软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(9):2078−2088 [doi: 10.3724/SP.J.1001.2013.04404]

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

求解二维矩形 Packing 面积最小化问题的动态归约算法*

何 琨[1], 姬朋立[1], 李初民[1,2]

[1](华中科技大学 计算机科学与技术学院,湖北 武汉 430074)

[2](School of Computer Science and Technology, University of Picardie Jules Verne, Amiens 80039, France)

通讯作者: 何琨, E-mail: brooklet60@gmail.com

摘 要: 二维矩形 Packing 面积最小化问题(rectangle packing area minimization problem,简称 RPAMP)是具有 NP 难度的高复杂度的布局优化问题,也是大规模集成电路设计中 floorplanning 问题的一个核心问题.通过动态构造矩形框的宽和高,将求解一个 RPAMP 转化为求解一组二维矩形 Packing 判定问题(rectangle packing decision problem,简称 RPDP).在求解 RPDP 的最大适配度算法的基础上,进一步考虑了当前动作对全局紧凑性的影响,评估了当前动作对局部空间的损害程度,设计了求解 RPDP 的最小损害度算法.然后,结合矩形框宽、高的动态构造方法,设计得到求解 RPAMP 的最终算法.对 15 个相关的 RPAMP 算例(包括著名的 MCNC 算例和 GSRC 算例)进行了测试.更新了其中 9 个算例的最好记录,另有 2 个与当前的最好记录持平.得到了 98.50%的平均填充率,将国内外文献中已见报道的最高平均填充率提高了 0.85%.

关键词: NP 难度;布局优化;布图规划;面积最小化;启发式

中图法分类号: TP181 **文献标识码**: A

中文引用格式: 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,2013,24(9): 2078−2088. http://www.jos.org.cn/1000-9825/4404.htm

英文引用格式: He K, Ji PL, Li CM. Heuristics for solving the 2D rectangle packing area minimization problem basing on a dynamic reduction method. Ruan Jian Xue Bao/Journal of Software, 2013,24(9):2078−2088 (in Chinese). http://www.jos.org.cn/1000-9825/4404.htm

Heuristics for Solving the 2D Rectangle Packing Area Minimization Problem Basing on a Dynamic Reduction Method

HE Kun[1], JI Peng-Li[1], LI Chu-Min[1,2]

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

[2](School of Computer Science and Technology, University of Picardie Jules Verne, Amiens 80039, France)

Corresponding author: HE Kun, E-mail: brooklet60@gmail.com

Abstract: This paper addresses an NP-hard layout optimization problem with a high computational complexity: the two-dimensional rectangle packing area minimization problem (RPAMP), which is a core issue of floorplanning problem in the very-large-scale integration (VLSI) design. First, by dynamically designing the two dimensions of the large rectangular frame, the study reduces the solving of a RPAMP to the solving of a series of two-dimensional rectangle packing decision problems (RPDP). Then, based on a best-fit-degree approach for the RPDP, the designs a least-damage-first algorithm for the RPDP, which further takes the consideration of the current placement's impact on global compaction and of its negative effect on local space's integrity. Next, by combining the method of dynamically designing two dimensions of the rectangular frame, a final dynamic reduction algorithm is proposed for solving the RPAMP. Experiments were on 15 RPAMP instances (including the well-known MCNC instances and GSRC instances). Computational results show that the proposed algorithm refreshed the current best solutions on nine instances. At the same time it also matchs the current best records

* 基金项目: 国家自然科学基金(61173180, 61272014)
收稿时间: 2012-12-03; 修改时间: 2013-01-25; 定稿时间: 2013-03-22

on two other instances. The obtained average filling rate is 98.50%, which improved the current best results reported in the literature by 0.85%.

Key words: NP hard; layout optimization; floorplanning; area minimization; heuristic

Floorplanning 问题[1−3]是大规模集成电路(very large scale integration,简称 VLSI)设计的一个重要步骤,它研究如何将一组给定的矩形模块放置到一个矩形芯片上.模块间根据逻辑结构的设计要求有线相连,问题的目标是,最小化矩形芯片的面积和模块间的总连线长度.本文研究 floorplanning 问题的一个核心子问题,即只考虑最小化芯片面积的二维矩形 Packing 面积最小化问题(rectangle packing area minimization problem,简称 RPAMP)[4−7]. RPAMP 在 1977 年已被证明是一个 NP 难度问题[8].

由于 RPAMP 计算的高复杂度,目前,求解 RPAMP 的精确算法[9]最多能处理 30 个模块的数据.设计高效启发式算法成为国内外研究的主要趋势,代表性算法可分为随机型算法和确定型算法两大类.随机型算法着眼于设计能够表示各矩形块间位置关系的编码以及相应的操作,主要包括遗传算法[10−13]、模拟退火算法[4,5,14]、局部搜索算法[6]等;确定型算法着眼于设计矩形块紧密放置的策略,主要包括分枝限界法[7]和贪心算法[15,16]等.矩形 Packing 问题是 RPAMP 的一种简化形式,RPAMP 中待放矩形框的尺寸大小是开放型的,而矩形 Packing 中待放矩形框或容器的形状和大小是固定的.近几十年来,国内外学者对矩形 Packing 问题进行了深入研究,二维矩形 Packing 方面设计出了 BLD(bottom-left-decreasing)算法[17]、BFA(best-fit)算法[18]等;三维矩形 Packing 方面有分层法[19]、最大空间法[20]、最大匹配度法[21]等.

