maximum clique problems

钟以楠

山东大学泰山学堂

2025 年 12 月 10 日

背景与意义

- 核心地位：最大团问题 (Maximum Clique Problem, MCP) 是计算复杂性理论中最基础的 NP-hard 问题之一，也是 Karp (1972) 提出的 21 个经典 NPC 问题之一。
- 广泛应用：
  - 生物信息学：蛋白质结构比对、基因序列分析。
  - 社交网络：社区发现、紧密群体识别。
  - 编码理论：纠错码设计、汉明距离约束。
- 挑战性：不仅精确求解困难，其近似求解也极具挑战性 (Inapproximability)。

问题定义 (Formal Definition)

定义 (最大团 Maximum Clique)

给定无向图 $G = (V, E)$，团 (Clique) 是顶点集 V 的一个子集 $C \subseteq V$，满足：

$$\forall u, v \in C, u \neq v \implies \{u, v\} \in E$$

即 C 中的任意两个顶点都互为邻接点。

关键区分

- 极大团 **(Maximal Clique)**：无法通过添加顶点扩展的团（局部极大值）。
- 最大团 **(Maximum Clique)**：基数 $|C|$ 最大的团（全局最大值）。我们定义图 G 的团数为 $\omega(G)$。

核心概念：最大独立集 (MIS)

在讨论等价性之前，我们先定义补图中的对应概念：

最大独立集 (Maximum Independent Set)

- 定义：给定无向图 $G = (V, E)$，独立集是顶点集 $S \subseteq V$，其中任意两点均不相邻。

$$\forall u, v \in S, \{u, v\} \notin E$$

- 直观理解：一群“互不认识”的人，或者彼此之间没有连边的点。
- 记号：其最大基数记为 $\alpha(G)$。

核心概念：最小顶点覆盖 (MVC)

这是理解等价性的难点，也是“互补”的关键：

最小顶点覆盖 (Minimum Vertex Cover)

- 定义：顶点集 $K \subseteq V$，使得图中的每一条边至少有一个端点属于 K。

$$\forall \{u, v\} \in E, \{u, v\} \cap K \neq \emptyset$$

- 直观隐喻 **(Security Guards)**：想象在路口（顶点）设置安保人员，要求每一条街道（边）都必须处于至少一名安保人员的监控之下。

- 记号：其最小基数记为 $\beta(G)$。

等价性证明一：从团到独立集

转化核心：利用 补图 **(Complement Graph)** $\bar{G}$。

证明.

设 $C \subseteq V$ 是图 G 的一个顶点子集。

- C 是 G 的团 (全连接)
- $\iff \forall u, v \in C$, u, v 在 G 中有边相连
- $\iff \forall u, v \in C$, u, v 在 $\bar{G}$ 中无边相连 (根据定义)
- $\iff$ C 是 $\bar{G}$ 的独立集

□

结论 **1:** $\omega(G) = \alpha(\bar{G})$

等价性证明二：从独立集到顶点覆盖

转化核心：利用 集合的补 **(Set Complement)** $V \setminus S$。

证明.

($\Rightarrow$) 设 S 为独立集（内部无边）。

- 图中每条边的两个端点，不可能都在 S 中。
- $\Longrightarrow$ 每条边至少有一个端点在 S 的外面 (即 $V \setminus S$)。
- $\Longrightarrow$ $V \setminus S$ 覆盖了所有的边。

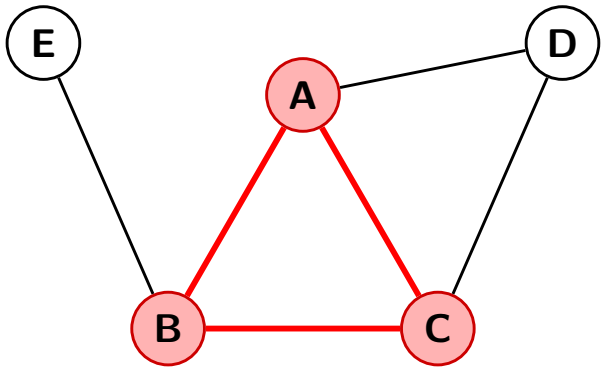
($\Leftarrow$) 反之亦然。要使独立集 $|S|$ 最大，必须使覆盖集 $|V \setminus S|$ 最小。 □

