ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.5, May 2010, pp.886−898
doi: 10.3724/SP.J.1001.2010.03524

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

加权 3-Set Packing 的改进算法*

冯启龙, 王建新+, 陈建二

(中南大学 信息科学与工程学院,湖南 长沙 410083)

Improved Algorithms for Weighted 3-Set Packing

FENG Qi-Long, WANG Jian-Xin+, CHEN Jian-Er

(School of Information Science and Engineering, Central South University, Changsha 410083, China)

+ Corresponding author: E-mail: jxwang@mail.csu.edu.cn

Feng QL, Wang JX, Chen JE. Improved algorithms for weighted 3-set packing. *Journal of Software*, 2010, 21(5):886−898. http://www.jos.org.cn/1000-9825/3524.htm

Abstract: Packing problems form an important class of NP-hard problems. In order to solve the weighted 3-set packing problem, this paper converts the problem to the weighted 3-set packing augmentation problem, and mainly works on how to construct a maximum weighted (k+1)-packing based on a maximum weighted k-packing. This paper gives a theoretical study on the structure of the problem and presents a deterministic algorithm of time $O^*(10.6^{3k})$ with color-coding, which significantly improves the previous best result $O^*(12.8^{3k})$. After further analyzing the structure of the problem and based on the set dividing method, the above result can be further reduced to $O^*(7.56^{3k})$.
Key words: weighted 3-set packing; weighted 3-set packing augmentation; color-coding

摘 要: Packing 问题构成了一类重要的 NP 难问题.对于加权 3-Set Packing 问题,把问题转化成加权 3-Set Packing Augmentation 问题进行求解,即主要讨论如何从一个已知的最大加权 k-packing 求得一个权值最大的(k+1)-packing.通过对问题结构的分析,结合 Color-Coding 技术,首先给出了一种时间复杂度为 $O^*(10.6^{3k})$的参数算法,极大地改进了目前文献中的最好结果 $O^*(12.8^{3k})$.通过对(k+1)-packing 结构的进一步分析,利用集合划分技术将上述结果降到 $O^*(7.56^{3k})$.
关键词: 加权 3-set packing;加权 3-set packing augmentation;color-coding
中图法分类号: TP301 **文献标识码**: A

在复杂性理论中,Packing 问题是一类重要的 NP 难问题,在调度和代码优化等领域有着重要的应用.其中,加权 3-Set Packing 问题是一个重要的NP完全问题.本文主要讨论该问题的参数算法设计.下面先给出几个主要问题的定义[1−3].

* Supported by the National Natural Science Foundation of China under Grant Nos.60433020, 60773111 (国家自然科学基金); the National Basic Research Program of China under Grant No.2008CB17107 (国家重点基础研究发展计划(973)); the Program for New Century Excellent Talents in University of China under Grant No.NCET-05-0683 (新世纪优秀人才支持计划); the Program for Cheung Kong Scholars and Innovative Research Team in University of China under Grant No.IRT0661 (长江学者和创新团队发展计划)

Received 2008-04-03; Accepted 2008-10-27

首先假设本文用到的元素都来自集合 U.

问题 1. Set Packing:给定一个含有 n 个集合的集合簇 S,寻找 S 的一个最大子集 S',使 S'中的任何两个集合之间没有公共元素.

问题 2. (参数化)加权 3-Set Packing:给定一个序对(S,k),其中,S 含有 n 个带权值的集合,每个集合包含 3 个元素,k 是一个整数.目标是构造一个权值和最大且含有 k 个集合的 packing,如果找不到,则返回不存在.

问题 3. 加权 3-Set Packing Augmentation:给定一个序对(S,P_k),其中,S 含有 n 个带权值的集合,每个集合含有 3 个元素,P_k 是 S 中一个权值和最大的 k-packing.目标是构造一个权值最大且含有 k+1 个集合的 packing,如果找不到,则返回不存在.

近年来,人们对加权 m-Set Packing 问题(每个集合包含 m 个元素)进行了大量的研究.

从近似算法角度,文献[4,5]都使用局部查找技术对加权 m-Set Packing 问题进行分析,得到了相同的近似率 $m-1+\varepsilon$.基于局部改进技术,文献[6]提出了一种近似率为$(m+1)/2$ 的近似算法.

参数计算理论被有效利用来求解加权 m-Set Packing 问题.文献[7]给出了一种时间复杂度为 $O^*(12.8^{mk})$的参数算法(本文中,我们用 $O^*(f(k)$表示界 $O(f(k)n^{O(1)})$).

对于 m-Set Packing 问题,当 m=2 时,2-Set Packing 与无向图的最大匹配问题等价.当 m=3 时,3-Set Packing 问题是一个重要的 NP 完全问题,人们针对该问题进行了大量的研究.文献[3]证明了 3-Set Packing 问题是固定参数可解的,并给出了一种时间复杂度为 $O^*((3k)!(3k)^{9k+1})$的参数算法.文献[8]把 3-Set Packing 的时间复杂度降到 $O^*((5.7k)^k)$.文献[9]给出了一种时间复杂度为 $O^*(10.88^{3k})$的随机算法,其确定性算法的时间复杂度为 $O^*(2^{O(k)})$,可以推得,该确定性算法的时间复杂度至少为 $O^*(32000^{3k})$.文献[10]给出了一种时间复杂度至少为 $O^*(12.67^{3k}T(k))$的算法,其中,$T(k)$是动态规划算法的运行时间(基于现在的技术,$T(k)$至少为 $O^*(10.4^{3k})$).文献[11]基于随机分治法给出了一种时间复杂度为 $O^*(16^{3k})$的确定化参数算法.文献[1]利用局部贪婪和着色技术给出了一种时间复杂度为 $O^*(4.61^{3k})$的确定性算法.文献[12]利用局部贪婪和随机分治法给出了一种时间复杂度为 $O^*(3.52^{3k})$的确定性算法,这个结果是当今 3-Set Packing 问题确定性算法的最佳结果.

对于加权 3-Set Packing 问题,除了利用文献[7]给出的结果得到时间复杂度为 $O^*(12.8^{3k})$算法以外,目前还没有系统的方法对该问题进行求解.本文从加权 3-Set Packing 问题的结构特性出发进行有效参数算法的设计,主要讨论怎样从一个已知的最大加权 k-packing 求得一个权值最大的$(k+1)$-packing,以此来求解加权 3-Set Packing 问题.通过对问题结构的进一步分析,使用 Color-Coding 技术得到了一个时间复杂度为 $O^*(10.6^{3k})$的参数算法.通过对$(k+1)$-packing 的结构进一步进行分析,并使用集合划分技术给出了时间复杂度为 $O^*(7.56^{3k})$的参数算法.该结果与文献中的最好结果 $O^*(12.8^{3k})$相比有了极大的改进.

1 相关术语和引理

在给出求解加权 3-Set Packing 的参数算法之前,先给出一些相关术语和引理.

引理 1. 对于任意的常数 c>1,加权 3-Set Packing Augmentation 问题可以在 $O^*(c^k)$时间内被求解,当且仅当加权 3-Set Packing 问题能够在 $O^*(c^k)$时间内被求解.

证明:充分性证明.假设加权 3-Set Packing 问题可以在 $O^*(c^k)$时间内被算法 A 求解.对于加权 3-Set Packing Augmentation 问题的任一个实例(S,P_k),如果 S 中存在$(k+1)$-packing,调用算法 A 求解权值最大的$(k+1)$-packing,时间复杂度为 $O^*(c^{k+1})=O^*(c^k)$.因此,加权 3-Set Packing Augmentation 问题可以在 $O^*(c^k)$时间内被求解.