本文利用待放块边长出现的频率,通过贪心地构造一组候选框宽,并动态地选取候选框宽和基于当前最好填充率设置候选框高,将求解一个 RPAMP 转化为求解一组 RPDP.基于求解 RPDP 的最大适配度算法[22],进一步考虑了当前动作对后续放入的影响,设计了求解 RPDP 的算法;然后,结合候选框宽、高的动态构造方法,得到最终的求解 RPAMP 的算法.测试了 15 个 RPAMP 算例,所设计的算法更新了 9 个算例的当前最优解,平均填充率比已见发表的最好结果高出了 0.85 个百分点,且在普通 PC 机上的平均计算时间仅为 200s.

本文第 1 节定义 RPAMP 以及将其归约为 RPDP 的方法.第 2 节详细阐述求解 RPDP 的最小损害度算法.在最小损害度算法的基础上,第 3 节提出求解 RPAMP 的动态归约算法.第 4 节进行实验计算并与求解 RPAMP 的代表性算法进行比较.最后,第 5 节进行总结和展望.

1 问题描述和归约方法

二维矩形 Packing 面积最小化问题是指:在二维欧氏空间中,已知 n 个宽、高分别任意给定的矩形块,问如何确定一个面积尽量小的矩形框,使得所有矩形块都能够合法地放入.合法是指放入后任意矩形块不超出框的边界、边平行于框的边,且矩形块间两两互不重叠.

将矩形框的左下角置于坐标原点,边平行于坐标轴.作为问题的解,应给出矩形框的宽和高以及每个矩形块的放置位置(矩形块的左下角坐标)和方向(竖直或水平).

图 1(a)中给出了一个 RPAMP 实例.图 1(b)和图 1(c)为该实例的两个合法解,它们对应的矩形框尺寸分别为 3×3 和 3×2.图 1(c)的填充率达到了 100%,所以不可能有面积更小的矩形框,是此实例的一个最优解.

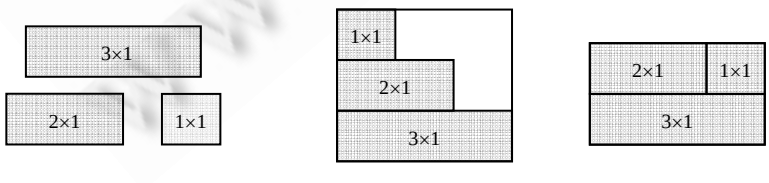


Fig.1 A RPAMP instance and its feasible solutions

图 1 一个 RPAMP 实例及可能的合法解

由于 RPAMP 中矩形框没有给定,直接求解有一定的困难,本文将其归约为一组矩形框给定的二维矩形 Packing 判定问题来求解.

二维矩形 Packing 判定问题是指:在二维欧氏空间中,已知一个宽、高分别任意给定的矩形框和 n 个宽、高分别任意给定的矩形块,问能否将矩形块全部合法地放入框内.合法的含义与 RPAMP 的相同.

归约的思路为:动态构造一组矩形框的候选宽和高$\{\langle w_1,h_1\rangle,...,\langle w_p,h_p\rangle\}$,$\forall i\in\{1,...,p\}$,$w_i\times h_i\geqslant S$($S$ 为所有矩形块面积之和),并分别判定能否将 n 个矩形块全部合法放入,从而得到 p 个 RPDP 实例.第 i+1 个实例的框宽和框高满足以下条件:若第 i 个 RPDP 实例的判定为真,则以适当的方式缩小第 i+1 个实例的框面积;否则,以适当的方式放大第 i+1 个实例的框面积,但仍小于上一次成功放入对应的框面积.用 RPDP 算法顺序计算这组 RPDP 实例,在此过程中,如果当前实例的判定为真,那么该实例的解加上它对应的框宽和框高就是 RPAMP 的当前最优解.如此迭代执行,计算结果为最后一个判定为真的 RPDP 实例的解和它对应的框宽和框高.

2 求解 RPDP 的最小损害度算法

本节以文献[22]的最大适配度算法为基础,提出了一种求解 RPDP 的改进算法——最小损害度算法(least-damage-first,简称 LDF).最大适配度算法的主要思想是:挑选一个矩形块放入到框内的某个角区,使其与其他已放入矩形块或框的边尽量靠拢,且对剩余空间的影响尽量小.改进的 RPDP 算法继承了上述思想,同时又考虑了格局的整体紧凑性和放入矩形块后对剩余空间的影响,使当前动作对后续放入的损害尽量小.第 2.1 节给出改进的角区、占角动作定义和本文提出的匹配度、损害度定义,第 2.2 节和第 2.3 节给出了求解 RPDP 的最小损害度基本算法和带回溯的树搜索算法.

2.1 相关概念的定义

定义 1(格局). 设某一时刻,若干个矩形块已合法放入到框内,还有若干个待放,这称为一个格局.初始格局时,框内未放入任何矩形块.当所有矩形块都已放入框中或框外的剩余矩形块均无法再放入时,称为终止格局.若终止格局将所有块全部放入,则称为一个成功布局.

定义 2(动作空间). 在当前格局下,将一个很小的虚拟矩形块合法放入到框中,然后,小矩形块向上、下、左、右 4 个方向分别膨胀,使膨胀后的虚拟矩形块的每条边均与已放入矩形块或框的边相贴(两条边相贴是指它们的重叠长度大于 0),该虚拟矩形块所覆盖的区域就称为当前格局下的一个动作空间.如果当前所有待放矩形块都不能合法地放入某个动作空间,则称此空间为无效动作空间.

