16 - Further reductions in NP

从最优化到决策

- 优化问题：我们面对一个优化问题 P，通常是求最小化的问题
  - 举例来说，问题是找到一个图的最小顶点覆盖
- 我们引入一个阈值 k
- 决策问题：决策版本的问题被记作 P_d，具体描述为：给定一个优化问题 P 的实例和一个阈值 k，询问是否存在一个解，使得 P 的值至多为 k
  - 例如，在这里的上下文中，问题变为是否存在一个大小最多为 k 的顶点覆盖

优化问题与决策问题

- 如果我们可以在多项式时间内解决优化问题 P，那么我们也可以在多项式时间内解决其决策版本 P_d
  - 这是因为决策问题通常是检查是否存在某种满足特定条件的解。如果我们能高效地找到最优解，那么比较这个解与给定的阈值 k 通常是简单的
- 同样，如果决策问题是 NPH 的，那么其对应的优化问题通常也是 NPH 的
  - 因为决策问题的解通常可以用来构建优化问题的解

顶点覆盖大小问题

顶点覆盖大小问题 (优化版本)

- 输入：一个图 $G = (V, E)$
- 输出：最小的顶点覆盖的大小

顶点覆盖问题 (决策版本)

- 输入：一个图 $G = (V, E)$ 和一个数字 k
- 输出：是否存在大小至多为 k 的顶点覆盖

解决顶点覆盖问题：

- 使用适用于顶点覆盖决策问题 VC_d 的算法来找到最小顶点覆盖的值 k^*
- 在图中选择一个顶点 v
  - 将顶点 v 及其连接的边移除，得到新的图 $G - \{v\}$
  - 如果顶点 v 在任何最小顶点覆盖中，那么新的图 $G - \{v\}$ 将具有大小为 $k^* - 1$ 的最小顶点覆盖
  - 检查新的图 $G - \{v\}$ 是否有大小至多为 $k^* - 1$ 的顶点覆盖
    - 如果存在这样的顶点覆盖，包含 v 在顶点覆盖中
    - 如果不存在这样的顶点覆盖，不包含 v 在顶点覆盖中
    - 继续选择下一个顶点，重复上述步骤，直到处理完所有顶点

子集和问题

优化版本：

- 描述：

    - 我们给定一个包含 n 个物品的集合 $\{1, 2, \ldots, n\}$
    - 每个物品 i 有一个非负整数权重 w_i
    - 我们还给定一个整数上限 W
- 目标：
    - 选择一个物品的子集 S，使得
        - 子集中所有物品的权重之和不超过 W，即 $\sum_{i \in S} w_i \leq W$
        - 子集中所有物品的权重之和最大化，即最大化 $\sum_{i \in S} w_i$

决策版本：

- 描述：
    - 我们给定一个包含 n 个物品的集合 T：$\{1, 2, \ldots, n\}$
    - 每个物品 i 有一个非负整数权重 w_i
    - 我们还给定一个整数上限 W
- 目标：
    - 决定是否存在一个物品的子集 S，使得该子集的总重量恰好等于 W，即 $\sum_{i \in S} w_i = W$

子集和问题的 P/NP

- 属于 NP
    - NP 类问题是指能够在多项式时间内验证其解的正确性的问题
    - 如果我们给定一个候选解 S（即一个子集），我们可以轻松验证这个子集是否满足以下条件：

$$\sum_{i \in S} w_i = W$$

- 属于 NPhard
    - 我们将通过从 3-SAT 问题进行归约来证明这一点
    - 给定一个 3CNF 公式 φ（包含 m 个子句和 n 个变量），我们将构造一个子集和问题的实例 $\langle T, W \rangle$，使得
        - φ 是可满足的，当且仅当存在一个子集 S 使得 T 的元素和恰好为 W
    - 不失一般性，我们假设：
        - 每个变量都会在某个子句中出现
        - 每个子句中不同时包含同一个变量的肯定与否定形式

规约过程

我们将创建包含 $m + n$ 位数字的整数，其中 m 表示子句的数量，n 表示变量的数量

$$\varphi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

- 将目标和 W 设置为在所有“变量位”上为 1，在所有“子句位”上为 4
    - 在每个变量位上，我们期望选择的子集中的元素权重和为 1
    - 在每个子句位上，我们期望选择的子集中的元素权重和为 4
- 为每个变量 x_i 创建两个整数 z_i 和 y_i
    - 这两个整数分别代表变量 x_i 及其否定 $\neg x_i$
    - 每个整数在对应的“变量位”上为 1，在其他“变量位”上为 0
        - 例如，整数 z_i 和 y_i 在第 1 位上为 1，其余变量位上为 0
    - 如果文字 x_i 出现在子句 C_j 中，整数 z_i 在对应的“子句位”上为 1

