arXiv:2302.00458v1 [cs.DS] 1 Feb 2023

最大权团的改进精确算法与启发式算法

Roman Erhardt* Kathrin Hanauer† Nils Kriege†‡ Christian Schulz*Darren Strash§

摘要

我们提出了用于求解最大权团问题的改进精确算法与启发式算法。最大权团问题是图论中的一个著名问题，具有广泛的应用。我们的算法将相关研究中的有效技术与新颖的数据约简规则交织运用，利用图的局部结构识别并移除顶点和边，同时保留最优解。我们在一系列合成图和真实世界图上评估了这些算法，发现它们在大多数输入实例上都优于当前最先进水平。我们提出的数据约简方法始终能生成比单独使用现有数据约简方法更小的约简图。因此，我们的精确算法 MWCRedu 在具有自然权重的中等规模地图标注图和随机双曲图上，能以快几个数量级的速度找到解。我们的启发式算法 MWCPeel 在这些实例上的表现优于竞争算法，但在极其稠密或大规模实例上的效果略逊一筹。

1 引言

在图中寻找团是图论中的经典问题，有着广泛的应用。在社交网络中，可以借助团来预测群体行为 [47]。在生物化学中，团可用于研究分子之间的相互作用，从而为药物发现提供信息 [33]。顶点加权图及其对应的最大权团问题 （MWC）可用于更广泛的应用，包括视频对象协同分割 [51], 编码理论 [52], 组合拍卖 [49], 和基因组学 [4]。

求解最大（无权）团问题一直是广泛研究的课题 [9, 31, 41,42, 48, 53], ，其中最有效的求解器将分支定界与 MaxSAT 推理结合起来进行剪枝 [30, 38]。然而，最先进水平的算法仍难以在合理的时间限制内为某些实例找到解。事实上，仍有尚未解决的实例，而近期才解决的实例耗时超过一年计算时间 [50]。近期研究重点在于求解加权变体的**NP**困难图问题 [7, 29, 45], ，这些问题在实践中更难求解。

解决**NP**难图问题的一种有力方法是使用数据约简规则，通过删除或收缩局部图结构，将输入实例约简为一个等价且规模更小的实例。数据约简规则最初是作为参数化算法的工具而开发的 [13]„ 在实际中已有效用于计算（无权）最大独立集 [11, 28, 39]/最小顶点覆盖 [2], 最大团 [10, 43], 和最大 k-团 [12, 25], ， 以及解决图着色 [32, 43] 和团覆盖问题 [19, 40], 等 [1]。然而，近期研究对加权 问题的探索仍只是冰山一角。La mm等人 [29], 、Gellner等人 [17], 和Gu等人 [20]近期为最大权独立集问题（MWIS）提出了一系列有效的数据归约方法，而Wang 等人 [45] 则针对加权图着色进行了数据归约。

然而，据我们所知，最大权团唯一的数据约简规则只是根据邻域的权重或邻居的最大权重来删除顶点 [7]。因此，进一步约简输入实例、使其更适合精确求解的潜力尚未得到充分挖掘。一种策略是对输入的补图 应用最大权独立集约简；然而，最大权独立集约简在大型稀疏实例上最为有效，而此处考虑的图的补图是稠密的，且不太可能装入内存。

我们的结果。 我们为最大权团开发了一套新颖的精确和启发式数据约简规则，旨在减少输入图中的顶点和边的数量，同时保持解质量。据我们所知，我们的数据约简规则是首批利用局部图结构解决最大权团问题的规则。我们还提出了专门用于删除图中边 的数据约简规则；据我们所知，此前尚未有人针对类似问题采用这种做法。约简图后，我们对剩余实例应用启发式或精确算法，以获得原始输入的解。我们扩展了近期提出的约简与剥离框架

*海德堡大学
†维也纳大学计算机科学学院，奥地利维也纳
‡维也纳大学数据科学研究网络，奥地利维也纳§汉密尔顿学院计算机科学系

该框架最初用于最大独立集和最大权独立集问题，并设计了如何以及何时应用约简、何时切换到精确求解器的方法。实验结果表明，我们的算法优于最先进水平。

2 预备知识

2.1 基本概念。我们考虑一个简单的加权无向图 $G=(V,E,w)$，其中包含 $n=|V|$和 $m=|E|$，$V=\{1,\dots,n\}$ 是顶点集合，$E\subseteq\{\{u,v\}\mid u,v\in V\}$ 是二元边集合，并且w: $V\to\mathbb{R}_{>0}$ 是一个为每个顶点赋予正实数权重的函数。我们将 w 扩展到集合，使得对于 $V'\subseteq V$, $w(V')=\sum_{v\in V'}w(v)$。 V' 的最大权重记为 $w^*(V')=\max_{v\in V'}w(v)$。若 $\{u,v\}\in E$，两个顶点 u 和 v 相邻（也称为邻居）。顶点$v\in V$的（开）邻域 $N(v)$ 定义为 $N(v)=\{u\in V\mid\{u,v\}\in E\}$，其闭邻域为 $N[v]=N(v)\cup\{v\}$。这两个定义都可以直接扩展到顶点集合 $V'\subset V$ 的邻域 $N(V')$，即$N(V')=\cup_{v\in V'}N(v)\setminus V'$和 $N[V']=N(V')\cup V'$。顶点的度 $\deg(v)$ 是其邻居数量，即 $\deg(v)=|N(v)|$；而 $\Delta:=\Delta(G)$ 表示最大度 $\max_{v\in V}\deg(v)$。 G 的补图定义为 $G=(V,E)$，其中 $E=\{\{u,v\}\mid u,v\in V\wedge u\neq v\wedge\{u,v\}\notin E\}$ 是 G中不存在的边的集合。G 的密度 $\rho:=\rho(G)$是已存在边数与可能存在边数之比， $\rho(G)=\frac{2m}{n(n-1)}$。由子集 $V'\subseteq V$诱导的子图记为 $G[V']=(V',E')$，其中 $E'=\{\{v_i,v_j\}\in E\mid v_i,v_j\in V'\}$。若对于所有顶点对 $u,v\in V'$均有 $\{u,v\}\notin E$，则称集合$V'\subseteq V$ 为独立集。

团是一个集合 $Q\subseteq V$，其中所有顶点两两相邻。补图 G 中的团对应于原图 G中的独立集，反之亦然。最大权团问题（MWC）是寻找权重最大的团。如果 $w\equiv 1$，我们得到最大基数团问题（MCC）（更简洁地称为最大团问题）。最大独立集问题（MIS）是寻找基数最大的独立集，而最大权独立集问题（MWIS）则要求寻找总权重最大的独立集。独立集的补集是一个顶点覆盖，即一个子集 $C\subseteq V$，使得每条边 $e\in E$ 都至少与其中一个顶点 C关联。最小顶点覆盖问题要求找到基数最小的顶点覆盖，因此它与最大独立集问题互为对偶。通过补图 G，最大团问题也与最大独立集问题和最小顶点覆盖问题互为对偶。通过推广而言，加权独立集与加权团也互为对偶。

顶点着色问题要求为每个顶点分配一个颜色标签 $c\in\mathbb{Z}$，使得任意两个相邻顶点的标签都不同，且所用不同颜色的数量最少。团中的所有顶点都必须分配不同的颜色。因此，如果一个图可以用 k 种颜色进行顶点着色，那么任何团的基数最多为 k。所有这些问题都是**NP**难问题。

2.2 相关工作。本文总结并扩展了硕士论文 [14]。针对最大团和最大权团问题，已有大量研究。由于本文侧重于加权版本，我们只介绍最大权团问题的研究成果；除非相关求解器和结果已扩展到加权情形，否则不详细讨论基数版本。关于最大团问题方法的详细综述，可参见 Wu 和 Hao [48]，以及 Abu-Khzam 等人 [1] 关于数据归约的研究。

2.2.1 精确求解器。大多数最大团问题的精确求解器采用分支定界框架 [9]，，该框架维护一个当前团 C 以及一个用于扩展该团的顶点候选集 $P=N(C)\setminus C$。快速求解器通过迅速计算一个紧的上界来剪枝搜索空间，该上界对应于将 P 中的顶点加入 C 后可得到的团大小。一种行之有效的方法是在$G[P]$ 上计算贪心启发式顶点着色，并将颜色数作为上界。随后，Kumlander [27]将这一方法扩展至最大权团问题：给定 $G[P]$ 的一个有效顶点着色，该着色使用 k 种颜色，并将 V 划分为颜色类 $\mathcal{D}=D_1\sqcup D_2\sqcup\cdots\sqcup D_k$，假设每个颜色类贡献一个权重最大的顶点，则可计算上界 $ub(\mathcal{D})=\sum_{j=1}^{k}w^*(D_j)$。

Fang 等人 [15] 率先将 MCC 求解器 MaxCLQ [31] 提出的 MaxSAT 推理思想应用于最大权团（MWC）。Jiang 等人 [24] 同样依赖 MaxSAT 推理，并提出了一种高效的ˆ 预处理步骤，用于计算初始团 C以及顶点分支顺序。此外，该步骤还会计算一个简单的上界，估计每个顶点 v 可能参与构成的最大权团的权重，即 $w(N[v])$ ˆ；并在 $w(N[v])\leq w(C)$ 时移除 v 。TSM-MWC [23]进一步改进了这一方法，采用两阶段 MaxSAT 推理方法，先应用成本较低的 MaxSAT 技术来减少分支顶点的数量，再全面搜索互不相交的冲突软子句。目前，TSM-MWC 在多种图实例上取得了最佳结果，尤其是在大型稀疏真实世界图实例上表现突出，是当前最先进的最大权团精确求解器。

2.2.2 启发式求解器。MCC 的局部搜索算法通常采用以下方案：从单个顶点开始构造一个团C，并反复添加与 C 中所有顶点都相邻的顶点，具体使用某种评估函数进行选择。同样，候选顶点是指那些可能被添加到 C 中的顶点。当无法再执行添加操作时，可以尝试移除一些顶点，以构造更大的团。

Gendrau 等人 [18] 基于这一策略提出了两种用于最大团的算法：一种是确定性方案，先加入度最高的顶点；当无法再加入顶点时，则移除能使候选顶点集最大的顶点。第二种算法会随机选择要加入当前解的顶点。Pullan [35]提出在主要搜索过程中加入交换算子。该算子会寻找一个与当前候选团 C中除一个顶点以外的所有顶点都相连的顶点。此外，该算法还会通过加入一个随机顶点，并从团中移除所有不相邻的顶点，来扰动当前候选团。