图 2(a)中给出了一个格局,图 2(b)、图 2(c)中标注的 A,B,C,D 就为该格局下全部的动作空间.由于 A 的一条边长度小于框外矩形块 a 和 b 的任意一条边,所以 a 和 b 都不能合法地放入其中,因此,A 是一个无效动作空间.

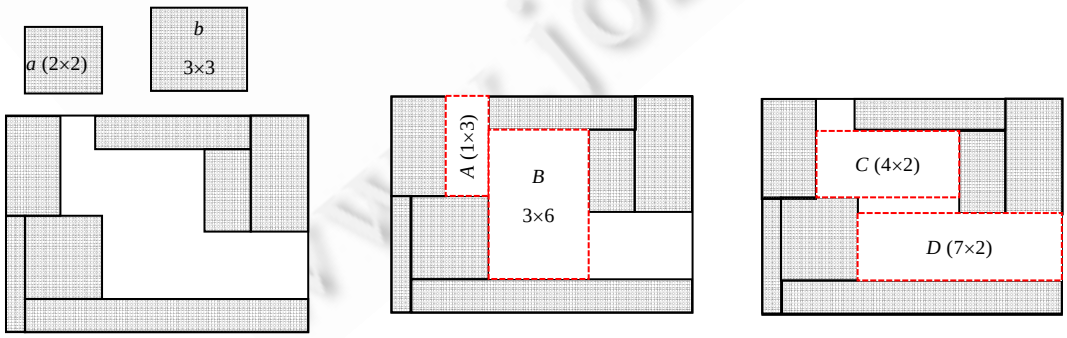


Fig.2 Action spaces at the current pattern

图 2 当前格局下的动作空间

定义 3(角区). 每个动作空间的每个角均称为一个角区.

根据构成角区的边的类型,角区可分为两类:

(1) 实角区:构成角区的两条边均是已放入矩形块或框的边;

(2) 虚角区:构成角区的两条边中至少有一条是已放入矩形块边的延长线.

如图 2(b)所示,动作空间 B 的左下角和右下角对应的分别为实角区和虚角区.

定义 4(占角动作). 在当前格局下,若将一个矩形块放入到动作空间的一个角区,使矩形块的顶点和角区的顶点重合,顶点所对应的边分别与角区的两条边相贴,且该放入矩形块不超出动作空间,则称此动作为一个占角动作.

根据放入矩形块是否可以部分移出相应的动作空间,占角动作可分为两类:

(1) 实占角动作:放入块不能沿着放入的角区至少部分地移出该动作空间;

(2) 虚占角动作:放入块可以沿着放入的角区至少部分地移出该动作空间.

在图 2(b)中,若将 b 放到 B 的右下角,因为 b 向右水平、向下竖直都不能部分地移出动作空间 B,所以该动作是一个实占角动作;若将 a 放到 B 的右下角,a 可以向右水平部分地移出动作空间 B,所以该动作是一个虚占角动作.

定义 5(匹配度). 匹配度是用来衡量动作空间和矩形块尺寸相似程度的一个概念.

动作空间与矩形块的匹配度分为 3 个级别:

级别 1:它们的两条边分别等长;

级别 2:一条边等长,且动作空间的另外一条边大于矩形块的另外一条边;

级别 3:其他情况.

定义 6(损害度). 损害度表示做完一个动作对后续放入的损害程度,定义为一个五元组,即⟨未贴边数,剩余动作空间数,角区优先级,占角动作的实虚度,动作匹配度⟩.

(1) 未贴边数 p_i:放入矩形块与当前动作空间相贴的边数目 $k_i\in\{2,3,4\}$,未贴边数 $p_i=4,\ldots,k_i$;

(2) 剩余动作空间数 n_i:表示矩形块放入后,新格局下动作空间的数目;

(3) 角区优先级 c_i:动作空间的左下角和右下角的优先级定义为 1,左上角和右上角的优先级定义为 2. $c_i\in\{1,2\}$;

(4) 占角动作的实虚度 l_i:根据占角动作的实虚和对应角区的实虚,定义占角动作的实虚度优先级. $l_i\in\{1,2,3\}$:

优先级 1:占角动作对应实角区;

优先级 2:占角动作对应虚角区,但该动作为实占角动作;

优先级 3:占角动作为虚占角动作;

(5) 动作匹配度 m_i:表示执行完一个动作后新生成的 0 至多个动作空间与框外剩余矩形块之间的匹配程度.取这些动作空间和框外剩余矩形块的最小匹配度作为该动作的匹配度,若执行完一个动作后没有新生成的动作空间,则定义它的匹配度为 1.$m_i\in\{1,2,3\}$.

两损害度的比较为按字典序依次比较相应的元素.例如,损害度 a:⟨1,2,1,2,3⟩和损害度 b:⟨1,2,2,1,1⟩的前两项分别相等,a 的第 3 项小于 b 的第 3 项,所以 $a<b$.

2.2 基本算法

基本算法 LDF_0 的执行过程是:从当前格局开始,每一步挑选一个损害度最小的占角动作来做,往框内放入一个矩形块,从而形成新的格局.如此反复,直至终止格局.若终止格局中所有矩形块都已放入框内则返回真,否则返回假.

算法 LDF_0.

