3	730	90	60	10	100	<0.1
10C6P1	970	80	120	10	100	<0.2
10C6P2	970	80	120	10	100	<0.2
10C6P3	970	80	120	10	100	<0.4
10C7P1	1 960	240	160	10	100	<3.7
10C7P2	1 970	240	160	10	100	<0.3
10C7P3	1 960	240	160	10	100	<0.8

在表 4 所示的实验结果中,21 个 10C 系列案例均能获得利用率为 100%的最优解,表明算法对无废料测试案例可以很快找到其最优解,具有良好的爬山特性.部分测试案例的排样图如图 5 所示.

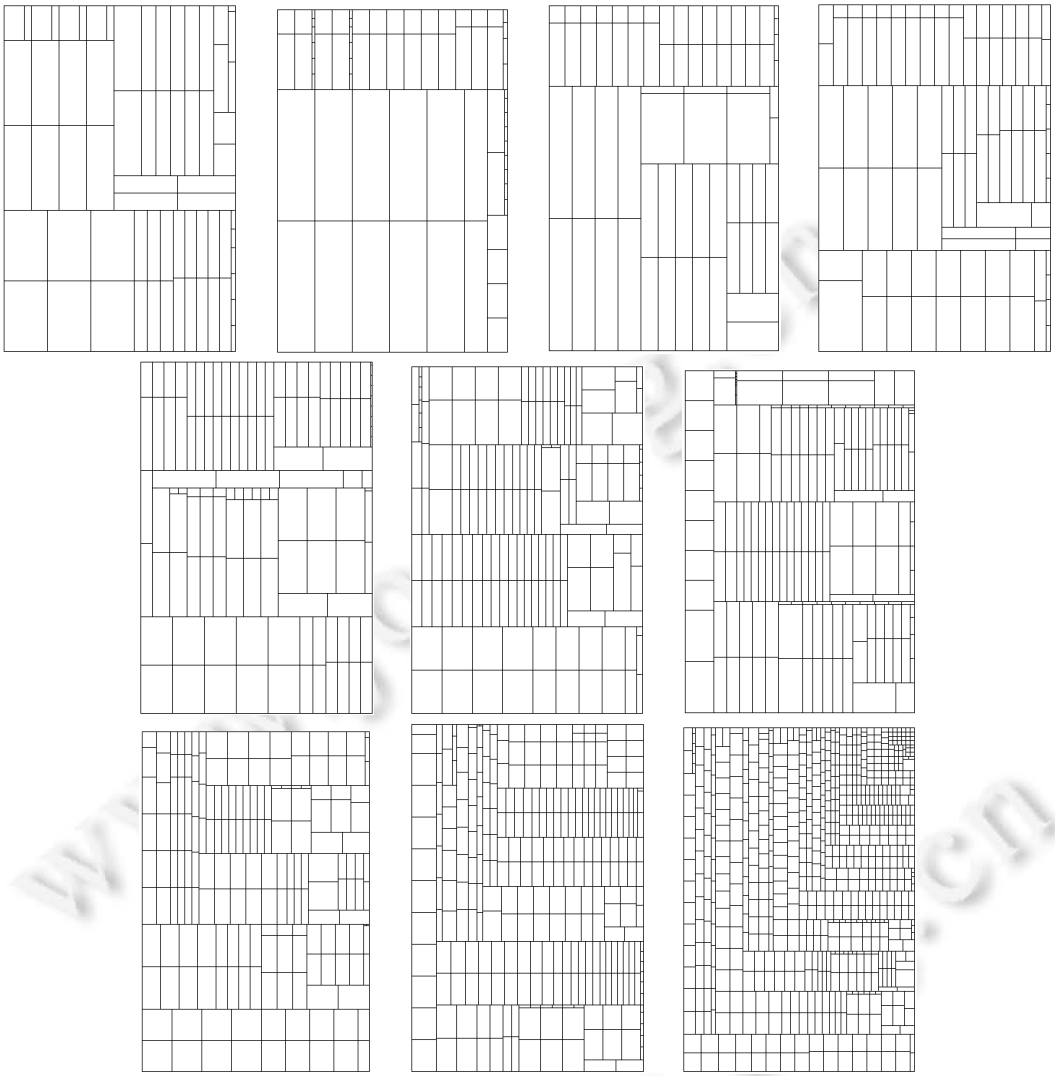

Fig.5 Packing result of 10C7P3

图 5 10C7P3 的排样结果

4 算法参数分析

BSDBF 算法中涉及到如下两个重要参数:一是二分法排样的上下限精度 e,二是 $A|B$ 类型待排矩形区搜索组块的步长 $limit$.