结论 (Final Equivalence)

$$\omega(G) = \alpha(\bar{G}) = |V| - \beta(\bar{G})$$

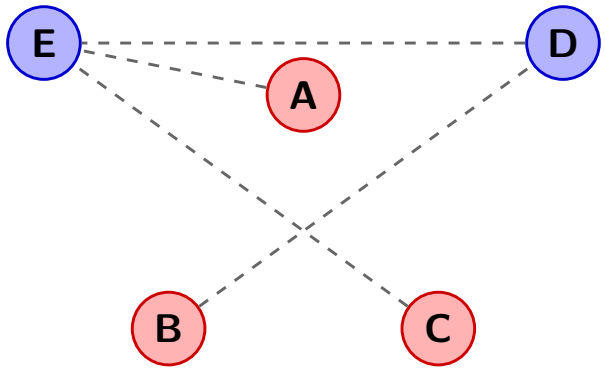
最大团、最大独立集与最小顶点覆盖的关系

原图 G: 最大团



最大团 (Max Clique)
$\{A, B, C\}$ 两两相连

补图 $\bar{G}$: MIS 与 MVC



最大独立集 (MIS): $\{A, B, C\}$ (互不相连)
最小点覆盖 (MVC): $\{D, E\}$ (覆盖所有虚线边)
注: $MVC = V \setminus MIS$

计算复杂性 (Computational Complexity)

- **NP-Complete**: 判定是否存在大小为 k 的团是 NPC 问题 (Karp, 1972)。
- 不可近似性 **(Inapproximability)**:
  - 除非 $P = NP$，否则不存在多项式时间近似方案 (PTAS)。
  - Håstad (1999) 证明：对于任意 $\epsilon > 0$，在 $n^{1-\epsilon}$ 因子内近似 MCP 是 NP-hard 的。
- 参数复杂性: MCP 是 $W[1]$-complete 的，这意味着即使固定参数 k，也不太可能存在 $f(k) \cdot n^{O(1)}$ 的 FPT 算法。

基础框架：状态定义

几乎所有现代精确算法都基于通用的分支定界 (Branch-and-Bound) 框架。

核心递归状态 Clique(C, P)

算法维护两个关键集合和一个全局最优解：

- 当前团 C **(Current Clique)**：
  递归路径上已经确定选入团的顶点集合。

- 候选集 P **(Candidate Set)**：
  $P = \{v \in V \setminus C \mid \forall u \in C, (u, v) \in E\}$。
  即：剩余顶点中，与 C 中所有顶点都相连的点集。

- 当前最优解 C^* **(Incumbent)**：
  搜索过程中发现的最大的团，用于剪枝比较。

基础框架：Carraghan & Pardalos (CP) 算法

- **CP 算法 (1990)**
- DFS 搜索的经典实现。
- 核心思想是利用剪枝策略减少搜索空间。

Algorithm 1: CP Algorithm Structure

Input: 当前团 C, 候选集 P
Global: 全局最优解 C^*

```
// 1. 剪枝 (Bounding)
if |C| + |P| ≤ |C*| then
    return                    // 无法更新最优解
end
// 2. 更新 (Update)
if |C| > |C*| then
    C* ← C
end
// 3. 分支 (Branching)
while P ≠ ∅ do
    Pick v ∈ P, P ← P \ {v}
    Clique(C ∪ {v}, P ∩ N(v))
end
```

核心剪枝：基于着色的上界原理

Theorem: Weak Duality of Graphs

对于任意图 G，其最大团大小 $\omega(G)$ 总是小于等于其色数 $\chi(G)$：

$$\omega(G) \leq \chi(G)$$

直观理解 (Intuition)

- 团的性质：团 C 中的任意两个顶点都是相连的。
- 着色约束：相连的顶点不能染相同颜色。
- 鸽巢原理：如果你有一个大小为 k 的团，你至少需要 k 种不同的颜色来区分它们。

为什么有效？

- 相比于简单的 $|P|$ 上界，色数利用了图的结构信息。
- 颜色类 (Color Class) 实际上是一个独立集 (Independent Set)。
- 一个团在每个独立集中至多只能选取 1 个顶点。