输入:矩形框的宽、高和矩形块集合;

输出:终止格局对应的布局和判定结果.

Step 1. 在当前格局下,考虑每个角区、框外剩余的每个块、块的不同放置方向,将块依据给定方向放入到角区,若构成一个占角动作,则计算其损害度.

Step 2. 依字典序,按如下规则挑选一个对后续放入损害最小的占角动作来做:

(1) 损害度:小优先;

(2) 放入块的面积:大优先;

(3) 块的长边长:大优先;

(4) 块左下顶点的 x 坐标:小优先;

(5) 块左下顶点的 y 坐标:小优先;

(6) 块的方向:躺优先;

Step 3. 依据如下规则更新动作空间链表,得到一个新的格局:

(1) 检查和更新与放入矩形块相嵌的动作空间;

(2) 去掉有包含关系的多余动作空间和无效动作空间;

Step 4. 重复 Step 1~Step 3,直至终止格局.

Step 5. 若终止格局中所有块都已放入框内,则判定为真,否则判定为假.同时,返回终止格局对应的布局.

在 Step 2 中,规则(1)选择一个损害度最小的动作来做,当通过损害度选出的动作不唯一时,规则(2)~规则(6)用来确定唯一一个占角动作:规则(2)和规则(3)确定了块的形状大小,规则(4)和规则(5)指明了块的放入位置,规则(6)明确了块的放置方向.

2.3 回溯策略

在 LDF_0 的基础上加入回溯策略,得到增强算法 LDF_1.LDF_1 每一步对损害度较小的若干个占角动作做进一步的观察,将做此动作后若执行基本算法 LDF_0 可得到的面积利用率作为预期结果,然后从中选择一个预期结果最好的动作来做.

算法 LDF_1.

输入:矩形框的宽、高和矩形块集合;

输出:终止格局对应的布局和判定结果.

Step 1. 在当前格局下,考虑每个角区、框外剩余的每个块、块的不同放置方向,将块依据给定方向放入到角区,若构成一个占角动作,则计算其损害度.

Step 2. 用 LDF_0 的排序规则对当前所有占角动作排序,选取排在前面的 N 个占角动作,其中,

$$N=\lfloor \text{当前格局下所有占角动作的数目}\times k\% \rfloor.$$

若 $N<lowerbound$,则 $N=lowerbound$;

否则,若 $N>upperbound$,则 $N=upperbound$.

Step 3. 对选取的前 N 个占角动作,将做此动作后若执行基本算法 LDF_0 可得到的面积利用率作为预期结果.选择预期结果最好的占角动作来做,如果预期结果最好的动作有多个,就选择排在最前面的一个动作.

Step 4. 更新动作空间链表,得到一个新的格局.

Step 5. 重复 Step 1~Step 4,直至终止格局.

Step 6. 若终止格局中所有块都已放入框内,则判定为真,否则判定为假.同时,返回终止格局对应的布局.

在增强算法中,我们并没有考虑所有的占角动作,只是按一定比例选取损害度较小的动作进行评估.因为一般而言,当前较优的动作得到较好的最终格局的可能性也较大.这样优中选优的策略就可以在保证解优度的情况下减少计算时间.另外,当候选的占角动作较少时,为了避免搜索的范围过小,我们对 N 的取值设定了一个下界;而当候选的占角动作过多时,考虑到计算时间,则只选择损害度较小的固定数量的动作进行回溯.

3 求解 RPAMP 的动态归约算法

求解 RPAMP 的动态归约算法(dynamic reduction algorithm,简称 DRA)的思路是:根据 RPAMP 的待放矩形

块集合,初始化一组候选框宽和一个期望填充率;动态构造 RPDP 实例并用 LDF_0 和 LDF_1 计算,每次构造由当前候选宽、期望填充率和公式(1)可得到一个 RPDP 实例,若该实例判定为真,则用所得布局及相应的框宽、框高更新 RPAMP 的当前最优解,根据本次计算结果调整当前候选框宽和期望填充率,使得下次构造的 RPDP 实例若判定为真,所得布局的填充率大于 RPAMP 的当前最优解对应的填充率.动态构造结束,直到最后一个候选框宽不能更新当前 RPAMP 最优解.

$$\text{框高}=\frac{\text{所有矩形块面积之和}}{\text{期望填充率}\times\text{框宽}} \tag{1}$$

下面给出若干概念的定义和动态归约算法的具体实现.

3.1 相关概念的定义

定义 7(*k* 级组合长度). k 个矩形块各自提供一条边,这些边的长度之和,称为一个 k 级组合长度.

给定 n 个矩形块,可以构成的所有 $k(k\leqslant n)$级组合长度的个数为 $C_n^k\cdot 2^k$,其中,C_n^k 是从 n 个矩形块中选 k 个所有可能的组合,2^k 是从 k 个矩形块的每一块各取一条边的不同取法的个数.

定义 8(向下紧凑动作). 布局中,所有矩形块按其左下角 x 坐标非降序排列,并按此顺序,将矩形块逐个向下竖直移动,直至其底部边与某矩形块的顶部边或框的底部边相贴.

定义 9(向左紧凑动作). 布局中,所有矩形块按其左下角 y 坐标非降序排列,并按此顺序,将矩形块逐个向左水平移动,直至其左部边与某矩形块的右部边或框的左部边相贴.

定义 10(紧凑布局). 若一个布局中框的每条边均与矩形块的边相贴,且框内任意块均不能向左水平或向下竖直移动,则称此布局为紧凑布局,否则为松散布局.

