处理难解问题的策略

1　对问题施加限制

2　固定参数算法

3　改进指数时间算法

4　启发式方法

5　平均情形的复杂性

二元可满足性问题

二元合取范式：合取范式的每一个简单析取式最多只有2个文字

二元可满足性：限制 SAT 实例中的合取范式为 2元合取范式

设 2元合取范式 F 有n个变元 $x_1, x_2, \ldots, x_n$和 m 个简单析取式 $C_1,C_2,\ldots,C_m$, 每个 C_j 至多有2个文字.

例子：

$C_1= x_1\vee \neg x_2$, $C_2= x_1$, $C_3= \neg x_1\vee \neg x_2$, $C_4= \neg x_2$, $C_5= x_3\vee x_4$, $C_6= \neg x_3\vee \neg x_4$, $C_7= \neg x_3\vee x_4$.

存在多项式时间的 2SAT 算法，$2SAT\in P$

2SAT算法

如果存在只有一个文字的简单析取式 C_j，则先进行化简.

化简过程：

首先使 C_j 为真，得到 C_j 中变量 x_i 的取值.

将 x_i 中的值代入 F，按照逻辑规则进行计算，只要有一个文字的简单析取式，则不断进行化简，直到出现如下情况 .

(1) 如果出现成假字句，则说明 F 不可满足

(2) 如果删除了所有简单析取式，则说明 F 可满足，可满足赋值为当前赋值

(3) 如果剩下的简单析取式都有 2 个文字，则化简过程结束.

2SAT算法（续）

化简后，如果剩下的简单析取式都有2个文字，则任取一个剩下的变元 x, 令 $t(x)=1$ 和 $t(x)=0$ 分别进行化简.

如果两种情况 F 都不可满足，则 F 不可满足，否则 F 可满足.

递归进行该算法.

该算法时间复杂度的分析：

每次化简显然可在 $O(m)$ 步内完成

每一个变元至多经过 2 次化简, 从而总共至多进行 $2n$ 次化简

故算法的时间复杂度为 $O(nm)$.

2SAT算法的实例

$C_1= x_1\vee \neg x_2$, $C_2= x_1$, $C_3= \neg x_1\vee \neg x_2$, $C_4= \neg x_2$, $C_5= x_3\vee x_4$, $C_6= \neg x_3\vee \neg x_4$. $C_7= \neg x_3\vee x_4$.

首先进行化简：

C_2只含一个文字 x_1, 令 $t(x_1)=1$, 删去 C_1 和 C_2, 把 C_3 改成 $C_3= \neg x_2$.

令 $t(\neg x_2)=1$, 即 $t(x_2)=0$, 删去 C_3 和 C_4. 剩下 $C_5= x_3\vee x_4$, $C_6=\neg x_3\vee\neg x_4$, $C_7= \neg x_3\vee x_4$.

不能化简后，随意取一个变量进行尝试，如：

令 $t(x_3)=0$, 重新进行化简, 删去 C_6 和 C_7, 把 C_5 改成 $C_5= x_4$. 令 $t(x_4)=1$, 删去C_5, $t(x_1)=1$, $t(x_2)=0$, $t(x_3)=0$, $t(x_4)=1$, 这是 F 的一个满足赋值, 故 F 是可满足的.

霍恩公式可满足性问题

霍恩 (Horn)子句：每个子句中正文字（不带否定号的变量）至多出现一次

霍恩公式：由霍恩子句构成的公式

霍恩 SAT 问题：

输入限制为霍恩公式的 SAT问题，简记为**HornSAT**.

HornSAT 具有一个简单的多项式时间算法，且该算法直接给出可满足赋值.

实例：$n=4, m=5$

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

$$\wedge(\neg x_1\vee\neg x_2\vee\neg x_3\vee x_4)\wedge x_4$$

求解HornSAT的多项式时间算法

算法12.1

输入：一组霍恩子句

输出：一个可满足赋值或宣布没有可满足赋值

过程：

1. **把所有变量赋值为假；**
2. **循环检查所有蕴含子句 (即带一个正文字的子句), 只要还有不满足的，就把该子句中唯一的正文字的变量赋值为真；**
3. **检查所有的纯负子句 (即不含正文字的子句)，若没有不满足的，则输出当前赋值，算法结束；否则 ，宣布没有可满足赋值.**

定理12.1

定理12.1 上述算法在公式长度的平方时间内正确求解 **HornSAT.**

证　公式长度（即公式中变量出现的总次数）记为 s.不同变量的总数为 n，则 n 总是不超过 s.

首先证明算法在公式长度的平方时间结束.

算法的第 1 步为每个变量赋值，至多花费 $O(n)$步;

第 2 步循环检查总次数不超过蕴含子句的个数，每个循环内部不超过线性时间 $O(s)$，因此时间不超过公式长度的平方时间 $O(s^2)$;

第 3 步检查是否有不满足的纯负子句，$O(s)$时间.

所以算法总时间为 $O(s^2)$.

证 首先证明上述算法在公式长度的平方时间内运行，在这里用公式长度 s（即公式中变量出现的总次数）作为输入规模的度量. 上述算法的第 1 步为每个变量赋值，至多花费 $O(n)$ 步，其中 n 是不同变量的总数，注意 n 总是不超过 s. 第 2 步循环检查的总次数不超过蕴涵子句的个数，因为每次发现一个不可满足的蕴涵子句，就把该子句中的唯一正文字的变量赋值为真，从而满足了该子句，而这样被满足的蕴涵子句不会重新变得不满足，因为算法从不把赋值为真的变量再次赋值为假，所以每个蕴涵子句至多有一次被发现不满足，因此循环的总次数不超过蕴涵子句的个数，而蕴涵子句的个数不超过公式的总长度 s. 在每个循环内部，检查所有的蕴涵子句是否都被满足，只需要扫描一遍输入公式，检查每个蕴涵子句中是否至少有一个文字取值为真，以及相应地为变量赋值，这些都不超过线性时间 $O(s)$，因此整个第 2 步花费的时间不超过公式长度的平方时间 $O(s^2)$. 第 3 步检查是否有不满足的纯负子句，与第 2 步的一次循环类似，也只花费 $O(s)$ 时间. 所以算法总的运行时间是公式长度的平方时间 $O(s^2)$.

定理12.1证明（续）

- 算法的正确性：
  - 当算法返回可满足赋值时，第2步保证了一定没有不满足的蕴涵子句，第3步则只在确认没有不满足的纯负子句时，才返回赋值，显然该赋值是正确的
  - 当算法宣布没有可满足赋值时，也是正确的，因为第二步中“把该子句中唯一的正文字的变量赋值为真”是必须的

其次证明上述算法的正确性. 如果算法返回一个赋值，那么这个赋值一定是可满足赋值，这是因为算法在返回赋值前，已经先确认了没有不满足的子句. 注意霍恩公式中的子句要么是蕴涵子句，要么是纯负子句，第 2 步保证了一定没有不满足的蕴涵子句，第 3 步则只在确认没有不满足的纯负子句时，才返回赋值，所以算法返回的赋值一定是满足赋值.

接下来需要证明，当算法宣布没有满足赋值时，也是正确的. 为此，考虑算法的执行过程. 当算法在第 1 步把所有变量都赋值为假时，所有包含有负文字(即带否定号的变量)的子句就都被满足了. 这时在第 2 步最早检查出的不满足子句都只包含正文字，而根据霍恩子句的定义，每个子句中至多包含一个正文字，因此这个子句恰好由一个正文字构成. 于是为了满足这个子句，在第 2 步就必须给这个变量赋值为真. 而在第 2 步循环体内，随着一些变量被赋值为真，可能有些原来通过负文字取值为真来满足的蕴涵子句又变得不满足，这时就必须把这些蕴涵子句中的正文字赋值为真. 换句话说，在第 2 步中每次把一个变量的赋值由假改为真，都是必需的，即在所有可能的满足赋值中，这些变量都必须被赋值为真，因为只有这样，才能满足相关的那些蕴涵子句. 这样一来，如果这样的赋值还不能满足所有的纯负子句，那么就不可能再有其他的满足赋值了，因此算法的第 3 步确实给出正确的结果.

“对问题限制”的其他例子

- 图的着色问题，当图的色数不超过 2 时，就是易解的.
- 限制图的顶点度数都不超过一个固定的常数 d, 那么求最大团的问题是多项式时间可解的.
  - 只需简单穷举即可，因为顶点度数不超过 d, 最大团中最多有 $d+1$ 个顶点, 故只需要检查所有 $d+1$ 个顶点就能找到最大团.
  - 设有 n 个顶点, 任取 $d+1$ 个顶点有 $\binom{n}{d+1}$ 种可能.
  - 检查 $d+1$个顶点之间的相邻关系只需 $O((d+1)^2) = O(1)$ 次运算, 共需

$$O\left(\binom{n}{d+1}\right) = O(n^{d+1})$$

次运算.

12.2 固定参数算法

通常把优化问题转化为判定问题时，都会在输入中引入一个参数，这是固定参数算法中参数的来源之一.

例子：图的最小顶点覆盖问题. 输入是一个无向图 G，输出是一个最小的顶点集，使得每条边都至少有一个端点在这个集合中.

把这个最小化问题转化为判定问题，
输入为图 (G, k)，其中 k 是 0 与 n 之间的整数，n 是 G的顶点数，而问题则变成判断 G 中是否有一个大小不超过 k 的顶点覆盖.

固定参数算法

- 固定参数算法（Fixed-Parameter Algorithms）或参数化算法（Parameterized Algorithms）：输入中带有一个参数 k，当输入规模为 n 时运行时间为 $O(f(k)n^c)$ 的算法，这里的 $f(k)$ 是与 n 无关的函数，c 是与 n 和 k 都无关的常数.

- “穷举”vs“固定参数”：顶点覆盖判定问题

  穷举所有 k 元顶点子集，看看有没有顶点覆盖. 复杂度

$$O(kn\binom{n}{k}) = O(kn^{k+1})$$

  固定参数算法的运行时间为 $O(2^k kn)$

算法12.2

算法12.2 顶点覆盖问题的固定参数算法

输入：一个 n 阶无向简单图 G

参数：希望找到的顶点覆盖的规模上界 k，其中 $0 \le k < n$

输出：一个大小不超过 k 的顶点覆盖或宣布没这种顶点覆盖

1. 检查G的边集, 若没边, 则输出空集为顶点覆盖, 算法结束;
2. 若边数超过 $k(n-1)$，则宣布没有大小不超过 k 的顶点覆盖，算法结束;
3. 任选一条边 (u,v)，递归检查 $G-\{u\}$和 $G-\{v\}$是否具有大小不超过 $k-1$ 的顶点覆盖;

   3.1 若都没有，则G没有大小不超k的顶点覆盖, 算法结束;

   3.2 若它们中至少一个有，比如说 $G-\{u\}$有大小不超过 $k-1$ 的顶点覆盖T，则输出 $T\cup\{u\}$为 G 的大小不超过 k 的顶点覆盖.

定理12.2

定理12.2 上述算法在 $O(2^k kn)$ 时间内正确地解决大小不超过 k 的顶点覆盖问题.

证　首先证明算法的正确性.

第 1 步显然是对的;

每个顶点最多关联 $n-1$ 条边，第 2 步也正确.

按照顶点覆盖的定义，对于任意一条边 (u,v)， u 和 v 中至少有一个要属于覆盖集，所以不难证明第 3 步也对.

显然算法会在有限步终止，正确性得证.