必要性证明.假设加权 3-Set Packing Augmentation 问题可以在 $O^*(c^k)$时间内被算法 B 求解.对于加权 3-Set Packing 问题的一个实例,首先在 S 中选择权值最大的一个集合作为权值最大的 1-packing P_1.对于每个实例(S,P_i),调用算法 B 在 S 中寻找权值最大的$(i+1)$-packing.如果对于实例$(S,P_i)i\le k-1$,算法 B 找不到一个$(i+1)$-packing,那么 S 中就不存在最大加权 k-packing;相反地,算法 B 将为实例(S,P_{k-1})找到相应的最大加权 k-packing.整个过程的时间复杂度为 $O^*(c^1)+O^*(c^2)+\ldots+O^*(c^{k-1})=O^*(c^k)$.因此,加权 3-Set Packing 问题可以在 $O^*(c^k)$时间内被求解. □

由引理 1 可知,加权 3-Set Packing 问题与加权 3-Set Packing Augmentation 问题的复杂度相同.因此,降低加权 3-Set Packing Augmentation 问题的复杂度是我们的目标.

本文将有效利用 Color-Coding 技术来设计加权 3-Set Packing 问题的参数算法.下面先给出与 Color-Coding 相关的定义和引理.

文献[13]首次提出了 Color-Coding 技术.假设 B 是包含许多元素的集合,B 的一种着色是一个映射,这个映射把 B 映射到自然数集$\{1,2,\ldots\}$,用 h 种颜色对 B 进行着色就是把 B 映射到$\{1,2,\ldots,h\}$.对于一种着色 f 和 B 的一个子集 B_1,如果 B_1 中的任意两个元素都没有被着上相同的颜色,那么称 B_1 被 f 正确着色.假设存在一个着色方案 C,使得对于 B 中的任意一个含有 h 个元素的子集,C 中总有一个 h 着色把该子集正确着色,就说 C 是一个用了 h 种颜色的着色方案.

下面引入文献[1]中关于 Color-Coding 的一个重要引理.

引理 2. 对于任意有限集 B 和整数 h,存在一个 h 种颜色的着色方案 C.对于集合 B,C 含有 $O^*(6.1^h)$个对 B 的 h 着色,而且 C 可在 $O^*(6.1^h)$时间内被构造.

2 基于 Color-Coding 的加权 3-Set Packing 参数算法

在本节中,我们将充分利用 Color-Coding 和动态规划技术来设计加权 3-Set Packing Augmentation 问题的参数算法.

对于加权 3-Set Packing Augmentation 问题的一个实例(S,P_k),其中,P_k 是 S 中权值最大的 k-packing.首先把 S 分成 S_1,S_2 两部分,具体过程如下:

对于 S 中的每一个集合ρ,如果ρ与 P_k 中的每个集合都没有公共元素,则将其放入 S_1 中,否则将其放入 S_2 中.

本文求解加权 3-Set Packing Augmentation 问题的算法思想是:把 S 分成 S_1 和 S_2 两部分.在 S_1 中寻找一个权值最大的集合,再加上 P_k 中的 k 个集合,即可构造一个$(k+1)$-packing,记为 PW_1,该过程可在多项式时间内完成.运用 Color-Coding 和动态规划技术在 S_2 中找到权值最大的$(k+1)$-packing,记为 PW_2.返回 PW_1 和 PW_2 中权值较大的一个.

对于实例(S,P_k),用 U_{P_k} 表示 P_k 包含 U 中的元素,用 $U_{P_{k+1}}$ 表示 P_{k+1} 包含 U 中的元素,用集合 $U_{S_2-P_k}$ 表示 S_2-P_k 中的元素.

首先,给出以下引理:

引理 3. 对于加权 3-Set Packing Augmentation 问题的一个实例(S,P_k),其中,P_k 是 S 中权值最大的 k-packing,如果 S 中存在$(k+1)$-packing,则权值最大的$(k+1)$-packing 是 PW_1 和 PW_2 中的权值较大者.

证明:S 被分成了 S_1 和 S_2 两部分.由 S_1 中的集合和 P_k 构造的所有$(k+1)$-packing 中,PW_1 是权值最大的一个$(k+1)$-packing,PW_2 是 S_2 中的一个权值最大的$(k+1)$-packing.假设 PW_2 的权值小于等于 PW_1 的权值,则需要证明的是,PW_1 是 S 中权值最大的$(k+1)$-packing.

假设 S 存在一个由 S_1 中的集合和 S_2 中的集合共同构造的$(k+1)$-packing P,且假设对于 P 中所有属于 S_1 的集合,ρ是权值最大的一个.显然,$P-\rho$是一个 k-packing.由于 P_k 是 S 中一个权值最大的 k-packing,故 $P-\rho$的权值不会大于 P_k 的权值.因此,由 P_k 和ρ构造的$(k+1)$-packing 的权值大于等于 P.由已知可得,在由 S_1 中的集合和 P_k 构造的所有$(k+1)$-packing 中,PW_1 是权值最大的一个$(k+1)$-packing.因此,$P_k+\rho$的权值不会大于 PW_1,故可得 PW_1 是 S 中权值最大的$(k+1)$-packing.对于当 PW_2 的权值大于 PW_1 的情况,证明过程类似. □

剩下的问题就是怎样在 S_2 中寻找一个权值最大的$(k+1)$-packing.首先给出以下引理.

引理 4. 对于加权 3-Set Packing Augmentation 问题的实例(S_2,P_k),其中,P_k 是 S_2 中的权值最大的 k-packing.假设 S_2 存在 P_{k+1},则 P_{k+1} 中的每个集合至少包含 U_{P_k} 中的 1 个元素,即 $U_{P_{k+1}}$ 至少包含 $k+1$ 个 U_{P_k} 中的元素.

证明:在把 S 分成 S_1,S_2 两部分的过程中,如果 S 中某集合ρ与 P_k 之间有公共元素,则将其放入 S_2 中,故 S_2 中的每个集合至少包含 U_{P_k} 中的 1 个元素.因此,如果 S_2 存在 P_{k+1},则该 P_{k+1} 中的每个集合至少包含 U_{P_k} 中的 1 个元素,即 $U_{P_{k+1}}$ 至少包含 $k+1$ 个 U_{P_k} 中的元素. □

由引理 4 可知,每一个 S_2 中的$(k+1)$-packing 至少包含了 $k+1$ 个 U_{P_k} 中的元素.因为 $U_{P_{k+1}}$ 总共有 $3k+3$ 个元素,所以那些在 $U_{P_{k+1}}$ 中出现但不在 U_{P_k} 中出现的元素的个数最多为 $2k+2$.用 Color-Coding 技术在 $U_{S_2-P_k}$ 中寻找这最多 $2k+2$ 个元素,同时引入另外 $3k$ 种颜色对 U_{P_k} 中的 $3k$ 个元素着色.这样,对于着色方案中的每一个着色,用到的总的颜色数为 $5k+2$.基于上面的着色,我们将有效运用动态规划技术在 S_2 中寻找一个权值最大的$(k+1)$-packing.

首先给出算法中要用到的几个符号表示.对于一个 packing P 和着色 f,如果 P 中任意两个元素都没有被着上相同的颜色,则称 P 被 f 正确着色,用 $cl(P)$表示 packing P 用到的颜色集合.

用动态规划在 S_2 中求最大加权$(k+1)$-packing 的算法思路如下:着色方案中的每一个着色 f 用了 $5k+2$ 种颜色对 S_2 中的元素进行了着色,用集合 Q_1 保存动态规划过程中可能产生的颜色集.对于 Q_1 中的 packing C 和 S_2 中的集合ρ_i,如果 $cl(C)$和 $cl(\rho_i)$没有相同颜色,则产生一个新的 packing $C'=C\cup\rho_i$.如果 C'中含有集合的个数不超过 $k+1$,且 Q_1 中没有一个 packing 用到的颜色与 $cl(C')$相同,则把 C'加入到 Q_1 中.如果 Q_1 中有一个 packing C_1 用到的颜色与 $cl(C')$相同,但 C'的权值大于 C_1 的权值,则用 C'替换 C_1.具体过程见算法 1.

算法 1. 3SPDP(S_2,k,f).

输入:S_2,k,f.

输出:如果 S_2 中存在权值最大的$(k+1)$-packing,则返回该 packing;否则,返回空集.