- 如果文字 $\neg x_i$ 出现在子句 C_j 中，整数 y_i 在对应的“子句位”上为 1
- 其他所有的“子句位”均设置为 0

- 对于每个子句 C_j，创建两个整数 s_j 和 t_j
  - 这两个整数在子句位 C_j 上有特定的值，在其他位上为 0
  - s_j 在对应的“子句位”上为 1
  - t_j 在对应的“子句位”上为 2

		x1	x2	x3	C1	C2	C3	C4
W		1	1	1	4	4	4	4

		x1	x2	x3	C1	C2	C3	C4
z1		1	0	0				
y1		1	0	0				
z2		0	1	0				
y2		0	1	0				
z3		0	0	1				
y3		0	0	1				
W		1	1	1	4	4	4	4

		x1	x2	x3	C1	C2	C3	C4
z1		1	0	0	1	0	0	1
y1		1	0	0	0	1	1	0
z2		0	1	0	0	0	0	1
y2		0	1	0	1	1	1	0
z3		0	0	1	0	0	1	1
y3		0	0	1	1	1	0	0
W		1	1	1	4	4	4	4

		x1	x2	x3	C1	C2	C3	C4
z1		1	0	0	1	0	0	1
y1		1	0	0	0	1	1	0
z2		0	1	0	0	0	0	1
y2		0	1	0	1	1	1	0
z3		0	0	1	0	0	1	1
y3		0	0	1	1	1	0	0
s1		0	0	0		0	0	0
t1		0	0	0		0	0	0
s2		0	0	0	0		0	0
t2		0	0	0	0		0	0
s3		0	0	0	0	0		0
t3		0	0	0	0	0		0
s4		0	0	0	0	0	0	
t4		0	0	0	0	0	0	
W		1	1	1	4	4	4	4

		x1	x2	x3	C1	C2	C3	C4
z1		1	0	0	1	0	0	1
y1		1	0	0	0	1	1	0
z2		0	1	0	0	0	0	1
y2		0	1	0	1	1	1	0
z3		0	0	1	0	0	1	1
y3		0	0	1	1	1	0	0
s1		0	0	0	**1**	0	0	0
t1		0	0	0	**2**	0	0	0
s2		0	0	0	0	**1**	0	0
t2		0	0	0	0	**2**	0	0
s3		0	0	0	0	0	**1**	0
t3		0	0	0	0	0	**2**	0
s4		0	0	0	0	0	0	**1**
t4		0	0	0	0	0	0	**2**
W		1	1	1	4	4	4	4

证明 NP-hard

其中一种思路

φ 是可满足的 $\Rightarrow$ 存在一个子集 S 使得 T 的元素和恰好为 W

$$\varphi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

通过 z 和 y 的构造，变量位的总和总是等于 111
因为所有子句都得到满足，所以我们在子句位上至少得到一个“1”，这个“1”来自于那些在 x 中被设置为“0”的变量
在这里我们假设 $\neg x_1, \neg x_2, x_3$ 是正确的

如果我们需要 1 个或 2 个更多的 “1” 让 C 的总和达到 “4”，我们需要选择合适的 s_i 或 y_i；同理需要 3 个.....

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	1	0	0	0
t_1		0	0	0	2	0	0	0
s_2		0	0	0	0	1	0	0
t_2		0	0	0	0	2	0	0
s_3		0	0	0	0	0	1	0
t_3		0	0	0	0	0	2	0
s_4		0	0	0	0	0	0	1
t_4		0	0	0	0	0	0	2
W		1	1	1	4	4	4	4

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	1	0	0	0
t_1		0	0	0	2	0	0	0
s_2		0	0	0	0	1	0	0
t_2		0	0	0	0	2	0	0
s_3		0	0	0	0	0	1	0
t_3		0	0	0	0	0	2	0
s_4		0	0	0	0	0	0	1
t_4		0	0	0	0	0	0	2
W		1	1	1	4	4	4	4

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	1	0	0	0
t_1		0	0	0	2	0	0	0
s_2		0	0	0	0	1	0	0
t_2		0	0	0	0	2	0	0
s_3		0	0	0	0	0	1	0
t_3		0	0	0	0	0	2	0
s_4		0	0	0	0	0	0	1
t_4		0	0	0	0	0	0	2
W		1	1	1	4	4	4	4

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	1	0	0	0
t_1		0	0	0	2	0	0	0
s_2		0	0	0	0	1	0	0
t_2		0	0	0	0	2	0	0
s_3		0	0	0	0	0	1	0
t_3		0	0	0	0	0	2	0
s_4		0	0	0	0	0	0	1
t_4		0	0	0	0	0	0	2
W		1	1	1	4	4	4	4

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	1	0	0	0
t_1		0	0	0	2	0	0	0
s_2		0	0	0	0	1	0	0
t_2		0	0	0	0	2	0	0
s_3		0	0	0	0	0	1	0
t_3		0	0	0	0	0	2	0
s_4		0	0	0	0	0	0	1
t_4		0	0	0	0	0	0	2
W		1	1	1	4	4	4	4