证 先来证明上述算法的正确性. 第 1 步显然是对的，因为若 G 中没有边，则按照顶点覆盖的定义，空集就是顶点覆盖. 再看第 2 步，由于简单图中一个顶点的最大度不能超过 $n-1$，则每个顶点最多关联 $n-1$ 条边，假如存在不超过 k 个顶点的顶点覆盖，则总边数不会超过 $k(n-1)$，所以第 2 步是对的.

下面证明第 3 步也是对的. 先假设 G 中存在大小不超过 k 的顶点覆盖 S，那么按照顶点覆盖的定义，对于任意一条边 (u,v) 来说，u 和 v 中至少有一个要属于 S，不妨设 u 属于 S，于是 $S-\{u\}$ 就要覆盖所有不与 u 关联的边，即 $S-\{u\}$ 是 $G-\{u\}$ 的大小不超过 $k-1$ 的顶点覆盖. 再假设 $G-\{u\}$ 和 $G-\{v\}$ 中至少有一个具有大小不超过 $k-1$ 的顶点覆盖，不妨设 $G-\{u\}$ 有大小不超过 $k-1$ 的顶点覆盖 T，于是 $T\cup\{u\}$ 就覆盖 G 中所有的边，即 G 中存在大小不超过 k 的顶点覆盖. 这样我们就证明了，G 中存在大小不超过 k 的顶点覆盖，当且仅当对于任意一条边 (u,v) 来说，$G-\{u\}$ 和 $G-\{v\}$ 中至少有一个具有大小不超过 $k-1$ 的顶点覆盖，所以第 3 步也是正确的.

定理12.2证明（续）

复杂度分析

设 $T(n, k)$ 表示算法在 n 阶图和参数 k 上的运行时间，则对于某个常数 c，$T(n, k)$ 满足下述递推公式：

$$T(n,1) \le cn$$

$$T(n, k) \le 2T(n-1, k-1) + ckn \le 2T(n, k-1) + ckn$$

对$k \ge 1$作归纳，可证明 $T(n, k) \le c2^k kn$，即假设

$$T(n, k-1) \le c2^{k-1}(k-1)n$$

则 $T(n, k) \le 2T(n, k-1) + ckn$ （递推公式）

$\le 2c2^{k-1}(k-1)n + ckn$ （归纳假设）

$= c2^k kn - c2^k n + ckn$

$\le c2^k kn$ （因为 $2^k > k$）

12.3 改进指数时间算法

- 符号

  O^*：表示忽略了多项式因子. 如 $O^*(2^n) = O(n^{O(1)}2^n)$

- 改进的指数时间算法

  当一个问题的平凡穷举算法为 $O^*(2^n)$ 时间时，则对于任何满足 $1 < c < 2$的常数 c，把时间复杂性为 $O^*(c^n)$ 的指数时间算法称为非平凡的指数时间算法，或改进的指数时间算法

- 例子：3SAT问题的改进的指数时间精确算法

3SAT的指数时间改进算法

算法12.3

输入：一个合取范式 F，每个子句最多包含 3 个文字
输出：一个满足 F 的赋值 t 或宣布 F 是不可满足的.

1. 令 $G = F$;
2. 若$G=\varnothing$, 则输出 t, 结束; 若G包含空字句, 则宣布G不可满足;
3. 任取子句$C=\alpha_1\vee\ldots\vee\alpha_k$，其中$\alpha_i$是文字, $1\le k\le 3$, 执行下述k步:
   - 3.1 令 $t(\alpha_1)=1$, 删去含α_1的子句, 把含$\neg\alpha_1$的子句$D=\neg\alpha_1\vee D'$换成D', D'可能是空子句, 即$\neg\alpha_1=\neg\alpha_1\vee\varnothing$. 化简后公式为$G'$. 对$G'$递归运用算法 2~4;
   - 3.2 令 $t(\alpha_1)=0$, $t(\alpha_2)=1$, 类似上面化简得到公式G'. 对G'递归运用算法 2~4;
   - 3.3 令 $t(\alpha_1)=0$, $t(\alpha_2)=0$, $t(\alpha_3)=1$, 类似上面化简得到公式为G'. 对G'递归运用算法 2~4;

   若上述 k 步都宣布G'是不可满足的，则宣布G不可满足.
4. 若宣布 F 是不可满足的，则计算结束.

定理12.3

定理12.3 上述算法在$O^*(1.8393^n)$时间内正确求解 3SAT.

证明：首先证明算法的正确性.

当算法在第 2 步输出一个赋值时，显然这个赋值的确是满足赋值.

下面证明当算法宣布没有可满足赋值时，也是正确的. 如果输入公式中含有某个子句 $C=\alpha_1\vee\alpha_2\vee\alpha_3$，则所有的满足赋值被划分为算法第 3 步检查的那三类，故算法不会漏掉满足赋值.

由于每次递归至少为一个变量赋值，因此每个递归检查的深度至多为 n，即算法不存在死循环. 这样就证明了算法是正确的.

定理12.3证明（续）

下面分析算法的复杂性：

设 $T(n)$ 表示算法在 n 个变量的公式上的运行时间.

设公式中有 m 个子句，则 $m=O(n^3)$，因为每个子句至多含有三个文字，不同子句的个数至多为 $O(n^3)$.

关于 $T(n)$ 有下列递推公式：

$$T(n) \le T(n-1) + T(n-2) + T(n-3) + O(n+m)$$

递推公式的解是

$$T(n) = O^*(1.8393^n)$$

其中 1.8393 是方程 $\lambda^3 = \lambda^2 + \lambda + 1$ 的最大根.

其他一些结论

- 截止到 2010 年底，目前 3SAT 的指数时间精确算法的最好结果是： 一个$O^*(1.321^n)$ 时间的随机算法，和一个$O^*(1.439^n)$ 时间的确定型算法.
- 任意色数的图的顶点着色问题都有 $O^*(3^n)$ 的算法.
- 背包问题有比 $O^*(2^{n/2})$ 更好的算法.
- 货郎问题也有比 $O^*(2^n)$ 更好的算法.

- 指数时间假设 (Exponential Time Hypothesis，简称ETH)：对于每个正整数 $k > 2$，都存在常数 $c_k > 0$ 使得求解 kSAT 的精确算法时间复杂性不低于 $O^*(2^{c_k n})$.

12.4启发式方法

- 启发式方法（Heuristics）：人们发现还有一类方法，目前无法从理论上给出任何性能保证，但在实践中却效果良好，就把这类方法统称为启发式方法（Heuristics）.
- 常用的启发式方法主要包括：回溯法、分支限界法、局部搜索法、遗传算法等.

局部搜索算法

- 一般步骤：
  1. 产生初始解
  2. 局部对解进行修改，如果好就代替原来的解，否则保留原来的解
  3. 重复执行步骤2，直到预定次数为止
- 可能出现的问题：
  陷入局部最优而出不来
- 改进策略：

  随机化策略：随机选择初始解

  重启策略：在不同的初始解上多次运行

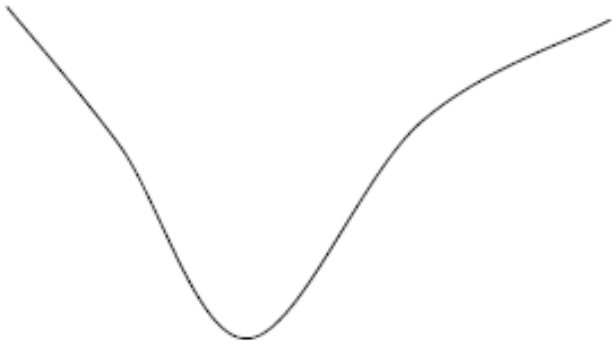
  模拟退火：根据温度取退步解代替当前解

Local Search

Local search. Algorithm that explores the space of possible solutions in sequential fashion, moving from a current solution to a "nearby" one.

Neighbor relation. Let $S \sim S'$ be a neighbor relation for the problem.

Gradient descent. Let S denote current solution. If there is a neighbor S' of S with strictly lower cost, replace S with the neighbor whose cost is as small as possible. Otherwise, terminate the algorithm.



A funnel

A jagged funnel

Metropolis Algorithm

Metropolis algorithm. [Metropolis, Rosenbluth, Rosenbluth, Teller, Teller 1953]

- Simulate behavior of a physical system according to principles of statistical mechanics.
- Globally biased toward "downhill" steps, but occasionally makes "uphill" steps to break out of local minima.

Gibbs-Boltzmann function. The probability of finding a physical system in a state with energy E is proportional to $e^{-E / (kT)}$, where $T > 0$ is temperature and k is a constant.

- For any temperature $T > 0$, function is monotone decreasing function of energy E.
- System more likely to be in a lower energy state than higher one.
  - T large: high and low energy states have roughly same probability
  - T small: low energy states are much more probable

Metropolis Algorithm

Metropolis algorithm.

- Given a fixed temperature T, maintain current state S.
- Randomly perturb current state S to new state $S' \in N(S)$.
- If $E(S') \leq E(S)$, update current state to S'
  Otherwise, update current state to S' with probability $e^{-\Delta E / (kT)}$,
  where $\Delta E = E(S') - E(S) > 0$.

Theorem. Let $f_S(t)$ be fraction of first t steps in which simulation is in state S. Then, assuming some technical conditions, with probability 1:

$$\lim_{t \to \infty} f_S(t) = \frac{1}{Z} e^{-E(S)/(kT)},$$

$$\text{where } Z = \sum_{S \in N(S)} e^{-E(S)/(kT)}.$$

Intuition. Simulation spends roughly the right amount of time in each state, according to Gibbs-Boltzmann equation.

Simulated Annealing

Simulated annealing.

- T large $\Rightarrow$ probability of accepting an uphill move is large.
- T small $\Rightarrow$ uphill moves are almost never accepted.
- Idea: turn knob to control T.
- Cooling schedule: T = T(i) at iteration i.

Physical analog.

- Take solid and raise it to high temperature, we do not expect it to maintain a nice crystal structure.
- Take a molten solid and freeze it very abruptly, we do not expect to get a perfect crystal either.
- Annealing: cool material gradually from high temperature, allowing it to reach equilibrium at succession of intermediate lower temperatures.

模拟退火法

算法12.4

输入：一个优化问题的实例

输出：一个可行解

1. 生成一个初始解，设定初始温度 T_0；
2. 循环执行下列步骤，直到 $T=0$；
   ① 从当前解的邻域中随机选择一个新解；
   ② 若新解是改进解，则代替当前解；否则设新解的退步幅度为Δ，以概率 $e^{-\frac{\Delta}{T}}$用新解代替当前解；
   ③ 更新温度 T；
3. 输出见到的最好的解.

Hopfield Neural Networks

Hopfield networks. Simple model of an associative memory, in which a large collection of units are connected by an underlying network, and neighboring units try to correlate their states.

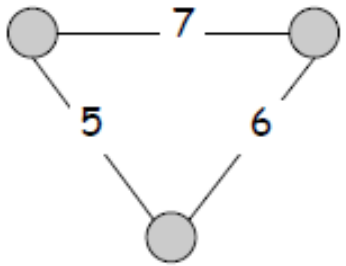
Input: Graph $G = (V, E)$ with integer edge weights w.

positive or negative

Configuration. Node assignment $s_u = \pm 1$.

Intuition. If $w_{uv} < 0$, then u and v want to have the same state; if $w_{uv} > 0$ then u and v want different states.

Note. In general, no configuration respects all constraints.