1. 如果 S_2 中某个集合中的两个元素被着上了同一种颜色,则把该集合删掉;
2. 设 S_2 中剩余的集合为$\rho_1,\rho_2,\ldots,\rho_m$;
3. $Q_1=\{\varnothing\}$;
4. for i=1 to m do
5. for Q_1 中的每个 packing C do

5.1. if $cl(C)$和 $cl(\rho_i)$没有相同颜色 then $C'=C\cup\rho_i$;

5.2. if C'中含有的集合个数不超过 $k+1$,且 Q_1 中没有一个 packing 用到的颜色集合与 $cl(C')$相同 then 把 C'加入到 Q_1 中;

5.3. if Q_1 中有一个 packing C_1 用到的颜色集合与 $cl(C')$相同,但 C_1 的权值小于 C'的权值 then 用 C'替换 C_1;

6. 如果 Q_1 中存在一个最大加权$(k+1)$-packing,则返回该 packing;否则,返回空集.

定理 1. 如果 S_2 中存在一个权值最大的$(k+1)$-packing,则算法 3SPDP(S_2,k,f)必定在 $O^*(2^{5k})$时间内返回该 $k+1$-packing.

证明:从算法 1 中的第 5.1 步~5.3 步中可以看出,加入到 Q_1 中的 C'是一个正确着色的 packing.通过对第 4 步 for 循环中的 i 进行归纳来证明以下结论:如果 S_2 中存在一个最大权值的$(k+1)$-packing P_{k+1},则算法执行完毕后,Q_1 必包含一个$(k+1)$-packing P'_{k+1},使得 $cl(P_{k+1})=cl(P'_{k+1})$,且权值相同.

对于任意的 $i(1\le i\le m)$,假设 X_i 表示 S_2 中前 i 个集合,$X_i\subseteq S_2$.因此,仅需证明以下命题成立:

命题 1. 如果 X_i 中存在一个最大加权 j-packing P_j,则在第 i 次执行完算法 1 第 4 步的 for 循环后,Q_1 中必定包含了一个 j-packing P'_j,使得 $cl(P_j)=cl(P'_j)$,且 P_j 与 P'_j 的权值相同.

算法 1 第 3 步中 $Q_1=\{\varnothing\}$,如果 X_i 中有一个 0-packing,则显然命题成立.

当 $i\ge1$ 时,假设 X_i 存在一个最大加权 j-packing $P_j=\{\rho_{l_1},\rho_{l_2},\ldots,\rho_{l_j}\}$,其中,$1\le l_1<l_2<\ldots<l_j\le i$,则在 X_{l_j-1} 中必包含了一个$(j-1)$-packing $P_{j-1}=\{\rho'_{l_1},\rho'_{l_2},\ldots,\rho'_{l_j-1}\}$,使得 $cl(P_{j-1})=cl(P_j-\rho_{l_j})$,且权值相同.由归纳假设可知,执行完第$(l_j-1)$次算法 1 第 4 步的 for 循环后,$Q_1$ 包含了一个最大加权$(j-1)$-packing P'_{j-1},使得 $cl(P'_{j-1})=cl(P_{j-1})$,且 P'_{j-1} 与 P_{j-1} 的权值相同.由前面假设知,X_i 中存在一个权值最大的 j-packing P_j.当 ρ_{l_j} 在第 5.1 步被考虑时,$cl(P'_{j-1})$和 $cl(\rho_{l_j})$之间没有相同的颜色.因此,如果 Q_1 中不存在使用了 $cl(P'_{j-1}\cup\{\rho_{l_j}\})$的 j-packing,则 j-packing $P'_{j-1}\cup\{\rho_{l_j}\}$ 将被加入到 Q_1 中.如果 Q_1 中存在一个 packing C_1 使用了颜色 $cl(P'_{j-1}\cup\{\rho_{l_j}\})$但 C_1 的权值小于 $P'_{j-1}\cup\{\rho_{l_j}\}$ 的权值,则用

$P'_{j-1} \cup \{\rho_{l_j}\}$ 取代 C_1.因为 Q_1 中的所有 packing 在整个过程中没有被移出 Q_1 且 $l_j \le i$,因此在执行完 i 次第 4 步的 for 循环后,Q_1 必包含一个与 P_j 使用了相同颜色集合且权值相同的 j-packing.当 $i=m$ 时,如果 S_2 中存在一个最大加权$(k+1)$-packing P_{k+1},则 Q_1 必包含一个$(k+1)$-packing P'_{k+1},使得 $cl(P_{k+1})=cl(P'_{k+1})$,且权值相同.

最后,对算法的复杂度进行分析.对于任意的 $j(0 \le j \le 5k+2)$和任意的 $3j$ 种颜色,Q_1 最多只包含 1 个用了这 $3j$ 种颜色的 packing.因此,Q_1 中最多包含了 $\sum_{j=0}^{k+1}\binom{5k+2}{3j}$ 个 packing,故算法 3SPDP 的复杂度为

$$O\left(mk\sum_{j=0}^{k+1}\binom{5k+2}{3j}\right)=O^*\left(\sum_{j=0}^{k+1}\binom{5k+2}{3j}\right)=O^*(2^{5k}).$$ □

基于引理 3 和定理 1,算法 2 给出了求解加权 3-Set Packing Augmentation 问题的算法.

算法 2. 3SPG(S,P_k).

输入:S,P_k.

输出:如果 S 中存在权值最大的$(k+1)$-packing,则返回该 packing;否则,返回空集.

1. $PW_1=\{\varnothing\}$;
2. 在 S 中找到所有与 P_k 没有公共元素的集合,记为 S_1,用 S_2 表示 $S-S_1$ 中的集合;
3. if S_1 不为空 then
   用 S_1 中权值最大的集合和 P_k 中的 k 个集合构造一个$(k+1)$-packing,记为 PW_1;
4. if S_2 不为空 then

4.1. 用 $2k+2$ 种颜色对 $U_{S_2-P_k}$ 中的元素进行着色,得到规模为 $O^*(6.1)^{2k}$ 的着色方案 F;

4.2. 用另外 $3k$ 种颜色对 U_{P_k} 中的元素进行着色,这样,着色方案 F 中的每个着色 f 用了 $5k+2$ 种颜色;

4.3. for F 中的每一个着色 f do

   PW_2=3SPDP(S_2,k,f);

   if PW_2 的权值大于 PW_1 的权值 then $PW_1=PW_2$;

5. 返回 PW_1.

定理 2. 对于加权 3-Set Packing Augmentation 问题的一个实例(S,P_k),如果存在 P_{k+1},则算法 3SPG(S,P_k)必定在 $O^*(10.6^{3k})$时间内返回一个权值最大的$(k+1)$-packing.

证明:对于加权 3-Set Packing Augmentation 问题的一个实例(S,P_k),如果存在 P_{k+1},则把 S 分成 S_1 和 S_2 两部分处理.用 S_1 中权值最大的集合和 P_k 中的 k 个集合构造一个$(k+1)$-packing,记为 PW_1.

基于引理 4,每一个 S_2 中的$(k+1)$-packing 包含了至少 $k+1$ 个 U_{P_k} 中的元素.因此,$U_{P_{k+1}}$ 最多包含 $2k+2$ 个不是 U_{P_k} 中的元素.用 Color-Coding 技术在 $U_{S_2-P_k}$ 中查找那最多 $2k+2$ 个元素,根据引理 2,可以得到一个大小为 $O^*(6.1^{2k})$的着色方案 F.再引入另外 $3k$ 种颜色对 P_k 中的元素进行着色,则每一个着色总共用了 $5k+2$ 种颜色.如果 S_2 中存在权值最大$(k+1)$-packing,则该$(k+1)$-packing 将被 F 中的某个着色 f 正确着色.由定理 1 知,算法 3SPDP(S_2,k,f)必定返回该$(k+1)$-packing.最后,对 PW_1 和算法 3SPDP 返回的 packing 进行比较,返回权值较大者.

最后,对算法的复杂度进行分析.对于 PW_1,可在多项式时间内处理完毕.对于 F 中每一个着色 f,都需要调用算法 3SPDP.由引理 2 可知,F 中的着色方案数为 $O^*(6.1^{2k})$.由定理 1 可知,算法 3SPDP 的时间复杂度为 $O^*(2^{5k})$.因此,算法 3SPG 总的时间复杂度为 $O^*(6.1^{2k})\times O^*(2^{5k})=O^*(10.6^{3k})$. □

基于引理 1,可以得到如下推论:

推论 1. 加权 3-Set Packing 问题可以在 $O^*(10.6^{3k})$时间内被求解.