使用基于遍历 e 值的改进二分法排样,e 值在算法中的影响分析如下:第一,e 值的大小和获得的排样方案利用率有一定关系,通过遍历 e,更容易找到较好的排样结果;第二,在不同排样工件种类情形下,二分法会始终朝着利用率较高的方向寻找排样结果.算法中精度 e 越大,就越容易找到单块板材排样方案,但方案的废料率往往偏大,精度 e 越小,单块板材排样方案的废料率通常更小,但是面积较小的工件会被提前大量使用.e 通过公式计算取值为 0.12、0.08、0.05、0.03、0.02,通过遍历 e,有利于兼顾单块板材废料率和整体利用率的关系,并为后处

理算法留下优化余地.由于在算法中采用遍历 e 值的策略,因此大大降低了 e 值对算法的影响.

对单块板材排样,在不考虑切割选择的影响这一情况下,假设每次只选择一种切割方式,则搜索组块数量为 $k\times limit^{(m-1)}$.对 $A|B$ 类型待排矩形区执行排样时,$limit$ 取值越大,搜索组块的范围越大,则越容易找到利用率高的排样方案,但会极大地增加搜索时间.因此搜索步长 $limit$ 的取值,对算法搜索时间有显著影响,需要设置合理的 $limit$ 取值,兼顾排样方案质量和搜索效率的关系.在算法中,对无废料排样,$limit$ 取值为 2,对有废料排样,$limit$ 取值可以放宽一些,有利于找到利用率高的排样结果.

5 结 论

本文提出了求解单规格一刀切矩形排样问题的BSDBF启发式搜索算法,该算法基于组化规则,大胆地使用了二元组块排样,获得了良好的局部子解特性,同时避免了组块空间的无限扩大;提出的“大小工件分治择优匹配”策略和组块快速举荐算法,是对组化机制的关键补充,弥补了组化机制的不足;基于废料率的递归排样搜索,同时兼顾了局部解和最优解的关系;最后提出的基于“大小工件分治策略和利用率优劣集划分策略”相结合的排样结果后处理算法,利用局部二次重排,使算法具备了全局搜索能力.因为算法不具有随机特性,获得的排样结果可以复现,提高了算法的工程应用能力.对国际上大量 Benchmark 案例的计算结果表明,相对于其他算法,在排样质量上具有一定的优势.

为了保证算法的收敛性,算法设计了两种松弛机制.

其一,在对 C 类型和 $A|B$ 类型的板材进行排样时,组块的推荐范围逐步扩大,保证了算法的全局搜索特性.

其二,基于动态二分法的单块板材排样方案寻优,预设二分法上下限精度适当放宽,以降低板材利用率,加快算法收敛.

后续的研究包括:

(1) 进一步研究组化理论及相关技术,包括组块快速生成机制、组块重要性评估方法、组块互换性理论等.

(2) 在排样方案后处理机制上,进一步研究板材拆分策略,将已排样板材和拆分后的工件进行小规模重排,不断置换出数量更多的小面积工件,并迭代执行,二次优化重组,提高整体排样质量.

(3) 将算法进一步应用于多规格矩形件排样问题和三维装箱问题,拓展算法的工程应用前景.

(4) 在工程应用中,算法还需要根据不同工艺和原材料属性,进一步考虑边界补偿问题.例如,在用激光切割、水刀切割钢板时,过宽的切缝必须看成派生排样工件,进入待排样队列.再例如,玻璃切割后还需要进行粗细磨边,边界补偿可以包括在“磨边余量”这一变量中.具体应用赋予了排样问题新的特征,这也将成为排样问题新的研究方向.

致谢 在此,我们向本文引用参考文献的作者们及给予本文研究工作支持和建议的同行们表示衷心的感谢.

References:

[1] Wäscher G, Haußner H, Schumann H. An improved typology of cutting and packing problems. European Journal of Operational Research, 2007,183(3):1109–1130. [doi: 10.1016/j.ejor.2005.12.047]

[2] Lodi A, Martello S, Vigo D. Heuristics and metaheuristic approaches for a class of two-dimensional bin packing problems. Informs Journal on Computing, 1999,11(4):345–357. [doi: 10.1287/ijoc.11.4.345]

[3] George JA, George JM, Lamar BW. Packing different-sized circles into a rectangular container. European Journal of Operational Research, 1995,84(3):693–712. [doi: 10.1016/0377-2217(95)00032-L]

[4] Garey MR, Johnson DS. Computer and Intractability: A Guide of Theory of NP-Completeness. San Francisco: W. H. Freeman & Co. Ltd., 1979.