核心剪枝：贪心着色与工程实现

挑战：计算精确的 $\chi(G[P])$ 也是 NP-hard 问题。
对策：使用贪心着色 **(Greedy Coloring)** 计算近似值。

Algorithm 2: Greedy Coloring Heuristic

Input: 候选集 P
Output: 颜色数 k
$k \leftarrow 0$
while $P \neq \emptyset$ **do**
  $k \leftarrow k + 1$
  $I \leftarrow \text{MaxIndependentSet}(P)$ // 贪心取
  // 将颜色 k 分配给 I 中的点
  $P \leftarrow P \setminus I$
end
return k

剪枝逻辑 **(Pruning Logic)**:

Let k_{greedy} be the result.

$$\omega(G[P]) \leq \chi(G[P]) \leq k_{greedy}$$

剪枝条件

If $|C| + k_{greedy} \leq |C^*|$
$\Longrightarrow$ **Prune / Return**

复杂度：通常为 $O(|P| + |E_P|)$，权衡了计算耗时与剪枝效果。

MCQ 算法详解：着色与定界 (Numbering)

MCQ 的核心是一个名为 NUMBER 的过程。它不只是简单的排序，而是通过贪心着色计算每个点在当前子图中的最大潜力。

Algorithm 3: Procedure NUMBER(P)

Input: 候选集 P
Output: 排序后的点集 $\{v_1, \ldots, v_k\}$ 及各点色号

$i \leftarrow 1$, $max\_color \leftarrow 0$
while $P \neq \emptyset$ **do**
  Pick $v \in P$ (e.g., max degree in P)
  // 给 v 分配最小可用颜色
  $k \leftarrow \min\{c \geq 1 : c \notin$ used_colors$(N(v) \cap P)\}$
  $color[v] \leftarrow k$
  $P \leftarrow P \setminus \{v\}$, **push** v to $Stack$
  if $k > max\_color$ **then**
    $max\_color \leftarrow k$
  end
end

过程解析：

- 动态着色：在每一层递归中，我们都对当前的 P 重新着色。
- 颜色即上界：这意味着包含 v 的最大团，在当前剩余的 P 中，最多还能再选 $color[v] - 1$ 个点。
- 结果：得到一个有序序列 $v_1, v_2, \ldots$（栈的弹出顺序），通常颜色号大的点排在后面。

MCQ 算法详解：递归与分支 (EXPAND)

Input: P **Global:** C, C^*

Procedure EXPAND(P)
$\{v_1, \ldots, v_k\} \leftarrow$ NUMBER(P)
for $i \leftarrow k$ ***downto*** 1 **do**
  $v \leftarrow v_i$
  if $|C| + color[v] \leq |C^*|$ **then return**
  // 3. 递归与回溯
  $P_{new} \leftarrow P \cap N(v)$
  if $P_{new} = \emptyset$ **then**
    if $|C| + 1 > |C^*|$ **then** $C^* \leftarrow C \cup \{v\}$
  end
  else
    EXPAND(P_{new})
  end
  $P \leftarrow P \setminus \{v\}$
end

- 单行剪枝：一旦 'color[v]' 降到阈值以下，循环直接结束（因为后续点颜色更小）。
- 基准情况：当 P_{new} 为空时，尝试更新全局最优解 C^*。

MCQ 算法详解：原理证明与工程优化

1. 为什么 Color Bound 有效？

- 独立集性质：着色本质上是将图划分为若干个独立集（颜色类）。
- 限制：一个团在每个独立集中至多包含 1 个顶点。
- 结论：如果 P 能被 k 种颜色着色，则 $\omega(P) \leq k$。

2. 逆序分支的数学意义

- 按 $color[v]$ 降序搜索 $(k \to 1)$。
- 快速收敛：$color[v]$ 是当前分支的“硬上限”。随着循环进行，上限迅速下降。

3. 工程瓶颈：集合求交

- 算法中最频繁的操作是计算新候选集：

$$P_{new} \leftarrow P \cap N(v)$$

- 在稠密图中，这是 $O(N)$ 的操作。

关键优化：Bitset

利用 CPU 位并行特性加速集合运算。

- 存储：邻接矩阵用 `std::bitset` 或 `unsigned long long[]` 存储。
- 运算：交集变为按位与 (&)。

精确算法：基于 DP 思想的子问题求解

除了着色剪枝，另一类重要方法利用了动态规划 **(Dynamic Programming)** 的思想来复用计算结果。

Cliquer 算法 (Östergård, 2002)