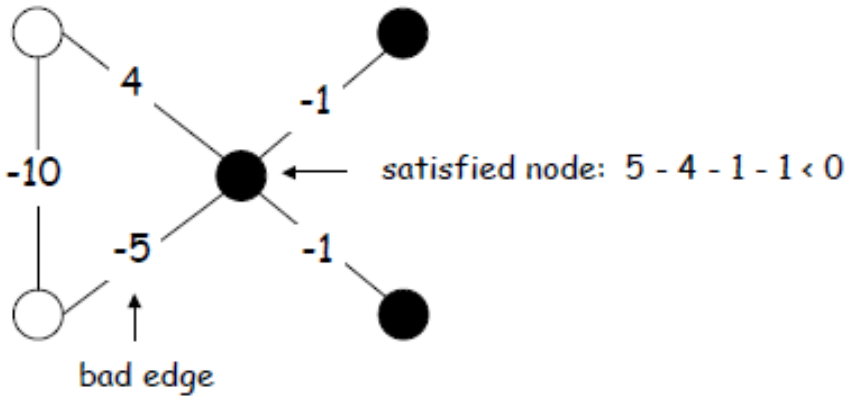
Hopfield Neural Networks

Def. With respect to a configuration S, edge e = (u, v) is good if $w_e s_u s_v < 0$. That is, if $w_e < 0$ then $s_u = s_v$; if $w_e > 0$, $s_u \neq s_v$.

Def. With respect to a configuration S, a node u is satisfied if the weight of incident good edges ≥ weight of incident bad edges.

$$\sum_{v:\, e=(u,v)\in E} w_e s_u s_v \;\leq\; 0$$

Def. A configuration is stable if all nodes are satisfied.



Goal. Find a stable configuration, if such a configuration exists.

Hopfield Neural Networks

Goal. Find a stable configuration, if such a configuration exists.

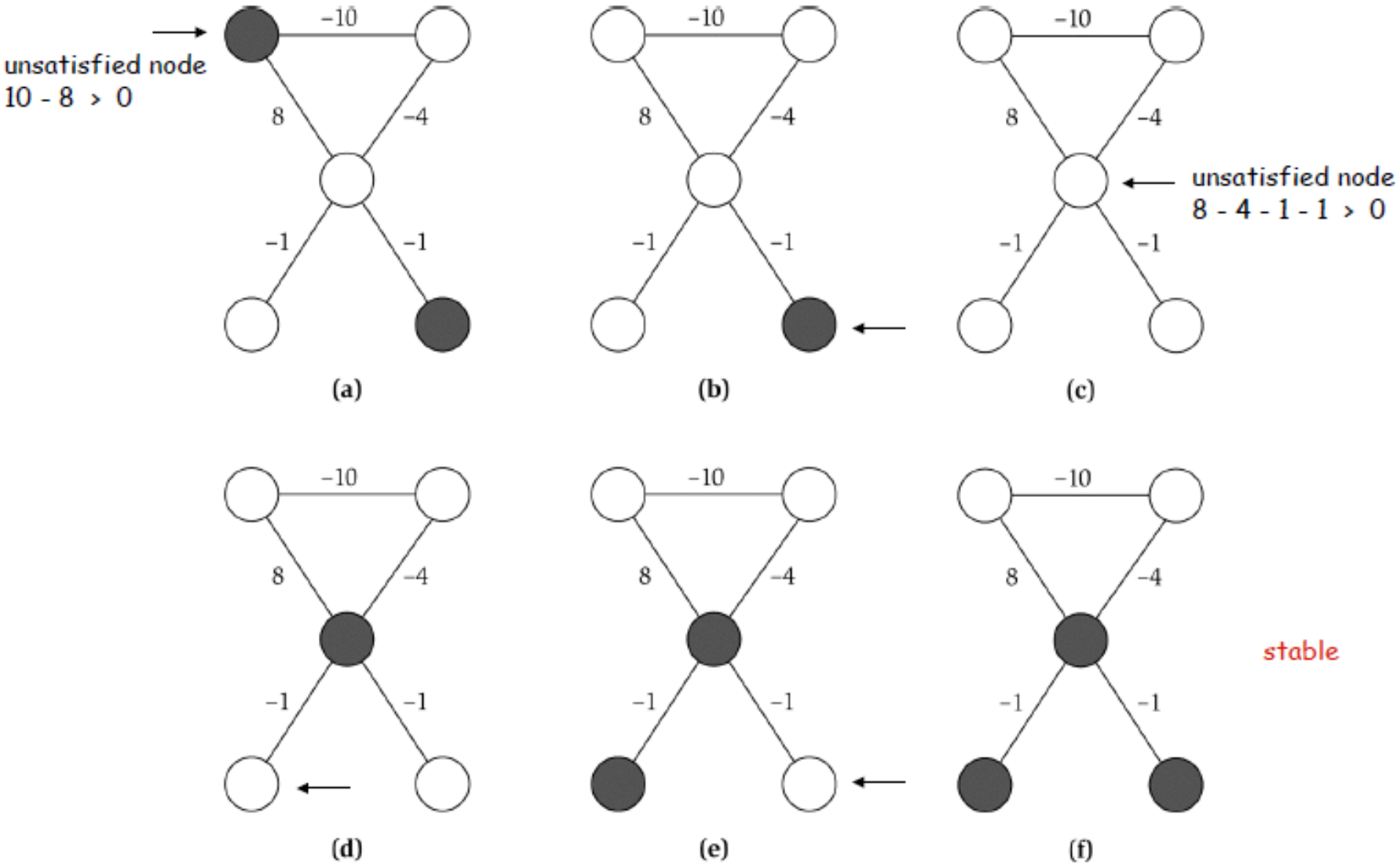
State-flipping algorithm. Repeated flip state of an unsatisfied node.

```
Hopfield-Flip(G, w) {
    S ← arbitrary configuration

    while (current configuration is not stable) {
        u ← unsatisfied node
        s_u = -s_u
    }

    return S
}
```

State Flipping Algorithm



Hopfield Neural Networks

Claim. State-flipping algorithm terminates with a stable configuration after at most $W = \Sigma_e |w_e|$ iterations.

Pf attempt. Consider measure of progress $\Phi(S)$ = # satisfied nodes.

Hopfield Neural Networks

Claim. State-flipping algorithm terminates with a stable configuration after at most $W = \Sigma_e |w_e|$ iterations.

Pf. Consider measure of progress $\Phi(S) = \Sigma_{e \text{ good}} |w_e|$.

- Clearly $0 \leq \Phi(S) \leq W$.
- We show $\Phi(S)$ increases by at least 1 after each flip.
  When u flips state:
  - all good edges incident to u become bad
  - all bad edges incident to u become good
  - all other edges remain the same

$$\Phi(S') = \Phi(S) - \sum_{\substack{e:\ e=(u,v)\in E \\ e \text{ is bad}}} |w_e| + \sum_{\substack{e:\ e=(u,v)\in E \\ e \text{ is good}}} |w_e| \geq \Phi(S) + 1$$

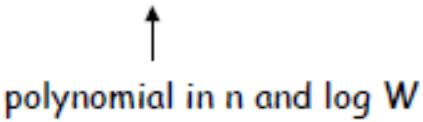
u is unsatisfied

Complexity of Hopfield Neural Network

Hopfield network search problem. Given a weighted graph, find a stable configuration if one exists.

Hopfield network decision problem. Given a weighted graph, does there exist a stable configuration?

Remark. The decision problem is trivially solvable (always yes), but there is no known poly-time algorithm for the search problem.



Maximum Cut

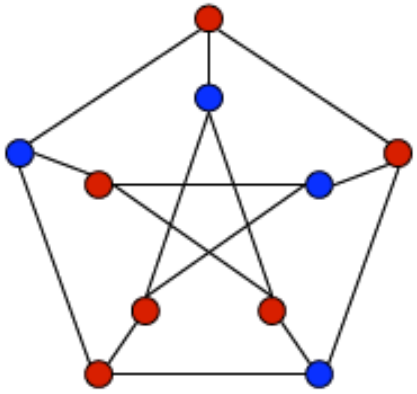
Maximum cut. Given an undirected graph $G = (V, E)$ with positive integer edge weights w_e, find a node partition (A, B) such that the total weight of edges crossing the cut is maximized.

$$w(A, B) := \sum_{u \in A,\ v \in B} w_{uv}$$

Toy application.

- n activities, m people.
- Each person wants to participate in two of the activities.
- Schedule each activity in the morning or afternoon to maximize number of people that can enjoy both activities.

Real applications. Circuit layout, statistical physics.



Maximum Cut

Single-flip neighborhood. Given a partition (A, B), move one node from A to B, or one from B to A if it improves the solution.

Greedy algorithm.

```
Max-Cut-Local (G, w) {
   Pick a random node partition (A, B)

   while (∃ improving node v) {
      if (v is in A) move v to B
      else           move v to A
   }

   return (A, B)
}
```

Maximum Cut: Local Search Analysis

Theorem. Let (A, B) be a locally optimal partition and let (A^*, B^*) be optimal partition. Then $w(A, B) \geq \frac{1}{2} \sum_e w_e \geq \frac{1}{2} w(A^*, B^*)$.

weights are nonnegative

Pf.

- Local optimality implies that for all $u \in A$: $\sum_{v \in A} w_{uv} \leq \sum_{v \in B} w_{uv}$
  Adding up all these inequalities yields:

$$2 \sum_{\{u,v\} \subseteq A} w_{uv} \leq \sum_{u \in A,\, v \in B} w_{uv} = w(A, B)$$

- Similarly $$2 \sum_{\{u,v\} \subseteq B} w_{uv} \leq \sum_{u \in A,\, v \in B} w_{uv} = w(A, B)$$

- Now,

each edge counted once

$$\sum_{e \in E} w_e = \underbrace{\sum_{\{u,v\} \subseteq A} w_{uv}}_{\leq \frac{1}{2} w(A, B)} + \underbrace{\sum_{u \in A,\, v \in B} w_{uv}}_{w(A, B)} + \underbrace{\sum_{\{u,v\} \subseteq A} w_{uv}}_{\leq \frac{1}{2} w(A, B)} \leq 2w(A, B) \quad \blacksquare$$

Maximum Cut: Big Improvement Flips

Local search. Within a factor of 2 for MAX-CUT, but not poly-time!

Big-improvement-flip algorithm. Only choose a node which, when flipped, increases the cut value by at least $\frac{2\varepsilon}{n} w(A, B)$

Claim. Upon termination, big-improvement-flip algorithm returns a cut (A, B) with $(2 + \varepsilon)\, w(A, B) \geq w(A^*, B^*)$.

Pf idea. Add $\frac{2\varepsilon}{n} w(A, B)$ to each inequality in original proof.

Claim. Big-improvement-flip algorithm terminates after $O(\varepsilon^{-1} n \log W)$ flips, where $W = \Sigma_e w_e$.

- Each flip improves cut value by at least a factor of $(1 + \varepsilon/n)$.
- After n/ε iterations the cut value improves by a factor of 2.
- Cut value can be doubled at most log W times.

if $x \geq 1$, $(1 + 1/x)^x \geq 2$

Maximum Cut: Context

Theorem. [Sahni-Gonzales 1976] There exists a $\frac{1}{2}$-approximation algorithm for MAX-CUT.

Theorem. [Goemans-Williamson 1995] There exists an 0.878567-approximation algorithm for MAX-CUT.

$$\min_{0 \le \theta \le \pi} \frac{2}{\pi} \frac{\theta}{1 - \cos\theta}$$

Theorem. [Håstad 1997] Unless P = NP, no 16/17 approximation algorithm for MAX-CUT.

0.941176

Neighbor Relations for Max Cut

1-flip neighborhood. (A, B) and (A', B') differ in exactly one node.

k-flip neighborhood. (A, B) and (A', B') differ in at most k nodes.