Pullan [36] 通过仅从权重最高的顶点中随机选择一个顶点并将其加入，将该算法扩展到了最大权团。Wang 等人 [46] 加入了一条基于配置检查的禁忌规则。Cai [5]进一步改进了该算法，采用更好的策略来决定从候选集中加入哪个顶点。该策略通过随机抽取 k 个不同的候选顶点，并根据某个收益估计函数选出最佳顶点来实现。Cai 和 Lin [7] 在其求解器 FastWCLq 中将该算法与数据约简规则相结合。他们使用的约简规则会计算每个顶点的上界；如果某个计算出的上界小于当前最佳团的权重，就移除该顶点。每当局部搜索找到更优解时，都会重新应用这些约简规则，从而提高局部搜索找到最优解的可能性。

SCCWalk4l [44] 采用了此前提出的配置检查策略以及数据归约。作者还提出了一种称为游走扰动的技术：当搜索陷入停滞时，向解中添加一个随机顶点，并从候选集中移除所有因这次扰动而变为无效的顶点。Cai 等人 [8]进一步改进了 FastWCLq，也采用了一种基于顶点着色的归约与爬山法。

SCCWalk4l 和 FastWCLq 是当前最先进的最大权团启发式求解器；前者尤其擅长处理小型稠密网络，例如 DIMACS 和 BHOSLIB 挑战中的图 [44]，而后者在大型稀疏真实网络中表现最佳 [8]。

3 数据归约

迄今为止，已知可用于最大权团的归约方法很少。然而，尤其对于大型实例，应用精确数据约简是缩小问题规模的一项非常重要的技术。一般而言，归约可将顶点归类为以下三种情况之一：（1）属于某个解；（2）不属于解；或（3）暂缓处理，即对该顶点的判定取决于稍后获得的有关邻居的额外信息。我们将最终得到的约简图记为 $\mathcal{K}$，此时已不再适用任何归约规则。接下来，我们回顾现有方法，并介绍一大批针对最大权团的新归约方法。

3.1 邻域权重约简。文献中常见的一种简单而有效的约简 [7, 8, 23,24, 44]，基于任意包含 $v \in V$的团的上界 $w(N[v])$。

Reduction Rule 1. ([7]) 设$\hat{C}$ 为目前找到的权重最高的$\hat{}$团，并设 $v \in V$，使得 $w(N[v]) \leq w(C)$。那么，可以将 v 从图中移除，而不降低最大解权重。

若在整个归约过程中存储并维护邻域权重，则可在 $\mathcal{O}(1)$时间内对顶点 $v \in V$ 应用该规则。

3.2 最大权重邻居约简。Cai等人 [7]通过纳入或排除邻域中权重最高的顶点 u^*，强化了邻域权重约简规则。

Reduction Rule 2. ([7]) 设$\hat{C}$ 为目前找到的权重最高的$\hat{}$团，设 $v \in V \setminus C$，并设 $u^* = \arg\max^{u \in N(v)} w(u)$。若$\max\{w(N[v]) - w(u^*)\ w(N[v] \cap N[u^*])\} \leq w(\hat{C})$，则可以将 v 从图中移除，而不降低最大解权重。

要对顶点 $v \in V$ 应用此规则，首先在 $\mathcal{O}(\deg(v))$ 时间内找出其权重最高的邻居 u^*，然后在$\mathcal{O}(\min\{\deg(v), \deg(u^*)\})$ 时间内计算它们邻域的交集，总体耗时为 $\mathcal{O}(\deg(v))$。计算邻域交集是应用此归约规则以及下文所述若干其他规则的关键操作。计算 $N(u) \cap N(v)$ 的运行时间取决于图表示。假设可以在常数时间内检查两个顶点是否相邻，我们可以遍历较小的集合，并找出其中也与另一个顶点相邻的顶点，耗时为$\mathcal{O}(\min\{\deg(u), \deg(v)\})$。对于

大型稀疏图，我们使用邻接表，并通过遍历两个顶点的邻居，使用指示器在 $\mathcal{O}(\deg(u)+\deg(v))$ 时间内完成此操作。

3.3 孪生约简。现在，我们介绍第一条基于孪生顶点的新数据约简规则。考虑两个相邻顶点 u 和 v，它们具有相同的闭邻域。这类顶点称为孪生顶点。如果其中一个顶点在解中，那么另一个也必须在解中。图1给出了一个示例。

Reduction Rule 3. 设 $u、v \in V、u \neq v$ 和 $N[u] = N[v]$。那么，可以将 u 和 v 收缩为一个权重为 $\{u,v\}$ $(\{u,v\}) = w(u) + w(v)$ 的新顶点，并且 $N(\{u,v\}) = N(u) \cap N(v)$ 不会降低最大解权重。

证明。假设存在一个最优解 C^*，不失一般性地，包含 u，但不包含 v。由于 v 与 u 的所有邻居都相连，因此总可以将 v 加入该解，从而得到一个权重更大的解。因此，每个最优解要么同时包含 u 和 v，要么两者都不包含。 □

要检查两个顶点 $u,v \in V$ 的前置条件（其中 $\deg(u) = \deg(v)$），可以使用标记方案，在 $\mathcal{O}(\deg(v))$ 时间内求得它们邻域的交集。

3.4 支配约简。若顶点 $u \in V$ 满足 $N(v) \subseteq N(u)$，则称其支配 $v \in V$。此外，如果 $w(v) \leq w(u)$，那么包含 u 的极大团权重将大于或等于包含 v 的极大团权重。基于这一观察，得到以下约简规则：

Reduction Rule 4. 设 $u, v \in V, \{u,v\} \notin E, N(v) \subseteq N(u)$，且 $w(v) \leq w(u)$。那么，可以从图中移除 v，而不降低最大解权重。

证明。假设存在一个最优解 C^*，不失一般性地，包含 v 但不包含 u。由于 u 与 v 的所有邻居都相邻，因此总可以在解中用 u 替换 v，从而得到权重至少相同的解，因为 $w(v) \leq w(u)$。由于 $\{u,v\} \notin E$，没有团能同时包含 u 和 v。因此，至少存在一个不包含 v 的最优解。

□

给定 v，我们按如下方式寻找满足 $N(u) \supseteq N(v)$ 的顶点 u：任意选择 $x \in N(v)$，并遍历所有 $u' \in N(x)$。如果 $\{u',v\} \notin E$、$\deg(u') \geq \deg(v)$，且 $w(u') \geq w(v)$，我们就用 $\mathcal{O}(\deg(v))$ 时间测试 $N(v) \subseteq N(u')$ 是否成立。对于给定顶点 v，该方法能在 $\mathcal{O}(\deg(v) \cdot \Delta)$ 时间内找出满足引理4中归约规则条件的所有顶点 u'。

现在我们介绍第一个归约，其目的是从图中移除边。如果 u 和 v 相邻，也可以采用类似的归约。然而，不能简单地移除 v，因为 v 可能属于一个包含 u 的团。因此，我们将 u 的权重加到 v 上，然后移除边 $\{u,v\}$，这样既保留了 v 和 u 处于同一团中时可获得的最优解，又能同时缩小图的规模。

Reduction Rule 5. 令 $u、v \in V、\{v,u\} \in E$ 和 $N(v) \subseteq N[u]$。那么，将 $w(v)$ 增加到 $w'(v) = w(v) + w(u)$ 并从图中删除边 $\{u,v\}$，不会降低最大解权重。

证明。设 C^* 是<替换>图中的一个最优解。假设 C^* 包含 v，但不包含 u。那么可以将 u 加入 C^*，从而获得更高的权重，这与 C^* 是最优解的假设相矛盾。因此，如果 C^* 包含 v，它也包含 u。还需考虑以下两种情况：

情况 1： 如果 C^* 同时包含 u 和 v，那么 $w(C^*) \leq w(u) + w(v) + w(N(v) \setminus \{u\}) = w'(v) + w(N(v) \setminus \{u\})$，因此在约简图中存在一个等价解，仅包含 v。

情况 2： 如果 C^* 包含 u 但不包含 v，则 $w(C^*) \leq w(u) + w(N(u))$，并且约简图中存在相同的解。

□

该约简可以类似地应用孪生约简（约简规则 3）来实现。

3.5 边界规约。该规则是约简规则 2 的自然扩展，它利用计算出的界限不仅判断某个顶点是否可以被删除，还判断连接该顶点与其权重最高的邻居的边。给定一个顶点 $v \in V$ 及其权重最高的邻居 $u^* \in N(v)$，令 $\mathrm{ub}_{inc}(v,u^*)$ 表示包含上界 $w(v) + w(u^*) + w(N(v) \cap N(u^*))$，并令 ub_{exc} 表示排除上界 $w(N[v]) - w(u^*)$。约简规则 2 表明，当以下两个条件同时成立时，可以删除 v：$\hat{}\hat{}\hat{}\,\mathrm{ub}_{inc}(v,u^*) \leq w(C)$ 以及 $\mathrm{ub}_{exc}(v,u^*) \leq w(C)$，其中 C 是当前的最优解。边界规约的扩展基于以下观察：如果 $\mathrm{ub}_{exc}(v,u^*) > w(C)$，但 $\mathrm{ub}_{inc}(v,u^*) \leq \hat{}w(C)$，则可以删除边 $\{v,u^*\}$。我们将该规则扩展为适用于 v 的所有邻居：

Reduction Rule 6. 令 $v \in V, u \in N(v)$，并令 $\hat{}_C\hat{}$ 为目前找到的最佳团。若 $ub_{inc}(v,u) < w(C)$，则可以从图中移除边 $\{v,u\}$，而不会降低最大解权重。