定义 11(紧凑动作). 对布局做以下操作:首先,交替执行向下紧凑动作和向左紧凑动作,直到两次连续的动作都没有移动任何矩形块;然后收缩矩形框,使它的框宽和框高分别为所有矩形块右上角 x 坐标的最大值和 y 坐标的最大值.

例如,对图 3(a)中的松散布局,执行向下紧凑动作可得到图 3(b)中的布局.接着,执行向左紧凑动作并收缩矩形框后可得到图 3(c)中的布局.可见,该布局是一个紧凑布局,且填充率比图 3(a)的要大.

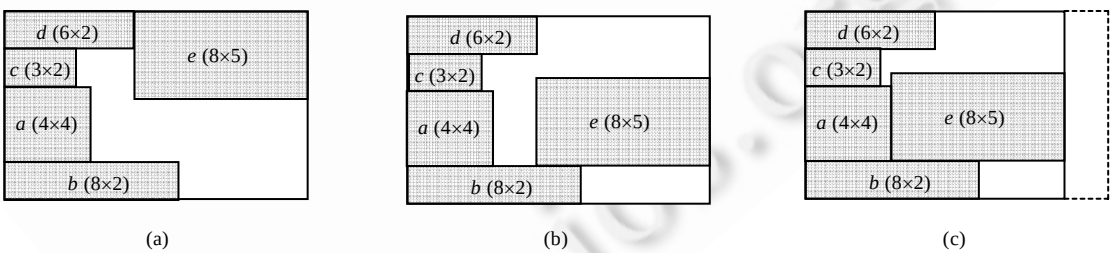


Fig.3 Transforming loose pattern to compact pattern

图 3 松散布局转化为紧凑布局

对任意布局执行以上紧凑动作,都会得到一个填充率不低于该布局的紧凑布局.在实际计算时,算法 LDF_0 和算法 LDF_1 计算出的布局有可能不是紧凑的.因此,可在动态归约算法每一步迭代结束时,对成功的布局进行紧凑操作.

3.2 算法DRA

算法 DRA 首先根据 RPAMP 的待放矩形块集合,确定一组组合长度作为候选框宽,并初始一个期望填充率,然后进行迭代计算.在每一步迭代,利用当前框宽、期望填充率和公式(1)算出当前框高,则〈当前框宽,当前框高,期望填充率〉构成一个 RPDP 实例;DRA 接着利用 LDF_0 或 LDF_1 计算该实例:若判定为真,紧凑所得布局,且若所得紧凑布局的填充率大于当前最优填充率,则紧凑布局及相应的框宽、框高为 RAPMP 的当前最优解.在下一次

迭代开始时,根据本次迭代的判定结果和当前填充率,判断是否取下一个候选框宽作为当前框宽,并相应地调整期望填充率.如此反复执行,直至最后一个候选框宽不能更新 RPAMP 的当前最优解.具体描述如下:

算法 DRA.

输入:组合长度级别上限 k,矩形块集合 M;

输出:计算出的最优解.

Step 1. 初始化:

计算所有 1~k 级组合长度,从中选取出现频率最高的前 l 个存放在数组 $W(1,\ldots,l)$中作为候选框宽;

初始化填充率增幅的递减数组 $I(1,\ldots,m)$;

框宽指针 i=1,填充率增幅指针 j=1;

初始化期望填充率 fr,当前最优填充率 $fr\_best=fr$;

Step 2. 检验初始期望填充率的合法性:

```
While (true) {
  由 w1 和 fr 构造 RPDP 实例;
  用算法 LDF0 计算该实例得到布局 X;
  if (所有矩形块都处于 X 中)
    break;
  else{
    if (fr>0.1)
      fr=fr−I1;
    else
      fr=fr−Im;}
}
```

Step 3. 由 w_i 和 fr 构造 RPDP 实例,并用最小损害度算法进行计算:

```
利用公式(1)计算当前框高 h;
if (矩形块规模<100)
  用算法 LDF1 计算该实例得到布局 X;
else
  用算法 LDF0 计算该实例得到布局 X;
```

Step 4. 根据计算结果更新当前最优解,并适当调整框宽指针和填充率增幅指针:

```
If (所有矩形块都处于 X 中){
  紧凑 X 得到布局 X′;
  if (X′的填充率>fr_best){
    更新最优解为 X′及其对应的框宽、框高;
    fr_best=X′的填充率;
  }
  if (j=m 且 wi 是第 1 次作为框宽进行计算)
    j=m−1;
  fr=fr+Ij;          //更新期望填充率
} else {
  fr=fr−Ij;          //恢复期望填充率
  if (j=m){
    if (i=l)
```

```
            退出程序,并返回最优解;
          else
            i=i+1;          //更新框宽
        } else
          j=j+1;
        fr=fr+I_j;          //更新期望填充率
      }
```

Step 5. 跳转到 Step 3.

一般认为,由已知矩形的边长以较高的频率组合而成的长度,作为框宽计算可得到较好的解.在算法 DRA 中,选取 1~k 级组合长度中出现频率最高的若干个作为候选框宽.

图 4(a)中给出了一个 RPAMP 实例.在它的所有 1 级、2 级组合长度中,出现频率最高的长度为 10 和 6,它们出现的频率分别为 6,4.将 10 和 6 作为框宽,分别用算法 DRA 计算,得到的布局如图 4(b)、图 4(c)所示.显然,长度 10 作为框宽得到了更优的解.

