

组合优化

张伯雷

南京邮电大学 计算机学院、通达学院

<https://bolei-zhang.github.io/course/opt.html>



目录

- 组合优化
 - 贪心算法
 - 分治算法
- NP完全问题
 - 背包问题
 - 集合覆盖问题
 - 近似算法

一般优化问题

- 优化问题

$$\min f_0(x).$$

$$\text{s. t. } f_i(x) \leq 0, \quad i = 1, \dots, m$$

$$h_i(x) = 0, \quad i = 1, \dots, p$$

- 其中

- $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$: 优化变量(optimization variable)
- $f_0: \mathbb{R}^n \rightarrow \mathbb{R}$: 目标/损失/效用函数(Objective/loss/utility function)
- $f_i: \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \dots, m$: 不等式约束(Inequality constraint)
- $h_i: \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \dots, p$: 等式约束(Equality constraint)
- $m=p=0$: 无约束问题

组合优化问题

- 优化问题

$$\min f_0(x).$$

$$\text{s.t. } f_i(x) \leq 0, \quad i = 1, \dots, m$$

$$h_i(x) = 0, \quad i = 1, \dots, p$$

- 其中 x 需要满足整数约束
- 好消息
 - 如果 x 的取值是有限的，则总可以通过遍历所有 x 的取值得到最优解
- 坏消息
 - 遍历所有 x 的取值的复杂度是指数级的

算法复杂性

- 算法复杂度一般表示为输入大小的函数
- 大O：当对于足够大的 x ， $f(x)$ 最多为 $g(x)$ 的常数倍，则 $f(x) = O(g(x))$
- 最差时间复杂度
- 平均时间复杂度



常用的算法设计模式

- 分治法
- 贪心法
- 动态规划
- 对偶
- 归约
- 局部搜索
- 随机

分治算法

- 分治

- 将原始问题划分为几个同类的子问题
- 递归解决每个子问题
- 将子问题的结果合并

- 效果

- 原始问题: $O(n^2)$
- 分治法: $O(n \log n)$

排序问题

- 对以下的字母序列进行排序

A	L	G	O	R	I	T	H	M	S
---	---	---	---	---	---	---	---	---	---

世界纪录！腾讯云98.8秒处理100TB数据



2016-11-10 23:57:00 出处：快科技 作者：[上方文Q](#) 编辑：[上方文Q](#) 人气：[6906 次](#) 

[腾讯](#) [云计算](#)

11月10日，有计算奥运会之称的Sort Benchmark全球排序竞赛公布了2016年最终成绩，腾讯云大数据联合团队用时仅仅98.8秒，就完成了100TB的数据排序。

这打破了阿里云去年创造的329秒纪录，而在更早的时候，百度曾经的纪录是716秒，Hadoop的纪录是4222秒。

在这次竞赛中，腾讯云数智分布式计算平台夺得了GraySort、MinuteSort两个项目的冠军，同时创造四项世界纪录，将阿里云去年的纪录整体提高了2-5倍。

合并排序

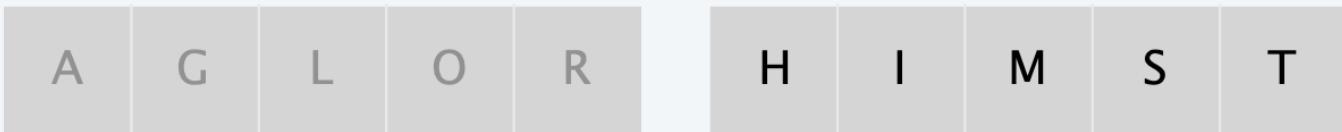
input



sort left half



sort right half



merge results





合并排序算法

MERGE-SORT(L)

IF (list L has one element)

RETURN L .

Divide the list into two halves A and B .

$A \leftarrow \text{MERGE-SORT}(A).$ $\longleftarrow T(n / 2)$

$B \leftarrow \text{MERGE-SORT}(B).$ $\longleftarrow T(n / 2)$

$L \leftarrow \text{MERGE}(A, B).$ $\longleftarrow \Theta(n)$

RETURN L .

合并排序的算法复杂度

$$T(n) \leq \begin{cases} 0 & \text{if } n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n & \text{if } n > 1 \end{cases}$$

↑
between $\lfloor n/2 \rfloor$ and $n - 1$ compares

- $T(n)$ 的复杂度为 $O(n \log n)$

快速排序

- 随机挑选一个pivot
- 根据pivot将算法分为左、右两个部分
- 递归解决左右两个子问题

the array A 7 6 12 3 11 8 9 1 4 10 2

p

partition A 3 1 4 2 6 7 12 11 8 9 10

sort L 1 2 3 4 6 7 12 11 8 9 10

sort R 1 2 3 4 6 7 8 9 10 11 12

the sorted array A 1 2 3 4 6 7 8 9 10 11 12



快速排序

RANDOMIZED-QUICKSORT(A)

IF (array A has zero or one element)

RETURN.

Pick pivot $p \in A$ uniformly at random.

$(L, M, R) \leftarrow \text{PARTITION-3-WAY}(A, p).$ $\longleftarrow \Theta(n)$

RANDOMIZED-QUICKSORT(L). $\longleftarrow T(i)$

RANDOMIZED-QUICKSORT(R). $\longleftarrow T(n - i - 1)$

贪心算法

• 贪心

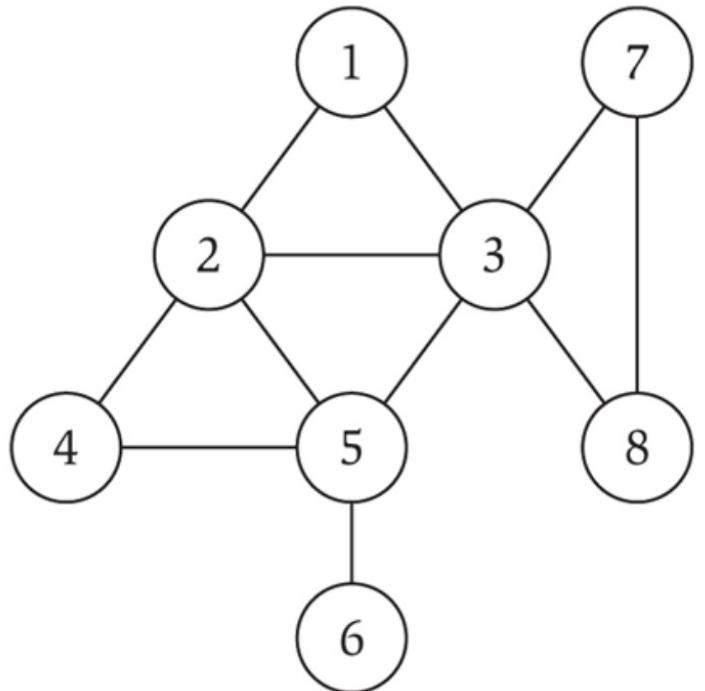
- 在算法的每一步，选择一个局部最优、或者单步最好的解
- 对算法进行迭代，直到产生最终正确的解
- 在一些特殊的情况下，可以达到全局最优

图的基本定义

- 图的表示. $G = (V, E)$

- $V =$ 节点集合
- $E =$ 节点之间的边的集合.

