图着色问题

问题背景： 图着色问题（Graph Coloring Problem）是一种经典的组合优化问题，通常定义为：给定一个无向图 $G=(V,E)$ 和整数 k，求一种将顶点 V 染色的方案，使得相邻的两个顶点不能使用相同的颜色，并且使用的颜色数量不超过 k。确定是否存在这样一种方案属于 NP 完全问题。

编码成 Partial MaxSAT：

- 变量定义：对于每个顶点 $v_i \in V$ 和每个颜色 $j \in \{1,2,\ldots,k\}$，定义布尔变量 $x_{i,j}$ 表示顶点 v_i 是否被赋予颜色 j。

$$x_{i,j} = \begin{cases} 1, & \text{如果顶点 } v_i \text{ 被赋予颜色 } j \\ 0, & \text{否则} \end{cases}$$

- 硬约束（Hard Clauses）：

  1. 每个顶点至少要被赋予一种颜色：

  $$\bigvee_{j=1}^{k} x_{i,j}, \quad \forall v_i \in V$$

  2. 每个顶点最多被赋予一种颜色：

  $$\neg x_{i,j} \vee \neg x_{i,l}, \quad \forall v_i \in V, \forall j,l \in \{1,\ldots,k\}, j<l$$

  3. 相邻顶点不能使用相同颜色：

  $$\neg x_{i,j} \vee \neg x_{p,j}, \quad \forall (v_i,v_p) \in E, \forall j \in \{1,\ldots,k\}$$

- 软约束（Soft Clauses）：

  希望使用尽可能少的颜色，定义软约束来最小化所用颜色数目。可引入变量 y_j 表示颜色 j 是否被使用，并添加软约束使得使用颜色的数量最少：

  $$\neg y_j \vee x_{i,j}, \quad \forall v_i \in V, \forall j \in \{1,\ldots,k\}$$

  并且在目标函数中添加如下软约束以最小化颜色使用：

  $$\text{Minimize: } \sum_{j=1}^{k} y_j$$

最大团问题

问题背景：最大团问题（Maximum Clique Problem）是图论中的一个经典 NP 难问题。给定一个无向图 $G=(V,E)$，一个**团（clique）**是指图中一个完全连接的顶点子集，即子集中的每对顶点之间都存在边。最大团问题的目标是找到一个最大的团（包含顶点数目最多的团）。该问题广泛应用于社交网络分析、生物信息学、通信网络等领域。

编码成 Partial MaxSAT：

- 变量定义：对于每个顶点 $v_i \in V$，定义布尔变量 x_i，表示顶点 v_i 是否属于所选的团：

$$x_i = \begin{cases} 1, & \text{若顶点 } v_i \text{ 属于团中} \\ 0, & \text{否则} \end{cases}$$

- 硬约束（Hard Clauses）：团要求所有被选中的顶点两两之间必须有边相连。因此，对于任意一对不相邻的顶点 $(v_i, v_j) \notin E$，添加硬约束：

$$\neg x_i \vee \neg x_j, \quad \forall (v_i, v_j) \notin E$$

  该约束确保选中的顶点集必定形成一个团。

- 软约束（Soft Clauses）：目标是最大化团的规模，因此对每个顶点 $v_i \in V$，添加软约束，以鼓励选择更多顶点进入团中：

$$(x_i), \quad \forall v_i \in V$$

  最终，Partial MaxSAT 求解器的目标是满足所有硬约束，同时最大化被满足的软约束数量，即选择更多顶点以形成最大团。

  因此，该 Partial MaxSAT 问题的目标函数可表示为：

$$\text{Maximize: } \sum_{v_i \in V} x_i$$

最大割问题

问题背景：最大割问题（Maximum Cut Problem）是组合优化领域中的经典问题之一。给定一个无向加权图 $G=(V,E,w)$，其中 $w:E\to\mathbb{R}^+$ 是边的权重函数。最大割问题的目标是将顶点集合 V 划分为两个不相交的子集，使得连接两个子集的边的权重之和最大化。最大割问题属于 NP 难问题，在统计物理、组合优化和电路设计等领域有重要的应用。

形式化地，最大割问题可表述为：

$$\max_{S\subseteq V}\sum_{\substack{(u,v)\in E\\ u\in S,v\in V\setminus S}} w(u,v)$$

编码成 Partial MaxSAT：

- 变量定义：对每个顶点 $v_i\in V$，定义布尔变量 x_i 表示该顶点所在的分区：

$$x_i=\begin{cases}1, & \text{若顶点 } v_i \text{ 属于第一个子集 } S\\ 0, & \text{若顶点 } v_i \text{ 属于第二个子集 } V\setminus S\end{cases}$$