- $\Theta(n^k)$ neighbors.

KL-neighborhood. [Kernighan-Lin 1970]

cut value of (A_1, B_1) may be worse than (A, B)

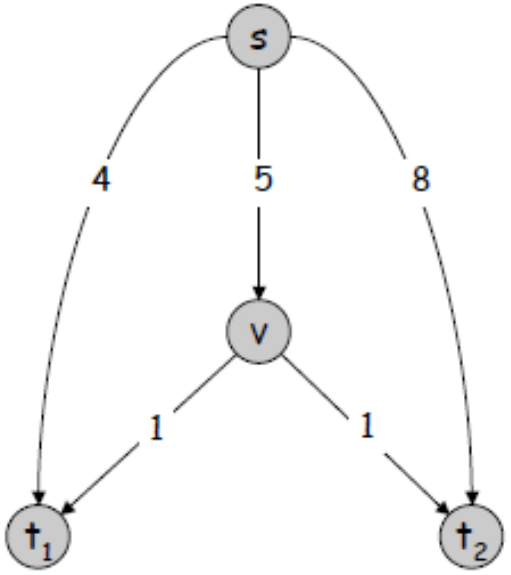
- To form neighborhood of (A, B):
  - Iteration 1: flip node from (A, B) that results in best cut value (A_1, B_1), and mark that node.
  - Iteration i: flip node from (A_{i-1}, B_{i-1}) that results in best cut value (A_i, B_i) among all nodes not yet marked.
- Neighborhood of (A, B) = (A_1, B_1), …, (A_{n-1}, B_{n-1}).
- Neighborhood includes some very long sequences of flips, but without the computational overhead of a k-flip neighborhood.
- Practice: powerful and useful framework.
- Theory: explain and understand its success in practice.

Multicast Routing

Multicast routing. Given a directed graph $G = (V, E)$ with edge costs $c_e \geq 0$, a source node s, and k agents located at terminal nodes $t_1, \ldots, t_k$. Agent j must construct a path P_j from node s to its terminal t_j.

Fair share. If x agents use edge e, they each pay c_e / x.

1	2	1 pays	2 pays
outer	outer	4	8
outer	middle	4	5 + 1
middle	outer	5 + 1	8
middle	middle	5/2 + 1	5/2 + 1



Nash Equilibrium

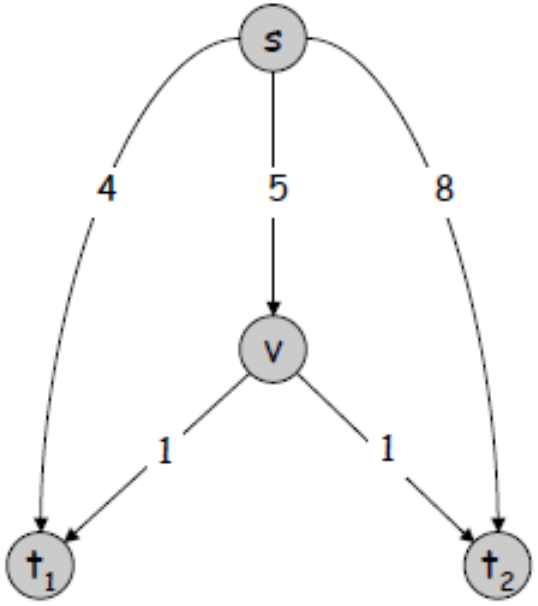
Best response dynamics. Each agent is continually prepared to improve its solution in response to changes made by other agents.

Nash equilibrium. Solution where no agent has an incentive to switch.

Fundamental question. When do Nash equilibria exist?

Ex:

- Two agents start with outer paths.
- Agent 1 has no incentive to switch paths (since $4 < 5 + 1$), but agent 2 does (since $8 > 5 + 1$).
- Once this happens, agent 1 prefers middle path (since $4 > 5/2 + 1$).
- Both agents using middle path is a Nash equilibrium.



Nash Equilibrium and Local Search

Local search algorithm. Each agent is continually prepared to improve its solution in response to changes made by other agents.

Analogies.

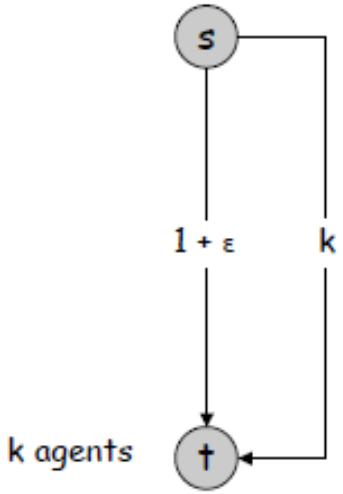
- Nash equilibrium : local search.
- Best response dynamics : local search algorithm.
- Unilateral move by single agent : local neighborhood.

Contrast. Best-response dynamics need not terminate since no single objective function is being optimized.

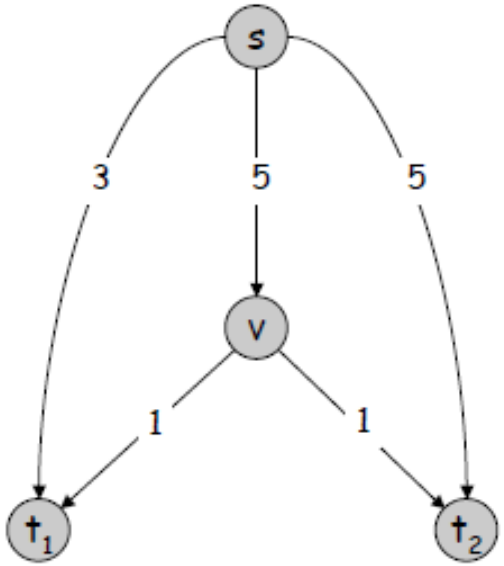
Socially Optimum

Social optimum. Minimizes total cost to all agent.

Observation. In general, there can be many Nash equilibria. Even when its unique, it does not necessarily equal the social optimum.



Social optimum = 1 + ε
Nash equilibrium A = 1 + ε
Nash equilibrium B = k



Social optimum = 7
Unique Nash equilibrium = 8

Price of Stability

Price of stability. Ratio of best Nash equilibrium to social optimum.

Fundamental question. What is price of stability?

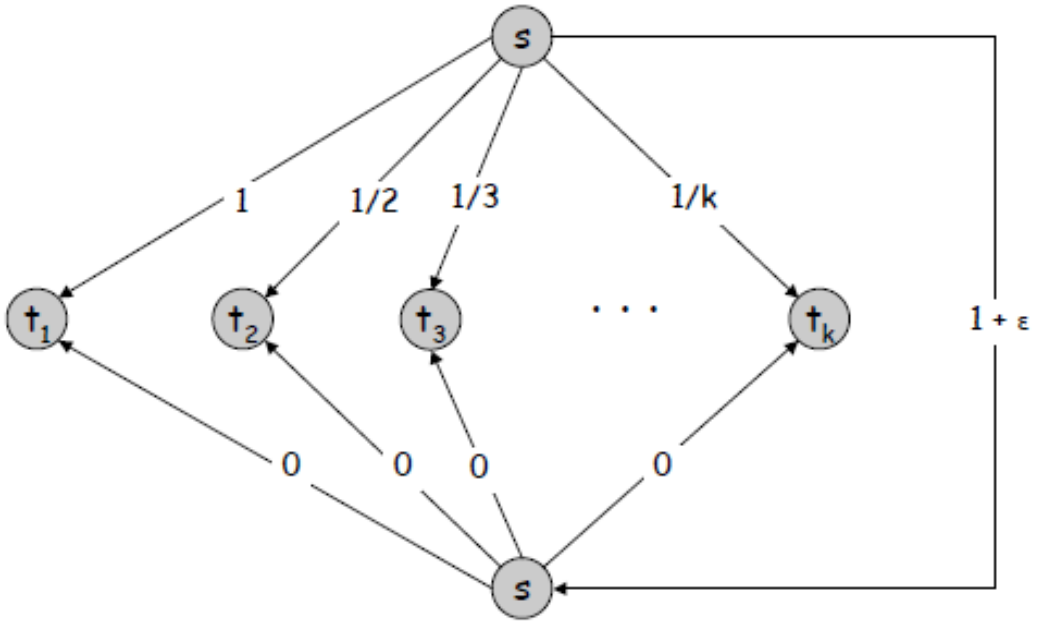
Ex: Price of stability = $\Theta(\log k)$.

Social optimum. Everyone takes bottom paths.

Unique Nash equilibrium. Everyone takes top paths.

Price of stability. $H(k) / (1 + \varepsilon)$.

$1 + 1/2 + \ldots + 1/k$



Finding a Nash Equilibrium

Theorem. The following algorithm terminates with a Nash equilibrium.

```
Best-Response-Dynamics(G, c) {
    Pick a path for each agent

    while (not a Nash equilibrium) {
        Pick an agent i who can improve by switching paths
        Switch path of agent i
    }
}
```

Pf. Consider a set of paths $P_1, \ldots, P_k$.

- Let x_e denote the number of paths that use edge e.
- Let $\Phi(P_1, \ldots, P_k) = \sum_{e \in E} c_e \cdot H(x_e)$ be a potential function.
- Since there are only finitely many sets of paths, it suffices to show that Φ strictly decreases in each step.

$H(0) = 0,\ H(k) = \sum_{i=1}^{k} \frac{1}{i}$

Finding a Nash Equilibrium

Pf. (continued)

- Consider agent j switching from path P_j to path P_j'.
- Agent j switches because

$$\underbrace{\sum_{f \in P_j' - P_j} \frac{c_f}{x_f + 1}}_{\text{newly incurred cost}} < \underbrace{\sum_{e \in P_j - P_j'} \frac{c_e}{x_e}}_{\text{cost saved}}$$

- Φ increases by $\sum_{f \in P_j' - P_j} c_f \left[H(x_f + 1) - H(x_f) \right] = \sum_{f \in P_j' - P_j} \frac{c_f}{x_f + 1}$
- Φ decreases by $\sum_{e \in P_j - P_j'} c_e \left[H(x_e) - H(x_e - 1) \right] = \sum_{e \in P_j - P_j'} \frac{c_e}{x_e}$
- Thus, net change in Φ is negative. ▪

Bounding the Price of Stability

Claim. Let $C(P_1, \ldots, P_k)$ denote the total cost of selecting paths $P_1, \ldots, P_k$. For any set of paths $P_1, \ldots, P_k$, we have

$$C(P_1, \ldots, P_k) \;\le\; \Phi(P_1, \ldots, P_k) \;\le\; H(k) \cdot C(P_1, \ldots, P_k)$$

Pf. Let x_e denote the number of paths containing edge e.

- Let E^+ denote set of edges that belong to at least one of the paths.

$$C(P_1, \ldots, P_k) \;=\; \sum_{e \in E^+} c_e \;\le\; \underbrace{\sum_{e \in E^+} c_e \, H(x_e)}_{\Phi(P_1, \ldots, P_k)} \;\le\; \sum_{e \in E^+} c_e \, H(k) \;=\; H(k)\; C(P_1, \ldots, P_k)$$

Bounding the Price of Stability

Theorem. There is a Nash equilibrium for which the total cost to all agents exceeds that of the social optimum by at most a factor of H(k).

Pf.