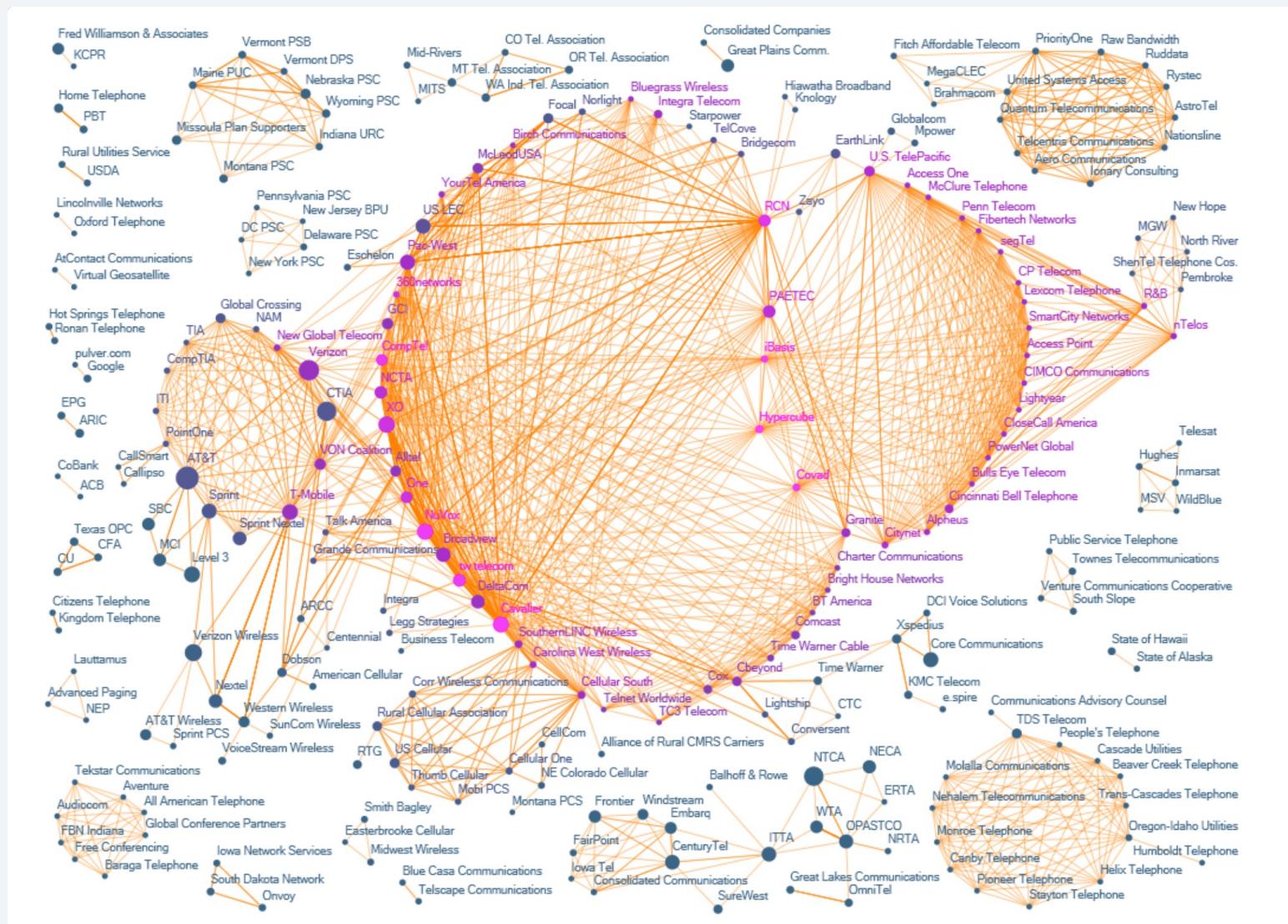
- 图可以刻画不同物体之间的关系



$$V = \{1, 2, 3, 4, 5, 6, 7, 8\}$$

$$E = \{1-2, 1-3, 2-3, 2-4, 2-5, 3-5, 3-7, 3-8, 4-5, 5-6, 7-8\} \quad m = 11, n = 8$$

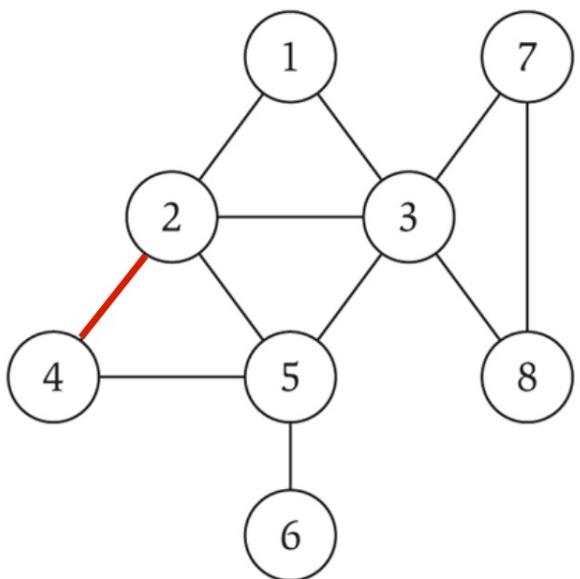
文献引用网络



图的表示：邻接矩阵

- 邻接矩阵
 - $A_{uv} = 1$ 如果 (u, v) 之间有边

- 空间复杂度为 n^2 .
- 检查两个点是否有边相连的复杂度为 $O(1)$

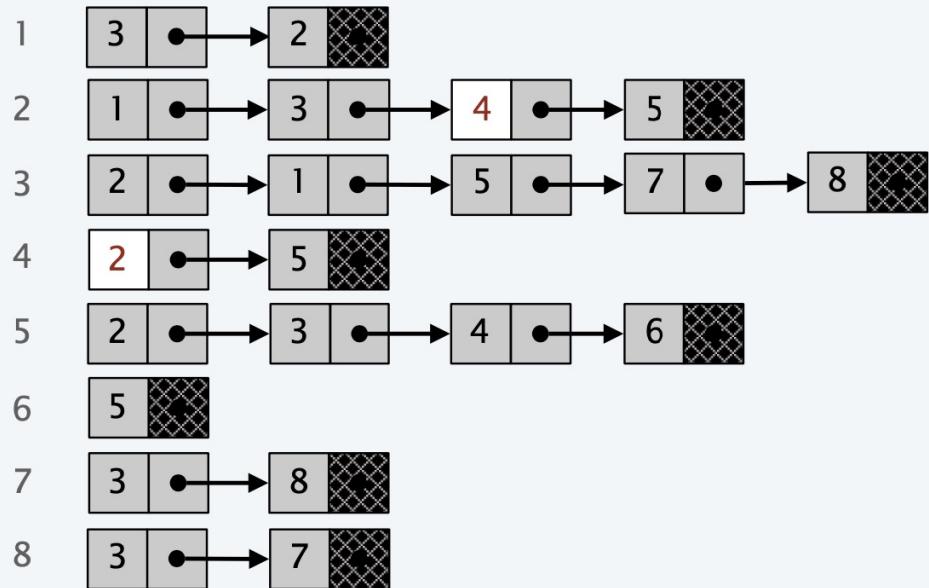
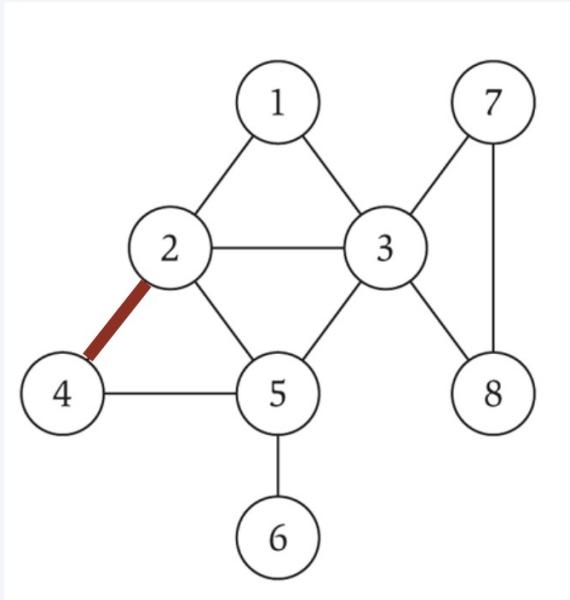


	1	2	3	4	5	6	7	8
1	0	1	1	0	0	0	0	0
2	1	0	1	1	1	0	0	0
3	1	1	0	0	1	0	1	1
4	0	1	0	0	1	0	0	0
5	0	1	1	1	0	1	0	0
6	0	0	0	0	1	0	0	0
7	0	0	1	0	0	0	0	1
8	0	0	1	0	0	0	1	0