另外一种思路

存在一个子集 S 使得 T 的元素和恰好为 $W \Rightarrow \varphi$ 是可满足的

$$\varphi = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

集合 S 必须在每个索引 i 上只包含 z_i 或 y_i 之一，不能同时包含或都不包含。否则，总和将无法等于 W
如果集合 S 包含 z_i，则设 $x_i = 1$；否则，设 $x_i = 0$

考虑任意子句 C_j；考虑对应的“子句位”
因为这些子句位的总和必须达到 4，它们必定至少从选择的 z 或 y 数字中接收到一个“1”
这意味着在每个子句位上，至少有一个文字为真

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	1	0	0	0
t_1		0	0	0	2	0	0	0
s_2		0	0	0	0	1	0	0
t_2		0	0	0	0	2	0	0
s_3		0	0	0	0	0	1	0
t_3		0	0	0	0	0	2	0
s_4		0	0	0	0	0	0	1
t_4		0	0	0	0	0	0	2
W		1	1	1	4	4	4	4

		x_1	x_2	x_3	C_1	C_2	C_3	C_4
z_1		1	0	0	1	0	0	1
y_1		1	0	0	0	1	1	0
z_2		0	1	0	0	0	0	1
y_2		0	1	0	1	1	1	0
z_3		0	0	1	0	0	1	1
y_3		0	0	1	1	1	0	0
s_1		0	0	0	1	0	0	0
t_1		0	0	0	2	0	0	0
s_2		0	0	0	0	1	0	0
t_2		0	0	0	0	2	0	0
s_3		0	0	0	0	0	1	0
t_3		0	0	0	0	0	2	0
s_4		0	0	0	0	0	0	1
t_4		0	0	0	0	0	0	2
W		1	1	1	4	4	4	4

其他 NPC 问题

0/1 背包问题

- 定义决策问题，包含性很容易看出
- 我们如何证明其难度?
- 我们应该从哪个问题进行归约?

独立集问题 (Independent Set)

- 图G中的独立集：在图中找一个节点集合，使得集合中的任意两个节点之间没有边
- 最大独立集：给定图G，找到一个最大独立集
- 最大独立集的决策版本：给定图G和一个整数k，是否存在一个独立集，其大小至少为k?

集合打包问题 (Set Packing)

- 集合打包：给定一个元素集合U，一个子集集合S1, ..., Sm和一个整数k，是否存在至少k个集合，它们两两不相交

集合覆盖问题 (Set Cover)

- 集合覆盖：给定一个元素集合U，一个子集集合S1, ..., Sm和一个整数k，是否存在至多k个集合，它们的并集等于U

三维匹配问题 (3-Dimensional Matching)

- 三维匹配：给定不相交的集合X, Y, Z（每个集合大小为n）和一个有序三元组的集合T（T是X x Y x Z的子集），是否存在n个三元组，使得X, Y, Z中的每个元素都正好在其中一个三元组中出现一次

图的k着色问题 (k-Colouring)

- 图的k着色：给定图G，将图中的节点着色，使得任意相邻的节点颜色不同，并且使用的颜色数量不超过k
- 一个函数 f 将图 G 的节点集合 V 映射到集合 $\{1, \ldots, k\}$ 中的元素，使得对于图中的每条边 (u, v)，我们有 $f(u) \neq f(v)$

图的3着色问题 (3-Colouring)

- 图的3着色：给定图G，是否存在一个3着色方案，使得任意相邻的节点颜色不同?

有向图中的哈密顿回路：

- 在有向图中访问每个顶点恰好一次的回路。

有向图中的哈密顿路径：

- 在有向图中访问每个顶点恰好一次的路径。

哈密顿回路：

- 给定一个有向图 GG，是否存在一个哈密顿回路?

哈密顿路径：

- 给定一个有向图 GG，是否存在一个哈密顿路径?

旅行商问题：

- 参考Kleinberg和Tardos的定义（第474页）。

NP完全性问题分类法

包装问题 (Packing problems)

- 独立集 (Independent Set)
- 集合打包 (Set Packing)

覆盖问题 (Covering problems)

- 顶点覆盖 (Vertex Cover)
- 集合覆盖 (Set Cover)

分割问题 (Partitioning problems)

- 三维匹配 (3D-Matching)
- 图着色 (Graph Colouring)

序列问题 (Sequencing problems)

- 哈密顿回路 (Hamiltonian Cycle)
- 哈密顿路径 (Hamiltonian Path)
- 旅行商问题 (Traveling Salesman)

数值问题 (Numerical problems)

- 子集和 (Subset Sum)
- 背包问题 (Knapsack)

约束满足问题 (Constraint Satisfaction problems)

- 3-SAT