- Let $(P_1^*, \ldots, P_k^*)$ denote set of socially optimal paths.
- Run best-response dynamics algorithm starting from P*.
- Since Φ is monotone decreasing $\Phi(P_1, \ldots, P_k) \le \Phi(P_1^*, \ldots, P_k^*)$.

$$C(P_1,\ldots,P_k) \le \Phi(P_1,\ldots,P_k) \le \Phi(P_1^*,\ldots,P_k^*) \le H(k)\cdot C(P_1^*,\ldots,P_k^*)$$

previous claim applied to P

previous claim applied to P*

Summary

Existence. Nash equilibria always exist for k-agent multicast routing with fair sharing.

Price of stability. Best Nash equilibrium is never more than a factor of H(k) worse than the social optimum.

Fundamental open problem. Find any Nash equilibria in poly-time.

12.5 平均情形的复杂性

- NP完全性理论是基于最坏情形下的复杂性度量
- 考虑平均情形的复杂性度量时，发现有些 NP完全问题在平均复杂性度量下是易解的.
- 在平均情形的复杂度性度量下，哈密顿回路问题有多项式时间的算法.
- 为了考虑平均情形的复杂性度量，首先要在所有输入上定义一个概率分布.
- 随机图模型 $G(n, p)$：图的顶点数为 n，然后在任何两个不同顶点之间独立地以概率 p 连一条边.

利用切诺夫界，G 以大概率满足以下三条性质 (设 $N(v)$ 表示顶点 v 的邻域)：

I. 对每个顶点 v，$|N(v)|$ 介于 $n/2 \pm n/50$ 之间.

II. 对每对顶点 u 和 v, $|N(u) \cup N(v)|$ 介于 $3n/4 \pm n/50$ 之间.

III. 对每三个顶点 u、v 和 w，$|N(u) \cup N(v) \cup N(w)|$ 介于 $7n/8 \pm n/50$ 之间.

切诺夫界

- 如果随机变量 $X_1, X_2, \ldots, X_n$ 是完全独立的，并且每个 X_i 取值在{0,1}中，设 $\mu = \sum_{i=1}^{n} \mathrm{E}[X_i]$，则对于任何 $\delta > 0$，有

$$\Pr[\sum_{i=1}^{n} X_i \geq (1+\delta)\mu] \leq \left[\frac{\mathrm{e}^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu}$$

$$\Pr[\sum_{i=1}^{n} X_i \leq (1-\delta)\mu] \leq \left[\frac{\mathrm{e}^{-\delta}}{(1-\delta)^{(1-\delta)}}\right]^{\mu}$$

以及对任何 $c > 0$，有

$$\Pr[|\sum_{i=1}^{n} X_i - \mu| \geq c\mu] \leq 2 \cdot \mathrm{e}^{-\min\{c^2/4, c/2\}\mu}$$

三个不等式统称为切诺夫界.

12.9 对每个顶点 v，其余 $n-1$ 个顶点与 v 是否有边相连是独立的，且服从 0-1 分布，且有边的概率为 1/2，设 X_i 为第 i 个顶点与 v 相连的边数，$i=1,2,\cdots,n-1$.

性质一的证明：根据切诺夫界，

$$\Pr\left[\left|\sum_{i=1}^{n-1}X_i-\frac{n}{2}\right|\geqslant\frac{n}{50}\right]\leqslant\Pr\left[\left|\sum_{i=1}^{n-1}X_i-\frac{n-1}{2}\right|\geqslant\frac{n-1}{100}\right]\quad(\text{当 } n \text{ 充分大时})$$

$$=\Pr\left[\left|\sum_{i=1}^{n-1}X_i-\mathrm{E}\left[\sum_{i=1}^{n-1}X_i\right]\right|\geqslant\frac{1}{50}\mathrm{E}\left[\sum_{i=1}^{n-1}X_i\right]\right]$$

$$\leqslant 2\exp\left(-\min\left\{\frac{4}{50^2},\frac{1}{100}\right\}\mathrm{E}\left[\sum_{i=1}^{n-1}X_i\right]\right)$$

$$=2\exp\left\{-\frac{4}{2500}\times\frac{n-1}{2}\right\}\xrightarrow{n\to\infty}0$$

上边只证明了对于固定的顶点 v，满足性质一.

$$\Pr[\forall\, v,v \text{ 满足性质一}]=1-\Pr[\exists\, v,v \text{ 不满足性质一}]\quad(\text{并的界})$$

$$\geqslant 1-n\Pr[v \text{ 不满足性质一}]\xrightarrow{n\to\infty}1$$

所以，对每个顶点 v，$|N(v)|$ 介于 $n/2\pm n/50$ 之间.

性质二的证明：对每对顶点 u 和 v，$N(u)\cup N(v)$ 是 u 和 v 邻域的并，由于在 $G(n,p)$ 模型中，每条边是否出现是独立的. 所以对第三个点 w，w 是否属于 $N(u)\cup N(v)$ 是独立同分布的 0-1 随机变量，且属于的概率为 3/4，类似性质一的证明，容易得到性质二的证明.

性质三的证明：对 $V\backslash\{u,v,w\}$ 中的任一顶点，是否属于 $N(u)\cup N(v)\cup N(w)$ 是独立同分布的，且属于的概率为 7/8，类似 1 的计算，可得证.

哈密顿回路算法

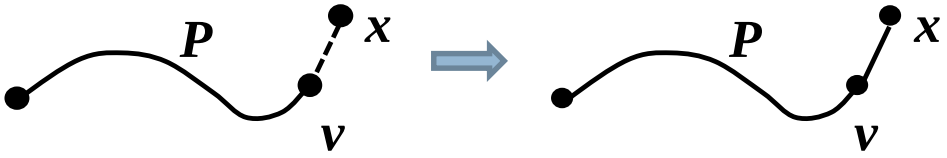
算法12.5

输入：在 $G(n,1/2)$ 分布下产生的一个随机图 G

输出：G 中的一条哈密顿回路

1. 反复利用下面规则构造路径 P，直到 P 不能再延长为止；

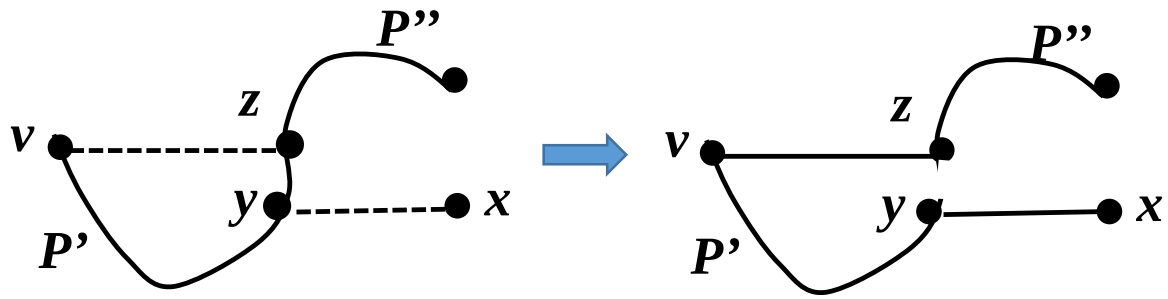
   1.1 若 P 外顶点 x 与 P 的端点 v 相邻，则用边 (x, v) 把 x 加入 P，即令 $P = xP$；



分析：这时，如果利用 1.1 已经不能延长该路径，则说明对 P 外的顶点 x，x 不与 P 的两端相邻，按照性质二，这时，以大概率，路径上至少已经有 $3n/4-n/50$ 个顶点.

哈密顿回路算法

1.2 若 $P = v\,P'y\,z\,P''$ 且 v 与 z 相邻、y 与 P 外顶点 x 相邻，则令 $P = x\,y\,P'\,v\,z\,P''$；

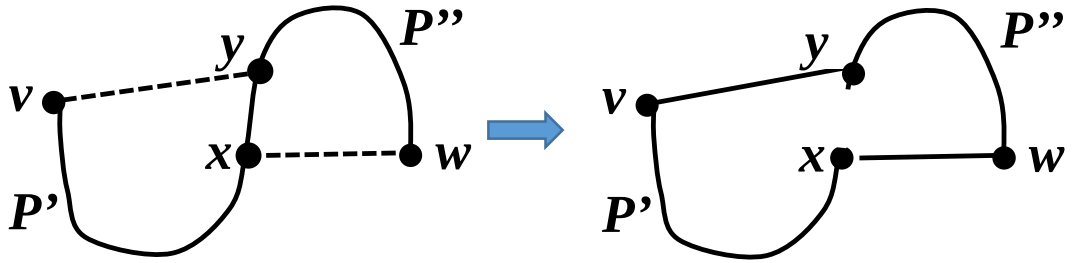


分析：假设已经没有符合 1.1 的 x，如上图，按照性质一，至少有 $n/2-n/50$ 个顶点与 v 相邻，这些点不能在已有路径外，也就是说，以大概率，可以找到如图的 z，所以不能加入路径上的点也不能与 y 相邻，所以步骤 1.2 结束后，路径外的点至少不能和该路径上的三个点相邻，按照性质三，路径上至少有 $7n/8-n/50$ 个顶点.

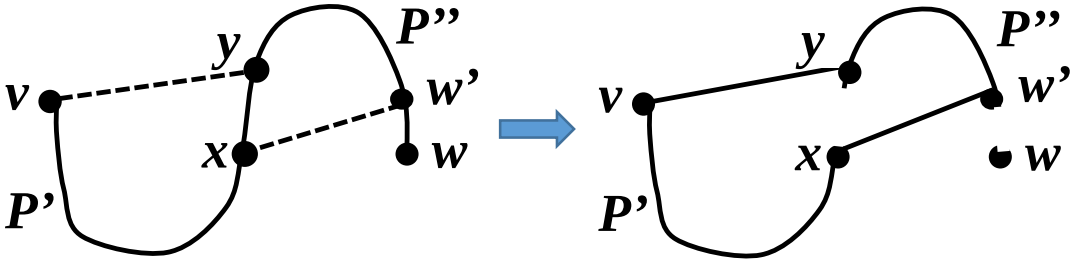
算法12.5（续）

2. 反复利用下面规则构造一个圈C.

2.1 若 $P = v\,P'\,x\,y\,P''\,w$且 v 与 y 相邻、w 与 x 相邻，则令 $C = w\,x\,P'\,v\,y\,P''\,w$;



2.2 若 $P = v\,P'\,x\,y\,P''\,w'\,w$ 且 v 与 y 相邻、w'与 x 相邻，则令 $C = w'\,x\,P'\,v\,y\,P''\,w'$;

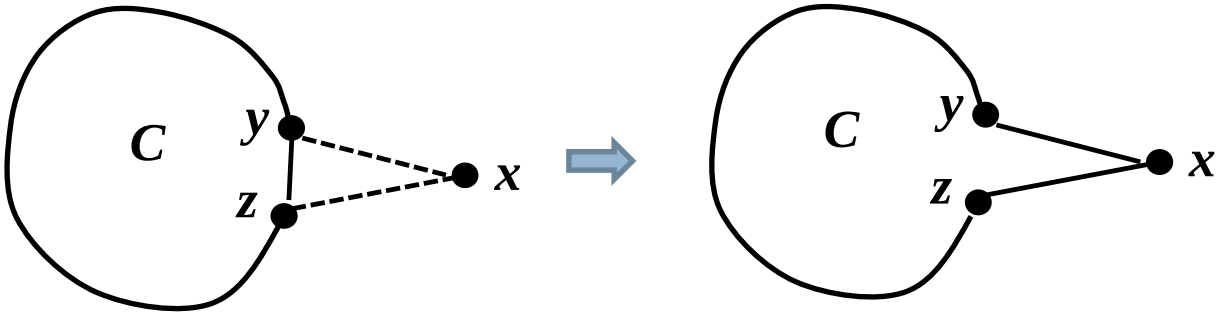


分析：以大概率，v 和 w 都有 $n/2-n/50$ 个相邻顶点在该路径中，如上图，假设 v 与 y 相邻，则相应的 y 有 $n/2-n/50$ 个，因为 x 可能就是 v，所以相应的 x 也至少 $n/2-n/50-1$ 个，而按照性质二，w 和 w' 的邻域至少有 $3n/4-n/50$ 个顶点，由于 $3n/4-n/50>n-(n/2-n/50-1)$，所以必然会构成一个圈 C，且 C 上最多比第一步构造的路径少一个顶点，即至少有 $7n/8-n/50-1$ 个顶点.