图的表示：邻接列表

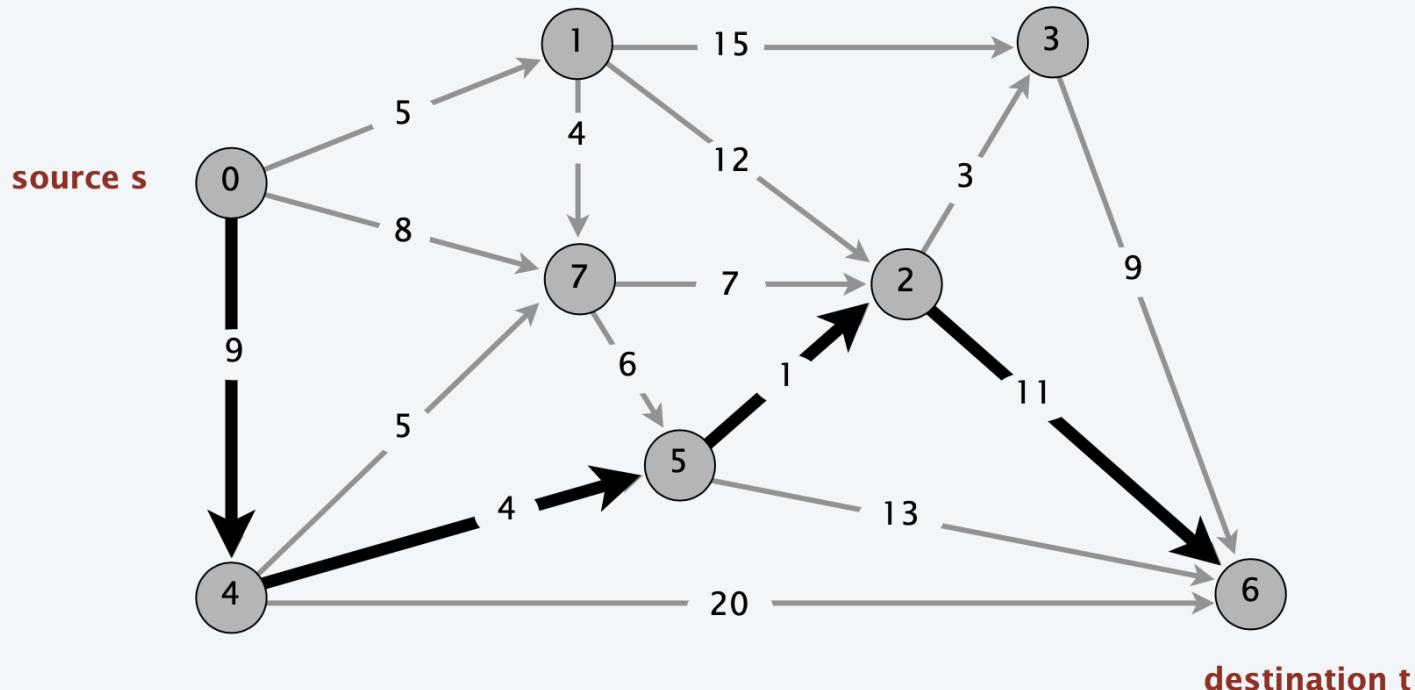
- 邻接链表

- 空间复杂度为 $\Theta(m + n)$.
- 检查两个点是否有边相连的复杂度为 $O(d(u))$



最短路径问题

- 给定一个图 $G=(V, E)$, 找到源节点 s 到目的节点 v 的最短路径



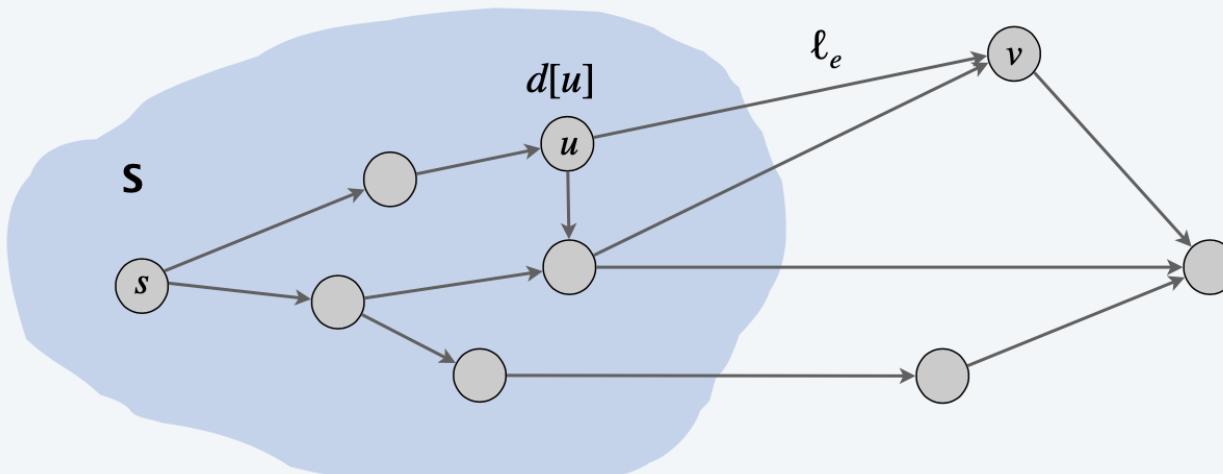
迪杰斯特拉算法 Dijkstra's algorithm

- 贪婪的方法:

- 维护一组已探索的节点 S , 其中算法已确定 $d[u] = \text{最短 } s \rightarrow u \text{ 路径的长度}.$
- 初始化 $S \leftarrow \{s\}$, $d[s] \leftarrow 0$. 重复选择未探索的节点 $v \notin S$, 从而最小化

$$\pi(v) = \min_{e = (u,v) : u \in S} d[u] + \ell_e$$

the length of a shortest path from s to some node u in explored part S , followed by a single edge $e = (u, v)$



Dijkstra's algorithm

- 贪婪的方法:

- 维护一组已探索的节点 S , 其中算法已确定 $d[u] = \text{最短 } s \rightarrow u \text{ 路径的长度。}$
- 初始化 $S \leftarrow \{s\}$, $d[s] \leftarrow 0$ 。重复选择未探索的节点 $v \notin S$, 从而最小化

$$\pi(v) = \min_{e = (u,v) : u \in S} d[u] + \ell_e$$

the length of a shortest path from s to some node u in explored part S , followed by a single edge $e = (u, v)$

- 将 v 添加到 S , 并设置 $d[v] \leftarrow \pi(v)$ 。
- 要恢复路径, 设置 $\text{pred}[v] \leftarrow e$ 即可达到最小值。



思考

- 1.一个txt文件里包含100亿个整数，如何进行排序
- 2.如何找出数组中最大的k个数字

组合优化

张伯雷

南京邮电大学 计算机学院、通达学院

<https://bolei-zhang.github.io/course/opt.html>



多项式归约

- 假设我们可以在多项式时间内解决问题 Y。 我们还能在多项式时间内解决什么问题？
- 归约：如果问题 X 的**任意**实例可以使用以下方法求解，则问题 X 多项式时间可归约为问题 Y：
 - 多项式次数的标准计算（加减乘除等），加上
 - 多项式次数的解决问题 Y 的 oracle （查询结果）调用
- 符号 $X \leq_P Y$ 。

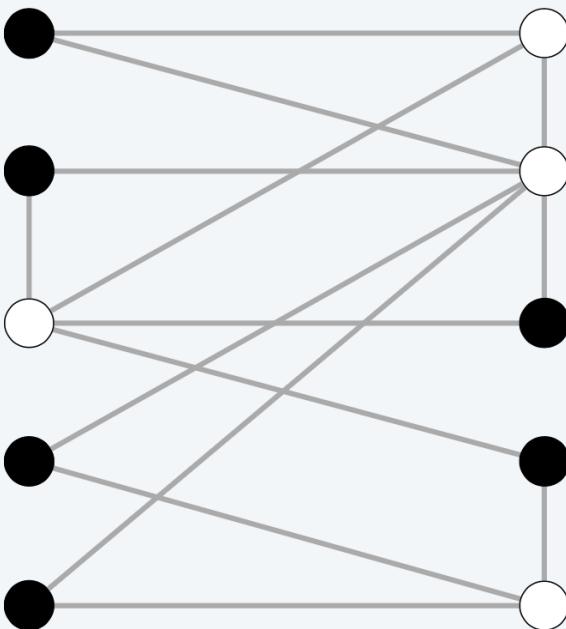


多项式归约

- 设计算法：如果 $X \leq_P Y$ 并且 Y 可以在多项式时间内求解，则 X 可以在多项式时间内求解。
- 不可解性 (Intractability) : $X \leq_P Y$ 并且 X 无法在多项式时间内求解，则 Y 也无法在多项式时间内求解。
- 等价性 (Equivalency) : 如果 $X \leq_P Y$ 且 $Y \leq_P X$ ，使用符号 $X \equiv_P Y$ 。在这种情况下， X 可以在多项式时间内求解，当且仅当 Y 可以。

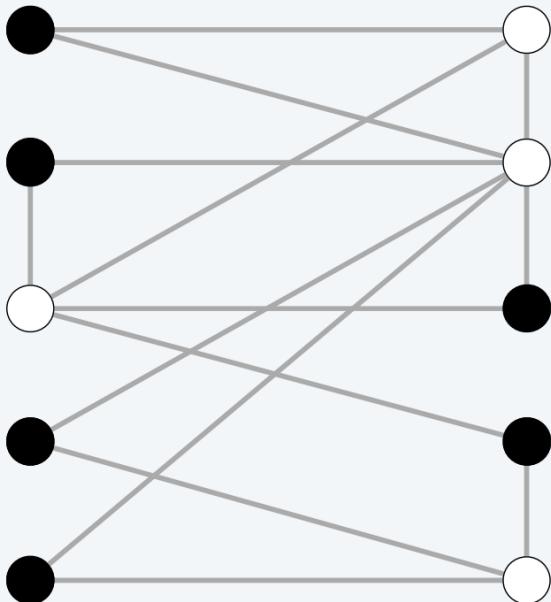
独立集合问题

- 独立集合问题：给定一个图 $G = (V, E)$ 和一个整数 k , 是否存在 k 个（或更多）顶点的子集使得没有两个顶点相邻？
- 例：是否存在大小 ≥ 6 的独立集合？
- 例：是否存在大小 ≥ 7 的独立集合？



顶点覆盖问题

- 顶点覆盖：给定一个图 $G = (V, E)$ 和一个整数 k ，是否存在 k 个（或更少）个顶点的子集，使得每条边都与子集中的至少一个顶点相连？





顶点覆盖与独立集合可以互相归约

- 定理：独立集合 \equiv_P 顶点覆盖。
- 证明：我们证明 S 是大小为 k 的独立集合，当且仅当 $V - S$ 是大小为 $n - k$ 的顶点覆盖。
 - ($=>$)
 - 令 S 为任意大小为 k 的独立集。
 - $V - S$ 的大小为 $n - k$ 。
 - 考虑任意边 $(u, v) \in E$ 。
 - S 独立
 - $\Rightarrow u \notin S$, 或 $v \notin S$, 或同时成立。
 - $\Rightarrow u \in V - S$, 或 $v \in V - S$, 或同时成立。
 - 因此, $V - S$ 覆盖 (u, v) 。

顶点覆盖与独立集合可以互相归约

- (\Leftarrow)
 - 令 $V - S$ 为任意大小为 $n - k$ 的顶点覆盖。
 - S 的大小为 k 。
 - 考虑任意边 $(u, v) \in E$ 。
 - $V - S$ 是顶点覆盖
 - $\Rightarrow u \in V - S$ ，或 $v \in V - S$ ，或同时成立。
 - $\Rightarrow u \notin S$ ，或 $v \notin S$ ，或同时成立。
 - 因此， S 是独立集。



布尔可满足性问题(SAT)

- 命题逻辑公式

- 也称为布尔表达式，由变量，运算符AND（连接，也用 \wedge 表示），OR（分离， \vee ），NOT（否定， \neg ）和括号构成。如果通过为其变量分配适当的逻辑值（即TRUE, FALSE）可以使公式为TRUE，则称该公式是可满足的。
给定公式，布尔可满足性问题（SAT）是检查它是否可满足。

- 3-SAT

- 3-SAT问题是指出对一个布尔表达式，其中每个子句（clause）中只包含3个布尔变量或它们的否定形式，判断是否存在一组布尔变量取值，使得整个布尔表达式的值为TRUE。

可满足性

- 文字 (Literal) : 布尔变量或其否定。

x_i or $\overline{x_i}$

- 子句 (Clause) : 文字的析取。

$C_j = x_1 \vee \overline{x_2} \vee x_3$

- 合取范式 (Conjunctive normal Form, CNF) : 一个由

$\Phi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

多个子句通过合取形成的范式

- SAT: 给定一个 CNF 公式 Φ , 它是否有满足合取范式为真的分配?