证明。值 $ub_{inc}(v, u)$ 是包含 v 和 u 的任何团的权重上界。如果一个团

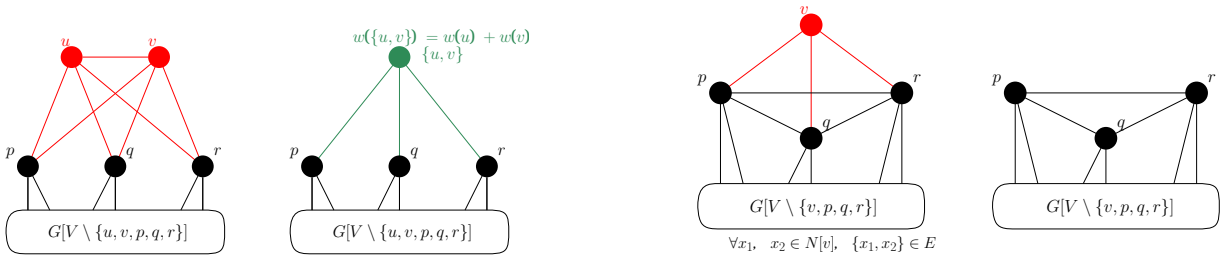

图1：MWC 的孪生约简（左）和单纯顶点删除归约（右）。

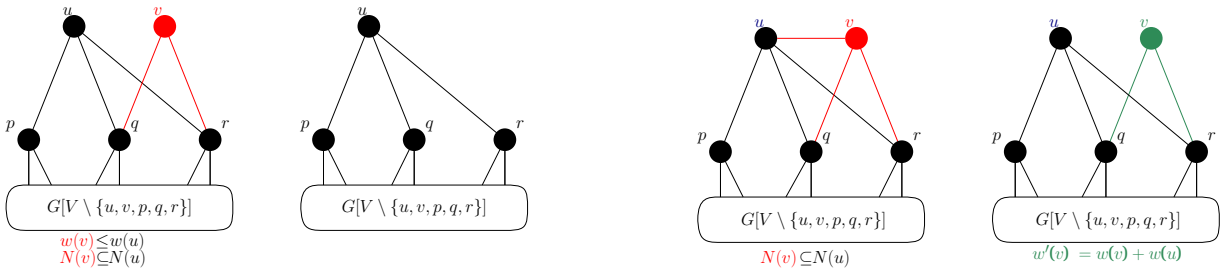

图2：支配　　归约：通过应用约简规则 4（左）和约简规则 5（右）对 MWC 进行约简　　).

$\hat{C}$ 的权重为 $w(\hat{C}) > ub_{inc}(v, u)$ 已知，那么至少存在一个不同时包含 v 和 u 的最优解 C^*。因此，在搜索更高权重的解时，边 $\{v, u\}$ 无关紧要。□

给定一条边 $\{v, u\}$，时间复杂度为$\mathcal{O}(\min\{\deg(v), \deg(u)\})$，与约简规则2相同。

3.6 单纯顶点删除归约。如果一个顶点 v 的闭邻域构成一个团 C_v，则称其为单纯形 顶点，即 $\forall x_1, x_2 \in N[v], \{x_1, x_2\} \in E$。在应用最大权团求解器之前，也可以删除单纯形顶点：一旦确定了一个单纯形v，它可能属于的最大团为 C_v，其 $w(C_v) = w(N[v])$。如果该权重高于当前已知的最高权重团，则更新下界。

Reduction Rule 7. 设 $v \in V$ 为一个单纯形顶点$\hat{}$，设 C 为目前找到的最佳团。仅当$\hat{}$ $\hat{}w(N[v]) > w(C)$时，设置 $C = N[v]$。无论如何，从图中删除v 都不会降低最大解权重。

证明。 如果 $w(N[v]) \leq w(\hat{C})$， v 就不可能属于严格更优的解。否则，如果 $w(N[v]) > w(C)$，在当前最佳解更新为 $\hat{} C = N[v]$ 后，同样的结论也成立。□

在 $N[v]$ 中测试每对顶点是否邻接，最坏情况下需要 $\mathcal{O}(\deg(v)^2)$ 。

算法1 通过数据归约简化图 G，并改进团 C；如果设置了lim，则使用顶点度限制。

Reduce($G = (V, E, w)$, $\hat{C}$, lim) 如果 lim，则将顶点度上限设为 0.1Δ初始化 D_i 对于每条归约规则 r_i重复执行所有归约 r_i 如果 r_i 未暂停且 $D_i \neq \emptyset$，则对 D_i ▷ 中的所有顶点应用 r_i，参见第 3.7 节 更新 D_i如果未达到约简率，则暂停 $r_i$$\hat{}$通过局部搜索更新 C，参见第 4.1 节 否则，如果 r_i 已暂停且 $\wedge G$ 已充分约简，则取消暂停 r_i ▷，参见第3.7节 如果 lim 且所有归约均已暂停，则提高顶点度上限，并更新 D_i，直到所有归约均暂停且度数不受限制 $\hat{}$返回 G, C

请注意，与其他归约不同，单纯形顶点归约可能会直接改进当前的最佳解 C。

3.7 应用归约。为应用本节提出的精确归约规则，我们采用了 Hespe等人 [22]策略的一个改编版本，其中包含依赖检查和归约跟踪。具体而言，归约集合 $\{r_i\}$

会被反复遍历，其中每条规则 r_i 都会应用于其可行顶点集合 D_i，该集合初始设为 $D_i = V$。经过初步实验，我们确定了以下归约顺序：邻域权重、孪生顶点、单纯形顶点、边界定（包括最大权重邻居）、支配情形1和支配情形2。每当某条规则 r_i 无法归约 一个顶点，即无法将其从图中移除时，该顶点便会从可行候选集合 D_i 中移除。否则，每条规则 r_j 的集合会更新为 $D_j = D_j \cup N(v)$，并将适用的顶点或边从图中移除。这样可以减少冗余计算，同时不影响约简图 [22]的最终规模。

归约跟踪旨在跟踪归约的有效性。与原始策略略有不同，归约跟踪会在某项归约未能达到每秒至少约简当前顶点数或边数 1% 的约简率时暂停该归约，直至其他归约将图约简至相同程度。归约跟踪不仅会在不同归约规则的应用之间进行检查，也会在遍历候选顶点期间定期检查，以防止单项归约拖慢求解器，并让效率更高的归约或精确求解器接手。Hespe等人策略的另一项改进，是对尝试应用归约的顶点度数设置动态上限。初始上限设为最高度数的 10%，每当上一层已穷尽应用所有归约时，上限便增加 10%。这确保优先应用适用于低度数顶点的归约，因为这类归约通常效率更高。当度数不再受限且所有归约均已暂停时，循环终止；此时，我们会在约简图上运行精确求解器或启发式求解器。

4 MWCRedu：一种新的精确算法

我们的精确算法 MWCRedu 分两个阶段运行：首先，使用第 3 节中的精确约简规则集对图进行约简。其次，将约简图交由精确分支定界求解器，以计算最终解。

4.1 计算下界。归约规则1、2和6的应用依赖于当前最优解 C 。计算界时通常优先采用快速启发式方法，因为投入更多时间改进初始解通常会带来递减的收益。Jiang 等人 [24]采用的方法很适合用于计算初始下界：反复从图中移除顶点度最小的顶点，直到剩余顶点两两相邻并构成初始团 C，由此得到初始下界 $w(C)$。

之后，通过单纯形顶点归约（归约规则7）和 FastWCLq 的局部搜索算法 [8], ，不断改进 C ；后者在检查各归约规则之间应用于约简图。随后， C 为归约规则1、2和6提供下界，同时也作为应用于约简图的求解器的初始解。算法1给出了概述。

4.2 分支定界。使用分支定界范式求解约简图。由于该过程的时间复杂度呈指数分布，因此，选择良好的排序并通过计算紧上界来缩小分支顶点集合至关重要。我们采用与 Jiang 等人相同的排序，即顶点按 $v_1 < v_2 < ... < v_n$ 排序，其中 v_1 的顶点度最小，移除 v_1 后 v_2 的顶点度最小，依此类推。这种排序称为图的退化排序。

为计算紧上界并缩小分支顶点集合，我们在整个搜索过程中采用高效的基于最大独立集和最大可满足性的 [24, 23] 方法。回顾第 2.2 节，对于将 V 划分为颜色类 $\mathcal{D} = D_1 \sqcup D_2 \sqcup \cdots \sqcup D_k$ 的任意顶点着色，每个颜色类都构成一个独立集，并且 $ub(\mathcal{D}) = \sum_{j=1}^{k} w^*(D_j)$ 是最大团权重的上界。随后，通过 TSM-MWC [23] 中的两阶段 MaxSAT 推理方法进一步缩小分支顶点集合。

在第一阶段（作者称之为二元 MaxSAT 推理）中，通过尽可能多地将顶点加入独立集，使分支顶点集合缩小，条件为 $\sum_{j=1}^{k'} w^*(D_j) \leq w(\hat{C})$。由于这些顶点本身无法组成权重超过 $w(C)$ 的团，因此可以将它们从分支顶点集合中移除。如果顶点 $v_i \in V$ 在所有现有独立集中都有邻居，但 上界 $+ w(v_i) \leq w(\hat{C})$ 成立，则将其作为一个新的独立集加入。否则，我们尝试将其权重拆分到不包含其任何邻居的独立集中：将权重为 $w^*(S_j)$ 的 v_i 加入独立集 S_j，并将权重更新为 $w(v_i) = w(v_i) - w^*(S_j)$，其中 $j = 1, 2, ..., k'$；重复此过程，直到其剩余权重为 $\delta = w(v_i) - \sum_{j=1}^{k'} w^*(S_j)$。如果 $\delta > 0$ 且 上界 $+ \delta \leq w(\hat{C})$，则将 v_i 作为权重为 δ 的新独立集加入；否则撤销权重拆分过程，并将 v_i 保留在分支顶点集合中。

在第二阶段，即有序最大可满足性推理中，通过检测独立集的互不相交冲突子集，进一步缩减分支顶点集合。首先，再次将分支顶点 v_i 的权重分摊到不包含其任何邻居的独立集 $\{S_1, S_2, ..., S_{k'}\}$ 上，从而得到剩余权重——

剩余权重 $w(v_i) = \delta > 0$，因为该顶点在第一阶段并未从分支顶点集合中移除。之后，算法尝试找到一组独立集 $\{U_1, U_2, ..., U_r\}$，其中每个独立集都恰好包含 v_i 的一个邻居 u。接着，它寻找一个独立集 D_q，使得对于任意 U_j，$D_q \cap N(v_i) \cap N(u) = \emptyset$，从而证明这些集合 $\{\{v_i\}, U_j, D_q\}$ 彼此冲突。在这种情况下，上界 可以进一步改进为 上界 $+ \delta - \beta$，其中 $\beta = \min(\delta, w^*(U_j) w^*(D_q))$ [23]。