3 基于集合划分的加权 3-Set Packing 参数算法

在本节,我们将使用集合划分和动态规划技术进一步降低加权 3-Set Packing Augmentation 问题的复杂度.由于 PW_1 可在多项式时间内获得,本节主要讨论如何在 S_2 中找到权值最大的$(k+1)$-packing.

基于引理 4,可得到如下引理:

引理 5. (S_2,P_k)是 3-Set Packing Augmentation 问题的一个实例,其中,P_k是 S_2 中一个权值最大的 k-packing.如果存在 P_{k+1},则每一个 P_{k+1} 最多有 r 个只包含 U_{P_k} 中 1 个元素的集合($0\le r\le k+1$),有 s 个只包含 U_{P_k} 中两个元素的集合($0\le s\le k+1$),有 t 个包含 U_{P_k} 中 3 个元素的集合($0\le t\le k-1$),其中,$r+s+t=k+1$.

证明:由引理 4 可知,如果存在 P_{k+1},则 P_{k+1} 中的每个集合至少包含 U_{P_k} 中的 1 个元素.这样,对于 P_{k+1} 中的每个集合来说,它包含 U_{P_k} 元素的个数只可能是 1 个、2 个或 3 个.因此,该 P_{k+1} 最多由引理 5 中给出的 3 种类型的集合组成. □

如果 S_2 中存在 P_{k+1},则用 $C_i(1\le i\le 3)$表示 S_2 中与 U_{P_k} 有 i 个公共元素的所有集合.C_2 的每个集合中有 2 个元素来自 U_{P_k},C_3 的每个集合中有 3 个元素来自 U_{P_k},用 $U_{C_2-P_k}$ 表示 C_2 中不在 U_{P_k} 中的元素集,则 $U_{C_2-P_k}\subseteq U_{S_2-P_k}$.

由引理 5 可知,如果 P_{k+1} 存在,则 P_{k+1} 中只包含 U_{P_k} 中一个元素的集合个数为 r 个,显然,这些集合被包含在 C_1 中.从 P_k 的 $3k$ 个元素中枚举出 r 个元素,这样的枚举方式共有$\binom{3k}{r}$种,并用 H 表示 C_1 中这 r 个元素所在的集合.

用集合 $U_{P_{k+1}-P_k}$ 表示是 $U_{P_{k+1}}$ 中的元素而不是 U_{P_k} 中的元素构成的集合,用 y 表示集合 $U_{P_{k+1}-P_k}$ 的大小,即 $y=|U_{P_{k+1}-P_k}|$.由引理 4 可知,U_{P_k} 中至少有 $k+1$ 个元素在 P_{k+1} 中,所以集合 $U_{P_{k+1}-P_k}$ 最多包含 $2k+2$ 个 $U_{S_2-P_k}$ 中的元素,即 $y\le 2k+2$.显然,$U_{P_{k+1}-P_k}$ 中的元素要么存在于 H 中,要么存在于 $C_2\cup C_3$ 中.假设 $U_{P_{k+1}-P_k}$ 中属于 H 中的元素组成的子集为 $U^H_{P_{k+1}-P_k}$.

下面先给出本节将要用到的相关术语和引理.

首先要指出的是,对某一集合进行划分,就是把该集合分成两部分.

集合 $U_{P_{k+1}-P_k}$ 的大小为 y,则有 2^y 种方式把 $U_{P_{k+1}-P_k}$ 分成两部分,其中必有一种划分正好使得子集 $U^H_{P_{k+1}-P_k}$ 被划分到 H 中,而 $U_{P_{k+1}-P_k}-U^H_{P_{k+1}-P_k}$ 被划分到 $C_2\cup C_3$ 中.但存在的问题是,我们不知道集合 $U_{P_{k+1}-P_k}$ 具体是由哪些元素组成的,只知道所有元素都来自集合 $U_{S_2-P_k}$.利用集合划分求解 S_2 中最大加权$(k+1)$-packing 的基本思想是:把集合 $U_{S_2-P_k}$ 划分成两部分,使得一部分包含 $U^H_{P_{k+1}-P_k}$,另一部分包含 $U_{P_{k+1}-P_k}-U^H_{P_{k+1}-P_k}$.即必有一种划分正好使得子集 $U^H_{P_{k+1}-P_k}$ 被划分到 H 中,而 $U_{P_{k+1}-P_k}-U^H_{P_{k+1}-P_k}$ 被划分到 $C_2\cup C_3$ 中.

首先给出划分函数的概念[14].

定义 1(划分函数). 集合 Z_n 的一个划分函数就是以 Z_n 为定义域的一个布尔函数(值域为$\{0,1\}$).

Z_n 的一个划分函数可以理解成对 Z_n 的一个划分(X_1,X_2)(把 Z_n 中所有 $f(x)=0$ 的 x 放入 X_1 中,把剩余的放入 X_2 中).Z_n 若干划分函数组成的集合称为 Z_n 的划分函数族.

假设 W 是 Z_n 中一个 k 大小的子集,设(W_1,W_2)为 W 的一个划分.对于 Z_n 的一个划分函数 f 来说,如果以下两种情况中有一种成立,则称划分函数 f 实现了划分(W_1,W_2):

(1) 对于所有的 $x\in W_1,f(x)=0$ 且对于所有的 $y\in W_2,f(y)=1$;

(2) 对于所有的 $x\in W_1,f(x)=1$ 且对于所有的 $y\in W_2,f(y)=0$.

下面给出(n,k)-universal set 的概念[15].

定义 2. (n,k)-universal set:对于 Z_n 的任意一个 k 大小的子集 W 和对 W 的任意一个划分(W_1,W_2),若 Z_n 的划分函数族 Z 中总存在一个划分函数实现了划分(W_1,W_2),则称 Z_n 的划分函数族 Z 为一个(n,k)-universal set.

需要指出的是:Z_n 的划分函数族 Z 中划分函数的个数为(n,k)-universal set Z 的大小.

假设 g 是定义在集合 Z_n 上的函数,对于 Z_n 的子集 V,如果对于 V 中任意的两个元素 x,y 都有 $g(x)\ne g(y)$,则称 g 在 V 上是单射的.

引理 6[15]. 给定集合 Z_n 和整数 k,存在大小最多为 $2n$ 的映射函数族 $\Psi_{n,k}$ 把 Z_n 映射到 Z_{k^2},且使得对于 Z_n 的任意 k 大小的子集 S,$\Psi_{n,k}$ 中必存在一个函数 g 在 S 上是单射的,且函数族 $\Psi_{n,k}$ 可在 $O(n)$时间内构造.

基于引理 6,首先构造基于 U 映射函数族 $\Psi_{n,2k+2}$,则 $\Psi_{n,2k+2}$ 中必存在一个映射函数 g,使得 g 在 $U_{P_{k+1}-P_k}$ 上是单射的.将映射函数族某个函数应用到实例(S_2,P_k)后,用 n'表示 S_2 包含的 U 中元素的数目,则 $n'\le(2k+2)^2$.基于文献[15]中给出的构造方法,可构造大小为 $O((2k+2)^2 2^{k+12\log^2 k+12\log k+6})=O(2^{k+12\log^2 k+12\log k+2\log(2k+2)+6})$ 的(n',k)-universal set,且可在 $O(2^{k+12\log^2 k+12\log k+2\log(2k+2)+6})$ 时间内完成构造.

利用集合划分求解加权 3-Set Packing Augmentation 问题的基本思想是:对于给定的一个加权 3-Set Packing Augmentation 问题的实例(S_2,P_k),如果存在最大加权 P_{k+1},由引理 4 可知,U_{P_k} 中至少有 k+1 个元素被包含在 $U_{P_{k+1}}$ 中,此时,$U_{P_{k+1}}$ 中最多含有 $2k$+2 个 $U_{S_2-P_k}$ 中的元素.当从 P_k 中的 $3k$ 个元素中枚举出 r 个元素时,把 P_{k+1} 分成两部分来处理:一部分包含在 H 中,另一部分包含在 $C_2\cup C_3$ 中.要想求得 P_{k+1},则必须把 $U_{S_2-P_k}$ 中属于 H 的那 $2r$ 个元素划分到 H 中,把其他的划分到 $C_2\cup C_3$ 中.构造$(|U_{S_2-P_k}|,2k+2)$-universal set,用动态规划在 $C_2\cup C_3$ 中寻找一个权值最大的$(k+1-r)$-packing.结合集合划分和分治法在 H 中寻找一个权值最大的 r-packing.