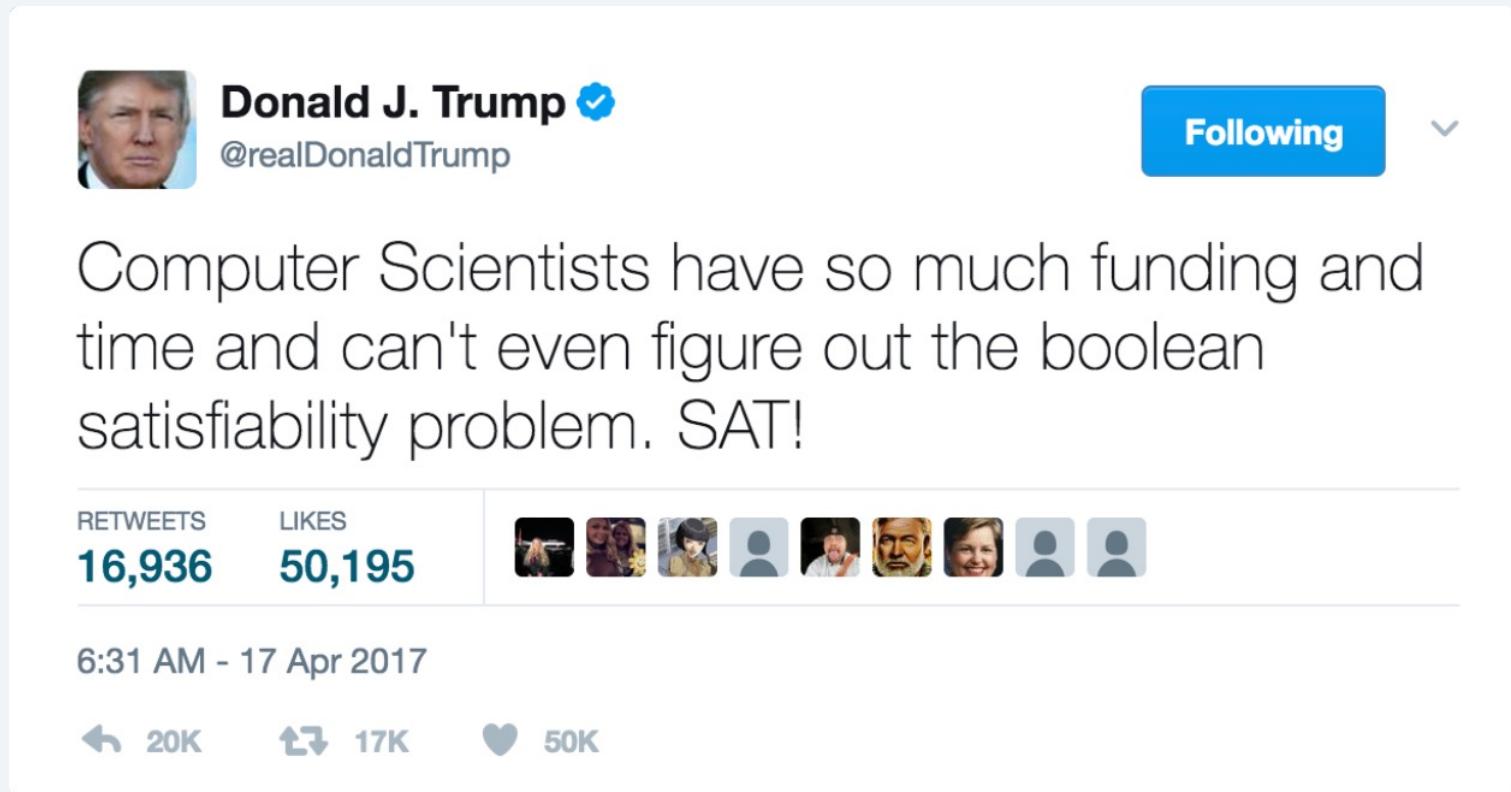
- 3-SAT: 其中每个子句恰好包含 3 个文字 (每个文字对应一个不同的变量)。

$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

yes instance: $x_1 = \text{true}, x_2 = \text{true}, x_3 = \text{false}, x_4 = \text{false}$

布尔可满足性问题

- 科学假设：3-SAT问题没有可行的多项式算法。



Donald J. Trump 
@realDonaldTrump

Following

Computer Scientists have so much funding and time and can't even figure out the boolean satisfiability problem. SAT!

RETWEETS 16,936 LIKES 50,195

6:31 AM - 17 Apr 2017

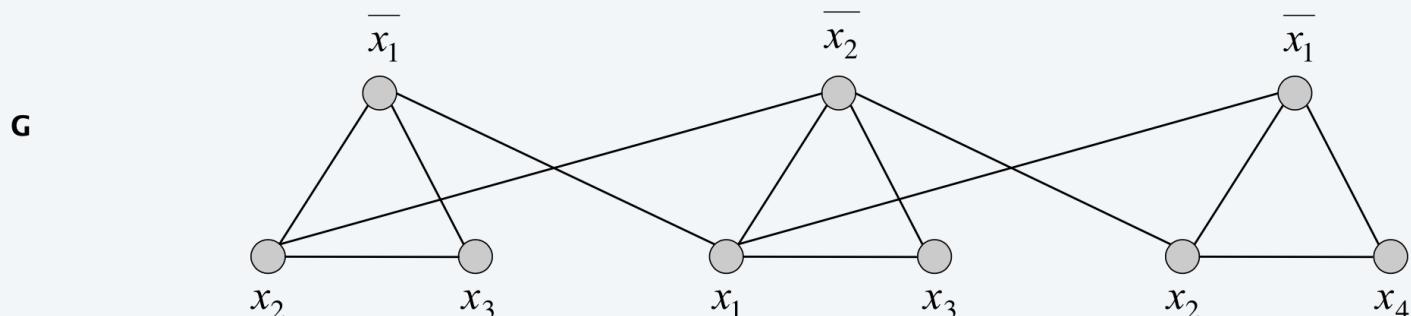
20K 17K 50K

3-SAT可以多项式归约到独立集合问题

- 定理. $3\text{-SAT} \leq_P \text{INDEPENDENT-SET}$.
- 证明：任意给定一个3-SAT的实例 Φ ，都可以构建一个关于独立集合的图 (G, k) ，使得该独立集合具有大小为 $k = |\Phi|$ 的解时，当且仅 $|\Phi|$ 是可满足的.

• 构建

- G 中的每个子句包含3个节点，每个节点代表一个文字
- 将三个子句连接为一个三角形
- 将每个文字与其他文字的否定连接起来



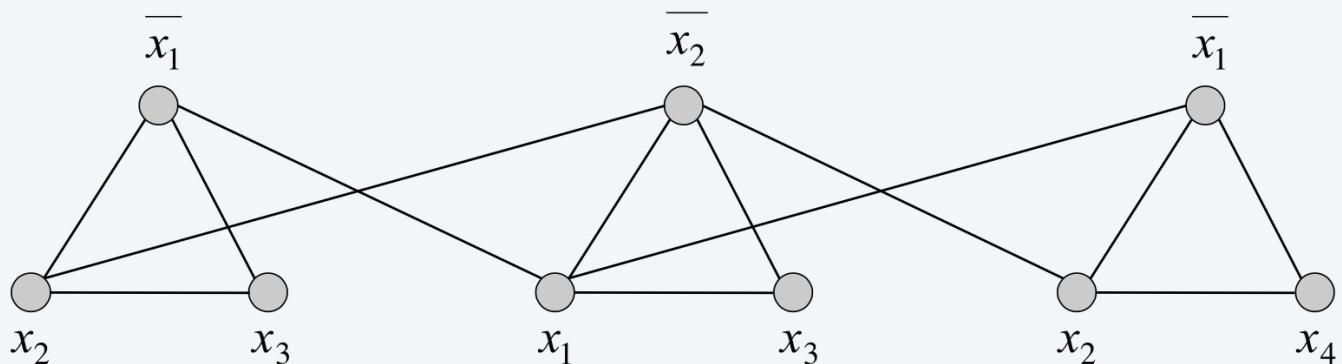
$k = 3$

$$\Phi = (\overline{x_1} \vee x_2 \vee x_3) \wedge (x_1 \vee \overline{x_2} \vee x_3) \wedge (\overline{x_1} \vee x_2 \vee x_4)$$

3-SAT可以多项式归约到独立集合问题

- 引理: Φ 是可满足的当且仅当 G 包含有大小为 $k = |\Phi|$ 的独立集合
- 证明: \Rightarrow 考虑任意一个3-SAT问题的分配
 - 从每个子句 (三角形) 中选择一个为真的文字 (节点)
 - 则这个解为一个独立集合

G



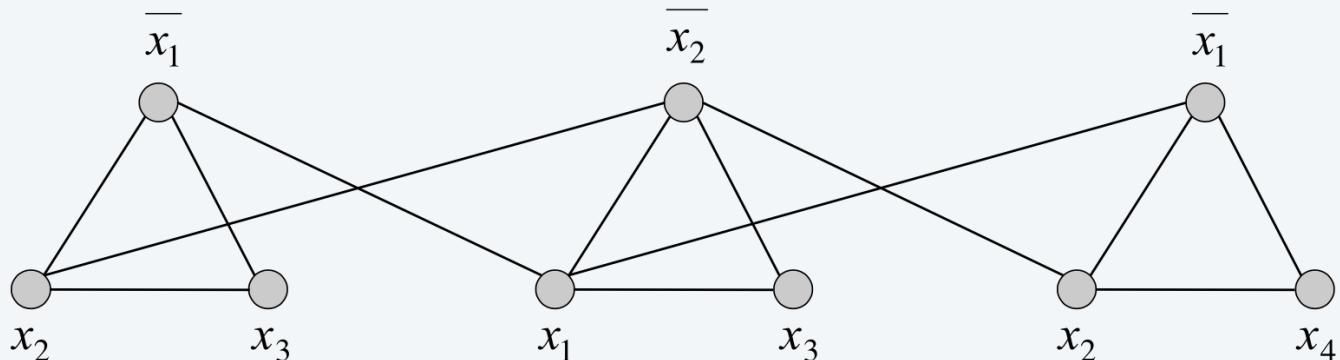
k = 3

$$\Phi = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4)$$

3-SAT可以多项式归约到独立集合问题

- 引理: Φ 是可满足的当且仅当 G 包含有大小为 $k = |\Phi|$ 的独立集合
- 证明: \Leftarrow 考虑任意一个大小为 k 的独立集合 S
 - 在每个三角形中, S 肯定包含唯一一个为真的节点
 - 将这些文字设置为真 (其余的可以按照一致性设置)
 - Φ 所有的子句都可以被满足