最后，如果考虑所有 $U_j \in \{U_1, U_2, ..., U_r\} ub$ 后仍高于下界，则通过单元传播识别冲突子集来降低ub；这种方法最初用于最大权团 [15]。单元传播基于这样一种思想：包含更多文字的子句更有可能得到满足，因此被视为较弱的子句。因此，单元子句是最强的子句，因为它只有一种可能的真值赋值。算法会反复满足此类子句，并从其他子句中删除其中所含文字的所有出现。如果剩下一个空子句，就可判定这组子句存在冲突。每当识别出一组冲突子句 $\{S_0, S_1, ..., S_r\}$时，上界就可以减少 $\delta = \min\{w^*(S_1) \ldots, w^*(S_r)\}$。为了进一步收紧上界，将每个 $S_j (0 \le j \le r)$拆分为 S'_j 和 S''_j，使得 $w^*(S'_j) = \delta$ 且 $w^*(S''_j) = w^*(S_j) - \delta$。 S'_j 于是表示目前发现的冲突子集，而进一步的冲突则可以从 S''_j [15]推导出来。

该过程会在求解器的每个分支中运行，以减少需要完成的工作量。当所有分支都已探索或剪枝，或者达到时间限制时，算法便会终止；在这种情况下，将报告找到的最佳解。

5 MWCPeel：一种新的启发式算法

对于我们新提出的启发式算法 MWCPeel，我们研究了顶点剥离技术，即根据某种启发式规则，将分数最低的顶点从图中移除。因此，该规则必须尽可能准确地反映某个顶点属于解的可能性。对于 MCC，使用顶点度是显而易见的选择，因为度较高的顶点更有可能构成大团。此外，顶点 v 所属的团不可能大于 $\deg(v)$。为了使该指标在 MWC 的情境下仍然保持为上界，我们将每个顶点邻域的权重纳入考虑。剥离步骤中使用了由此得到的简单直观的评分指标 $w(N[v])$。

总体而言，我们的启发式求解器与第4节所述的精确方法 MWCRedu 类似，但在前文介绍的精确约简基础上实现了剥离约简：我们首先穷尽地运行精确约简。在约简图上，

算法2 启发式求解器 MWCPeel

```
MWCPeel(G = (V, E, w))
  计算初始团 Ĉ                                   ▷ 第4.1节
  重复                                           ▷ 算法1
    G, C ← Reduce(G, Ĉ, isFirstIteration)
    N ← 个待剥离的顶点                            ▷ 第5.1节
    移除 N 个分数最低的顶点 v  w(N[v])
  直到 满足停止准则                              ▷ 第5.2节
  返回 TSM-MWC(G, ^C)
```

我们应用剥离策略，移除不太可能属于大团的顶点。我们重复这一过程，直到剩余图足够小，或剥离约简的分数不够高，然后在剩余图上应用精确算法。算法2概述了这一过程。

5.1 剥离策略。Chang等人 [11] 提出了一种约简与剥离启发式技术，通过反复从图中移除最小度顶点，并将其加入不断扩大的独立集中。对于最大权团，一种直接的方法是移除分数最低的顶点，并将其排除在解之外。更具体地说，我们在每个剥离步骤中移除当前剩余顶点中的固定百分比。每一步要剥离的顶点数量 $\mathcal{N}$ 按如下方式动态确定：

$$\mathcal{N} = \begin{cases} 0.1n & \text{if } n > 50{,}000, \\ \max\{0.01n, \frac{0.01}{50{,}000} n\} & \text{otherwise,} \end{cases}$$

其中， n 始终表示当前顶点数量，而在初步实验中，50,000这一阈值已被证明是合适的。如果不区分较大图和较小图，精确约简往往会在许多顶点上重复应用，从而显著降低求解器的速度。此外，由于真实世界图实例中的顶点度通常遵循幂律分布， [21], 对于较大的图，最优解的规模占图的比例较小。每次剥离步骤后，都会更新可行候选集，并再次应用精确约简。

5.2 停止准则。 另一个重要决策是何时停止应用剥离约简；过早停止可能导致在约简图上应用的求解器运行时间大幅增加，而过晚停止则可能对解质量产生负面影响。由于需要约简的最优顶点数量高度依赖于图结构，静态停止准则不太可能是一个好的策略。因此，我们采用了一种动态策略，该策略

通过将当前计算出的分数与之前计算出的分数进行比较来运行。

第一个停止准则是最大分值相对于总最大分值下降到某一阈值以下。这表明剥离约简开始削减最大解。

如果最低分与最高分之间的差值缩小到某一阈值以下，则触发第二个停止准则。这表明评分模型已无法清晰区分高质量顶点和低质量顶点。

我们将两个阈值都设为 90 %，以在加速与解质量之间取得良好平衡。作为故障保护措施，在应用启发式约简之前会备份当前图状态；如果图被约简至空，则可以重新加载该备份。完成约简过程后，在约简图上运行分支定界求解器，以获得最终结果。

6 实验评估

我们实现了新的求解器 MWCRedu 和 MWCPeel，并在大量不同的实例上，将它们与各自类别中最先进的求解器进行评估。具体而言，我们在每个数据集上将精确求解器 MWCRedu 与目前最佳的精确求解器 TSM-MWC 进行比较，并将启发式求解器 MWCPeel 与目前最佳的启发式求解器 FastWCLq 和 SCCWalk4l 进行比较。

方法。 实验在配备 16 核 CPU、运行频率为 2.10GHz 的英特尔至强 Silver 4216 处理器上进行，操作系统为 Linux，内存为 95 GB。所有求解器均使用 C/C++ 实现，并通过 GNU g++编译，且启用了完整优化（-O3）。每个求解器最多并行处理 16 个图实例。由于求解器独占该机器运行，因此并行求解与顺序求解图实例并无显著差异。ˆ 我们始终报告解质量 $w(C)$以及找到该解所用的时间 t_{sol}。对于精确求解器，我们还会给出证明该解最优所需的时间 t_{prv}。使用随机数生成的求解器会采用不同种子运行五次，并报告其解的平均值，以更好地反映其整体表现。如果精确算法在 3,600 秒的时间限制内未能完成，则会停止运行，并输出目前找到的最佳解。启发式算法在运行 1,000 秒后停止。

实例。 我们在多种图上评估算法，这些图涵盖不同的规模、密度、赋权方式和应用领域。其中一些

图最初是无权的，因此人为赋予了权重。对于每个无权图，权重

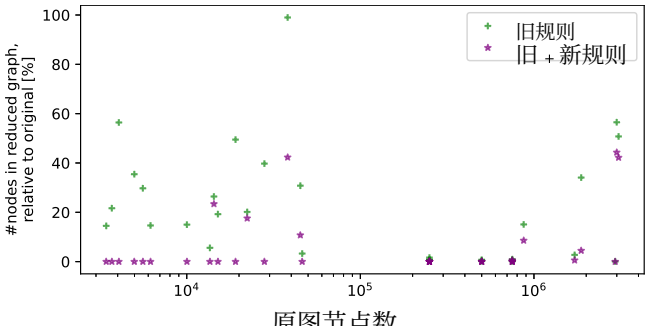

图3：旧归约和旧 +新归约对应的原图规模与约简图规模。

均匀取自范围 $[1, 200]$.[1]

我们编制了58个实例：OSM包含Cai 等人 [6]，使用 Barth 等人 [3]的方法，从 OpenStreetMap 数据生成的12个自然加权地图标注实例。REP中的10个实例是来自网络数据仓库 [37], 的真实世界图，DIMACS中的23个实例则取自第二届 DIMACS 实现挑战赛 [26]。此外，我们使用了13个随机双曲图（RHG）。这些图是随机生成的，其顶点度遵循幂律分布 [34] ，由 KaGen 框架 [16]生成。我们将幂律指数设定在1.75至2.25之间，并将平均度设定在100至500之间。对于REP、DIMACS和RHG，我们按照上述说明分配了人工权重。各实例的详细统计数据见附录 D中的表8。

6.1 新数据约简规则的影响。 我们首先研究约简规则对各实例的影响，并将加入我们提出的“新”规则后所产生的效果，与现有文献中描述的“旧”规则进行比较。表3展示了所有实例的约简图大小，表1则展示了部分实例的约简图大小。

对于 DIMACS 实例，新数据约简规则并未帮助生成规模更小的约简图（因此未列入表中）。这在意料之中，因为这些实例较为稠密，而数据约简规则往往在稀疏实例上效果更好。对于其他实例，额外采用新数据约简规则后，约简图的规模显著缩小。

REP 实例集中的最大归约出现在 web-wikipedia link it 上，新归约规则将其约简图缩减为空图，也就是说，仅通过归约便完全解决了该实例。

[1]其他权重分布（如幂律分布和指数分布）得到了类似的结果，因篇幅限制而未列出。

	约简图大小			
	旧+新归约		仅旧约简	
图	绝对值	n_0的百分比	绝对值	n_0的百分比
	REP			
bio-human-gene1	**3,915**	**17.57**	4,485	20.13
sc-TSOPF-RS-b2383	**16,123**	**42.29**	37,737	98.99
soc-orkut	**1,264,963**	**42.21**	1,521,404	50.76
web-wikipedia_link_it	**0**	**0.00**	1,214	0.04
维基百科增长数据集	**83,724**	**4.48**	637,483	34.08
	RHG			
RHG_250k_100_1.75	**7**	**0.00**	1,061	0.42
RHG_500k_500_2.25	**0**	**0.00**	1,761	0.35
RHG_750k_250_2.25	**15**	**0.00**	1,062	0.14
rhg_750k_500_1.75	**4,445**	**0.59**	7,341	0.98
rhg_750k_500_2.25	**12**	**0.00**	2,651	0.35
	OSM			
哥伦比亚特区-AM2	**0**	**0.00**	759	5.58
格陵兰-AM3	**0**	**0.00**	1,768	35.46
爱达荷-AM3	**0**	**0.00**	2,293	56.42
马萨诸塞州-AM3	**0**	**0.00**	802	21.66
弗吉尼亚州-AM3	**0**	**0.00**	907	14.66

表1：应用新旧数据归约规则时所选实例及约简图规模（节点数），以及仅运行现有文献中的归约规则时得到的约简图规模。数值越小越好。 n_0 表示初始节点数。

最大的改进出现在 sc-TSOPF-RS-b2383 上，旧规则几乎不起作用，仅将节点数减少了约 1%。然而，结合新规则后，计算得到的约简图仅包含原始实例中 42.29 % 的节点。对于所有实例，在旧规则的基础上使用新规则，得到的约简图始终比单独使用旧规则时更小。平均而言，旧规则单独使用时将图的规模缩减了约 67%；与新规则结合后，这一比例提升到了 80% 以上。