3.1 用动态规划在$C_2\cup C_3$中寻找一个权值最大的$(k+1-r)$-packing

假设 $k'=k+1-r(k'\le k+1)$,在 $C_2\cup C_3$ 中用动态规划寻找一个权值最大的 k'-packing 的思想是:用 Q_3 保存动态规划过程可能产生的 packing,对于 Q_3 中的每个 packing C 和 $C_2\cup C_3$ 中的每个集合ρ_i,如果 C 和ρ_i中属于 U_{P_k} 的元素没有共同元素,则生成一个新的 packing $C'=C\cup\{\rho_i\}$.如果 C'中集合个数不大于 k'并且 Q_3 中没有一个 packing 用到的 U_{P_k} 元素与 C'所用到的 U_{P_k} 元素完全相同,则把 C'加到 Q_3 中.如果 Q_3 中有一个 packing P 用到的 U_{P_k} 元素与 C'所用到的 U_{P_k} 元素完全相同,且 P 的权值小于 C',则用 C'替换 P.具体过程见算法 3.

算法 3. SP(C_2,C_3,k',U_{P_k}).

输入:C_2,C_3,k',U_{P_k}.

输出:如果 $C_2\cup C_3$ 中存在 k'-packing,则返回权值最大的 k'-packing;否则,返回空集.

1. 设 $U_{C_2-P_k}$ 中的元素为 $x_1,x_2,\ldots,x_m$;
2. $Q_3=\{\varnothing\}$; $Q_{new2}=\{\varnothing\}$;
3. for i=1 to m do
3.1. for Q_3 中的每个 packing C do
3.2. for C_2 中包含元素 x_i 的每个集合ρ do
3.3. if C 与ρ没有公共元素 then $C'=C\cup\{\rho\}$
3.4. if C'中集合的个数不大于 k',并且 Q_{new2} 中没有一个 packing 用到的 U_{P_k} 元素与 C'所用到的 U_{P_k} 元素完全相同
then 将 C'加到 Q_{new2} 中;
3.5. if Q_{new2} 中存在一个 packing P 用到的 U_{P_k} 元素与 C'所用到的 U_{P_k} 元素完全相同,但 P 的权值小于 C'
then 用 C'取代 P;
3.6. $Q_3=Q_{new2}$;
4. 设 C_3 中的元组为 $z_1,z_2,\ldots,z_l$;
5. for h=1 to l do
5.1. for Q_3 中的每个 packing C do
5.2. if C 中没有元素与 z_h 中的元素相同 then $C'=C\cup\{z_h\}$
5.3. if C'中集合的个数不大于 k',并且 Q_{new2} 中没有一个 packing 用到的 U_{P_k} 元素与 C'所用的 U_{P_k} 元素完全相同
then 将 C'加到 Q_{new2} 中;
5.4. if Q_{new2} 中存在一个 packing P^*用到的 U_{P_k} 元素与 C'所用到的 U_{P_k} 元素完全相同,但 P^*的权值不大于 C'

then 用 C'取代 P^*;

5.5. $Q_3=Q_{new2}$;

6. 如果 Q_3 存在 k'-packing,则返回权值最大 k'-packing;否则,返回空集.

定理 3. 如果 $C_2\cup C_3$ 中存在权值最大的 k'-packing,算法 SP(C_2,C_3,k',U_{P_k})必定返回一个权值最大的 k'-packing,其时间复杂度为 $O^*(2^{3k})$.

证明:如果 $C_2\cup C_3$ 中存在权值最大的 k'-packing,假设 C_2 中构成该 k'-packing 的集合个数为 k'',这 k''个 C_2 中的集合构成一个 k''-packing $P_{k''}$,$0\le k''\le k'$.需要证明以下两部分:

① 算法 3 执行完第 3 步的 for 循环后,Q_3 必包含一个与 $P_{k''}$用了相同 U_{P_k} 元素且权值相等的 k''-packing;

② 算法 3 执行完第 5 步的 for 循环后,Q_3 必包含一个权值最大的 k'-packing.

第①部分的证明过程如下:

对算法 3 中第 3 步的 for 循环中的 i 进行归纳以证明:如果 C_2 中存在 k''-packing $P_{k''}$,则 Q_3 中必包含一个 $P'_{k''}$,使得 $P'_{k''}$ 与 $P_{k''}$用了相同的 $2j$ 个 U_{P_k} 元素,且权值相同.

已知 $U_{C_2-P_k}$ 中有 m 个不同的元素 $x_1,x_2,\ldots,x_m$,对于任意的 $i(1\le i\le m)$,用 X_i 表示 $x_1,x_2,\ldots,x_i$ 这 i 个元素所在的集合,只需证明以下命题:如果 X_i 存在一个 j-packing P_j,则在第 i 次执行完算法 3 中第 3 步的 for 循环后,Q_3 中必定包含了一个 j-packing P'_j,使得 P_j 和 P'_j 用了相同的 $2j$ 个 U_{P_k} 元素,且 P_j 与 P'_j 的权值相同.

算法 3 的第 2 步中 $Q_3=\{\phi\}$,如果 X_i 中有一个 0-packing,则显然命题成立.

当 $i\ge 1$ 时,假设 X_i 存在了一个 j-packing $P_j=\{\varphi_{l_1},\varphi_{l_2},\ldots,\varphi_{l_j}\}$,其中,$1\le l_1<l_2<\ldots<l_j\le i$,则在 X_{l_j-1} 中必包含了一个$(j-1)$-packing $P_{j-1}=\{\varphi'_{l_1},\varphi'_{l_2},\ldots,\varphi'_{l_j-1}\}$,使得 P_{j-1} 和$\{P_j-\varphi_{l_j}\}$用了相同的 $2(j-1)$个 U_{P_k} 元素,且 P_{j-1} 和$\{P_j-\varphi_{l_j}\}$的权值相同.由归纳假设可知,经过第(l_j-1)次执行算法 3 中第 3 步的 for 循环后,Q_3 包含了一个$(j-1)$-packing P'_{j-1},使得 P_{j-1} 和 P'_{j-1} 用了相同的 $2(j-1)$个 U_{P_k} 中的元素,且权值相同.由假设可知,X_i 中存在 j-packing P_j,当 φ_{l_j} 在第 3.2 步被考虑时,P'_{j-1} 中的 U_{P_k} 元素和 φ_{l_j} 中的 U_{P_k} 元素完全不相同.因此,如果 Q_3 中不包含一个与 $P'_{j-1}\cup\{\varphi_{l_j}\}$ 用了相同 $2j$ 个 U_{P_k} 元素且权值相同的 j-packing,则 j-packing $P'_{j-1}\cup\{\varphi_{l_j}\}$ 将要被加入到 Q_3 中.因为 Q_3 中的 packing 在算法的执行过程中不会被移出 Q_3,并且 $l_j\le i$,所以在执行完 i 次第 3 步的 for 循环后,Q_3 一定含有一个 j-packing P'_j,使得 P'_j 与 P_j 用了相同的 $2j$ 个 U_{P_k} 的元素,且权值相同.当 $i=m$ 时,如果 C_2 中存在 k''-packing $P_{k''}$,则 Q_3 中必包含一个 $P'_{k''}$,使得 $P'_{k''}$ 与 $P_{k''}$用了相同 $2j$ 个 U_{P_k} 的元素,且权值相同.

第②部分的证明过程如下:

对算法 3 中第 5 步 for 循环中的 h 进行归纳以证明:如果 $C_2\cup C_3$ 中存在权值最大的 k'-packing $P_{k'}$,则 Q_3 中必包含一个 $P'_{k'}$,使得 $P'_{k'}$ 与 $P_{k'}$用了相同的 U_{P_k} 中的元素,且权值相同.