核心策略：反向归纳 (Iterative Deepening)

1. 将顶点排序 $v_1, v_2, \ldots, v_n$。
2. 定义后缀子图 $S_i = \{v_i, v_{i+1}, \ldots, v_n\}$。
3. 定义状态 $c(i)$：子图 S_i 中的最大团大小。

Algorithm 4: Östergård's Approach

Data: 顺序 $i = n \ldots 1$
$c(n) = 1$
for $i = n - 1$ **to** 1 **do**
  在 S_i 找含 v_i 最大团 C_{new}
  // 利用 $c(i+1)$ 剪枝
  if $|Curr| + c(i+1) \leq |Best|$ **then**
    Prune (剪枝)
  end
  $c(i) = \max(|C_{new}|, c(i+1))$
end

现代优化：位并行与重着色

在 $N \approx 100 \sim 4000$ 的稠密图上，常数级优化至关重要。

- 位并行 **(Bit-parallelism) - BB-MaxClique**：
  - 思路：利用 CPU 的 64 位寄存器 (或 SIMD) 存储邻接矩阵。
  - 操作：集合求交集 $P \cap N(v)$ 转化为位运算 `P & N[v]`。
  - 效果：相比传统数组实现，速度提升 10-30 倍，特别适合稠密图。
- 重着色 **(Recoloring) - MCS**：
  - 痛点：贪心着色可能不准，导致界不够紧。
  - 策略：尝试将某些点的颜色“修复”到更小的颜色类中。如果能成功减少总色数，就能触发更多剪枝。

代数建模：离散视角 (Integer Programming)

最大团问题最自然的建模是基于补图的独立集约束 (Nemhauser & Trotter, 1975)。

$$\text{Maximize} \quad |C| = \sum x_i \quad \text{s.t.} \quad x_i + x_j \leq 1,\ \forall \{i,j\} \notin E,\ x \in \{0,1\}^n$$

1. 约束含义

- $\{i,j\} \notin E \implies i,j$ 不相连。
- 团要求两两相连，故不连通的对子中至多选一个。
- 这等价于求补图 $\bar{G}$ 的最大独立集。

2. 线性松弛 (LP Relaxation)

- 若松弛为 $x_i \in [0,1]$，会出现巨大的 Integrality Gap。
- 例：C_5 (五边形)，$\omega(G) = 2$。
- LP 解可取全 0.5，目标值 2.5。
- 后果：直接用 LP Solver 做 Branch & Bound 效率极低。

代数建模：连续视角 (Motzkin-Straus)

1965 年，Motzkin-Straus 建立了组合优化与连续优化的桥梁。

Theorem: 在标准单纯形 $\Delta = \{x \in \mathbb{R}^n \mid \sum x_i = 1, x_i \geq 0\}$ 上最大化 $f(x) = x^T A x$。若最大值为 f^*，则：

$$\omega(G) = \frac{1}{1 - f^*} \iff f^* = 1 - \frac{1}{\omega(G)}$$

直观推导 (Intuition)

- 假设团 C 大小为 k，我们在 C 上均匀分配概率。
- 令 $x_i = 1/k$ (若 $i \in C$)，否则 0。
- 代入 $f(x) = \sum A_{ij} x_i x_j$。

计算过程

- 团内有 $k(k-1)$ 条有向边 $(A_{ij} = 1)$。
- 每一项贡献 $(1/k) \cdot (1/k) = 1/k^2$。
- 总和：$k(k-1) \times \frac{1}{k^2} = 1 - \frac{1}{k}$。

结论：团越大 ($k \uparrow$) $\implies$ 函数值越高 ($f \to 1$)

数学困境：非凸性的本质 (Non-Convexity)

试图最大化 $f(x) = x^T Ax$ $(x \in \Delta)$ 时，我们将面临极不友好的几何性质。

1. 谱性质：不定矩阵 (Indefinite Matrix)