- 硬约束（Hard Clauses）：最大割问题本质上并无强制约束需要满足，因此不需要添加硬约束。
- 硬约束（Soft Clauses）：对于图中的每一条边 $(v_i,v_j)\in E$，添加以下两个子句：

  1. 子句一：$(x_i\vee x_j)$
  2. 子句二：$(\neg x_i\vee\neg x_j)$

- 与最大割问题的关联目标：
  - 每条被切割的边（连接不同集合的顶点对），为 Max-SAT 问题贡献 2 个满足的子句。
  - 每条未被切割的边（连接相同集合的顶点对），为 Max-SAT 问题贡献 1 个满足的子句。

顶点覆盖问题

问题背景： 顶点覆盖问题（Vertex Cover Problem）是组合优化领域中一个典型的 NP 完全问题。给定一个无向图 $G=(V,E)$，顶点覆盖是顶点集的一个子集 $C \subseteq V$，满足对图中每条边 $(u,v) \in E$，至少有一个端点属于 C。顶点覆盖问题的目标是在图中找到一个最小的顶点覆盖集合，即包含最少顶点数的覆盖集合。该问题在网络安全、资源配置、任务调度等领域具有广泛的应用。

形式化地，顶点覆盖问题可表述为：

$$\min_{C \subseteq V} |C|, \quad \text{满足：} \forall (u,v) \in E,\ u \in C \vee v \in C$$

编码成 Partial MaxSAT：

- 变量定义：对每个顶点 $v_i \in V$，定义布尔变量 x_i 表示顶点 v_i 是否属于顶点覆盖集合 C：

$$x_i = \begin{cases} 1, & \text{若 } v_i \in C \\ 0, & \text{否则} \end{cases}$$

- 硬约束（Hard Clauses）：对于图中的每一条边 $(v_i, v_j) \in E$，必须至少有一个端点被选择进入集合 C，因此添加硬约束：

$$(x_i \vee x_j), \quad \forall (v_i, v_j) \in E$$

  这确保集合 C 为一个合法的顶点覆盖。

- 软约束（Soft Clauses）：目标是找到最小的顶点覆盖集合，因此需要对每个顶点 $v_i \in V$ 加入软约束以最小化被选中的顶点数目：

$$(\neg x_i), \quad \forall v_i \in V$$

  最终，Partial MaxSAT 求解器的目标是满足所有硬约束，并同时最大化被满足的软约束（即选择更少的顶点）。

  目标函数可表述为：

$$\text{Minimize: } \sum_{v_i \in V} x_i$$

最小支配集问题

问题背景：最小支配集问题（Minimum Dominating Set Problem）是组合优化中的经典 NP 完全问题之一。给定一个无向图 $G = (V, E)$，一个顶点子集 $D \subseteq V$ 称为支配集（Dominating Set），当且仅当对于图中每个顶点 $v \in V$，要么 $v \in D$，要么 v 邻接的至少一个顶点属于集合 D。最小支配集问题的目标是在图中找到一个顶点数目最少的支配集。该问题广泛应用于传感器网络、设施选址、路由和社会网络分析等场景。

形式化描述为：

$$\min_{D \subseteq V} |D|, \quad \text{满足：} \forall v \in V, \quad v \in D \vee (\exists u \in D, (u, v) \in E)$$

编码成 Partial MaxSAT：将最小支配集问题编码为 Partial MaxSAT 问题的过程如下：

- 变量定义：对于每个顶点 $v_i \in V$，定义布尔变量 x_i 表示该顶点是否属于支配集 D：

$$x_i = \begin{cases} 1, & \text{若 } v_i \in D \\ 0, & \text{否则} \end{cases}$$

- 硬约束（Hard Clauses）：为确保支配集定义满足，每个顶点 $v_i \in V$ 都必须被覆盖，即每个顶点或其邻居中至少有一个被选入集合。因此，对于每个顶点 $v_i \in V$ 添加如下硬约束：

$$\left(x_i \vee \bigvee_{(v_i, v_j) \in E} x_j \right), \quad \forall v_i \in V$$