新规则在RHG实例上同样表现出色，并且始终比仅使用旧规则生成更小的约简图。总体而言，归约在这些实例上非常高效。仅使用旧规则时，所得约简图的规模会缩减至原图规模的0.14 %至1.64 %。结合使用新规则后，这一范围为0%至0.59 %。在新旧规则结合使用时，有两个RHG实例被约简为零节点。平均而言，通过新旧规则共同得到的约简图仅为原图规模的0.05 %，而平均仅使用旧规则时，规模是其十倍以上。

新旧规则结合使用时，在所有OSM实例上都计算出了空的约简图；而仅使用文献中的规则时，从未出现过这种情况。平均而言，旧规则将顶点数量缩减至25.4%，其范围相对较大，从哥伦比亚特区-AM2上的5.58%到爱达荷-AM3上的56.42%不等。

总而言之，我们的新约简规则在所有 REP、RHG 和 OSM 实例上都能稳定且显著地生成更小的约简图，甚至在 15 个实例上将约简图化为空，而仅使用旧规则在我们收集的任何实例上都未能做到这一点。图3以直观方式总结了这一结果。

6.2 精确算法。我们讨论四个实例集各自的汇总结果（见表2）。

在 OSM 实例上（表 4），我们的算法 MWCRedu 在几何平均数上比 TSM-MWC 快一个数量级以上，无论是找到解的时间 t_{sol} 还是证明最优性t_{prv}。在该集合的十二个实例中，它也始终比 TSM-MWC 快。由于两者都是精确

实例集	t_{sol}		t_{prv}		$w(\hat{C})$	
	TSM-MWC	MWCRedu	TSM-MWC	MWCRedu	TSM-MWC	MWCRedu
	精确结果					
DIMACS	1,106.99	**946.34**	1,714.98	**1,650.88**	6,460	**6,587**
REP	**117.15**	134.60	**190.45**	259.14	14,092	**14,321**
RHG	95.93	**14.55**	128.16	**17.73**	106,210	**106,781**
OSM	27.62	**1.55**	31.01	**2.43**	537,149	**542,993**
	136.15	**41.14**	189.82	**65.55**	47,738	**48,359**

实例集	t_{sol}			$w(\hat{C})$		
	FastWCLq	SCCWalk4l	MWCPeel	FastWCLq	SCCWalk4l	MWCPeel
	启发式结果					
DIMACS	193.11	**4.46**	91.31	6,792	**6,968**	6,547
REP	190.36	345.88	**52.12**	**14,190**	9,382	14,056
RHG	40.36	513.37	**10.69**	**106,781**	63,647	106,731
OSM	4.45	64.29	**1.45**	**542,993**	528,956	**542,993**
	50.68	84.49	**16.49**	**48,620**	38,516	48,056

表 2：各图集上精确算法（上）和启发式算法（下）的结果概览，以几何平均数表示。

算法，除两种情况外，它们得到的解的权重都相同；在这两种情况下，TSM-MWC 未能在时间限制内找到最优解，便提前停止并给出了较差的结果。因此，MWCRedu 在此占据优势。

在 DIMACS（表 4）上，两个求解器的性能没有明显差异。MWCRedu 在时间限制内完成了 23 个实例中的 9 个，而 TSM-MWC 仅完成了 8 个实例。两者的运行时间通常非常接近，除 7 个情况外，解的权重都相同。之所以表现相似，是因为 MWCRedu 使用的新增精确约简无法从该图集中的任何实例移除顶点或边。因此，求解器很快便转而调用分支定界求解器，而后者使用的技术与 TSM-MWC 相同。应用约简规则所带来的开销，仅在较简单的实例上较为明显。在两者都能正常完成的实例中，MWCRedu 的平均表现略好，这可能是因为在约简阶段运行局部搜索获得了更好的初始解。

在 REP 实例（表 5）上的结果各有胜负。对于某些实例，MWCRedu 和 TSM-MWC 分别优于对方。在三个实例上，TSM-MWC 未能证明解的最优性，其中两次以次优结果终止。对于顶点数超过 1000000 的大型实例，TSM-MWC 的效率很高；而在规模较小、更加稠密的生物学图上，MWCRedu 则优于 TSM-MWC。

在 RHG（表 5）上，MWCRedu 明显优于其竞争对手 TSM-MWC。TSM-MWC 两次超时，并以次优解终止；而 MWCRedu 总能正常完成，且除一个实例外，都是运行速度更快的算法。其运行时间上的优势十分显著，最多可达两个数量级。MWCRedu 表现出色的原因可能在于这些实例的结构，使其能够通过高效的归约快速移除大多数顶点。

总之，在 OSM 和 RHG 实例上，MWCRedu 显然是更优的算法；在 DIMACS图上，其表现与 TSM-MWC 不相上下。对于规模较小且稠密的实例（例如 REP 集中的实例），TSM-MWC 可能是速度更快的算法；而 MWCRedu 则能在规模非常大的实例上发挥优势。值得注意的是，MWCRedu 在与 TSM-MWC 相同的实例以及更多实例上，均在时间限制内完成，因此是一个更可靠的候选算法。

6.3 启发式算法。现在，我们将启发式求解器 MWCPeel 与最先进的求解器 FastWCLq 和 SCCWalk4l 进行比较，并讨论它们在四个实例集上的差异。汇总结果见表 2。

如表6所示，在 12 个 OSM实例中，MWCPeel 有 11 个表现最佳。MWCPeel 和 FastWCLq 均为所有实例找到了最优解。

对于DIMACS图（表6），SCCWalk4l明显优于其竞争算法。在FastWCLq和MWCPeel之间，FastWCLq计算出的权重解大多略高一些，

不过计算这些解所需的时间更长。对于TSM-MWC未能找到最优解的实例，FastWCLq和MWCPeel在其中大多数实例上都能以短得多的时间获得更高权重的解。

如表 7 所示，启发式求解器在 REP图上的表现非常有竞争力。尽管所有算法得到最佳解的次数大致相同，但 SCCWalk4l 的平均解质量最低。综合考虑速度，MWCPeel 的表现相当不错。平均而言，MWCPeel 的速度是第二快算法 FastWCLq 的 3.7 倍；而 FastWCLq 得到的解平均比 MWCPeel 好 0.9%。不过值得注意的是，我们的精确求解器 MWCPeel 得到的解权重甚至高于 FastWCLq，而且平均速度也更快。

RHG 的结果见表 7。在 39 个实例中，MWCPeel 有 31 个实例的表现优于其他求解器。FastWCLq 有时能找到权重略高于 MWCPeel 的解，但其平均运行时间更长（相差 3.8 倍）。无论速度还是解质量，SCCWalk4l 都明显逊色。

7 结论

我们提出了一个名为 MWCRedu 的精确算法和一个名为 MWCPeel 的启发式算法，用于求解最大权团问题。我们的算法将相关研究中的有效技术与新颖的数据约简规则交替结合起来，利用局部图结构识别并移除顶点和边，同时保留最优解。在大量不同图上的实验表明，对于大多数输入，这些算法的表现优于当前最先进的求解器。尤其是，在自然加权的中等规模街道网络图和随机双曲图上，MWCRedu 的速度快了几个数量级。MWCPeel 在这些实例上的表现优于竞争算法，但在极其稠密或规模很大的实例上效果稍逊。未来，我们希望研究这些方法的并行化。鉴于这些算法取得的良好结果，我们计划将其作为开源软件发布。

致谢。我们感谢德国研究联合会（DFG）项目 SCHU 2567/3-1 的资助。N.K. 获得了维也纳科学与技术基金（WWTF）项目 VRG19-009 的支持。

参考文献

[1]Faisal N. Abu-Khzam、Sebastian Lamm、Matthias Mnich、Alexander Noe、Christian Schulz 和 Darren Strash。《实用数据约简的最新进展》。载于 Hannah Bast、Claudius Korzen、Ulrich Meyer 及 Manuel Penschuck，编者，*Algorithms for Big Data: DFG Priority Program 1736* 97 133 ，页码 – 。施普林格自然瑞士，Cham，2022。[2] T. Akiba 和 Y. I wata。实践中的分支与约简指数/FPT 算法：顶点覆盖案例研究。*Theor. Comput. Sci.*，609，第 1 部分：211–225，2016。[3] Lukas Barth、Benjamin Niedermann、Martin N¨ ollenburg 和 Darren Strash。时序地图标注：一种全新的统一框架及其实验。在 Siva Ravada、Mohammed Eunus Ali、Shawn D. Newsam、Matthias Renz 和 Goce Trajcevski 编，*Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2016, Burlingame, California, USA, October 31 - November 3*，2016，页码 23:1–23:10。ACM，2016。[4] Sergiy Butenko 和 Wilbert E Wilhelm。计算生物化学与基因组学中的团检测模型。*European Journal of Operational Research*，173 1 1 17，2006 ( ): – 。[5] Shaowei Cai。复杂度与质量之间的平衡：大规模图上的最小顶点覆盖局部搜索。载于 *Twenty-Fourth International Joint Conference on Artificial Intelligence* 2015 ，。[6] Shaowei Cai、Wenying Hou、Jinkun Lin 和 Yuanjie Li。通过动态策略改进最小权顶点覆盖的局部搜索。载于 *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI-18* 1412 1418 ，，页码 – 。国际人工智能联合会议组织，2018 年 7 月。[7] Shaowei Cai 和 Jinkun Lin。快速求解大规模图上的最大权团问题。载于 *IJCAI*，页码 568–574，2016。[8] Shaowei Cai、Jinkun Lin、Yiyuan Wang 和 Darren Strash。一种快速计算大型稀疏图中最大权团的半精确算法。*Journal of Artificial Intelligence Research* 72 39 67, 2021 ，：– 。[9] Randy Carraghan 和 Panos M Pardalos。最大团问题的精确算法。*Operations Research Letters* 9 6 375 382, 1990 ，( ): – 。[10] Lijun Chang。在大型稀疏图上高效计算和枚举最大团。*VLDB J.*，29 5 999 1022，2020 ( ): – 。[11] Lijun Chang、Wei Li 和 Wenjie Zhang。通过约简剥离在线性时间内计算近最大独立集。载于 *Proceedings of the 2017 ACM International Conference on Management of Data*，页码 1181–1196，2017。[12] Alessio Conte、Donatella Firmani、Maurizio Patrignani 和 Riccardo Torlone。一种用于寻找大型 k *Knowl. Inf. Syst.* 63 7 1745 1769, - 2021 plex 的元算法。，( ): – 。[13] Marek Cygan、Fedor V. Fomin、Lukasz Kowalik、Daniel Lokshtanov、D´aniel Marx、Marcin Pilipczuk、Michal Pilipczuk 和 Saket Saurabh。*Parameterized Algorithms* 2015 。施普林格，。[14] R. Erhardt。加权