G



k = 3

$$\Phi = (\bar{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee x_2 \vee x_4)$$



决策、搜索与优化

- 决策问题：是否存在一个小于等于 k 的节点覆盖集合
- 搜索问题：找到一个小于等于 k 的节点覆盖集合
- 优化问题：找到一个最小的节点覆盖集合
- 目标：以上的三个问题可以互相（多项式）归约



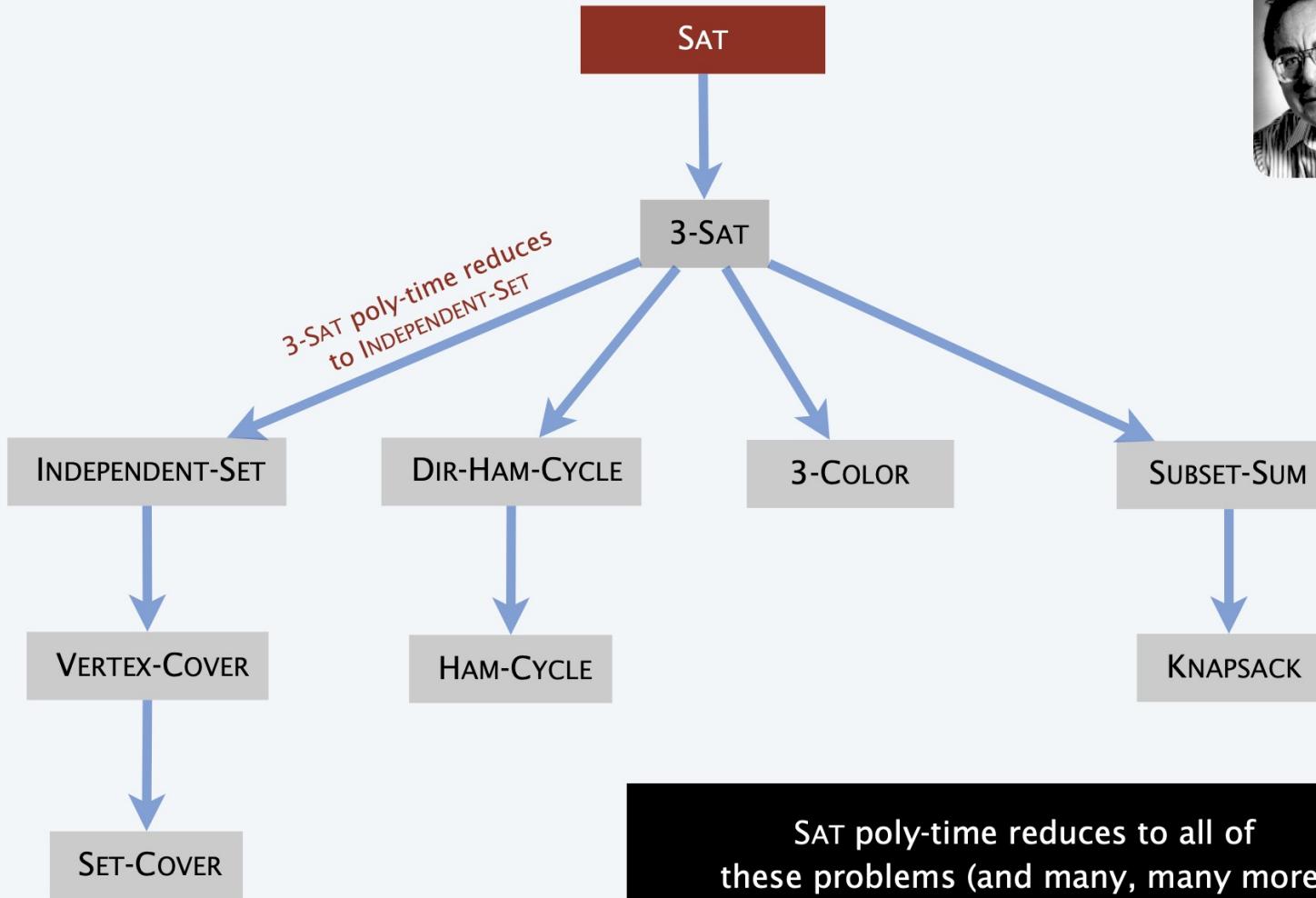
决策与搜索

- 顶点覆盖：是否存在大小 $\leq k$ 的顶点覆盖？
- 搜索顶点覆盖：找到大小 $\leq k$ 的顶点覆盖。
- 定理：顶点覆盖 \equiv_P 搜索顶点覆盖。
- 证明： \leq_P 决策问题是搜索问题的一个特例。
- 证明： \geq_P
 - 为了找到大小 $\leq k$ 的顶点覆盖：
 - 判断是否存在大小 $\leq k$ 的顶点覆盖。
 - 找到一个顶点 v ，使得 $G - \{v\}$ 的顶点覆盖大小 $\leq k - 1$ 。
(任何大小 $\leq k$ 的顶点覆盖中的任何顶点都将具有此属性)
 - 在顶点覆盖中包含 v 。
 - 在 $G - \{v\}$ 中递归查找大小 $\leq k - 1$ 的顶点覆盖

决策、搜索与优化

- 搜索顶点覆盖：找到大小 $\leq k$ 的顶点覆盖
- 最优顶点覆盖：找到最小的顶点覆盖。
- 定理：搜索顶点覆盖 \equiv_P 最优顶点覆盖。
- 证明： \leq_P 上搜索问题是优化问题的一个特例。
- 证明： \geq_P
 - 为了找到最小的顶点覆盖：
 - 通过二分搜索找到大小为 k^* 的最小顶点覆盖
 - 解决对于给定 k^* 的搜索问题

常见的一些问题



- **决策问题**

- 问题X为一个集合的字符串
- 实例s为其中的一个字符串
- 算法A可以解决问题X: $A(s) = \begin{cases} yes, & if s \in X \\ no, & if s \notin X \end{cases}$
- 定义: **P** 所有决策问题中具有多项式算法的问题集合.
- 解释: 一个决策问题是一个取一些字符串为输入，并要求输出为是或否的问题。若有一个算法（譬如图灵机）能够在最多 n^k 步内对一个串长度为n的输入给出正确答案，其中k是某个不依赖于输入串的常数，则我们称该问题可以在多项式时间内解决，并且将它置入类P



- 现在假设有一个算法 $A(w,C)$ 取两个参数，一个串 w ，也就是我们的决定问题的输入串，而另一个串 C 是“建议证明”(certifier)，并且使得 A 在最多 n^k 步内产生“是／否”答案（其中 n 是 w 的长度，而 k 不依赖于 w ）
- 然后假设： w 是一个答案为“是”的例子，当且仅当，存在 C 使得 $A(w,C)$ 返回“是”。则我们称这个问题可以在非决定性多项式时间内解决，且将它放入NP类。我们把算法 A 作为一个所建议的证明的检验器，它运行足够快。
- 不能确定在多项式时间内找到答案，但是可以在多项式时间内验证答案是否正确

P, NP, EXP

- **P:** 存在多时间算法的决策问题。
- **NP:** 存在多时间验证者的决策问题。
- **EXP:** 存在指数时间算法的决策问题。
- 命题： $P \subseteq NP$ 。
- 命题： $NP \subseteq EXP$ 。
- 事实： $P \neq EXP \Rightarrow P \neq NP$, 或 $NP \neq EXP$, 或同时成立。

计算机科学的核心问题

- $P = NP?$ [Cook 1971, Edmonds, Levin, Yablonski, Gödel]

$P \neq NP$

“I conjecture that there is no good algorithm for the traveling salesman problem. My reasons are the same as for any mathematical conjecture: (i) It is a legitimate mathematical possibility and (ii) I do not know.”