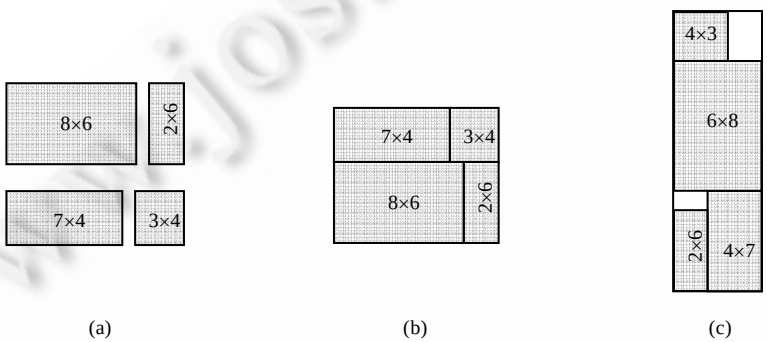


Fig.4 A RPAMP instance and its feasible solutions within different widths

图 4 RPAMP 实例和不同框宽对应的解

在 DRA 算法中,当前填充率增幅的变化策略主要是基于以下考虑:第 1 个候选框宽作为框宽,最优填充率一般能有较大幅度的优化;初始化当前填充率增幅为其上界,每次不能找到合法的 RPAMP 解时,依次减小是合理的.然而从第 2 个候选框宽开始,由于当前填充率已较高,不太可能大幅度提高当前最优填充率.如果每次更新框宽后,仍然将当前填充率增幅设为较大的值,就会计算许多无效的候选框宽和填充率组合.为了优化程序的速度,在每次更新框宽后,要先测试一下更新后的框宽是否可以继续优化当前最优解,也就是将当前填充率增幅设为其下界进行计算.若测试结果找到一个更优的 RPAMP 解,则下次迭代将当前填充率增幅设为一个次小的值;否则,继续测试下一个候选框宽.这样,就使得算法能够在保证解优度的同时尽量减少无效的计算,从而提高计算的速度.

4 计算结果

我们将算法 DRA 用 C++语言编程,并在 AMD Athlon (3.01GHz, 2.0GB,Windows XP)个人计算机上进行了计算.算法 LDF_1 的参数 k,*lowerbound*,*upperbound* 分别设为 25,60 和 100;算法 DRA 的参数 fr,m 分别设为 0.5 和 5,数组 $I(5)$=(0.100,0.050,0.020,0.005,0.001).针对 k=2 和 k=4 分别找出出现频率最高的前 50 个组合长度,从而得到 100 个组合长度作为候选框宽.

测试数据为 15 个 RPAMP 相关算例,Apte,Xerox,Hp,Ami33 和 Ami49 出自 MCNC[23],N10,N30,N50,N100,N200,N300 出自 GSRC[24],其余 4 个算例出自文献[6].根据待放矩形块数目是否小于 100,分为小规模算例和大规模算例;根据待放矩形块个数与矩形块种数的比例,分为弱异构算例和强异构算例.算例的规模越大、异构性

越强,计算的难度就越大.其中,MCNC 中都是小规模弱异构型算例;GSRC 包括小规模算例和大规模算例,小规模算例中任意两个矩形块都不相同是强异构类型,大规模算例存在较多的矩形块两两相同是弱异构类型;Imahori 中的算例都是大规模弱异构类型的.

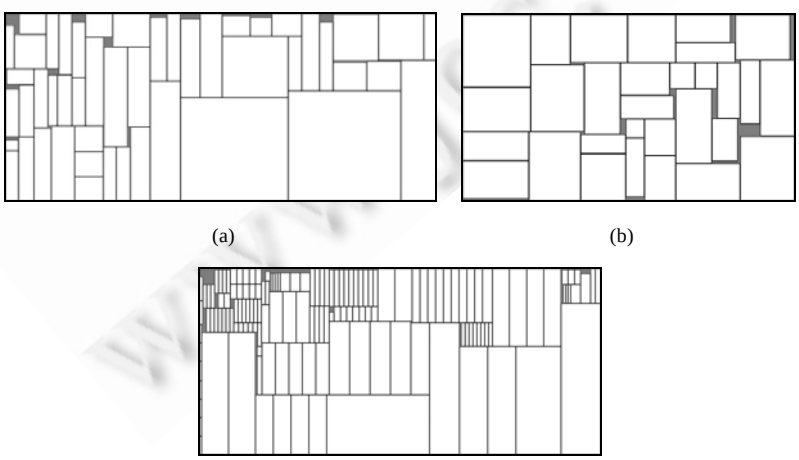
表 1 给出了 DRA 与 8 个代表性算法计算结果的比较,其中,*fr* 为填充率;DRA 结果中,追平当前最好记录的用粗体显示、超过的用粗斜体显示.由表 1 可见:DRA 有 9 个结果优于当前最好记录,2 个与之持平;针对 7 个大规模算例,DRA 更新了其中 6 个的最好结果,且填充率的更新幅度都在 0.7%之上.得到的平均填充率 98.50%比目前最好结果高出了 0.85 个百分点,因此,DRA 算法在计算优度上有较明显的优势,尤其是针对大规模算例.在计算速度方面,由于其他算法没有对这 15 个算例全部进行计算,所以不好比较.但总的来说,DRA 的计算速度基本处于中等.