最大团问题。硕士论文，海德堡大学，2022。[15] Zhi wen Fang、Chu-Min Li 和 Ke Xu。一种基于最大可满足性推理的最大权团问题精确算法。*Journal of Artificial Intelligence Research* 55 799 833, 2016 ，：–。

[16] Daniel Funke、Sebastian Lamm、Ulrich Meyer、Manuel Penschuck、Peter Sanders、Christian Schulz、Darren Strash 和 Moritz von Looz。无通信的大规模分布式图生成。 *Journal of Parallel and Distributed Computing* 131 200 217, ，：–。2019

[17] Alexander Gellner、Sebastian Lamm、Christian Schulz、Darren Strash 和 Bogd´an Zav´alnij。利用递增变换改进最大权独立集问题的数据约简。载于 *2021 Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)* 128 142 ，页码 –。工业与应用数学学会，2021。[18] Michel Gendreau、Patrick Soriano 和 Louis Salvail。使用禁忌搜索方法求解最大团问题。 *Annals of operations research*，41 4 385 403, 1993 ( ): –。[19] Jens Gramm、Jiong Guo、Falk H¨uffner 和 Rolf Niedermeier。团覆盖的数据约简与精确算法。 *ACM J. Exp. Algorithmics*，13，2009 年 2 月。[20] Jiewei Gu、Weiguo Zheng、Yuzheng Cai 和 Peng Peng。面向大规模图中近最大加权独立集的计算。载于 *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*，KDD '21，页码 467—-477，美国纽约，NY，2021。计算机协会。[21] Luca Gugelmann、Konstantinos Panagiotou 和 Ueli Peter。随机双曲图：度序列与聚类。载于 *International Colloquium on Automata Languages, and Programming*，页码 –。施普林格，2012。[22] Demian Hespe、Christian Schulz 和 Darren Strash。最大独立集的可扩展核化。 *Journal of Experimental Algorithmics (JEA)* 24 1 22, ，：–。2019

[23] Hua Jiang、Chu-Min Li、Yanli Liu 和 Felip Manya。一种用于最大权团问题的两阶段 MaxSAT 推理方法。载于 *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 2018 ，卷。

[24] Hua Jiang、Chu-Min Li 和 Felip Manya。大型图中最大权团问题的精确算法。载于 *Proceedings of the AAAI conference on artificial intelligence* 31, 2017 ，卷。

[25] Hua Jiang、Dongming Zhu、Zhichao Xie、Shaowen Yao 和 Zhang-Hua Fu。一种基于顶点划分的最大 k-plex 问题新上界。Zhi-Hua Zhou 编， *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21* 1689 1696 ，页码 –。国际人工智能联合会议组织，2021 年 8 月。主会场。

[26] David S Johnson 和 Michael A Trick. *Cliques, coloring and satisfiability: second DIMACS implementation challenge October 11-13 1993* 26 ，，，，第 . 卷。美国数学学会，1996 年。[27] Deniss Kumlander. 一种基于启发式顶点着色和回溯搜索的最大权团问题新型精确算法。载于 *Proc., 5th Intl Conf on Modelling, Computation and Optimization in Information Systems and Management Sci. ences* 202 208, 2004 ，第 –页。[28] Sebastian Lamm、Peter Sanders、Christian Schulz、Darren Strash 和 Renato F. Werneck. 大规模寻找近最优独立集。 *J. Heuristics*，23 卷 4 期，207–229 页，2017 年（）：–。[29] Sebastian Lamm、Christian Schulz、Darren Strash、Robert Williger 和 Huashuo Zhang. 在大型真实世界图上精确求解最大权独立集问题。载于 *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)* 144 158 2019 ，第 –页。工业与应用数学学会，。[30] Chu-Min Li、Hua Jiang 和 Felip Many`a. 关于最大团问题分支定界算法中分支数量的最小化。*Computers & Operations Research*，84：1–15，2017 年。[31] Chu-Min Li 和 Zhe Quan. 一种基于最大可满足性的高效分支定界算法，用于求解最大团问题。载于 *Twenty-fourth AAAI conference on artificial intelligence* 2010 ，。

[32] Jinkun Lin、Shaowei Cai、Chuan Luo 和 Kaile Su. 一种用于超大图着色的基于归约的方法。载于 *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*，第 517–523 页，2017 年。[33] Ingo Muegge 和 Matthias Rarey. 小分子对接与评分。 *Reviews in computational chemistry* 17 1 60, 2001 ，：–。

[34] Manuel Penschuck、Ulrik Brandes、Michael Hamann、Sebastian Lamm、Ulrich Meyer、Ilya Safro、Peter Sanders 和 Christian Schulz. 可扩展网络生成的最新进展。*arXiv:2003.00736* 2020 *arXiv preprint* ，。

[35] Wayne Pullan. 用于最大团问题的分阶段局部搜索。 *Journal of Combinatorial Optimization* 12 3 303 323, 2006 ，（）：–。

[36] Wayne Pullan. 用局部搜索近似求解最大顶点/边加权团。 *Journal of Heuristics* 14 2 117 134, 2008 ，（）：–。

[37] Ryan A. Rossi 和 Nesreen K. Ahmed. 网络数据仓库及其交互式图分析与可视化。载于 *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* 2015 ，。

[38] Pablo San Segundo、Fabio Furini、David ´Alvarez 和 Panos M. Pardalos. CliSAT：一种用于求解困难最大团问题的新型精确算法。 *European Journal of Operational Research* 2022 ，。

[39] Darren Strash. 简单归约对最大独立集问题的作用。载于 Thang N. Dinh 和 My T. Thai（编）， *Computing and Com-*

binatorics - 22nd International Conference, COCOON 2016, *Ho Chi Minh City, Vietnam, August 2-4*, 2016, *Proceedings* 9797 *Lecture Notes in Computer*，第 卷，*Science* 345 356 2016 第 – 页。施普林格，。[40] Darren Strash 和 Louise Thompson. *Effective Data Reduction for the Vertex Clique Cover Problem*，第 41–53 页。工业与应用数学学会，2022 年。[41] Etsuji Tomita 和 Tomokazu Seki. 一种用于寻找最大团的高效分支定界算法。载于*International conference on discrete mathematics and theoretical computer science* 278 289 ，第 – 页。施普林格，2003 年。[42] Etsuji Tomita 、 Yoichi Sutani 、 Takanori Higashi、Shinya Takahashi 和 Mitsuo Wakatsuki. 一种简单而更快速的最大团分支定界算法。载于*International Workshop on Algorithms and Computation* 191 203 2010 ，第 – 页。施普林格，。[43] Anurag Verma、Austin Buchanan 和 Sergiy Butenko. 在超大稀疏网络上求解最大团和顶点着色问题。*INFORMS Journal on Computing* 27 1 164 177, 2015 ，（）：–。[44] Yiyuan Wang、Shaowei Cai、Jiejiang Chen 和 Minghao Yin. Sccwalk：一种高效的最大权团局部搜索算法及其改进。 *Artificial Intelligence*，280：103230，2020 年。[45] Yiyuan Wang、Shaowei Cai、Shiwei Pan、Ximing Li 和 Monghao Yin. 加权图着色问题的归约与局部搜索。*Proceedings of the AAAI Conference on Artificial Intelligence*，34 卷 03 期，2433–02441 页，2020 年（）：–，4 月。[46] Yiyuan Wang、Shaowei Cai 和 Minghao Yin. 两种高效的最大权团问题局部搜索算法。载于*Thirtieth AAAI Conference on Artificial Intelligence* 2016 ，。[47] Stanley Wasserman、Katherine Faust 等. 社会网络分析：方法与应用。1994 年。[48] Qinghua Wu 和 Jin-Kao Hao. 最大团问题算法综述。*European Journal of Operational Research* 242 3 693 709, 2015 ，（）：–。[49] Qinghua Wu 和 Jin-Kao Hao. 通过加权最大团启发式算法求解赢家确定问题。 *Expert Systems with Applications*，42(1)：355–365，2015 年。[50] Jingen Xiang、Cong Guo 和 Ashraf Aboulnaga. 使用 MapReduce 进行可扩展的最大团计算。载于*2013 IEEE 29th International Conference on Data Engineering (ICDE)* 74 85, 2013 ，第 – 页。[51] Dong Zhang、Omar Javed 和 Mubarak Shah. 通过受约束最大权团进行视频对象协同分割。载于 *European Conference on Computer Vision*，第 551–566 页。施普林格，2014 年。[52] Hootan Zhian、Masoud Sabaei、Nastooh Taheri Javan 和 Omid Tavallaie. 使用最大权团增加编码机会。载于 *2013 5th Computer Science and Electronic Engineering Conference (CEEC)* 168 173 2013 ，第 – 页。电气电子工程师学会，。[53] Patric R.J. ¨Osterg˚ard. 一种求解最大团问题的快速算法。 *Discrete Applied Mathematics*，120(1):197–207，2002年。专门 посвящ于第六届特温特图与组合优化研讨会的特刊。