  该约束保证了每个顶点都被支配集覆盖。

- 软约束（Soft Clauses）：目标是最小化支配集的规模。因此对每个顶点 $v_i \in V$，添加如下软约束，鼓励选取更少顶点：

$$(\neg x_i), \quad \forall v_i \in V$$

  最终，Partial MaxSAT 求解器的目标是满足所有硬约束，同时最大化满足的软约束，即使支配集规模尽可能小。

  目标函数表达式可记为：

$$\text{Minimize: } \sum_{v_i \in V} x_i$$

最小加权顶点覆盖问题

问题背景：最小加权顶点覆盖问题（Minimum Weighted Vertex Cover Problem）是顶点覆盖问题的加权版本。给定一个无向图 $G = (V, E)$，以及顶点权重函数 $w : V \to \mathbb{R}^+$，加权顶点覆盖问题的目标是找到一个顶点覆盖集合 $C \subseteq V$，使得集合中顶点权重之和最小。即要求满足每条边至少有一个端点位于覆盖集合中的同时，最小化覆盖集合的总权重。

形式化地，问题可描述为：

$$\min_{C \subseteq V} \sum_{v_i \in C} w(v_i), \quad \text{满足：} \forall (u, v) \in E, \quad u \in C \vee v \in C$$

该问题在通信网络设计、资源分配、设施选址等实际应用领域均具有重要的现实意义。**编码成 Partial MaxSAT：**

- 变量定义：对每个顶点 $v_i \in V$，定义布尔变量 x_i 表示该顶点是否属于顶点覆盖集合 C：

$$x_i = \begin{cases} 1, & \text{若 } v_i \in C \\ 0, & \text{否则} \end{cases}$$

- 硬约束（Hard Clauses）：对于图中每条边 $(v_i, v_j) \in E$，至少有一个端点必须属于覆盖集合 C，因此添加如下硬约束：

$$(x_i \vee x_j), \quad \forall (v_i, v_j) \in E$$

- 软约束(Soft Clauses)：目标是最小化覆盖集合中顶点的权重之和，因此，对于每个顶点 $v_i \in V$，添加权重为 $w(v_i)$ 的如下软约束：

$$(\neg x_i) : w(v_i), \quad \forall v_i \in V$$

Partial MaxSAT 求解器的目标为满足所有硬约束的同时，尽量满足更多的软约束，从而最小化覆盖集合的权重。

因此，目标函数的表达式为：

$$\text{Minimize: } \sum_{v_i \in V} w(v_i) \cdot x_i$$

最小集合覆盖问题

问题背景：最小集合覆盖问题（Minimum Set Cover Problem）是经典的组合优化问题之一，属于 NP 完全问题。问题定义如下：给定一个全集 $U = \{u_1, u_2, \ldots, u_n\}$ 和一个集合族 $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$，其中每个集合 $S_j \subseteq U$，且 $\bigcup_{j=1}^{m} S_j = U$，最小集合覆盖问题的目标是在 $\mathcal{S}$ 中找到尽可能少的集合，使它们的并集能够覆盖整个全集 U。该问题在资源配置、设施选址、网络设计、任务调度等领域均有广泛应用。

问题可形式化为：

$$\min_{\mathcal{C} \subseteq \mathcal{S}} |\mathcal{C}|, \quad \text{满足：} \bigcup_{S_j \in \mathcal{C}} S_j = U$$

编码成 Partial MaxSAT：

- 变量定义：对集合族 $\mathcal{S}$ 中的每个集合 S_j 定义布尔变量 x_j，表示是否选取该集合：

$$x_j = \begin{cases} 1, & \text{若集合 } S_j \text{ 被选中} \\ 0, & \text{否则} \end{cases}$$

- 硬约束（Hard Clauses）：为了保证全集中每个元素都被覆盖，对于全集 U 中的每个元素 u_i，至少存在一个包含它的集合 S_j 被选取。因此，对于每个元素 $u_i \in U$，添加硬约束：

$$\bigvee_{j: u_i \in S_j} x_j, \quad \forall u_i \in U$$

- 软约束（Soft Clauses）：目标是最小化所选集合的数目，因此对每个集合 $S_j \in \mathcal{S}$ 添加如下软约束，以最小化所选集合数量：