对于任意的 $h(1\le h\le l)$,假设 h 个三元组 $z_1,z_2,\ldots,z_h$ 组成的集合为 $Z_h,Z_h\subseteq C_3$,需证明以下命题:如果 $C_2\cup Z_h$ 存在一个权值最大 j-packing P_j,那么第 h 次执行算法中第 5 步的 for 循环后,Q_3 一定包含了一个 j-packing P'_j,使得 P_j 与 P'_j 用了相同的 U_{P_k} 中的元素,且权值相同.

假设 $C_2\cup Z_h$ 存在权值最大的 j-packing $P_j=\{\varphi_{l_1},\varphi_{l_2},\ldots,\varphi_{l_j}\}$,其中,$1\le l_1<l_2<\ldots<l_j\le h$,则在 $C_2\cup Z_{l_j-1}$ 中必包含一个$(j-1)$-packing $P_{j-1}=\{\varphi'_{l_1},\varphi'_{l_2},\ldots,\varphi'_{l_j-1}\}$,使得 P_{j-1} 和$\{P_j-\varphi_{l_j}\}$用了相同的 U_{P_k} 元素,且权值相同.由归纳假设可知,经过第(l_j-1)次执行算法 3 中第 5 步的 for 循环后,Q_3 包含了一个$(j-1)$-packing P'_{j-1},使得 P_{j-1} 和 P'_{j-1} 用了相同的 U_{P_k} 中的元素,且权值相同.由假设可知,$C_2\cup Z_h$ 中存在权值最大的 j-packing P_j,当 φ_{l_j} 在第 5.2 步被考虑时,P'_{j-1} 中的 U_{P_k} 元素与 φ_{l_j} 中的 U_{P_k} 元素完全不同.因此,如果 Q_3 中不包含一个与 $P'_{j-1}\cup\{\varphi_{l_j}\}$ 用了相同 U_{P_k} 元素且权值相同的 j-packing,则 $P'_{j-1}\cup\{\varphi_{l_j}\}$ 将要被加入到 Q_3 中.因为 Q_3 中的 packing 在算法的执行过程中不会被移出 Q_3 并且 $l_j\le h$,所以在执行完 h 次第 5 步的 for 循环后,Q_3 一定含有一个 j-packing P'_j,使得 P'_j 与 P_j 用了相同的 U_{P_k} 元素,且权值相同.当 $h=l$ 时,如果 $C_2\cup C_3$ 中存在权值最大的 k'-packing $P_{k'}$,则 Q_3 中必包含一个 $P'_{k'}$,使得 $P'_{k'}$ 与 $P_{k'}$用了相

同U_{P_k}中的元素,且权值相同.

最后,对算法的时间复杂度进行分析.如果只对C_2来说,对于每一个$0\le j\le k'$和包含$2j$个属于U_{P_k}的元素的集合,Q_3仅最多保存1个用了这$2j$个元素的j-packing符号对.此时,Q_3最多包含$\sum_{j=0}^{k'+1}\binom{3k}{2j}$个packing符号对.如果只对$C_3$来说,对于每一个$0\le j\le k'$和包含$3j$个属于$U_{P_k}$的元素的集合,$Q_3$仅最多保存1个用了这$3j$个元素的$j$-packing.此时,$Q_3$最多包含$\sum_{j=0}^{k'-1}\binom{3k}{3j}$个packing.因此,算法SP的时间复杂度为

$$\max\left\{O^*\left(\sum_{j=0}^{k'+1}\binom{3k}{2j}\right),O^*\left(\sum_{j=0}^{k'-1}\binom{3k}{3j}\right)\right\}=O^*(2^{3k}).$$ □

3.2 在H中求一个权值最大的r-packing

假设已从P_k的$3k$个元素中取出了$r(0\le r\le k+1)$个元素,用H表示C_1中这r个元素所在的集合.

在H中用分治法和集合划分求一个权值最大的r-packing的算法思想是:从r个元素中任意选取$\left\lceil\frac{r}{2}\right\rceil$个元素,用$H_1$表示$H$中这$\left\lceil\frac{r}{2}\right\rceil$个元素所在的集合,用$H_2$表示$H-H_1$中的集合.对于$(|U_{S_2-P_k}|,2k+2)$-universal set中的某一个对$U_{S_2-P_k}$的划分,被划分到$H$中属于$U_{S_2-P_k}$元素集记为$U_{S_2-P_k}^H$.如果$H$中存在权值最大的$r$-packing P_r,则该r-packing中共有$2r$个属于$U_{S_2-P_k}$的元素,记为$U_{P_r-P_k}$.假设$U_{P_r-P_k}$中属于H_1的$U_{S_2-P_k}$的元素组成的集合为$U'_{P_r-P_k}$,要想求得该r-packing,则必须把$U'_{P_r-P_k}$划分到H_1中,把$U_{P_r-P_k}-U'_{P_r-P_k}$划分到H_2中.构造$(|U_{S_2-P_k}^H|,2r)$-universal set,然后递归在H_1,H_2中寻找相应的最大加权packing.具体过程见算法4.

算法4. RSP(H',D',r).

输入:H',D',r,其中,H'是H的一个子集,D'是H中r'-packing的集合且D'与H'没有公共元素.

输出:返回packing的集合D,且D中的每个packing是由D'中的packing和H'中的packing合并而成.

1. $D=\{\varnothing\}$;
2. if $r=1$ then
2.1. if $D'=\varnothing$ then 返回H'中的所有1-packing;
    else for D'中每一个r'-packing P和H'中的每个元组ρ do
2.2. if P中没有元素与ρ中的元素相同
    then $P'=P\cup\{\rho\}$
2.3. if D中不存在包含ρ的$(r'+1)$-packing then 将P'加到D中;
2.4. if D中存在包含ρ的$(r'+1)$-pakcing $\overline{P}$,但$\overline{P}$的权值不大于P'的权值
    then 用P'取代$\overline{P}$;
2.5. 返回D;
3. 从r个元素中任取$\left\lceil\frac{r}{2}\right\rceil$个元素,设$H$中这$\left\lceil\frac{r}{2}\right\rceil$个元素所在的集合为$H_1,H_2=H-H_1$;
4. for $(|U_{S_2-P_k}^H|,2r)$-universal set中的每一种划分 do
4.1. $H_1'=H_1;H_2'=H_2$;
4.2. 在H_1'中,如果某集合中属于$U_{S_2-P_k}$的元素被划分到H_2'中,则删除该集合;
4.3. 在H_2'中,如果某集合中属于$U_{S_2-P_k}$的元素被划分到H_1'中,则删除该集合;
4.4. $D_1=RSP\left(H_1',D',\left\lceil\frac{r}{2}\right\rceil\right)$;
4.5. if $D_1\ne\varnothing$ then

4.6. $D_2=RSP\left(H_2', D_1, r-\left\lceil \frac{r}{2} \right\rceil\right)$;

4.7. for D_2 中的每个$(r'+r)$-packing α do
  if D 中不存在$(r'+r)$-packing then 把α加入到 D 中;
  if D 中存在$(r'+r)$-packing $\bar{\alpha}$,但$\bar{\alpha}$的权值不大于α的权值
  then 用α取代$\bar{\alpha}$;

5. 返回 D.

定理 4. 如果 H 中存在权值最大的 r-packing,算法 RSP 必定返回一个包含该 r-packing 的集合 D,且该算法的时间复杂度为 $O(16^{k+O(\log^3 k)})$.

证明:本算法把 H 分成两部分来处理:H_1,H_2.如果 H 中存在权值最大的 r-packing P_r,则该 r-packing 中共有 $2r$ 个属于 $U_{S_2-P_k}$ 的元素,记为 $U_{P_r-P_k}$.假设 $U_{P_r-P_k}$ 中属于 H_1 的 $U_{S_2-P_k}$ 的元素组成的集合为 $U'_{P_r-P_k}$,要想求得该 r-packing,则必须把 $U'_{P_r-P_k}$ 划分到 H_1 中,把 $U_{P_r-P_k}-U'_{P_r-P_k}$ 划分到 H_2 中.根据(n,k)-universal set 的构造,可以构造$(|U^H_{S_2-P_k}|,2r)$-universal set,其大小不会超过 $O(2^{2k+2+12\log^2(2k+2)+12\log(2k+2)+2\log(4k+6)+6})$,总有一种划分会把 $U'_{P_r-P_k}$ 划分到 H_1 中,把 $U_{P_r-P_k}-U'_{P_r-P_k}$ 划分到 H_2 中.