— *Jack Edmonds 1966*



“In my view, there is no way to even make intelligent guesses about the answer to any of these questions. If I had to bet now, I would bet that P is not equal to NP. I estimate the half-life of this problem at 25–50 more years, but I wouldn’t bet on it being solved before 2100. ”

— *Bob Tarjan (2002)*



计算机科学的核心问题

P = NP

“ I think that in this respect I am on the loony fringe of the mathematical community: I think (not too strongly!) that P=NP and this will be proved within twenty years. Some years ago, Charles Read and I worked on it quite bit, and we even had a celebratory dinner in a good restaurant before we found an absolutely fatal mistake. ”

— Béla Bollobás (2002)

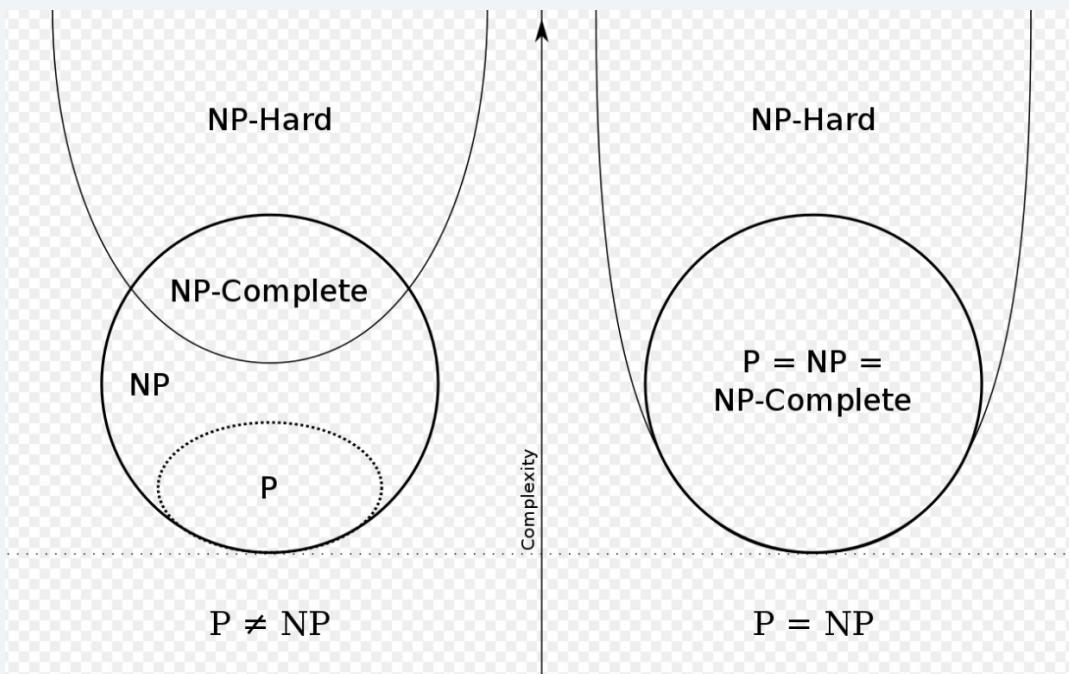


“ In my opinion this shouldn't really be a hard problem; it's just that we came late to this theory, and haven't yet developed any techniques for proving computations to be hard. Eventually, it will just be a footnote in the books. ” — John Conway



NP-complete

- NP 完全：问题 $Y \in NP$ 具有以下属性：对于每个问题 $X \in NP$, $X \leq_P Y$ 。
 - 1. Y 为一个NP问题
 - 2. 所有的NP问题都可以在多项式时间内归约到 Y
- 命题：假设 $Y \in NP$ 完全问题。那么， $Y \in P$ 当且仅当 $P = NP$



第一个NP完全问题

Theorem. [Cook 1971, Levin 1973] SAT \in NP-complete.

<p>The Complexity of Theorem-Proving Procedures Stephen A. Cook University of Toronto</p> <p>Summary</p> <p>It is shown that any recognition problem solved by a polynomial time-bounded nondeterministic Turing machine can be "reduced" to the problem of determining whether a given propositional formula is a tautology. Here "reduced" means, roughly speaking, that the first problem can be solved deterministically in polynomial time provided an oracle is available for solving the second. From this notion of reducible, polynomial degrees of difficulty are defined, and it is shown that the problem of determining tautologyhood has the same polynomial degree as the problem of determining whether the first of two given graphs is isomorphic to a subgraph of the second. Other examples are discussed. A method of measuring the complexity of proof procedures for the predicate calculus is introduced and discussed.</p> <p>Throughout this paper, a set of strings means a set of strings on some fixed, large, finite alphabet Σ. This alphabet is large enough to include symbols for all sets described here. All Turing machines are deterministic recognition devices, unless the contrary is explicitly stated.</p> <p>1. Tautologies and Polynomial Reducibility.</p> <p>Let us fix a formalism for the propositional calculus in which formulas are written as strings on Σ. Since we will require infinitely many proposition symbols (atoms), each such symbol will consist of a member of Σ followed by a number in binary notation to distinguish that symbol. Thus a formula of length n will have at least $1/n$ distinct function and predicate symbols. The logical connectives are \wedge (and), \vee (or), and \neg (not).</p> <p>The set of tautologies (denoted by {tautologies}) is a</p>	<p>certain recursive set of strings on this alphabet, and we are interested in the problem of finding a good lower bound on its possible recognition times. We provide no such lower bound here, but theorem 1 will give evidence that {tautologies} is a difficult set to recognize, since many apparently difficult problems can be reduced to determining tautologyhood. By reduced we mean, roughly speaking, that if tautologyhood could be decided instantly (by an "oracle") then these problems could be decided in polynomial time. In order to make this notion precise, we introduce query machines, which are like Turing machines with oracles in [1].</p> <p>A query machine is a multitape Turing machine with a distinguished tape called the query tape, and three distinguished states called the query state, yes state, and no state, respectively. If M is a query machine and T is a set of strings, then a T-computation of M is a computation of M in which initially M is in the initial state and has an input string w on its input tape, and each time M assumes the query state there is a string u on the query tape, and the next state M assumes is the yes state if $u \in T$ and the no state if $u \notin T$. We think of an "oracle", which knows T, placing M in the yes state or no state.</p> <p>Definition</p> <p>A set S of strings is P-reducible (P for polynomial) to a set T of strings iff there is some query machine M and a polynomial $Q(n)$ such that for each input string w, the T-computation of M with input w halts within $Q(w)$ steps (w is the length of w), and ends in an accepting state iff $w \in S$.</p> <p>It is not hard to see that P-reducibility is a transitive relation. Thus the relation \leq on</p>
---	--

ПРОБЛЕМЫ ПЕРЕДАЧИ ИНФОРМАЦИИ
Том IX 1973 Вып. 3

КРАТКИЕ СООБЩЕНИЯ

УДК 519.14

УНИВЕРСАЛЬНЫЕ ЗАДАЧИ ПЕРЕБОРА

Д. А. Лесин

В статье рассматривается несколько известных массовых задач «переборного типа» и доказывается, что эти задачи можно решать лишь за такое время, за которое можно решать вообще любые задачи указанного типа.

После уточнения понятия алгоритма была доказана алгоритмическая неразрешимость ряда классических массовых проблем (например, проблем тождества элементов групп, гомеоморфности многообразий, разрешимости диофантовых уравнений и других). Тогда же было предложено исследовать задачи о нахождении минимальной степени сплошного решения для задач, не связанных с решением аналогичных вопросов для них аналитического типа. Такова ситуация с так называемыми переборными задачами: минимизация булевых функций, поиска доказательства ограниченной длины, выяснение изоморфизмы графов и другими. Все эти задачи решаются тривиальными алгоритмами, состоящими в переборе всех возможностей. Однако эти алгоритмы требуют экспоненциального времени работы и у математиков сложилось убеждение, что более быстрое алгоритма для них нет. Был получен ряд серий аргументов в пользу этого, но спорят о том, что для доказательства утверждения не удалось никому. (Например, до сих пор не доказано, что для нахождения математических доказательств нужно больше времени, чем для их проверки.)

Однако если предположить, что вообще существует какое-нибудь (хотя бы искусственно построенное) массовая задача переборного типа, неразрешимая простыми (в смысле объема вычислений) алгоритмами, то можно показать, что этим же способом обладают и многие «классические» переборные задачи (в том числе задача минимизации задачи поиска доказательства и др.). В этом и состоят основные результаты статьи.

Функции $f(n)$ и $g(n)$ будем называть сравнимыми, если при некотором k

$$f(n) \leq (g(n+2))^k \text{ и } g(n) \leq (f(n+2))^k.$$

Аналогично будем называть задачи сравнимыми (или просто переборной задачей).

Определение. Задача называется данному x найти какое-нибудь y в Δ_n , сравнимой с данной x , такое, что выполняется $A(x, y)$, где $A(x, y)$ — какое-нибудь свойство, проверяемое алгоритмом, время работы которого сравнимо с данной x . (Под алгоритмом здесь можно понимать, например, алгоритмы Колмогорова — Успенского или машины Тьюринга, или нормальные алгоритмы x, y — двоичные слова). Квазиперебором называется задача, для которой x сравнимо с y в смысле перебора y .

Мы рассмотрим шесть задач этой группы. Рассматриваемые в них объекты кодируются естественным образом в виде двоичных слов. При этом выбор естественной кодировки не существует, так как все они дают сравнимые длины кодов.

Задача 1. Заданы список конечное множество и покрытие его 500-элементными подмножествами. Найти поддоктрины заданной мощности (соответственно выяснить существует ли она).

Задача 2. Таблично задана частичная булева функция. Найти заданного размера дизъюнктивную нормальную форму, реализующую эту функцию в области определения (соответственно выяснить существует ли она).

Задача 3. Выяснить, выводима или опровергнута данная формула исчисления высказываний. (Или, что то же самое, равна ли константе данная булева формула.)

Задача 4. Даны два графа. Найти гомоморфизм одного на другой (выяснить его существование).

Задача 5. Даны два графа. Найти изоморфизм одного в другой (на его часть).

Задача 6. Рассматривается матрица из целых чисел от 1 до 100 и некоторое условие о том, какие числа в них могут соседствовать по вертикали и какие по горизонтали. Заданы числа на границе и требуется предложить их на всю матрицу с соблюдением условий.