A 约简图大小

	约简图大小			
	old+new 归约		仅旧约简	
图	绝对值	% 的 n_0	绝对值	% 的 n_0
	REP			
aff-digg	**74,864**	**8.58**	131，624	15.08
bio-human-gene1	**3,915**	**17.57**	4，485	20.13
生物人类基因2	**3,353**	**23.38**	3，787	26.41
小鼠生物基因	**4,840**	**10.73**	13，888	30.79
sc-TSOPF-RS-b2383	**16,123**	**42.29**	37,737	98.99
soc-flickr-und	**9,465**	**0.55**	46,728	2.72
soc-orkut	**1,264,963**	**42.21**	1,521,404	50.76
soc-orkut-dir	**1,360,796**	**44.29**	1,736,368	56.51
web-wikipedia_link_it	**0**	**0.00**	1，214	0.04
维基百科增长数据集	**83,724**	**4.48**	637，483	34.08
算术平均数		**19.41**		33.55
	RHG			
RHG_250k_100_1.75	**7**	**0.00**	1，061	0.42
RHG_250k_100_2.25	**62**	**0.02**	782	0.31
RHG_250k_250_1.75	**11**	**0.00**	2，170	0.87
RHG_250k_250_2.25	**14**	**0.01**	983	0.39
RHG_250k_500_1.75	**32**	**0.01**	4，103	1.64
RHG_250k_500_2.25	**39**	**0.02**	1，250	0.50
RHG_500k_250_1.75	**2**	**0.00**	3，782	0.76
RHG_500k_250_2.25	**122**	**0.02**	1，138	0.23
RHG_500k_500_2.25	**0**	**0.00**	1，761	0.35
RHG_750k_250_1.75	**0**	**0.00**	4，365	0.58
RHG_750k_250_2.25	**15**	**0.00**	1，062	0.14
RHG_750k_500_1.75	**4,445**	**0.59**	7，341	0.98
RHG_750k_500_2.25	**12**	**0.00**	2，651	0.35
算术平均数		**0.05**		0.58
	OSM			
哥伦比亚特区-AM2	**0**	**0.00**	759	5.58
哥伦比亚特区-AM3	**0**	**0.00**	1,513	3.27
格陵兰-AM3	**0**	**0.00**	1，768	35.46
夏威夷-AM3	**0**	**0.00**	11,130	39.74
爱达荷-AM3	**0**	**0.00**	2,293	56.42
肯塔基-AM3	**0**	**0.00**	9，447	49.47
马萨诸塞州-AM3	**0**	**0.00**	802	21.66
俄勒冈-AM3	**0**	**0.00**	1，662	29.74
罗得岛州-AM3	**0**	**0.00**	2,913	19.26
佛蒙特州-AM3	**0**	**0.00**	498	14.49
弗吉尼亚州-AM3	**0**	**0.00**	907	14.66
washington-AM3	**0**	**0.00**	1，503	15.00
算术平均数		**0.00**		25.40

表3：同时应用新旧数据归约规则得到的约简图规模（节点数），与仅运行现有文献中的归约所得到的约简图规模。 n_0 表示初始节点数。越小越好。在 DIMACS 实例上，约简图规模没有变化。

B 精确算法的详细结果

图	t_{sol}		t_{prv}		$w(\hat{C})$	
	TSM-MWC	MWCRedu	TSM-MWC	MWCRedu	TSM-MWC	MWCRedu
	OSM精确结果					
district-of-col-AM2	0.80	**0.15**	0.88	**0.21**	**235,777**	**235,777**
哥伦比亚特区	1,937.83	**5.64**	1,938.06	**6.52**	**545,969**	**545,969**
格陵兰-AM3	10.31	**0.82**	11.91	**3.30**	**604，575**	**604，575**
夏威夷-AM3	3,598.87	**30.88**	-	**54.68**	1,110,978	**1,229,741**
爱达荷-AM3	218.66	**4.47**	220.42	**5.55**	**1,101,721**	**1,101,721**
肯塔基-AM3	3，580。19	**114.09**	-	**144.51**	1,808,419	**1，860，308**
马萨诸塞州-AM3	0.81	**0.03**	1.01	**0.13**	**115,636**	**115,636**
俄勒冈-AM3	8.93	**1.70**	11.07	**2.47**	**557，634**	**557，634**
罗得岛-AM3	81.30	**11.03**	93.69	**18.40**	**1,162,925**	**1,162,925**
佛蒙特州-AM3	4.57	**0.49**	4.90	**0.56**	**604,213**	**604,213**
弗吉尼亚州-AM3	0.13	**0.06**	0.21	**0.06**	**207，457**	**207，457**
washington-AM3	12.22	**1.01**	12.75	**1.06**	**356，314**	**356，314**
几何平均值	27.62	**1.55**	31.01	**2.43**	537,149	**542,993**

图	t_{sol}		t_{prv}		$w(\hat{C})$	
	TSM-MWC	MWCRedu	TSM-MWC	MWCRedu	TSM-MWC	MWCRedu
	DIMACS 精确结果					
brock800_1	**1,122.28**	1,191.32	**2,667.43**	2,820.98	**3,006**	**3,006**
brock800_2	**2,601.19**	2,752.57	**3,111.08**	3,285.93	**3,074**	**3,074**
brock800_3	**1,318.78**	1,400.10	**2,735.77**	2,896.13	**2,984**	**2,984**
brock800_4	**2,019.67**	2,137.40	-	-	**3,059**	**3,059**
C1000.9	2,889.65	**1,429.16**	-	-	7,338	**7,459**
C2000.5	**1,867.49**	1,908.40	-	-	**2,395**	**2,395**
C2000.9	2,248.93	**697.11**	-	-	7,898	**8,284**
C4000.5	2,804.98	**2,488.93**	-	-	**2,460**	2,437
C500.9	2,716.53	**2,611.48**	-	-	**6,789**	**6,789**
gen400_p0.9_55	**2，359。72**	2，378。30	-	-	**6，654**	**6，654**
gen400_p0.9_65	**2，768。34**	2,772.78	-	-	**6，535**	**6，535**
gen400_p0.9_75	1，420。53	**1,388.95**	2,522.75	**2，511。52**	**7，492**	**7，492**
汉明码10-4	3,110.88	**3,018.87**	-	-	**5，205**	5,125
约翰逊32-2-4	**1,983.31**	2,070.46	-	-	**2，935**	**2，935**
keller5	**2,801.07**	2,821.09	-	-	3,807	**3,827**
keller6	2,569.15	**2,301.13**	-	-	5,617	**6,175**
MANN_a27	2.67	**2.09**	2.81	**2.14**	**17,866**	**17,866**
MANN_a45	121.72	**59.03**	126.40	**63.31**	**49,459**	**49,459**
MANN_a81	3,599.91	**1,922.82**	-	**3,116.00**	118,898	**161,903**
p_hat1000-3	2，177。09	**2，068。98**	-	-	**8，261**	**8，261**
p_hat1500-2	952.79	**902.21**	2,491.91	**2，371。21**	**7，556**	**7，556**
p_hat1500-3	1，212。92	**1，111。24**	-	-	**10,796**	**10,796**
sanr400_0.7	**11.86**	12.69	**26.63**	28.43	**2,926**	**2,926**
几何平均值	1,106.99	**946.34**	1,714.98	**1,650.88**	6,460	**6,587**

表 4：各图的 OSM 和 DIMACS 精确结果及几何平均数。

图	t_{sol}		t_{prv}		$w(\hat{C})$	
	TSM-MWC	MWCRedu	TSM-MWC	MWCRedu	TSM-MWC	MWCRedu
	REP精确结果					
aff-digg	**16.55**	45.40	**244.04**	273.67	**3，829**	**3，829**
bio-human-gene1	3,598.98	**2，010。82**	-	**3,327.65**	136,325	**136,692**
生物人类基因2	2,298.36	**474.49**	-	**1，380。66**	**131，904**	**131，904**
小鼠生物基因	**1，789。70**	**198.47**	-	**240.97**	50，785	**59,476**
sc-TSOPF-RS-b2383	**9.70**	27.59	**9.70**	365.22	**913**	**913**
soc-flickr-und	**35.10**	133.73	**74.52**	162.42	**10,847**	**10,847**
soc-orkut	**56.42**	135.79	**67.48**	144.89	**5,832**	**5,832**
soc-orkut-dir	**46.26**	116.30	**60.22**	128.33	**5,261**	**5,261**
Web-Wiki 链接它	170.90	**45.99**	171.08	**46.15**	**87,175**	**87,175**
Web-Wiki增长	**13.09**	84.77	**17.30**	88.57	**3,334**	**3,334**
几何平均值	**117.15**	134.60	**190.45**	259.14	14,092	**14,321**
图	t_{sol}		t_{prv}		$w(\hat{C})$	
	TSM-MWC	MWCRedu	TSM-MWC	MWCRedu	TSM-MWC	MWCRedu
	RHG 精确结果					
rhg_25k_100_1.75	94.71	**2.26**	94.91	**2.72**	**99，839**	**99，839**
rhg_250k_100_2.25	2.53	**2.02**	2.75	**2.10**	**37，947**	**37,947**
RHG_250k_250_1.75	101.48	**4.74**	107.14	**4.76**	**112，769**	**112,769**
RHG_250k_250_2.25	21.87	**4.53**	22.60	**4.66**	**71，001**	**71，001**
RHG_250k_500_1.75	1,079.84	**35.47**	1,092.32	**38.43**	**137，234**	**137，234**
RHG_250k_500_2.25	51.02	**9.05**	52.12	**9.25**	**102，364**	**102，364**
RHG_500k_250_1.75	**3，093。33**	**37.09**	-	**37.70**	130,973	**131，559**
RHG_500k_250_2.25	88.77	**11.36**	89.79	**11.71**	**88,512**	**88,512**
RHG_500k_500_2.25	43.49	**15.11**	47.06	**21.09**	**122,781**	**122，781**
RHG_750k_250_1.75	**3，599.25**	**33.80**	-	**55.44**	150，676	**160，845**
RHG_750k_250_2.25	132.25	**18.15**	132.43	**18.33**	**96,362**	**96，362**
RHG_750k_500_1.75	**9.74**	311.73	**226.52**	1,173.96	**207，197**	**207，197**
rHG_750k_500_2.25	35.91	**34.13**	49.39	**34.19**	**119，936**	**119，936**
几何平均值	95.93	**14.55**	128.16	**17.73**	106,210	**106,781**