Table 1 Comparisons with other algorithms

表 1 相关算法的计算结果比较

算例	矩形块的个数	Chan& Markov[7]		Imahori, *et al.*[6]		Wu& Chan[16]		Kimura& Ida[10]		Chen& Huang[15]		Pisinger[4]		Fernando& Katkoori[11]		Li, *et al.*[5]		目前最好结果	DRA 的结果	
		fr(%)	*t*(s)	*fr*(%)	*t*(s)	*fr*(%)	*t*(s)	*fr*(%)	*t*(s)	*fr*(%)	*t*(s)	*fr*(%)	*t*(s)	*fr*(%)	*t*(s)	*fr*(%)	*t*(s)	*fr*(%)	*fr*(%)	*t*(s)
Apte	9	99.23	2.38	–	–	–	–	–	–	99.23	0.01	99.23	0.58	–	–	99.23	0.4	99.23	**99.23**	1
Xerox	10	97.75	9812	–	–	–	–	–	–	97.68	0.01	97.71	0.68	–	–	97.58	0.7	97.75	97.04	3
Hp	11	98.70	891	–	–	–	–	–	–	98.67	0.02	98.70	0.75	–	–	98.70	0.6	98.70	**98.70**	4
Ami33	33	92.52	1.73	–	–	98.25	10	74.29	1800	98.84	2.01	99.01	1 359	96.37	640	98.09	4.7	99.01	98.77	198
Ami49	49	92.84	3.01	97.37	100	97.81	77	96.77	1800	98.27	6.61	98.48	3 004	93.75	1 384	98.30	6.7	98.48	***98.58***	786
N10	10	–	–	–	–	–	–	–	–	–	–	–	–	95.00	181	–	–	95.00	***97.06***	5
N30	30	–	–	–	–	–	–	94.44	1800	–	–	–	–	92.74	–	–	–	94.44	***97.84***	261
N50	50	–	–	–	–	–	–	91.99	1800	–	–	–	–	91.51	–	98.50	7.1	98.50	98.42	1 110
N100	100	93.38	5.62	–	–	99.85	1	–	–	98.57	8.22	–	–	87.24	–	97.71	25.6	99.85	97.72	4
N200	200	91.97	7.09	–	–	95.91	6	–	–	98.64	9.70	–	–	81.63	4789	96.82	124	98.64	***99.38***	19
N300	300	91.97	11	–	–	96.52	17	–	–	98.75	37.23	–	–	81.20	–	96.16	215	98.75	***99.61***	68
Rp100	100	–	–	97.08	200	–	–	–	–	–	–	–	–	–	–	–	–	97.08	***97.86***	3
Pcb146	146	–	–	96.71	300	–	–	–	–	–	–	–	–	–	–	–	–	96.71	***98.74***	28
Rp200	200	–	–	96.30	400	–	–	–	–	–	–	–	–	–	–	–	–	96.30	***99.02***	20
Pcb500	500	–	–	96.28	1000	–	–	–	–	–	–	–	–	–	–	–	–	96.28	***99.61***	490
平均		–	1341.7	–	400	–	22.2	–	1800	–	14.2	–	873	–	1748.5	–	42.8	97.65	***98.50***	200

图 5 给出了 DRA 计算 Ami49,N30 和 Rp100 得到的布局图案,其填充率均超过了当前最好记录,它们对应的矩形框分别为 2912×12348,325×626 和 150×1397.



(a) (b)

(c)

Fig.5 Packing layouts for three instances

图 5 若干实例的布局图案

5 结束语

基于将复杂问题逐步拆分、分段解决的思路,本文深入研究了 RPAMP 与其归约形式 RPDP 之间的关系,提出了求解 RPAMP 的动态归约算法.在求解 RPDP 的最大适配度算法的基础上,通过完善和细化动作空间、角区和占角动作的定义,设计紧凑动作和定义紧凑布局,以及量化当前动作对局部空间的损害程度,设计了求解 RPDP 的最小损害度算法.然后,基于动态归约的方法进一步设计了求解 RPAMP 的高效求解算法.利用国际上公开通行的 15 个算例,与当前国内外文献中的代表性求解算法进行了比较.实验结果表明,所提出的动态归约算法具有较高的求解精度.

在下一步的工作中,我们将在保证求解质量的前提下设法提高算法的计算速度.另外,拟在此基础上深入研究求解 floorplanning 的目标优化问题的高效算法.

References:

[1] Murata H, Fujiyoshi K, Nakatake S, Kajitani Y. VLSI module placement based on rectangle-packing by the sequence-pair. Computer-Aided Design of Integrated Circuits and Systems, 1996,15(12):1518−1524. [doi: 10.1109/43.552084]

[2] Kahng AB. Classical floorplanning harmful? In: Wiesel M, Hill D, eds. Proc. of the ISPD 2000. New York: ACM Press, 2000. 207−213. [doi: 10.1145/332357.332401]

[3] Dong SQ, Hong XL, Huang G, Gu J. A non-slicing floorplanning and placement algorithm using a new constraint graph based model. Ruan Jian Xue Bao/Journal of Software, 2001,12(11):1586−1594 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/12/1586.htm

[4] Pisinger D. Denser packings obtained in $O(n\log\log n)$ time. Journal on Computing, 2007,19(3):395−405. [doi: 10.1287/ijoc.1060.0192]

[5] Li YM, Li Y, Zhou MT. Area optimization in floorplanning using AP-TCG. Journal of Convergence Information Technology, 2010,5(10):216−222.