NP-complete的证明

- 提示：一旦建立了第一个NP完全问题，可以进行推广。
- 方法：要证明 $Y \in \text{NP-complete}$:
 - Step 1. 证明 $Y \in \text{NP}$.
 - Step 2. 选择一个NP完全问题 X .
 - Step 3. 证明 $X \leq_P Y$

组合优化

张伯雷

南京邮电大学 计算机学院、通达学院

<https://bolei-zhang.github.io/course/opt.html>

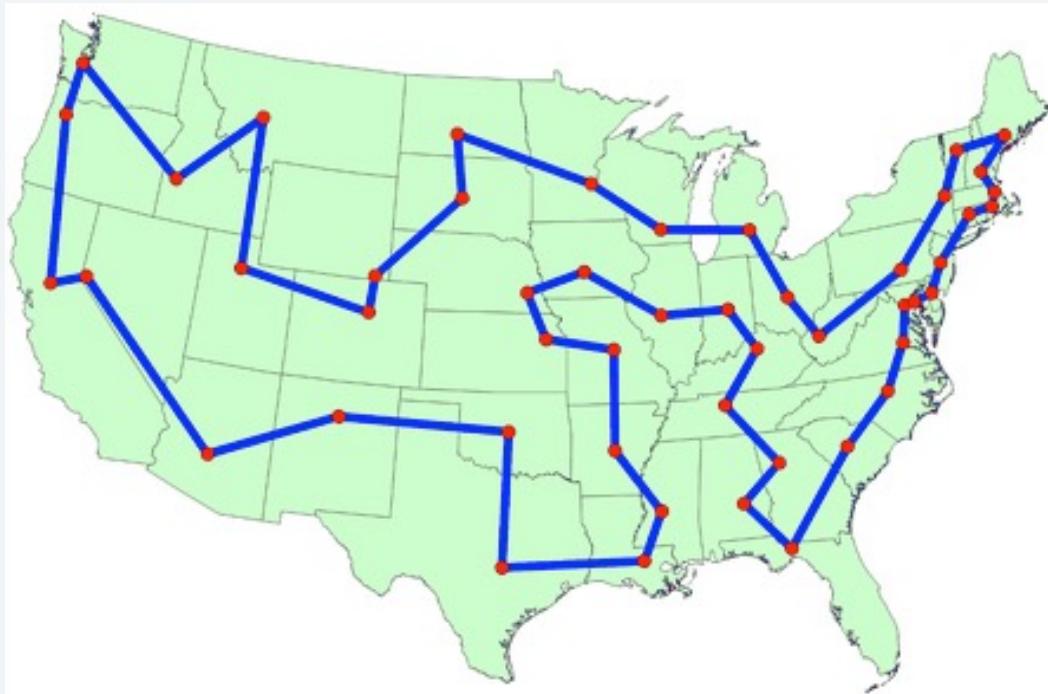


目录

- 组合优化
- NP完全问题
- 近似算法

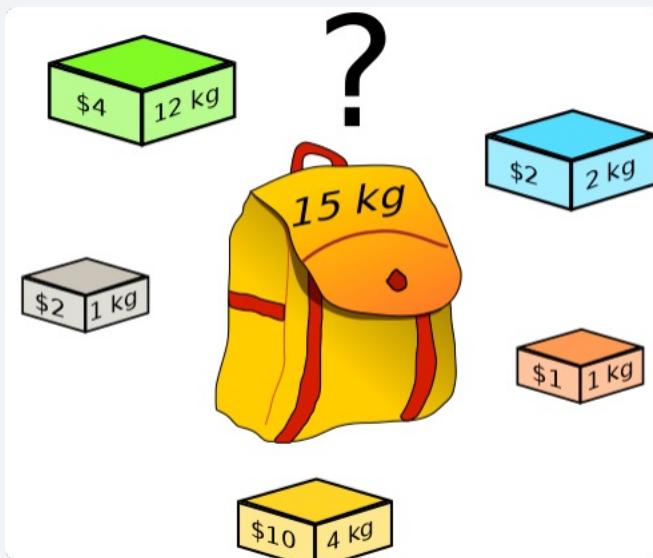
旅行商問題

- 問題：一個商品推銷員要去若干個城市推銷商品，該推銷員從一個城市出發，需要經過所有城市後，回到出發地。應如何選擇行進路線，以使總的行程最短。



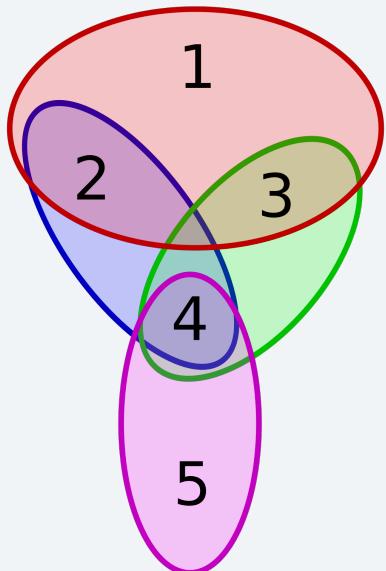
背包問題

- 給定一组物品，每种物品都有自己的重量和价格，在限定的总重量内，我们如何选择，才能使得物品的总价格最高。



集合覆盖问题

- 给定全集 U , 以及一个包含 n 个集合且这 n 个集合的并集为 U 的集合 S , 集合覆盖问题是找到 S 的一个最小子集, 使得他们的并集为 U



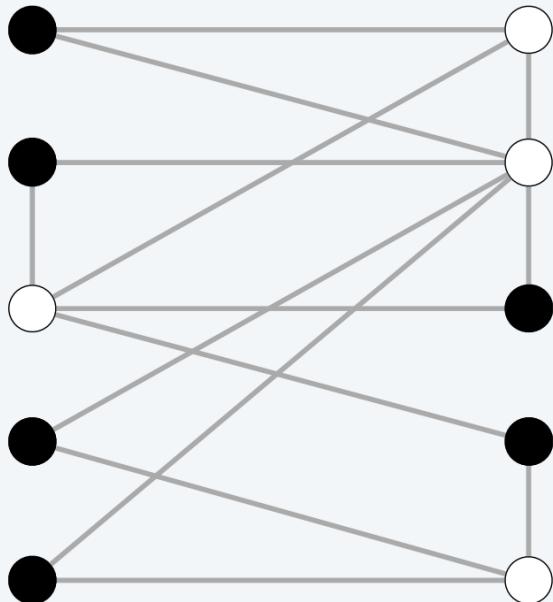


近似算法

- 贪心法
- 放松法
- 对偶法

顶点覆盖问题

- 顶点覆盖：给定一个图 $G = (V, E)$ 和一个整数 k ，是否存在 k 个（或更少）个顶点的子集，使得每条边都与子集中的至少一个顶点相连？





贪心法

Input: An undirected graph G

Result: A vertex cover.

Let $T = \emptyset$;

while Some edge $\{u, v\}$ of G is uncovered **do**

| Add both u and v to T ;

end

Output T .



贪心法分析

- 令 $\{e_1, e_2, \dots, e_k\}$ 贪心法算法选择的边。
- 根据算法， T 是这些边的端点集合，即 $|T| = 2k$ ，并且这些边中没有两条边共享顶点。
- 为了证明这一说法，只要证明 $OPT \geq k$ 就足够了。
- 考虑图 H 仅由这 k 个边组成（因此 H 有 $2k$ 个顶点和 k 个边）。
- 最优解也是 H 的顶点覆盖。但它必须有每条边的一个端点 e_1, e_2, \dots, e_k ，因为这些边不共享顶点。
- 因此，该算法在最坏情况下的近似率最多为 2。
- 贪心法为顶点覆盖的2-approximation的算法

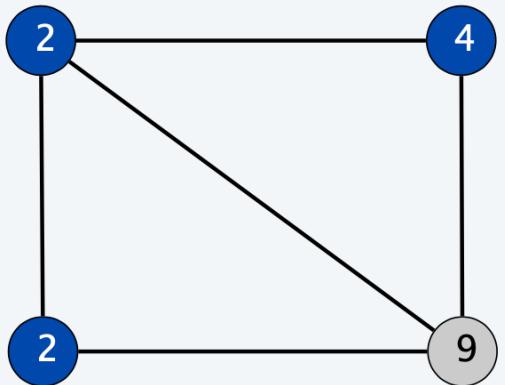


近似算法

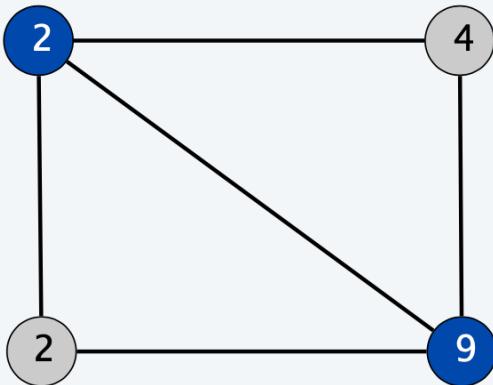
- 贪心法
- 放松法
- 对偶法

加权顶点覆盖问题

- 加权顶点覆盖： 给定一个图 $G = (V, E)$ ， 每个节点上都有一个权重， 找到最小的权重节点集合，使得所有的边都可以被覆盖？

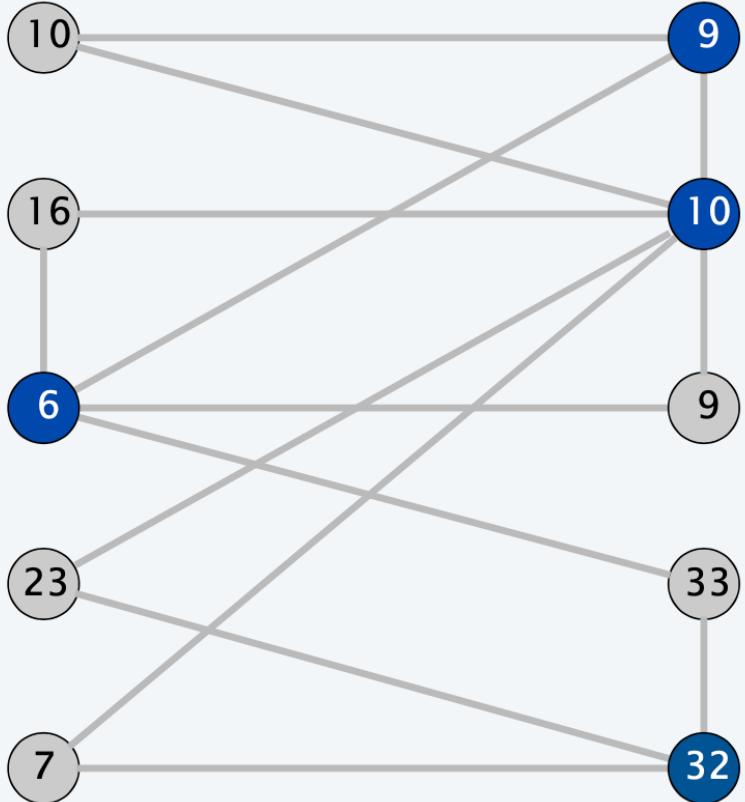


weight = 2 + 2 + 4



weight = 11

加权顶点覆盖问题



$$\text{total weight} = 6 + 9 + 10 + 32 = 57$$

整数优化问题

- 整数优化建模

- 对每个节点，用0/1变量来代表这个节点是否被选择

$$x_i = \begin{cases} 1, & \text{若 } x_i \text{ 在选择的集合中} \\ 0, & \text{否则} \end{cases}$$