$$(\neg x_j), \quad \forall S_j \in \mathcal{S}$$

  Partial MaxSAT 求解器的目标是满足所有硬约束并同时最大化被满足的软约束，从而实现最小化集合覆盖规模的目标。

  因此，目标函数的表达式为：

$$\text{Minimize:} \sum_{S_j \in \mathcal{S}} x_j$$

算法：改进的局部搜索顶点覆盖

```
Algorithm: ImprovedLocalSearchVertexCover(G, k, MaxIterations)
Input:
    G = (V, E)                // 给定的无向图
    k                         // 需要选取的顶点数
    MaxIterations             // 最大迭代次数限制
Output:
    BestCoverSet              // 覆盖尽可能多边的k个顶点集合

Initialize:
    CurrentSet ← 选取图G中度数最高的k个顶点
    BestCoverSet ← CurrentSet
    BestCoveredEdges ← CountCoveredEdges(G, BestCoverSet)

for iteration from 1 to MaxIterations do:
    improvement ← False

    for each vertex u in CurrentSet do:
        for each vertex v in (V - CurrentSet) do:
            NewSet ← CurrentSet - {u} + {v}
            CoveredEdges ← CountCoveredEdges(G, NewSet)

            if CoveredEdges > BestCoveredEdges then:
                CurrentSet ← NewSet
                BestCoveredEdges ← CoveredEdges
                BestCoverSet ← NewSet
                improvement ← True
                break  // 一旦找到改善的解，立即跳出内层循环

        if improvement then
            break  // 改善后重新开始外层循环

    if not improvement then:
        break  // 达到局部最优，终止搜索

return BestCoverSet

Function CountCoveredEdges(G, VertexSet):
    Covered ← 0
    for each edge (u, v) in E do:
        if u in VertexSet or v in VertexSet then:
            Covered ← Covered + 1
    return Covered
```

Listing 1: ImprovedLocalSearchVertexCover

局部最优解的改进策略

下面介绍三种常用的跳出局部最优的策略，每种策略先给出核心思想，再说明如何在算法里实现。

策略一：随机重启 (Random Restarts) 思想： 当算法陷入某个局部最优时，不立即停止，而是记录当前的最佳解，然后重新生成一个完全不同的初始解，再次运行局部搜索。

实现： 将主算法重复执行 N 次，每次从一个新的随机（或基于贪心的）初始解开始；最后返回这 N 次运行中找到的最优解。

策略二：允许“坏”移动 (Simulated Annealing) 思想： 模拟物理退火过程。在搜索初期，以一定概率接受一次“坏”的移动（即增益 $Gain < 0$），以跳出当前“山谷”并探索其他区域；随着迭代，接受坏移动的概率逐渐降低。

实现： 对每次候选交换计算增益 $Gain$：

- 若 $Gain > 0$，则始终接受该移动；
- 若 $Gain \leq 0$，则以概率 $p = \exp\bigl(Gain/T\bigr)$ 接受，

其中 T 为“温度”参数，随迭代次数按照预定退火表降温。

策略三：禁忌搜索 (Tabu Search) 思想： 为防止在几个解间来回振荡，引入“禁忌列表”(Tabu List)，禁止近期的反向移动，以强制算法探索新的邻域。

实现： 每当将顶点 v_{out} 从当前解移出时，将其加入禁忌列表；在接下来的若干迭代中，禁止将这些顶点重新加入解集中。列表可设固定长度，超过后按 FIFO 规则释放最旧元素。

策略四：迭代局部搜索 (Iterated Local Search, ILS) 思想： 在当前局部最优解的基础上，施加一次“扰动”生成一个新解，再对新解进行局部搜索，不断在扰动和精炼之间交替，以发现更优解。

实现： 每次从当前最优解执行轻度扰动（如随机交换若干顶点），得到候选解；然后对该解运行完整的局部搜索；重复 N 次，保留迭代过程中出现的全局最优。

策略五：贪婪随机自适应搜索 (GRASP) 思想： 将构造和局部搜索两阶段结合：在构造阶段使用“贪婪＋随机”策略生成初始解，再对其做局部搜索；多次重复，最终取最优。

实现：

1. *构造阶段：* 根据增益值构造候选列表 (RCL)，从中随机挑选元素加入解；
2. *局部搜索阶段：* 对构造解执行标准局部搜索；
3. 重复上述两阶段 N 次，保留最优解。

策略六：变邻域搜索 (Variable Neighborhood Search, VNS) 思想： 利用一系列不同规模或类型的邻域算子，从小邻域到大邻域逐步扩展，以跳出当前局部最优。

实现：

1. 依次定义多个邻域结构（如单顶点交换、双顶点交换、三顶点重组等）；
2. 在第 k 个邻域内进行扰动并局部搜索；若找到更优解则回到第一个邻域，否则进入下一个邻域；

3. 当所有邻域都尝试完毕后，重新从最优解开始。

策略七：基于学习的自适应搜索 (Adaptive Memory Programming) 思想： 利用历史搜索信息（如成功移动模式、频繁交换对等）来指导当前搜索，动态调整扰动或选择算子的概率。

实现：

- 对每种移动操作（如某种类型的顶点交换）维护一个评分；
- 每轮根据评分分布随机或贪心地选取扰动算子；
- 搜索过程中实时更新评分，强化有效算子的优先级。