表 5：每个图的 REP 和 RHG 精确结果及几何平均数。

C 启发式算法的详细结果

图	t_{sol}			$w(\hat{C})$		
	FastWCLq	SCCWalk4l	MWCPeel	FastWCLq	SCCWalk4l	MWCPeel
			OSM启发式结果			
哥伦比亚特区-AM2	0.32	4.91	**0.16**	**235,777**	234,219	**235,777**
哥伦比亚特区-AM3	16.87	208.46	**5.26**	**545,969**	**545,969**	**545,969**
格陵兰-AM3	2.96	39.33	**0.85**	**604，575**	**604，575**	**604，575**
夏威夷-AM3	86.47	727.42	**29.82**	**1,229,741**	1,224,690	**1,229,741**
爱达荷-AM3	15.76	162.51	**4.29**	**1,101,721**	1,098,044	**1,101,721**
肯塔基-AM3	374.57	997.09	**102.17**	**1，860，308**	1,437,770	**1，860，308**
马萨诸塞州-AM3	0.25	50.59	**0.03**	**115,636**	113,381	**115,636**
俄勒冈-AM3	6.06	239.62	**1.72**	**557，634**	546,314	**557，634**
罗得岛-AM3	34.16	252.86	**9.34**	**1,162,925**	1,162,920	**1,162,925**
佛蒙特州-AM3	**0.37**	2.32	0.39	**604,213**	602,793	**604,213**
弗吉尼亚州-AM3	0.20	6.38	**0.05**	**207，457**	**207，457**	**207，457**
washington-AM3	1.88	23.11	**0.96**	**356,314**	**356，314**	**356，314**
几何平均值	4.45	64.29	**1.45**	**542,993**	528,956	**542,993**
图	t_{sol}			$w(\hat{C})$		
	FastWCLq	SCCWalk4l	MWCPeel	FastWCLq	SCCWalk4l	MWCPeel
			DIMACS 启发式结果			
brock800_1	150.98	**0.33**	45.67	3,000	**3,006**	2,886
brock800_2	163.22	73.82	**59.84**	3,024	**3,074**	2,935
brock800_3	127.04	**0.36**	37.69	**2,984**	**2,984**	2,912
brock800_4	230.35	98.56	**48.01**	3,007	**3,059**	2,887
C1000.9	409.16	**1.45**	262.85	8,693	**9,058**	7,779
C2000.5	404.03	**8.25**	536.78	2,426	**2,467**	2,390
C2000.9	351.96	**105.48**	123.18	9,822	**10,874**	8,603
C4000.5	375.71	**82.45**	638.30	2,580	**2,787**	2,472
C500.9	251.22	**0.28**	474.51	7,277	**7,313**	6,964
gen400_p0.9_55	169.17	**0.09**	50.16	**6,781**	**6,781**	6，614
gen400_p0.9_65	312.92	**0.28**	59.97	6，869	**6，881**	6，654
gen400_p0.9_75	104.98	84.15	**10.78**	7，547	**7，551**	7,261
汉明码10-4	287.81	**1.42**	527.12	5，727	**5,917**	5,279
约翰逊32-2-4	459.61	**0.06**	537.90	3，004	**3，042**	3，020
keller5	390.66	**2.14**	326.39	3,811	**3,851**	3,545
keller6	347.22	**216.10**	400.16	6,727	**8,412**	6,103
MANN_a27	16.84	168.23	**1.63**	**17,866**	17,864	17,710
MANN_a45	79.35	**43.89**	93.78	**49,459**	**49,459**	49,312
MANN_a81	180.29	376.13	**7.17**	**161,903**	161,895	161,648
p_hat1000-3	375.61	**0.51**	404.89	8，248	**8，295**	8，223
p_hat1500-2	423.40	**1.22**	475.25	7，519	**7，556**	7，546
p_hat1500-3	356.96	**4.08**	206.37	10,725	**10,926**	10，801
sanr400_0.7	6.94	**0.06**	0.91	**2,926**	**2,926**	2,874
	193.11	**4.46**	91.31	6,792	**6,968**	6,547

表6：各图的 OSM 和 DIMACS 启发式结果及几何平均数 .

图	t_{sol}			$w(\hat{C})$		
	FastWCLq	SCCWalk4l	MWCPeel	FastWCLq	SCCWalk4l	MWCPeel
			REP启发式结果			
aff-digg	240.01	**30.73**	47.68	3，514	**3,829**	**3，829**
bio-human-gene1	719.13	640.92	**493.36**	136，581	136，647	**136，713**
生物人类基因2	457.56	534.75	**89.59**	131，763	131，862	**131，904**
小鼠生物基因	593.17	412.13	**13.43**	59,439	**59，473**	59，146
sc-TSOPF-RS-b2383	33.02	243.56	**1.43**	**913**	900	870
soc-flickr-und	601.63	252.98	**44.40**	10,806	8,968	**10,847**
soc-orkut	**135.88**	526.16	183.51	**5,832**	4,552	5,582
soc-orkut-dir	**157.93**	521.46	185.02	**5,261**	4,080	5,116
web-wikipedia_link_it	71.06	972.97	**36.59**	**87，175**	2，903	**87，175**
维基百科增长数据集	**44.03**	343.25	66.06	**3,334**	2，960	3，136
几何平均值	190.36	345.88	**52.12**	**14,190**	9,382	14,056
图	t_{sol}			$w(\hat{C})$		
	FastWCLq	SCCWalk4l	MWCPeel	FastWCLq	SCCWalk4l	MWCPeel
			RHG启发式结果			
rhg_250k_100_1.75	10.56	134.23	**2.67**	**99，839**	**99，839**	**99，839**
RHG_250k_100_2.25	2.76	59.67	**2.09**	**37，947**	**37，947**	**37，947**
RHG_250k_250_1.75	42.01	501.91	**4.24**	**112,769**	112,074	112，756
RHG_250k_250_2.25	9.82	204.35	**4.58**	**71，001**	**71，001**	**71，001**
RHG_250k_500_1.75	125.03	869.09	**18.39**	**137，234**	86，773	136，884
RHG_250k_500_2.25	36.98	624.17	**10.15**	**102，364**	100,087	**102，364**
RHG_500k_250_1.75	90.36	854.58	**22.69**	**131，559**	67，352	131，100
RHG_500k_250_2.25	22.04	647.54	**11.40**	**88,512**	85,244	**88,512**
RHG_500k_500_2.25	71.83	995.47	**15.33**	**122,781**	46，201	**122,781**
RHG_750k_250_1.75	122.79	998.01	**22.72**	**160，845**	36,006	**160，845**
RHG_750k_250_2.25	28.23	687.98	**17.00**	**96，362**	95，558	**96，362**
RHG_750k_500_1.75	375.30	1,017.89	**21.41**	**207，197**	27，079	**207，197**
RHG_750k_500_2.25	72.82	1,002.84	**36.00**	**119，936**	43，470	**119，936**
几何平均值	40.36	513.37	**10.69**	**106,781**	63,647	106,731

表 7：REP 和 RHG 启发式结果在各图上的表现及几何平均数。

D 实例的详细属性

数据集	实例	$\|V\|$	$\|E\|$	ρ
DIMACS	brock800 1	800	207,505	6.49×10^{-1}
	brock800 2	800	208,166	6.51×10^{-1}
	brock800 3	800	207,333	6.49×10^{-1}
	brock800 4	800	207,643	6.50×10^{-1}
	C1000.9	1,000	450,079	9.01×10^{-1}
	C2000.5	2,000	999,836	5.00×10^{-1}
	C2000.9	2,000	1,799,532	9.00×10^{-1}
	C4000.5	4,000	4,000,268	5.00×10^{-1}
	C500.9	500	112,332	9.00×10^{-1}
	gen400 p0.9 55	400	71，820	9×10^{-1}
	gen400 p0.9 65	400	71，820	9×10^{-1}
	gen400 p0.9 75	400	71,820	9×10^{-1}
	汉明码10-4	1,024	434，176	8.29×10^{-1}
	约翰逊32-2-4	496	107，880	8.79×10^{-1}
	keller5	776	225,990	7.52×10^{-1}
	keller6	3,361	4,619,898	8.18×10^{-1}
	MANN_a27	378	70,551	9.90×10^{-1}
	MANN_a45	1,035	533,115	9.96×10^{-1}
	MANN_a81	3,321	5,506,380	9.99×10^{-1}
	p_hat1000-3	1,000	371,746	7.44×10^{-1}
	p_hat1500-2	1,500	568，960	5.06×10^{-1}
	p_hat1500-3	1，500	847，244	7.54×10^{-1}
	sanr400 0.7	400	55,869	7.00×10^{-1}
OSM	哥伦比亚特区-AM2	13,597	1,609,795	8.05×10^{-2}
	哥伦比亚特区-AM3	46,221	27,729,137	1.93×10^{-1}
	格陵兰-AM3	4，986	3,652,361	1.74×10^{-2}
	夏威夷-AM3	28,006	49,444,921	1.87×10^{-1}
	爱达荷-AM3	4,064	3,924,080	4.75×10^{-1}
	肯塔基-AM3	19,095	59,533,630	2.94×10^{-1}
	马萨诸塞州-AM3	3,703	551,491	1.10×10^{-1}
	俄勒冈-AM3	5，588	2,912,701	2.60×10^{-2}
	罗得岛-AM3	15,124	12,622,219	1.26×10^{-1}
	佛蒙特州-AM3	3,436	1,136,164	3.27×10^{-1}
	弗吉尼亚州-AM3	6,185	665，903	3.48×10^{-2}
	washington-AM3	10,022	2,346,213	4.67×10^{-2}
REP	aff-digg	872,622	22,501,699	5.91×10^{-5}
	bio-human-gene1	22,283	12,323,680	4.96×10^{-2}
	生物人类基因2	14，340	9,027,024	8.78×10^{-2}
	小鼠生物基因	45，101	14,461,095	1.42×10^{-2}
	sc-TSOPF-RS-b2383	38,121	16,115,324	2.22×10^{-2}
	soc-flickr-und	1,715,256	15,555,040	1.06×10^{-5}
	soc-orkut	3,072,442	117,185,082	2.48×10^{-5}
	soc-orkut-dir	2,997,167	106,349,208	2.37×10^{-5}
	web-wikipedia_link_it	2,936,414	86，754，663	2.01×10^{-5}
	维基百科增长数据集	1,870,710	36,532,530	2.09×10^{-5}
RHG	rhg_250k_100_1.75	250，000	7,755,473	2.48×10^{-4}
	RHG_250k_100_2.25	250，000	10,546,938	3.38×10^{-4}
	RHG_250k_250_1.75	250,000	17,828,988	5.71×10^{-4}
	RHG_250k_250_2.25	250,000	24,036,880	7.69×10^{-4}
	RHG_250k_500_1.75	250,000	35，161，098	1.13×10^{-3}
	RHG_250k_500_2.25	250,000	47，230，197	1.51×10^{-3}
	RHG_500k_250_1.75	500，000	35,493,799	2.84×10^{-4}
	RHG_500k_250_2.25	500，000	49,954,694	4.00×10^{-4}
	RHG_500k_500_2.25	500，000	92,901,492	7.43×10^{-4}
	RHG_750k_250_1.75	750，000	53,201,080	1.89×10^{-4}
	RHG_750k_250_2.25	750，000	73,667,026	2.62×10^{-4}
	RHG_750k_500_1.75	750,000	102,363,505	3.64×10^{-4}
	RHG_750k_500_2.25	750，000	139,633,569	4.96×10^{-4}

表8：实例属性