算法12.5（续）

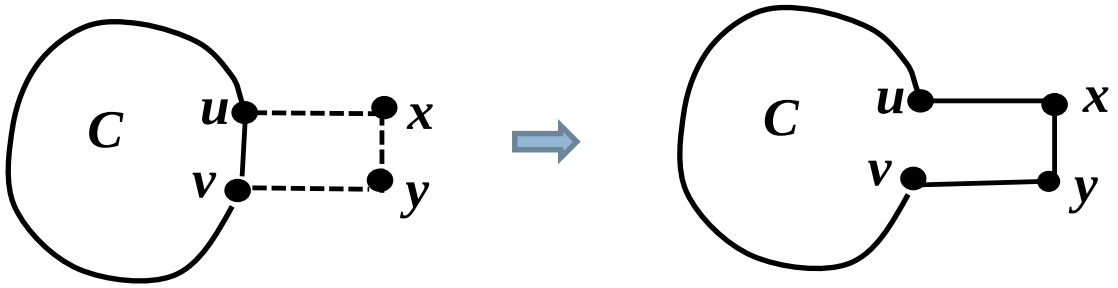
3. 反复利用下面三条规则把其余顶点加入 C，每次加入一个或两个顶点，直到没有剩余顶点为止；

3.1 若 C 外的顶点 x 与 C 上连续两点 y 和 z 都相邻，则令 $C = C \cup \{(y, x), (x, z)\} - \{(y, z)\}$；

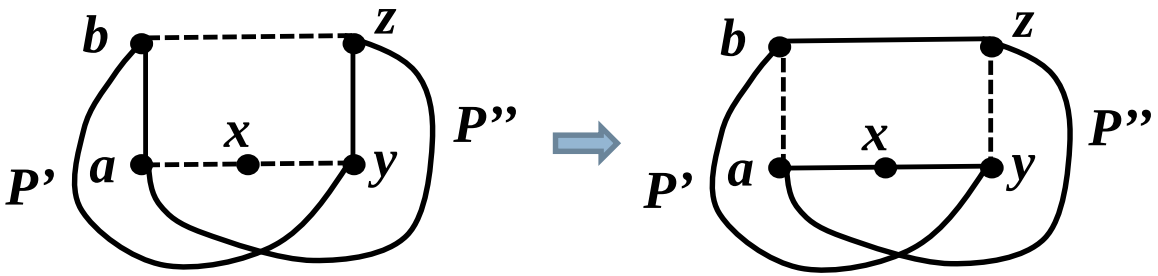


算法12.5（续）

3.2 若 C 外的两点 x 和 y 相邻且分别与 C 上连续两点 u 和 v 相邻，则令 $C = C \cup \{(u,x), (x,y), (y,v)\} - \{(u,v)\}$



3.3 若 $C = a\ b\ P'\ y\ z\ P''\ a$ 且 C 外的顶点 x 与 a 和 y 都相邻、b与 z 相邻，则令 $C = a\ x\ y\ P'\ b\ z\ P''\ a$;



4. 输出 C 作为哈密顿回路.

分析：首先，只要有边存在，如步骤 3.2 中的 x 和 y，则证明至少能用步骤 3.1 或步骤 3.2 中的一种方法. 下面假设两个方法都不能用. 由于与 x 和 y 之一相邻的点至少有 $3n/4-n/50$ 个，而这些点在 C 上至少有 $3n/4-n/50-(n-(7n/8-n/50-1))=5n/8-2n/50-1>n/2$，它们不可能出现在一个至多为 n 个顶点的圈，且两两不相邻. 假设圈外的顶点是一个独立集，且不能用 3.1，为了分析方便，我们利用 $G(n,1/2)$ 的独立集为 $O(n)$ 量级这一事实. 而相应的与 x 相邻的顶点至少有 $n/2-n/50$ 个，规定圈上的一个方向，则这些的顶点同侧的顶点也有 $n/2-n/50$ 个(因为这些与 x 相邻的顶点在圈上互不相邻)，所以它们必然存在着一条边，也就是如图中的 bz 边:

定理12.4

定理12.4 上述算法在 $G(n,1/2)$ 上以大概率在 $O(n^3)$ 时间内输出哈密顿回路.

“大概率”（with high probability，简称 w.h.p.）：
随着 n 趋于无穷，上述算法能有效找出哈密顿回路的 $G(n,1/2)$ 随机图的概率趋于 1.

1. 算法在第 1 步构造的路径 P 上至少有 $7n/8–n/50$个顶点；
2. 在第 2 步构造的圈 C 上至少有 $7n/8–n/50–1$个顶点；
3. 在第 3 步上只要还有边的两端都在圈 C 之外，则可以应用第 3.1 步或第 3.2 步，因此第 3 步总是可行的.

因此以大概率，上述算法找出 G 的哈密顿回路.

复杂性估计：算法的第 1-3 步每一步都可在 $O(n^3)$ 时间内实现，因此上述算法在 $O(n^3)$ 时间内运行.

算法复杂性分析：由于我们可以用一个标记数组，所以对于一个顶点，其是否在已有的路径或圈中可以在常数时间内得知. 所以，在第 1 步中，每加入一个顶点，可以在 $O(n)$时间内完成，顶点数至多为 n，故在有顶点加入时，复杂性不会超过 $O(n^2)$；而判断是否还有顶点可以加入，显然只需将所有顶点遍历一遍，对于每一个顶点，也可在 $O(n)$时间内判断是否可以用两种方法加入路径. 第 2 步只涉及 $O(1)$个顶点，所以可以在 $O(1)$时间内完成. 在第 3 步中，如果是利用步骤 3.1 或步骤 3.2，则每添加一个顶点，最多将图中顶点遍历一遍以及访问 $O(n)$条边，故这一部分时间复杂度不会超过 $O(n^2)$；对于步骤 3.3，要更进一步地分析. 设在圈上有与 x 相邻的按顺时针排列的两个不相邻的顶点 a 和 b，将沿顺时针方向排列的下一个顶点简称为后边的顶点，设 a 后边为 u，b 后边为 v，假设此时 x 所有邻域中的顶点都在圈上(否则可以利用步骤 3.1 或步骤 3.2 将 x 添入)，则设 x 的邻域为 $A=\{a,b,s_1,s_2,\cdots\}$，设这些顶点后边的顶点集为 $B=\{u,v,t_1,t_2,\cdots\}$，则与 u、v 之一相邻的顶点至少有 $3n/4-n/50$ 个，而 B 中的顶点至少有 $n/2-n/50$ 个，$3n/4-n/50+n/2-n/50>n$，所以 B 中有和 u 或 v 相邻的顶点. 所以步骤 3.3 的添加可以首先找到两个与 x 相邻顶点 a 和 b，这需要 $O(1)$的时间，然后遍历一遍圈，肯定有和 u 或 v 相邻的顶点，这时可以按步骤 3.3 将 x 加入圈，而这遍历需要 $O(n)$的时间，所以总的时间复杂度为 $O(n^2)$.

12.6 难解算例生成

12.6.1相变现象与难解性

- 相变现象：存在参数的一个临界值，实例的性质在临界点两侧发生了很大的突变，具有某个性质的概率在一侧随实例规模趋于无穷而渐近为 0，在另一侧则随实例规模趋于无穷而渐近为 1.
- 例如：可满足性相变是说，在临界值的一侧，以很大的概率产生有解的实例，而在临界值的另一侧，以很大的概率产生无解的实例.
- 人们通过实验发现，最难解的实例都集中在可满足性相变临界值的附近.因此人们就用临界值附近的实例作为难解算例.

约束满足问题

- 约束满足问题 是对于布尔公式可满足性、图着色等 NP 完全问题的一种自然推广．在约束满足问题的实例中，给定一些变量和这些变量之间的一些约束，每个约束规定了一些变量之间各种取值的相容组合，问题是寻找变量的赋值，以满足所有的约束.
- 为了产生约束满足问题的随机实例，人们提出过 A, B, C, D 四种模型，设 $0 < p_1 < 1$ 和 $0 < p_2 < 1$ 是两个控制参数，设有 n 个变量，设每个变量有 d 种取值，$d > 0$ 是正整数.

约束满足问题的随机模型

- **A模型** 在可能的 $n(n-1)/2$ 对变量中，每对变量以概率 p_1 成为约束，也就是在 $n(n-1)/2$ 对变量中，每对变量以概率 p_1 被选出组成约束. 对于每个给定的约束，在 d^2 对赋值中，每组赋值以概率 p_2 成为不相容赋值.
- **B模型** 在可能的 $n(n-1)/2$ 对变量中，随机选择 $p_1 n(n-1)/2$ 对变量成为约束，也就是在 $n(n-1)/2$ 对变量中，以等概率挑选一个有 $p_1 n(n-1)/2$ 对变量的子集组成约束集. 对于每个给定的约束，在 d^2 对赋值中，随机挑选 $p_2 d^2$ 个成为不相容赋值.

约束满足问题的随机模型

- **C模型** 在可能的 $n(n-1)/2$ 对变量中，每对变量以概率 p_1 成为约束，也就是在 $n(n-1)/2$ 对变量中，每对变量以概率 p_1 被选出组成约束．对于每个给定的约束，在 d^2 对赋值中，随机挑选 $p_2 d^2$ 个成为不相容赋值.

- **D模型** 在可能的 $n(n-1)/2$ 对变量中，随机选择 $p_1 n(n-1)/2$ 对变量成为约束，也就是在 $n(n-1)/2$ 对变量中，以等概率挑选一个有 $p_1 n(n-1)/2$ 对变量的子集组成约束集．对于每个给定的约束，在 d^2 对赋值中，每组赋值以概率 p_2 成为不相容赋值.

约束满足问题的随机模型（续）

- 四个模型共同的问题：渐进无解性. 就是说，当参数确定时，在变量个数变成充分大时，这些模型产生的实例基本上是没有解的，即解的个数将趋近于 **0.**

- **RB模型**：意思是对 **B模型**的修订，这是首个在理论上严格地证明了存在既有精确相变现象又有难解实例的 **NP** 完全问题的随机模型.

为RB模型产生随机实例

算法12.6

输入：参数组 $<k, n, \alpha, r, p>$，其中

每个约束由 k 个不重复的变元组成，k 是大于等于 2 的整数；

n 是变元个数；

每个变元的值域大小为 $d = n^{\alpha}$，α 是正常数；

至多有 $m = rn\ln n$个约束，r是正常数，控制约束密度；

p 是正常数，控制约束的紧度，$0 < p < 1$.

输出：在 RB 模型中产生的随机实例

1. 有重复地选择 $m = rn \ln n$ 个随机约束，每个约束由 k 个不重复的变元组成.

2. 对每个约束，均匀随机无重复地选择 $q = pd^k$ 个不相容赋值，因此相容赋值的个数为 $(1-p)d^k$个.