假设 T_r 为算法总的时间复杂度:

$$
\begin{aligned}
T_r &\le 2^{2k+2+12\log^2(2k+2)+12\log(2k+2)+2\log(4k+6)+6}\left(T_{\left\lceil \frac{r}{2}\right\rceil}+T_{\left\lfloor \frac{r}{2}\right\rfloor}\right)\\
&\le 2^{2k+2+12\log^2(2k+2)+12\log(2k+2)+2\log(4k+6)+7}\,T_{\left\lceil \frac{r}{2}\right\rceil}\\
&\le O(2^{2(2k+2)+12\log^3(2k+2)+12\log^2(2k+2)+2\log^2(2k+6)+7\log(2k+2)})\\
&= O(16^{k+O(\log^3(2k+2))}).
\end{aligned}
$$

下面对算法的空间复杂度进行分析.对于算法 RSP 的每一次递归调用,在算法的第 2.3 步和第 2.4 步,如果 D 中存在不包含ρ的 packing,才把新生成的 packing 加入到 D 中.D 中存在一个包含ρ的 pakcing $\bar{P}$,但 $\bar{P}$ 的权值不大于 P'的权值,用 P'取代 $\bar{P}$.因此,D_1,D_2 中最多包含了 nr 个 packing.由于算法的递归次数不会超过 $\log r$,因此该算法用到的空间复杂度为 $O(nr\log r)$. □

3.3 在S_2中求一个权值最大的$(k+1)$-packing

在 S_2 中求一个权值最大的$(k+1)$-packing 的算法思想是:由引理 5 可知,P_{k+1} 中只包含 U_{P_k} 中 1 个元素的集合个数为 $r(0\le r\le k+1)$.由于不知道 r 的具体取值,要枚举出 r 的所有可能取值.对于某一个 r 来说,从 P_k 中的 $3k$ 个元素中枚举出 r 个元素共有$\binom{3k}{r}$组合方式,用 H 表示 C_1 中含有这 r 个元素的集合.对于每种组合方式,需要进行如下过程:把 P_{k+1} 分成两部分来处理:一部分包含在 H 中,另一部分包含在 $C_2\cup C_3$ 中.对于 P_{k+1} 中被包含在 $C_2\cup C_3$ 中的那一部分,用动态规划求一个权值最大的$(k+1-r)$-packing;对于 H 中的那一部分,用分治法求一个权值最大的 r-packing.具体过程见算法 5.

算法 5. GSP(S_2,k).

输入:S_2,k.

输出:如果 S_2 中存在权值最大的$(k+1)$-packing,则返回该 packing;否则,返回不存在.

1. $Q=\{\varnothing\}$;
2. 构造基于 U 映射函数族 $\Psi_{n,2k+2}$;
3. for r=0 to k+1 do

3.1. 从 P_k 中的 $3k$ 个元素中枚举出 r 个元素,共有 $\binom{3k}{r}$ 组合方式;

3.2. for 每一种组合方式 do

用 H 表示 C_1 中这 r 个元素所在的集合;

3.3. for $\Psi_{n,2k+2}$ 中每一映射函数 g do

for ($|U_{S_2-P_k}|$,2k+2)-universal set 中的每一种划分 do

$C_2'=C_2;C_3'=C_3$;H'=H;

若 P_k 中某元素属于 H',则在 $C_2'\cup C_3'$ 中删除包含该元素的所有集合.

在 C_2' 中,如果某集合中属于 $U_{S_2-P_k}$ 的元素已被划分到 H'中,则删除该集合;

在 H'中,如果某集合属于 $U_{S_2-P_k}$ 的元素中有元素被划分到 C_2' 中,则删除该集合;

调用算法 Q_4=RSP(H',∅,r);

if Q_4 不为空 then

调用算法 Q_5=SP($C_2',C_3',k+1-r,U_{P_k}$);

if Q_5 不为空 then

合并 Q_4,Q_5 中的 packing,构造相应的(k+1)-packing,并把其加入 Q 中;

4. if Q 不为空 then

返回 Q 中权值最大的(k+1)-packing;

else 返回 S_2 中不存在(k+1)-packing.

定理 5. 如果 S_2 中存在权值最大的(k+1)-packing,则算法 GSP 必定返回该(k+1)-packing,且算法的时间复杂度为 $O^*(7.56^{3k})$.

证明:在算法 GSP 中,穷举 r 的取值,要考虑从 U_{P_k} 中取出 r 个元素的所有组合方式.如果 S_2 中存在权值最大的(k+1)-packing P_{k+1},则必有一个 r 和一种组合方式满足.本算法把 P_{k+1} 分成两部分来处理:一部分包含在 H 中,另一部分包含在 $C_2\cup C_3$ 中.基于 U 映射函数族 $\Psi_{n,2k+2}$ 中必存在一个映射函数 g,使得 g 在 $U_{P_{k+1}}$ 上是单射的.在构造的($|U_{S_2-P_k}|$,2k+2)-universal set 中,必有一种划分把集合 $U^H_{P_{k+1}-P_k}$ 元素恰好被划分到 H 中,且 $U_{P_{k+1}-P_k}-U^H_{P_{k+1}-P_k}$ 中的元素恰好被划分到 $C_2\cup C_3$ 中.由定理 3 可知,如果 $C_2\cup C_3$ 中存在权值最大的(k+1−r)-packing,则算法 SP 必定返回该(k+1−r)-packing.根据定理 4 可知,如果 H 中存在权值最大的 r-packing,则算法 RSP 必定返回一个包含该 r-packing.因此,如果 S_2 中存在权值最大的 P_{k+1},则算法 GSP 必定返回该(k+1)-packing.

下面对算法的复杂度进行分析.对于 r 的每一种取值,要枚举所有可能的 r 个元素,共有 $\binom{3k}{r}$ 组合.然后,对于($|U_{S_2-P_k}|$,2k+2)-universal set 中的每个划分函数,调用算法 RSP 和算法 SP 寻找相应的 r-packing 和(k+1−r)-packing.($|U_{S_2-P_k}|$,2k+2)-universal set 的大小不会超过 $O(2^{2k+2+12\log^2(2k+2)+12\log(2k+2)+2\log(4k+6)+6})$.从 U_{P_k} 中枚举出 r 个元素后,U_{P_k} 剩余 2k−r 个元素.由定理 3 可知,算法 SP 的时间复杂度为 $O^*(2^{3k-r})$.由定理 4 可知,算法 RSP 的时间复杂度为 $O(16^{k+O(\log^3 k)})$.因此,算法 GSP 的时间复杂度为

$$\begin{aligned}O\left(\sum_{r=0}^{k+1}\binom{3k}{r}(2^{2k+2+12\log^2(2k+2)+12\log(2k+2)+2\log(4k+6)+6}(2^{3k-r}+T_r))\right)&=O^*\left(\sum_{r=0}^{k+1}\binom{3k}{r}(2^{2k+O(\log^2 k)}(2^{3k-r}+T_r))\right)\\&=O^*\left(\sum_{r=0}^{k+1}\binom{3k}{r}(2^{2k+O(\log^2 k)}(2^{3k-r}+16^{k+O(\log^3 k)}))\right)\\&=O^*\left(2^{6k+O(\log^3 k)}\sum_{r=0}^{k+1}\binom{3k}{r}\right)\\&=O^*(7.56^{3k}).\end{aligned}$$ □

根据引理 1、定理 2 和定理 5,可得以下推论:

推论 2. 加权 3-Set Packing 问题可在 $O^*(7.56^{3k})$时间内被求解.