[5] Martello S, Vigo D. Exact solution of the two-dimensional finite bin packing problem. Management Science, 1998,44(3): 388–399. [doi: 10.1287/mnsc.44.3.388]

[6] Dell'Amico M, Martello S, Vigo D. A lower bound for the non-oriented two-dimensional bin packing problem. Discrete Applied Mathematics, 2002,118(1-2):13–24. [doi: 10.1016/S0166-218X(01)00253-0]

[7] Liu Y, Chu CB, Wang KL. A new heuristic algorithm for a class of two-dimensional bin-packing problems. The Int'l Journal of Advanced Manufacturing Technology, 2011,57(9):1235–1244. [doi: 10.1007/s00170-011-3351-1]

[8] Lee LS. A genetic algorithm for two-dimensional bin packing problem. Math Digest, 2008,2(1):34–39.

[9] Hong SH, Zhang DF, Lau HC, Zeng XX, Si YW. A hybrid heuristic algorithm for the 2D problem variable-sized bin packing. European Journal of Operational Research, 2014,238(1):95–103. [doi: 10.1016/j.ejor.2014.03.049]

[10] Sotelo-Figueroa MA, Soberanes HJP, Carpio JM, Huacuja HJF, Reyes LC, Soria-Alcaraz JA. Improving the bin packing heuristic through grammatical evolution based on swarm intelligence. Mathematical Problems in Engineering, 2014,2014(1): 1−12. [doi: 10.1155/2014/545191]

[11] Zhang DF, Kang Y, Deng A. A new heuristic recursive algorithm for the strip rectangular packing problem. Computers & Operations Research, 2006,33(8):2209–2217. [doi: 10.1016/j.cor.2005.01.009]

[12] Zhang DF, Han SH, Ye WG. A bricklaying heuristic algorithm for the orthogonal rectangular packing problem. Chinese Journal of Computers, 2008,31(3):509−515 (in Chinese with English abstract). [doi: 10.3321/j.issn:0254-4164.2008.03.017]

[13] Jiang XB, Lü XQ, Liu CC. Lowest-Level left align best-fit algorithm for the 2D rectangular strip packing problem. Ruan Jian Xue Bao/Journal of Software, 2009,20(6):1528–1538 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3395.htm [doi: 10.3724/SP.J.1001.2009.03395]

[14] Peng BT, Zhou YW. Recursive heuristic algorithm for the 2D rectangular strip packing problem. Ruan Jian Xue Bao/Journal of Software, 2012,23(10):2600–2611 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4187.htm [doi: 10.3724/SP.J.1001.2012.04187]

[15] Cao DY, Yang M, Kotov VM, Liu RT. Two-Stage heuristic algorithm for two-dimensional guillotion bin packing problem. Computer Integrated Manufacturing Systems, 2012,18(9):1954–1963 (in Chinese with English abstract). [doi: 10.13196/j.cims.2012.09.54.caody.013]

[16] Berkey JO, Wang PY. Two-Dimensional finite bin-packing algorithms. Journal of the Operational Research Society, 1987,38(5): 423–429. [doi: 10.1057/jors.1987.70]

[17] Lodi A, Martello S, Vigo D. Neighborhood search algorithm for the guillotine non-oriented two-dimensional bin packing problem. In: Voß S, Martello S, Osman I, Roucairol C, eds. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization. Kluwer Academic Publishers, 1999. 125–139. [doi: 10.1007/978-1-4615-5775-3_9]

[18] Polyakovsky S, M'Hallah R. An agent-based approach to the two-dimensional guillotine bin packing problem. European Journal of Operational Research, 2009,192:767–781. [doi: 10.1016/j.ejor.2007.10.020]

[19] Parreño F, Alvarez-Valdes R, Oliveira JF, Tamarit JM. A hybrid GRASP/VND algorithm for two- and three-dimensional bin packing. Annals of Operations Research, 2010,179(1):203–220. [doi: 10.1007/s10479-008-0449-4]