RB模型的相变结论

定理12.5 设 $k, \alpha > \frac{1}{k}, p \le \frac{k-1}{k}$ 都是常数, 令 $r_{cr} = \frac{-\alpha}{\ln(1-p)}$, 则

Pr[RB模型的随机实例是可满足的]

$$= \begin{cases} 1 & \text{当} r < r_{cr} \text{时} \\ 0 & \text{当} r > r_{cr} \text{时} \end{cases}$$

定理12.6 设 $k, \alpha > \frac{1}{k}, p \le \frac{\alpha}{\ln k}$ 都是常数, 令 $p_{cr} = 1 - e^{-\frac{\alpha}{r}}$, 则

Pr[RB模型的随机实例是可满足的]

$$= \begin{cases} 1 & \text{当} p < p_{cr} \text{时} \\ 0 & \text{当} p > p_{cr} \text{时} \end{cases}$$

12.6.2 隐藏解的难解算例

- 一个更难的问题是，不但要产生难解实例，而且要产生已知解的难解实例.
- 对于不完全算法来说，使用不知道解的算例来测试性能，就会遇到问题. 比如当算法没有找到解时，就无从判断算法的结果的正确性.
- 因此人们需要构造带有植入解(planted solution)的难解算例，就是预先隐藏了一个已知的解的难解算例. 对于优化问题来说，还要在算例中隐藏一个最优解.
- 在这个方面，RB模型也具有很好的性质，不仅能产生植入解的判定问题的难解算例，甚至还能产生隐藏着最优解的难解算例.

产生隐藏着解或最优解的算例

算法12.7

输入：一组控制参数

输出：隐藏解或最优解的算例

1. 选定一个要隐藏的解或最优解；
2. 按照控制参数规定的方式随机产生算例的各个约束. 每产生一个约束，就要检查这个约束是否与要隐藏的解矛盾，如果不矛盾则保留这个约束，否则就丢弃这个约束，直到产生足够多的约束得到一个算例为止；
3. 输出最后得到的算例.

用RB模型产生最大团难解算例

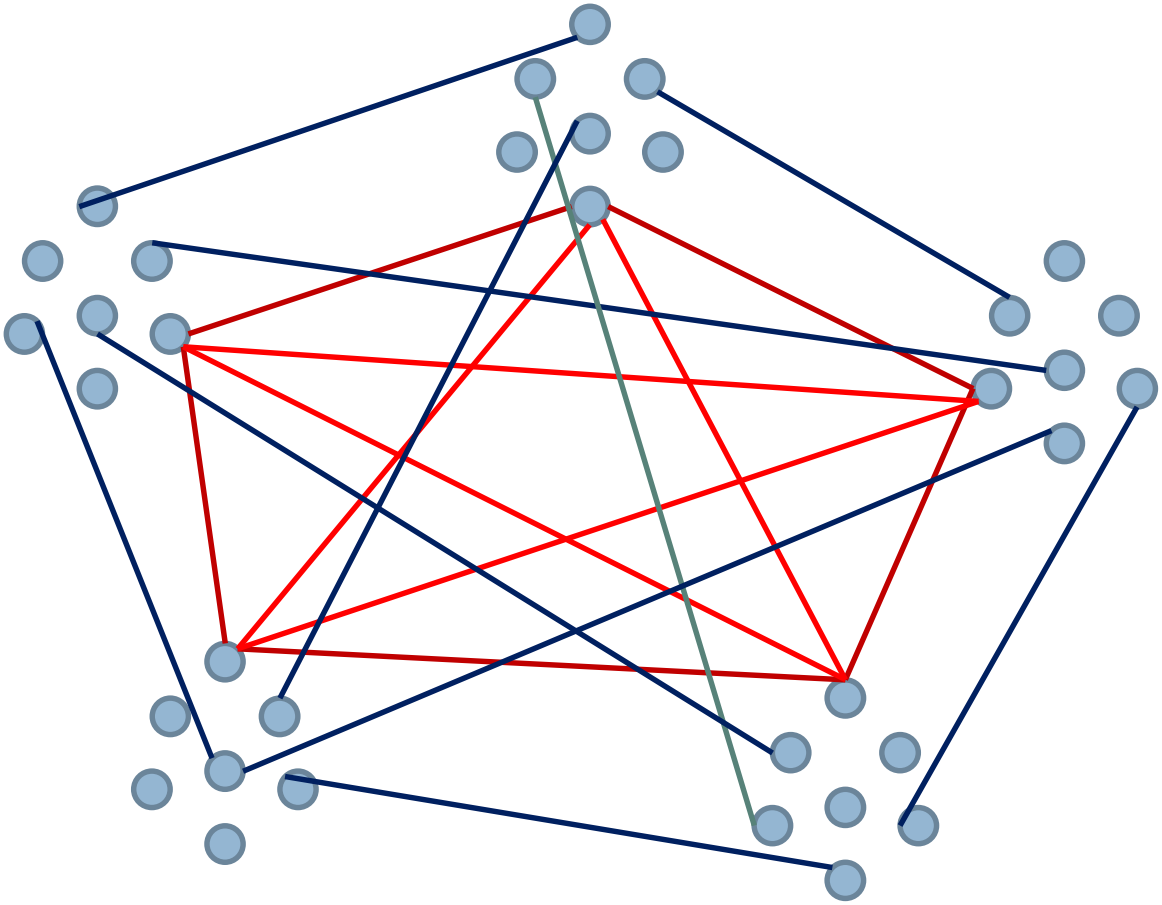
算法12.8

输入：图的顶点数 4000，要隐藏的最大团的大小100

输出：一个有 4000 个顶点的图，其中隐藏了 100 个顶点的最大团

1. 把 4000 个顶点分成 100 组，每组 40 个顶点；
2. 从每组随机挑选一个顶点，共挑出 100 个顶点，把这 100 个顶点相互之间都连上边，得到一个大小为 100 的团；
3. 然后在不同组的顶点之间随机连边，连的总边数 m 参照算法 10.6 和定理 10.5（或定理10.6）用参数 r 和 p 来定，其中 $n = 100$，$d = 40$（由此可推算出 α），$k = 2$.

算法12.8 示意图



12.7 基于统计物理的消息传递算法

12.7.1 消息传递算法与回溯法、局部搜索法的比较

- 在回溯法和局部搜索法中，除了检查赋值是否满足约束外，没有利用或很少利用实例的结构信息. 在消息传递算法中，变量只向它在其中出现的约束传递消息，约束也只跟它里面包含的变量传递消息，利用了实例本身的结构信息.
- 在消息传递算法中采用软决策，不直接为变量赋值，而是计算每个变量取某个值的概率分布，通过不断迭代来修正概率分布，直到取某个值的概率明显占优，则确定采用这个赋值；或者到概率分布没有明显改进为止，再采用其他方法来决定这个变量的取值，比如局部搜索法等.

消息传递算法(续)

- 更新概率的规则是基于统计物理模型来制定的. 根据消息的含义和更新规则，消息传递算法由简单到复杂又可分为警告传播算法（Warning Propagation，简称 WP 算法）、置信传播算法（Belief Propagation，简称 BP 算法）、调查传播算法（Survey Propagation，简称 SP 算法）等.

- 随机 3SAT 模型： 给定变元个数 n 和子句个数 m，从 $2n$ 个文字的所有长度为 3 的子句中均匀随机选择 m 个子句, 这些子句的合取就构成一个随机 3SAT公式.

12.7.2 基于统计物理的消息传递算法

算法12.9 随机 3SAT 的 SP 算法

算法的关键在于设计消息的含义和如何更新消息的机制.

$\gamma_{C\to l}$：子句C 给文字 l 发送消息 $\gamma_{C\to l}$，
消息的取值有真或假的含义.

输入：随机 3SAT 公式

输出：一个满足赋值

随机3SAT的SP算法（续）

选定某个临界值 $\varepsilon > 0$. 对每个子句 C 和每个文字 l，
若 l 在 C 中出现，则按照下列规则计算值 $\gamma_{C\to l}$：

1. 开始时，按照 [0,1] 上的均匀分布，为每个 $\gamma_{C\to l}$ 独立随机赋值；
2. 对每个子句 C，按照文字的随机排列顺序，根据下列规则 3-5 迭代更新 $\gamma_{C\to l}$；
3. 令

$$Z_l = \prod_{C':l\in C'} \gamma_{C'\to l}$$

$$Z_l^C = \prod_{C':l\in C'\wedge C'=C} \gamma_{C'\to l}$$

其中约定空的乘积等于1；

随机3SAT的SP算法（续）

4. 若 $C = j \vee k \vee l$，则令

$$\gamma_{C\to l} = 1 - \left(1 - \frac{Z_{\bar{j}}}{Z_{\bar{j}} + Z_j^C - Z_{\bar{j}} Z_j^C}\right)\left(1 - \frac{Z_{\bar{k}}}{Z_{\bar{k}} + Z_k^C - Z_{\bar{k}} Z_k^C}\right)$$

若$C = k \vee l$，则令

$$\gamma_{C\to l} = \frac{Z_{\bar{k}}}{Z_{\bar{k}} + Z_k^C - Z_{\bar{k}} Z_k^C}$$

5. 当所有 $\gamma_{C\to l}$ 的更新都小于 ε 时，停止迭代更新 $\gamma_{C\to l}$;

随机3SAT的SP算法 (续)

6. 找出 $\left| t(x) = \frac{-Z_x + Z_{\bar{x}}}{Z_x + Z_{\bar{x}} + Z_x Z_{\bar{x}}} \right|$ 最大的变量 x，如果这个绝对值小于ε，则运行随机游动算法；

7. 当 $t(x) > 0$ 时，把 x 赋值为真，当 $t(x) < 0$ 时，把 x 赋值为假，化简公式，回到第 2 步，对剩余的公式继续迭代更新 $\gamma_{C \to l}$.

12.8 量子算法简介

12.8.1 量子比特

几率幅 计算基 量子寄存器 线性叠加态

12.8.2 正交测量

12.8.3 量子门

定理10.7

12.8.4 一个量子算法

道奇算法

量子比特

用 $|\phi\rangle$ 记号表示量子比特，以区别于经典比特，这种记号称为迪拉克（Dirac）记号.

经典比特有两个取值 0 和 1，要么为 0、要么为1.

量子比特也有值 $|0\rangle$ 和 $|1\rangle$，其物理实现可以是电子的两个自旋方向、光子的两个偏振方向、原子的两个能级等.

但在量子世界中这些状态是可以叠加的，比如电子可以同时处在两个自旋方向上、光子可以同时具有两个偏振方向、原子可以同时处于两个能级等.

量子比特

量子比特还可以是 $|0\rangle$ 和 $|1\rangle$ 的线性叠加，即形如

$$\alpha|0\rangle + \beta|1\rangle$$

的叠加态，其中 α 和 β 是两个复数，称为几率幅，表示处在 $|0\rangle$ 的概率是 $|\alpha|^2$，处在 $|1\rangle$ 的概率是 $|\beta|^2$，这里 $|\alpha|$ 表示复数 α 的模，所以 α 和 β 要满足 $|\alpha|^2 + |\beta|^2 = 1$.

用线性代数的话来说，量子比特就是一个 2 维复系数线性空间中的单位向量.

计算基

如果用 $|0\rangle$ 和 $|1\rangle$ 作为一组基，称为计算基，简称基 (Computational bases)，相应的坐标表示是

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix},$$

于是 $|\phi\rangle$ 记号就表示了列向量.

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = \begin{pmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{pmatrix}$$

$$|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = \begin{pmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{pmatrix}$$

也可以作为一组基. 通常我们都用 $|0\rangle$ 和 $|1\rangle$ 作为基，除非另加说明.