4 结 论

本文主要讨论了如何从一个最大加权的 k-packing 着手构造权值最大的(k+1)-packing,最终求解加权 3-Set Packing 问题.对于加权 3-Set Packing Augmentation 问题,把 S 分成 S_1,S_2 两部分进行处理.在 S_1 中的集合和 P_k 中集合可构造的所有(k+1)-packing 中,可在多项式时间内找到一个权值最大的(k+1)-packing PW_1.对于 S_2 中的(k+1)-packing,可得到如下性质:S_2 中的 $U_{P_{k+1}}$ 至少包含 k+1 个 U_{P_k} 中的元素.基于上述性质,通过使用 Color-Coding 技术,可以在 $O^*(10.6^{3k})$时间内得到 S_2 中的权值最大的(k+1)-packing PW_2.并证明了 PW_1 和 PW_2 的权值较大者为 S 中的最大加权(k+1)-packing.本文通过对 S_2 中(k+1)-packing 的结构进一步进行分析并使用集合划分技术,给出了时间复杂度为 $O^*(7.56^{3k})$的参数算法,该结果与文献中的最好结果 $O^*(12.8^{3k})$相比有了极大的改进.本文给出的算法也可被用于求解图论中各种三角形 packing 问题[16].例如,加权边不相交三角形 packing 问题、加权点不相交三角形 packing 和加权 P_2-packing 问题都可用本文给出的算法进行求解,显著地降低了相应问题的时间复杂度.

References:

[1] Chen JE, Liu Y, Lu SJ, Sze SH. Greedy localization and color-coding: Improved matching and packing algorithms. In: Bodlaender H, *et al*., eds. Proc. of the 2nd Int'l Workshop on Parameterized and Exact Computation. LNCS 4169, Berlin: Springer-Verlag, 2006. 84−95.

[2] Chandra B, Halldorsson M. Greedy local improvement and weighted set packing approximation. Journal of Algorithms, 2001,39(2): 223−240. [doi: 10.1006/jagm.2000.1155]

[3] Downey R, Fellows M. Parameterized Complexity. New York: Springer-Verlag, 1999. 215−220.

[4] Arkin E, Hassin R. On local search for weighted *k*-set packing. In: Burkard R, *et al*., eds. Proc. of the European Symp. on Algorithms. Berlin: Springer-Verlag, 1997. 13−22.

[5] Bafna V, Narayan B, Ravi R. Nonoverlapping local alignments (weighted independent sets of axis-parallel rectangles). Discrete Applied Mathematics, 1996,71(1-3):41−53. [doi: 10.1016/S0166-218X(96)00063-7]

[6] Berman P. A *d*/2 approximation for maximum weight independent set in d-claw free graphs. In: Halldórsson M, ed. Proc. of the 7th Scandinavian Workshop on Algorithm Theory. LNCS 1851, Berlin: Springer-Verlag, 2000. 214−219.

[7] Liu YL, Chen JE, Wang JX. Parameterized algorithms for weighted matching and packing problems. In: Cai JY, *et al*., eds. Proc. of the 4th Annual Conf. on Theory and Applications of Models of Computation. LNCS 4484, Berlin: Springer-Verlag, 2007. 692−702.

[8] Jia WJ, Zhang CL, Chen JE. An efficient parameterized algorithm for *m*-set packing. Journal of Algorithms, 2004,50(1):106−117. [doi: 10.1016/j.jalgor.2003.07.001]

[9] Koutis I. A faster parameterized algorithm for set packing. Information Processing Letters, 2005,94(1):7−9. [doi: 10.1016/j.ipl.2004.12.005]

[10] Fellows MR, Knauer C, Nishimura N, Ragde P, Rosamond F, Stege U, Thilikos D, Whitesides S. Faster fixed-parameter tractable algorithms for matching and packing problems. In: Albers S, *et al*., eds. Proc. of the European Symp. on Algorithms. LNCS 3221, Berlin: Springer-Verlag, 2004. 311−322.

[11] Kneis J, Möelle D, Richter S, Rossmanith P. Divide-and-Color. In: Fomin F, ed. Proc. of the 32nd Int'l Workshop on Graph-Theoretical Concepts in Computer Science. LNCS 4271, Berlin: Springer-Verlag, 2006. 58−67.

[12] Wang JX, Feng QL. An $O^*(3.52^{3k})$ parameterized algorithm for 3-set packing. In: Agrawal M, *et al*., eds. Proc. of the 5th Annual Conf. on Theory and Applications of Models of Computation. LNCS 4978, Berlin: Springer-Verlag, 2008. 212−222.

[13] Alon N, Yuster R, Zwick U. Color-Coding. Journal of the ACM, 1995,42(4):844−856. [doi: 10.1145/210332.210337]

[14] Chen JE, Lu SJ. Improved parameterized set splitting algorithms: A probabilistic approach. Algorithmica, 2009,54(4):472−489. [doi: 10.1007/s00453-008-9206-y]

[15] Fredman M, Komlos J, Szemeredi E. Storing a sparse talbe with $O(1)$ worst case acess time. Journal of the ACM, 1984,31(3): 538−544. [doi: 10.1145/828.1884]

[16] Mathieson L, Prieto E, Shaw P. Packing edge disjoint triangles: A parameterized view. In: Downey R, *et al*., eds. Proc. of the Int'l Workshop on Parameterized and Exact Computation. LNCS 3162, Berlin: Springer-Verlag, 2004. 127−137.

冯启龙(1982－),男,山东临沂人,博士生,主要研究领域为参数计算.

陈建二(1954－),男,博士,教授,博士生导师,主要研究领域为生物信息学,计算机理论,计算复杂性及优化,计算机网络优化算法,计算机图形理论与算法.

王建新(1969－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为生物信息学,网络优化理论.

* *

2010 年全国开放式分布与并行计算学术年会

征 文 通 知

由中国计算机学会开放系统专业委员会主办、新疆大学软件学院承办的 2010 年全国开放式分布与并行计算学术年会(DPCS2010)将于 2010 年 8 月 19 日–21 日在新疆乌鲁木齐市新疆大学召开。本次年会录用的论文将以正刊方式发表在《微电子学与计算机》第 8 期。会议将评选优秀论文，予以奖励并推荐到一级学报发表。欢迎大家积极投稿。

一、征文范围(包括但不限于)

开放式分布与并行计算模型、体系结构、编程环境、算法及应用；开放式网络、数据通信、网络与信息安全、业务管理技术；开放式海量数据存储与 Internet 索引技术，分布与并行数据库及数据/Web 挖掘技术；开放式网格计算、云计算、Web 服务、P2P 网络及中间件技术；开放式无线网络、移动计算、传感器网络与自组网技术；分布式人工智能、多代理与决策支持技术；开放式虚拟现实技术与分布式仿真；开放式多媒体技术与流媒体服务，包括媒体压缩、内容分送、缓存代理、服务发现与管理技术。

二、来稿要求

1．论文必须是未正式发表的，或者未正式等待刊发的研究成果。稿件格式应包括题目、作者、所属单位、摘要、关键词、正文和参考文献等，具体格式请参照网站提供的样式。

2．请务必附上第一作者简历(姓名、性别、出生年月、出生地、职称、学位、研究方向等)、通信地址、邮政编码、联系电话和电子信箱。同时，请注明论文所属领域。来稿一律不退，请自留底稿。

3．论文投稿通过会议网站(http://cs.nju.edu.cn/dpcs)提交，也可按如下地址提交激光打印稿一式 2 份和电子版(Word 文件)：

联系人及地址：830008 新疆乌鲁木齐市西北路 134 号 新疆大学(北校区)软件学院，于炯 院长

E-mail：DPCS2010@sina.com，DPCS2010@sohu.com

三、重要日期

征文投稿截止日期：2010 年 6 月 5 日　　录用通知发出日期：2010 年 6 月 20 日

四、联系方式

1．会议承办方

新疆大学软件学院 于炯，电话：0991-4556262，13150469888；E-mail：DPCS2010@sina.com

新疆大学软件学院 田园，电话：0991-4558654，13999108264；E-mail：tianyuan0628@sohu.com

2．专委会

南京大学计算机系 陈贵海，电话：13951985532；E-mail：gchen@nju.edu.cn