对于任意非平凡图，邻接矩阵 A 既非正定也非半正定。

- 这意味着特征值 λ 有正有负。
- 导致优化曲面呈现复杂的“马鞍面”特征，而非碗状。

2. 几何后果：崎岖的优化地形

由于矩阵不定，目标函数 $f(x)$ 非凸非凹，导致：

- 局部陷阱：空间中充满指数级数量的局部极大值。
- 致命的对应关系：

$$\text{局部极值} \iff \text{极大团 (Maximal)}$$

$$\text{全局极值} \iff \text{最大团 (Maximum)}$$

- 结论：简单的梯度上升只能带你找到“极大团”，几乎不可能翻越山岭找到“最大团”。

数学困境：微积分与线性代数的失效

既然函数性质不好，标准数学工具的表现如何？

1. 梯度上升 (Gradient Ascent)

- 梯度 $\nabla f(x) = 2Ax$ 指向当前增长最快的方向。
- 缺陷：对于多峰函数，梯度上升是“短视”的。它会迅速收敛到最近的极大团（Local Optima），然后停滞不前。

2. 谱分析 (Spectral Analysis)

- **Wilf Bound**: $\omega(G) \leq \lambda_{max} + 1$。
- 缺陷：这只是一个上界。特征向量对应的是实数域的最优解，很难直接映射回离散的 0/1 团结构。

直观理解：登山悖论

想象一片连绵起伏的山脉：

- 极大团 = 无数个小山头。
- 最大团 = 珠穆朗玛峰。

微积分（梯度）就像蒙着眼的登山者，只能感知脚下的坡度。他大概率会爬上家门口的小土坡，然后以为自己到了最高点。

$\therefore$ 必须引入扰动与跳跃 (启发式)

现代启发式框架：罚函数策略 (k-fixed Penalty)

这类算法（如 NuMVC, FastStep）将优化问题转化为一系列判定问题：能否找到大小为 k 的团？

1. 核心机制：允许“犯错”

- 状态定义：搜索空间包含非团子图。我们在 V 中任选 k 个点作为集合 S。
- 能量函数 **(Cost Function)**：

$$f(S) = |\{(u,v) \in S \times S \mid (u,v) \notin E\}|$$

即 S 中缺失边的数量（Missing Edges）。
- 目标：通过局部调整使得 $f(S) = 0$。

2. 深度直觉：填海造陆

- 困境：合法团之间往往被“非团结构”隔开（像海洋隔开群岛）。如果只允许走合法团，很容易卡死。
- 策略：罚函数允许算法暂时进入“非法区域”（海里），以此为桥梁走到另一个更好的团。
- **Configuration Checking**：为了防止在海里迷路（循环震荡），必须引入禁忌机制 (Tabu)。

现代启发式框架：合法策略 (Legal Strategy)

这一流派始终维护当前解 C 为合法团。其算子设计深受连续优化梯度 $(\nabla f = 2Ax)$ 的启发。

三大核心算子 (Operators)

1. **Add (扩展)** 若存在点 v 与 C 全连通，直接加入。*(对应贪心上升)*
2. **Swap (交换) - 梯度的离散化身** 当无法 Add 时（陷入极大团），我们需要换点。
   - 如何选点？计算候选集中每个点 v 的权重（通常基于 $x^T Ax$ 的梯度增益）。
   - 操作：踢出一个“最差”的点，换入一个“潜力最大”的点。
   - 意义：这是在离散空间中模拟沿着梯度方向修正搜索路径。
3. **Drop (扰动)** 如果 Swap 多次后仍无进展，强制随机丢弃 C 中部分节点。*(对应跳出局部最优 Basin)*

总结：合法策略更像是一个“带随机扰动的爬山算法”。

现代启发式框架：进化算法的陷阱与出路

在最大团问题 (MCP) 上，纯粹的遗传算法 (GA) 往往不如简单的局部搜索。

1. 结构脆弱性与交叉失效

- 核心矛盾：团是一个高内聚结构。
- 交叉算子 **(Crossover)**：假设父代 P_1 和 P_2 都是团。
- 语义破坏：简单的杂交（如各取一半）产生的子代，往往内部极其稀疏，支离破碎。
- 后果：子代不再是团，修复它需要耗费巨大计算量。算法退化为随机搜索。