- 节点集合覆盖问题，可以建模为以下的优化问题：

$$\begin{aligned} & \min \sum_{i \in V} w_i x_i \\ \text{s. t. } & x_i + x_j \geq 1, \forall (i, j) \in E \\ & x_i \in \{0, 1\}, i \in V \end{aligned}$$

- 如果 x^* 为以上整数优化问题的最优解，则 $S^* = \{i \in V : x_i^* = 1\}$ 是一个最小权重的顶点覆盖

整数线性规划问题

- 一般的整数线性规划问题 (ILP)

$$\min c^T x$$

$$\text{s. t. } Ax \geq b,$$

$$x \geq 0,$$

x 为整数

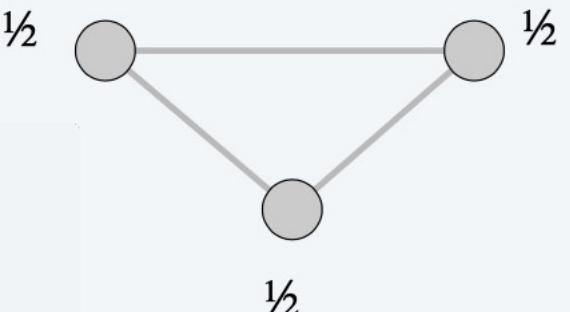
- 根据加权顶点覆盖问题，整数线性规划问题的一个特例为NP-hard的问题，所以一般的整数线性规划问题也是NP-hard的

线性放松 (LP relaxation)

- 线性规划放松 (LP) :

$$\begin{aligned} & \min \sum_{i \in V} w_i x_i \\ \text{s. t. } & x_i + x_j \geq 1, \forall (i, j) \in E \\ & x_i \geq 0 \end{aligned}$$

- LP的最优值小于等于ILP的最优值
- LP的最优解可能不是一个整数解



取整 (rounding)

- 引理：如果 \bar{x}^* 为以上LP问题的最优解，则 $\bar{S}^* = \{i \in V: x_i^* \geq \frac{1}{2}\}$ 所代表的顶点覆盖，最多是最优顶点覆盖 x^* 的2倍
- 证明：



近似算法

- 贪心法
- 放松法
- 对偶法

线性规划对偶问题

Primal LP:

$$\text{maximize } \mathbf{c}^T \mathbf{x}$$

$$\begin{aligned} A\mathbf{x} &\leq \mathbf{b} \\ \mathbf{x} &\geq 0 \end{aligned}$$

Dual LP:

$$\text{minimize } \mathbf{y}^T \mathbf{b}$$

$$\begin{aligned} \mathbf{y}^T A &\geq \mathbf{c}^T \\ \mathbf{y} &\geq 0 \end{aligned}$$

对偶

- 原始问题

$$\begin{aligned}
 & \min \sum_{i \in V} w_i x_i \\
 \text{s. t. } & x_i + x_j \geq 1, \forall (i, j) \in E \\
 & x_i \geq 0
 \end{aligned}$$

- 对偶问题

$$\begin{aligned}
 & \max \sum_{e \in E} y_e \\
 \text{s. t. } & \sum_{(i, j) \in E} y_{ij} \leq w_i, \forall i \in V \\
 & y_e \geq 0, \forall e \in E
 \end{aligned}$$

算法

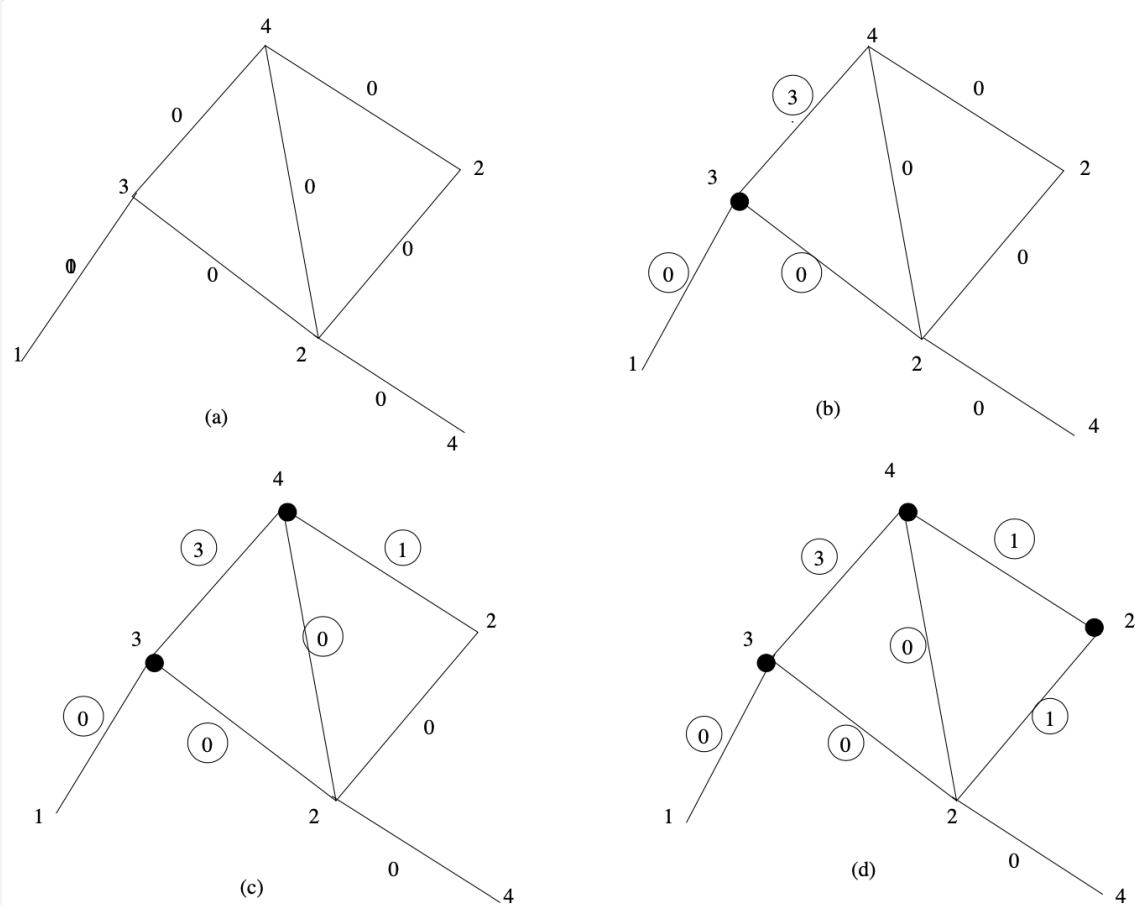
Algorithm 2 顶点覆盖的原始对偶算法

Input: 无向图 $G = (V, E)$, 顶点费用 c .

Output: (x, y) 和购买的顶点集 $A = \{v | x_v = 1\}$.

- 1: **for all** $E \neq \emptyset$ **do**
- 2: 选择任意非空的 $E' \subseteq E$.
- 3: 对每个 $e \in E'$ 均匀提高 y_e , 直到某个对偶约束变紧.
- 4: 假设 S 是对应于对偶约束的顶点集合.
- 5: 对于每个 $v \in S$, 设 $x_v = 1$, 从 E 中删除所有与 S 中顶点相关的边.
- 6: **end for**
- 7: **return** A

例子



选取一条未冻结 y_e 的边，并且提高 y_e 的值，直到相应的对偶约束变紧。然后，所有入射到该顶点的边都将被冻结，并且 x_v 的值将升高到1。

可行性

- 引理： x 是原始可行的， y 是对偶可行的
- 证明：
 - 从 E 中删除每条边都与某个顶点相关联，使得 $x_v = 1$ ，只有当每条边都被删除时，算法才会终止。因此，对于所有的 $e \in E$ ，有 $\sum_{v \in E} x_v \geq 1$ ，因此 x 是可行的；
 - 对于 y ，由于其没有违反任何约束，因为一旦约束变紧，约束中的边将被删除，因此他们的 y_e 的值不会进一步提高。



近似性

- 原始-对偶算法是2-approximation
- 证明：



不可近似性*

- 定理：如果 $P \neq NP$ ，则加权顶点覆盖问题不存在 k -approximation 的算法，其中 $k < 1.3606$

On the Hardness of Approximating Minimum Vertex Cover

Irit Dinur* Samuel Safra†

May 26, 2004

Abstract

We prove the Minimum Vertex Cover problem to be NP-hard to approximate to within a factor of 1.3606, extending on previous PCP and hardness of approximation technique. To that end, one needs to develop a new proof framework, and borrow and extend ideas from several fields.



作业

- 对集合覆盖问题，建模为一个整数线性规划问题，并写出其对偶问题

谢谢！！