[20] Alvelos F, Chan TM, Vilaca P, Gomes T, Silva E, Valério de Carvalho JM. Sequence based heuristics for two-dimensional bin packing problems. Engineering Optimization, 2009,41(8):773–791. [doi: 10.1080/03052150902835960]

[21] Chan TM, Alvelos F, Silva E, Valério de Carvalho JM. Heuristics with stochastic neighborhood structures for two-dimensional bin packing and cutting stock problems. Journal of Operational Research, 2011,28(2):255–278. [doi: 10.1142/S0217595911003168]

[22] Alvarez-Valdes R, Parreño F, Tamarit JM. A GRASP/Path relinking algorithm for two- and three-dimensional multiple bin-size bin packing problems. Computers & Operational Research, 2013,40(12):3081–3090. [doi: 10.1016/j.cor.2012.03.016]

[23] Charalambous C, Fleszar K. A constructive bin-oriented heuristic for the two-dimensional bin packing problem with guillotine cuts. Computers & Operations Research, 2011,38(10):1443–1451. [doi: 10.1016/j.cor.2010.12.013]

[24] Krzysztof F. Three insertion heuristics and a justification improvement heuristic for two-dimensional bin packing with guillotine cuts. Computers & Operations Research, 2013,40(1):463–474. [doi: 10.1016/j.cor.2012.07.016]

[25] Kröger B. Guillontineable bin-packing: A genetic approach. European Journal of Operational Research, 1995,84:645–661. [doi: 10.1016/0377-2217(95)00029-P]

[26] Cui YD. An optimum algorithm for stock layout of square blank. Modern Manufacturing Engineering, 1998,6:32–33 (in Chinese with English abstract). [doi: 10.16731/j.cnki.1671-3133.1998.06.017]

[27] He L. Research on scroll packing problem with guillotine constraint [MS. Thesis]. Guangzhou: Guangdong University of Technology, 2014 (in Chinese with English abstract). [doi: 10.7666/d.Y2581651]

[28] Christofides N, Whitlock C. An algorithm for two-dimensional cutting problems. Operations Research, 1977,25 (1):30–44. [doi: 10.1287/opre.25.1.30]

[29] Beasley JE. Algorithms for unconstrained two-dimensional guillotine cutting. Journal of the Operational Research Society, 1985a,36:297–306. [doi: 10.1057/jors.1985.51]

[30] Beasley JE. An exact two-dimensional non-guillotine cutting tree search procedure. Operations Research, 1985b,33(1):49–64. [doi: 10.1287/opre.33.1.49]

[31] Clautiaux F, Jouglet A, Hayek JE. A new lower bound for the non-oriented two-dimensional bin-packing problem. Operations Research Letters, 2007,35(3):365–73. [doi: 10.1016/j.orl.2006.07.001]

[32] Hopper E, 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. [doi: 10.1016/S0377-2217(99)00357-4]

附中文参考文献:

[12] 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,2008,31(3):509–515. [doi: 10.3321/j.issn:0254-4164.2008.03.017]

[13] 蒋兴波,吕肖庆,刘成城.二维矩形条带装箱问题的底部左齐择优匹配算法.软件学报,2009,20(6):1528–1538. http://www.jos.org.cn/1000-9825/3395.htm [doi: 10.3724/SP.J.1001.2009.03395]

[14] 彭碧涛,周永务.求解 2D 条带矩形 Packing 问题的迭代启发式算法.软件学报,2012,23(10):2600–2611. http://www.jos.org.cn/1000-9825/4187.htm [doi: 10.3724/SP.J.1001.2012.04187]

[15] 曹大勇,杨梅,科托夫·弗拉基米尔·米哈伊拉维,刘润涛.二维一刀切装箱问题的两阶段启发式算法.计算机集成制造系统,2012,18(9):1954–1963. [doi: 10.13196/j.cims.2012.09.54.caody.013]

[26] 崔耀东.矩形毛坯下料排样的一种优化算法.现代制造工程,1998,6:32–33. [doi: 10.16731/j.cnki.1671-3133.1998.06.017]

[27] 何霖.满足一刀切约束的卷型材矩形件排样方法研究[硕士学位论文].广州:广东工业大学,2014. [doi: 10.7666/d.Y2581651]

王磊(1981－),男,河南开封人,博士,讲师,主要研究领域为计算机集成制造,智能算法,排样优化.

刘强(1978－),男,博士,教授,主要研究领域为智能制造系统建模与优化.

陈新(1960－),男,博士,教授,博士生导师,主要研究领域为网络化制造,微电子制造及优化.