2. 破局：模因算法 (Memetic)

- 既然 GA 搜得慢，局部搜索 (LS) 容易卡住，不如结合。
- 框架：

  GA (负责跳跃)+LS (负责挖掘)

- 专用重组算子：现代算法 (如 Map-Elites) 设计了保留团结构的特殊交叉算子，而非简单的基因片段交换。

Key Insight：不要直接套用通用 AI 算法。算法设计必须尊重问题的拓扑结构 **(Topology)**。

现实应用：网络分析与生物计算

1. 社交网络 (SNA)

场景：在海量数据中挖掘核心圈层。

- 社区发现：寻找高密度的 **Cohesive Subgroups**。
- 反恐与风控：恐怖组织或黑产团伙通常表现为全连通子图（团）。
- 难点：图规模极大（百万节点），需结合启发式算法（如 k-core 预处理）进行剪枝。

2. 生物信息 (Bioinformatics)

场景：分子结构的相似性比对。

- 蛋白质结构预测：判断两个蛋白质在空间上是否同构。
- 建模手段：构建对应图 **(Correspondence Graph)**。
  - 节点：原子对 (u, v)。
  - 边：若距离约束兼容则连线。
- 目标：求最大团即求最大公共子结构 (MCS)。

现实应用：计算机视觉与信息论

3. 计算机视觉 (CV)

场景：图像匹配与物体识别。

- 特征匹配：寻找两幅图中几何一致的特征点集。
- 关联图 **(Association Graph)**：
  - 节点：假设匹配对 (u, v)。
  - 边：若几何约束兼容则连线。
- 本质：在含噪图中寻找最大相容子集。

4. 编码理论 (Coding)

场景：零误差信道传输。

- 纠错设计：选一组信号，使任意两个都不会因干扰混淆。
- 香农容量 **(Shannon Capacity)**：
  - 建“混淆图” G (若混淆则连边)。
  - 问题转化为求 $\alpha(G)$ (最大独立集)。
- 理论：Lovász ϑ 函数为此提供紧上界。

Insight：现实中的“兼容性 (Compatibility)”与“互不冲突”，在数学上往往等价于图论中的最大团或独立集问题。

全景回顾：精度与速度的权衡

最大团问题的求解历史，本质上是在搜索空间剪枝与局部极值跳出之间寻找平衡。

1. 精确算法 (Exact: B&B)

代表：MCQ, MaxCliqueDyn

- 核心：利用数学界（着色、MaxSAT）进行剪枝。
- 优势：保证最优性 (C^*)。
- 劣势：最坏复杂度 $O(3^{n/3})$，难以处理 $N > 4000$ 的稠密图。

2. 启发式 (Heuristic: LS)

代表：NuMVC, FastStep

- 核心：利用扰动机制（Penalty, Swap）跳出局部最优。
- 优势：能在巨大图上迅速找到高质量解。
- 劣势：无最优性证明。

3. 现代趋势：混合求解 (Hybrid Solving)

现在的顶级求解器倾向于结合两者：

- 使用启发式快速找到一个大的 LB (Lower Bound)。
- 将此 LB 注入精确算法作为初始界，大幅提升剪枝效率。

前沿展望：当组合优化遇上 AI

随着图神经网络 (GNN) 和深度强化学习 (DRL) 的兴起，MCP 研究正在进入新的范式。

1. Learning to Branch / Prune

- 传统 MCQ 使用贪心着色来排序分支。
- **AI** 思路：训练 GNN 预测哪个节点最可能在最大团中，从而指导分支顺序。
- 优势：在特定分布的图上，比人工设计的 *Heuristic* 快数个数量级。

2. 连续优化的进阶 (SDP)

- 除了 Motzkin-Straus，半正定规划 **(SDP)** 提供了更紧的界（Lovász Theta function $\vartheta(G)$）。
- 这是一个凸松弛 (Convex Relaxation)，可以在多项式时间内求解。

Open Problems

1. 大规模稀疏图：社交网络图的团挖掘 (Web-scale)。
2. 端到端求解：能否完全用神经网络替代搜索？(目前还很难)。
3. 硬件加速：利用 GPU/FPGA 并行化 Bitset 运算。

Thanks!