[6] Imahori S, Yagiura M, Ibaraki T. Improved local search algorithms for the rectangle problem with general spatial costs. European Journal of Operational Research, 2005,167(1):48−67. [doi: 10.1016/j.ejor.2004.02.020]

[7] Chan HH, Markov IL. Practical slicing and non-slicing block-packing without simulated annealing. In: Garrett D, Lach J, Zukowski CA, eds. Proc. of the 14th ACM Great Lakes Symp. on VLSI. New York: ACM Press, 2004. 282−287. [doi: 10.1145/988952.989020]

[8] 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]

[9] Korf RE, Moffitt MD, Pollack ME. Optimal rectangle packing. Annuals of Operations Research, 2010,179(1):261−295. [doi: 10.1007/s10479-008-0463-6]

[10] Kimura Y, Ida K. Improved genetic algorithm for VLSI floorplan design with non-slicing structure. Computers and Industrial Engineering, 2006,50(4):528−540. [doi: 10.1016/j.cie.2005.01.023]

[11] Fernando P, Katkoori S. An elitist non-dominated sorting based genetic algorithm for simultaneous area and wirelength minimization in VLSI floorplanning. In: Conner LO, ed. Proc. of the 21st Int'l Conf. on VLSI Design. Los Alamitos: IEEE Computer Society CPS, 2008. 337−342. [doi: 10.1109/VLSI.2008.97]

[12] Chen JL, Zhu WX. A hybrid genetic algorithm for VLSI floorplanning. In: Chen W, Li SH, eds. Proc. of the 2010 IEEE Int'l Conf. on Intelligent Systems (ICIS). Xiamen: IEEE Press, 2010. 128−132. [doi: 10.1109/ICICISYS.2010.5658785]

[13] Yang L, Ma YC, Hong XL, Dong SQ, Zhou Q. An incremental algorithm for non-slicing floorplan based on corner block list representation. Chinese Journal of Smeiconductors, 2005,26(12):2335−2343 (in Chinese with English abstract).

[14] Chen JL, Zhu WX, Ali MM. A hybrid simulated annealing algorithm for nonslicing VLSI floorplanning. IEEE Trans. on Systems, Man, and Cybernetics—Part C: Application and review, 2011,41(4):544−553. [doi: 10.1109/TSMCC.2010.2066560]

[15] Chen M, Huang WQ. A two-level search algorithm for 2D rectangular packing problem. Computer & Industrial Engineering, 2007, 53(1):123−136. [doi: 10.1016/j.cie.2007.04.007]

[16] Wu YL, Chan CH. On improved least flexibility first heuristics superior for packing and stock cutting problems. In: Lupanov OB, Kasim-Zade OM, Chaskin AV, Steinhofel K, eds. Proc. of the 3rd Int'l Symp. on Stochastic algorithms: Foundations and Applications. Heidelberg: Springer-Verlag, 2005. 70−81. [doi: 10.1007/11571155_8]

[17] Hopper E. Two-Dimensional packing utilizing evolutionary algorithms and other meta-heuristic methods [Ph.D. Thesis]. Cardiff: Cardiff University, 2000.

[18] He K, Huang WQ, Jin Y. An efficient deterministic heuristic for two-dimensional rectangular packing. Computers and Operations Research, 2012,39(7):1355−1363. [doi: 10.1016/j.cor.2011.08.005]

[19] Bortfeldt A, Gehring H. A hybrid genetic algorithm for the container loading problem. Europe Journal of Operations Research, 2001,131(1):143−161. [doi: 10.1016/S0377-2217(00)00055-2]

[20] Parreno F, Alvarez-Valdes R, Oliveira JF. Neighborhood structures for the container loading problem: A VNS implementation. Journal of Heuristics, 2010,16(1):1−22. [doi: 10.1007/s10732-008-9081-3]

[21] He K, Huang WQ. An efficient placement heuristic for three-dimensional rectangular packing. Computers and Operations Research, 2011,38(1):227−233. [doi: 10.1016/j.cor.2010.04.015]

[22] 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

[23] The MCNC benchmark problems for VLSI flooplanning. http://www.mcnc.org

[24] GSRC floorplan benchmark suite. http://www.cse.ucsc.edu/research/surf/GSRC/GSRCbench.html

附中文参考文献:

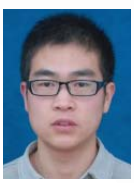
[3] 董社勤,洪先龙,黄钢,顾均.基于新约束图模型的布局规划和布局算法.软件学报,2001,12(11):1586−1594. http://www.jos.org.cn/1000-9825/12/1586.htm

[13] 杨柳,马昱春,洪先龙,董社勤,周强.基于角模块布图表示的增量式布图规划算法.半导体学报,2005,26(12):2335−2343.

[22] 何琨,黄文奇,金燕.基于动作空间求解二维矩形 Packing 问题的高效算法.软件学报,2012,23(5):1037−1044. http://www.jos.org.cn/1000-9825/3986.htm

何琨(1972－),女,北京人,博士,副教授,CCF 高级会员,主要研究领域为 NP 难度问题现实求解,组合优化.
E-mail: brooklet60@gmail.com

李初民(1962－),男,博士,教授,博士生导师,主要研究领域为 NP 难度问题现实求解,组合优化.
E-mail: chu-min.li@u-picardie.fr



姬朋立(1988－),男,博士生,主要研究领域为 NP 难度问题现实求解,组合优化.
E-mail: jipengli8@gmail.com