张量积

张量积：是两个以上线性空间或两个以上向量之间的一种运算.

两个线性空间的张量积还是一个线性空间，它的维数等于原来两个线性空间维数之积，它的基等于原来两个线性空间的基的卡氏积，即对于原来两个线性空间的每一对基 $|i\rangle$ 和 $|j\rangle$，在张量积空间有一个对应的基，记作 $|i\rangle \otimes |j\rangle$，也简记作 $|i\rangle\,|j\rangle$、$|i,j\rangle$ 或 $|ij\rangle$.

例子：向量 $\alpha_1|0\rangle + \beta_1|1\rangle$ 和 $\alpha_2|0\rangle + \beta_2|1\rangle$ 的张量积就是

$$(\alpha_1|0\rangle + \beta_1|1\rangle) \otimes (\alpha_2|0\rangle + \beta_2|1\rangle)$$

$$= \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle$$

量子寄存器

两个或多个量子比特可以组成量子寄存器，量子寄存器的状态就是各个量子比特状态的张量积.

两个经典比特构成的经典寄存器总共有四个状态 00、01、10、11，两个量子比特构成的量子寄存器也有四个基 $|00\rangle$、$|01\rangle$、$|10\rangle$、$|11\rangle$，以及这四个基的线性叠加态

$$\alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

其中α、β、γ、δ都是复数，也称为几率幅，满足

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$$

量子寄存器

n 个量子比特构成的量子寄存器有 2^n 个基 $|00\ldots0\rangle$、$|00\ldots1\rangle$、…、$|11\ldots1\rangle$，每个基对应一个长度为 n 的 0-1 串，量子寄存器状态是 2^n 个基态的叠加态.

而 2^n 个几率幅也满足模的平方和等于 1 的条件，描述这样的一个状态就需要 2^n 个 几率幅.

在直观上，若用经典寄存器来模拟 n 个量子比特构成的量子寄存器的状态，就需要保存 2^n 个几率幅，这个开销是指数增长的.

由于叠加态的存在，使得量子计算机不仅有可能在存储能力上比经典计算机更强，而且还具备了并行处理能力.

内积

在量子比特的线性空间上可以定义一个内积.对于两个量子比特 $|\phi\rangle$ 和 $|\varphi\rangle$，令内积

$$\langle\phi,\varphi\rangle = (\alpha^*,\beta^*)\begin{pmatrix}\gamma\\ \delta\end{pmatrix} = \alpha^*\gamma + \beta^*\delta,$$

α^* 是 α 的共轭复数，线性空间称为希尔伯特空间.

我们约定对于一般的列向量 $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$，让 $\langle\phi|$ 表示 $|\phi\rangle$ 的共轭转置，即

$$\langle 0| = (1,0)，\langle 1| = (0,1)，\langle\phi| = \alpha^*\langle 0| + \beta^*\langle 1| = (\alpha^*,\beta^*)$$

于是内积 $\langle\phi,\varphi\rangle$ 可以记作 $\langle\phi\,||\varphi\rangle$，或简记作 $\langle\phi\,|\,\varphi\rangle$

$|0\rangle$ 和 $|1\rangle$ 是一组标准正交基，同样可以验证 $|+\rangle$ 和 $|-\rangle$ 是另一组标准正交基.

正交测量

虽然量子比特可以具有叠加态，几率幅 α 和 β 作为复数里面包含有无穷多位的信息，但不能通过物理测量来得到几率幅，因此无法直接利用这些信息，对量子比特最简单的测量是下面介绍的正交测量.

每个正交测量由一组正交基表示，一次测量的结果就是一个随机变量，取值就是测量所用的各个正交基，概率就是对应几率幅的模的平方.

例如，在 $|0\rangle$ 和 $|1\rangle$ 下测量 $|\phi\rangle = \alpha|0\rangle + \beta|1\rangle$，结果就是以概率 $|\alpha|^2$ 得到 $|0\rangle$，以概率 $|\beta|^2$ 得到 $|1\rangle$，这正是前面说量子比特可以同时处在 $|0\rangle$ 和 $|1\rangle$ 上的精确含义.

正交测量

在基 |0⟩ 和 |1⟩ 下测量 |+⟩ = 1/(|0⟩ + |1⟩)，就能以等概率1/2分别得到 |0⟩ 或 |1⟩. 这相当于抛掷一枚均匀硬币的结果，所以量子算法天然就能轻松地模拟经典随机算法.

对于量子算法而言，几率幅只需取实数就够了

在测量之后，原来的量子比特就变成测量结果中出现的那个正交基，而原来的状态就消失了.

量子测量的这种性质在一定程度上抵消了量子叠加态带来的好处，给量子算法的性能带来了严重限制.

量子门

采用著名计算机科学家姚期智先生提出的量子电路模型来描述量子算法，量子电路的基本运算单元是量子门.

在量子计算中，量子比特是希尔伯特空间中的单位向量.

在量子比特上的基本操作除了上述的正交测量之外，还有保持向量长度不变的线性变换，就是数学上所说的酉变换（物理学上叫么正变换），这些变换就称为量子门.

量子门

量子门可以用酉方阵来表示，即满足

$$U^\dagger U = UU^\dagger = I$$

的方阵 U，其中 $U^\dagger$ 是 U 的共轭转置，I 是单位阵.

一个量子门 U 作用在量子比特 $|\phi\rangle$ 上的结果，就得到量子比特 $U|\phi\rangle$.

注意量子门作为酉变换，都是可逆的，而经典与门和或门都不是可逆的，只有经典非门是可逆的.

单比特量子门

单比特量子门：

$$I=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \quad X=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \quad Z=\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} \quad H=\begin{pmatrix}\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}\\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}}\end{pmatrix}$$

I 是恒等映射

$X|0\rangle = |1\rangle$，$X|1\rangle = |0\rangle$，相当非门，起翻转比特作用；

Z 对于另一组基 $|+\rangle$ 和 $|-\rangle$ 起非门的作用，即

$$Z|0\rangle = |0\rangle,\ Z|1\rangle = -|1\rangle,\ Z|+\rangle = |-\rangle,\ Z|-\rangle = |+\rangle,$$

H 称为哈达玛门，在两组不同基之间起坐标变换作用，即

$$H|0\rangle = |+\rangle,\ H|1\rangle = |-\rangle,\ H|+\rangle = |0\rangle,\ H|-\rangle = |1\rangle$$

两比特量子门

两比特量子门：

约定空白处都是 0

$$CN = \begin{pmatrix} 1 & & \\ & & X \end{pmatrix} = \begin{pmatrix} 1 & 0 & & \\ 0 & 1 & & \\ & & 0 & 1 \\ & & 1 & 0 \end{pmatrix}$$

受控非门 *CN* (Controlled-Not)

$$CN\,|00\rangle = |00\rangle,\ CN\,|01\rangle = |01\rangle,\ CN\,|10\rangle = |11\rangle,\ CN\,|11\rangle = |10\rangle$$

当第一个量子比特等于 1 时，就对第 2 个量子比特做非运算，否则什么都不作.

第一个量子比特称为控制比特，第二个比特称为目标比特，我们把这样的门记作 $CN_{1,2}$，即 $CN_{1,2}\,|a,b\rangle = |a, a\oplus b\rangle$，其中⊕ 表示异或运算或模 2 加法.

类似地可定义 $CN_{2,1}$，让第二个比特作控制，第一个比特作目标，即 $CN_{2,1}\,|a,b\rangle = |a\oplus b, b\rangle$.

定理12.7

定理12.7 不存在两比特量子门 $COPY$，使得对于任意量子比特 $|\phi\rangle$，$COPY\,|\phi\rangle\,|0\rangle = |\phi\rangle\,|\phi\rangle$.

证 用反证法，假设存在这样的两比特量子门 $COPY$，则按照 $COPY$ 的定义，就有

$$COPY\,|0\rangle\,|0\rangle = |0\rangle\,|0\rangle,\quad COPY\,|1\rangle\,|0\rangle = |1\rangle\,|1\rangle.$$

当 $|\phi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle$ 时，按照 $COPY$ 的定义，就有

$$\begin{aligned} COPY\,|\phi\rangle\,|0\rangle &= |\phi\rangle\,|\phi\rangle \\ &= (\alpha\,|0\rangle + \beta\,|1\rangle)\,(\alpha\,|0\rangle + \beta\,|1\rangle) \\ &= \alpha^2\,|0\rangle\,|0\rangle + \alpha\beta\,|0\rangle\,|1\rangle + \beta\alpha\,|1\rangle\,|0\rangle + \beta^2\,|1\rangle\,|1\rangle \end{aligned}$$

定理12.7证明（续）

按照 $COPY$ 的线性性质（量子门都是线性变换）

$$COPY\ |\phi\rangle\,|0\rangle = COPY\,(\alpha\,|0\rangle + \beta\,|1\rangle\,)\,|0\rangle$$

$$= COPY\,(\alpha\,|0\rangle\,|0\rangle + \beta\,|1\rangle\,|0\rangle\,) = \alpha\,|0\rangle\,|0\rangle + \beta\,|1\rangle\,|1\rangle.$$

对于任意的 α 和 β 上述结果不会总是相同的，矛盾！

上述定理表明量子信息不可克隆.

但我们可以交换两个变量的赋值，即存在两比特量子门 $SWAP$，使得 $SWAP\,|a\rangle\,|b\rangle = |b\rangle\,|a\rangle$:

$$CN_{1,2}\,|a\rangle\,|b\rangle = |a\rangle\,|a\oplus b\rangle,$$

$$CN_{2,1}\,|a\rangle\,|a\oplus b\rangle = |a\oplus(a\oplus b)\rangle\,|a\oplus b\rangle = |b\rangle\,|a\oplus b\rangle,$$

$$CN_{1,2}\,|b\rangle\,|a\oplus b\rangle = |b\rangle\,|(a\oplus b)\oplus b\rangle = |b\rangle\,|a\rangle.$$

道奇算法

算法12.10

输入：以暗箱形式给出的函数 $f:\{0,1\}\to\{0,1\}$，

两比特量子寄存器初始状态 $|0\rangle|1\rangle$

输出：宣布 $f(0)=f(1)$，或者宣布 $f(0)\neq f(1)$

1. 对寄存器的两个量子比特分别作用哈达玛门 H；
2. 对寄存器的两个量子比特共同作用函数门 O_f；
3. 对寄存器的第一个量子比特作用哈达玛门 H；
4. 测量寄存器的第一个量子比特，如果测量结果是 $|0\rangle$ 就宣布 $f(0)=f(1)$，如果是 $|1\rangle$ 就宣布 $f(0)\neq f(1)$.

定理12.8 道奇算法正确解决上述问题，且只求一次函数值.

量子算法

道奇算法：是人们找到的第一个比经典算法加速一倍的量子算法.

格罗乌（Grove）量子算法：从N个输入（每个输入带标记或不带标记）中找出一个带标记的输入，只花费$O(\sqrt{N})$时间，比$O(N)$时间的经典算法有平方级的加速.

肖尔（Shor）量子算法：能在$O(n^2 \ln n \ln\ln n)$时间内完成n位整数的因子分解，这比已知最快的$\exp(\Theta(n^{1/3}\ln^{2/3} n))$时间的经典算法具有指数倍加速.

对于某些特定问题，量子算法比经典算法更强. 但是有证据表明，量子算法大概和概率算法一样，都不能在多项式时间内解决NP完全问题.

小结

NP难问题的处理策略

- 对问题施加限制：参数化分析
- 改进指数时间算法：降低指数时间复杂度函数的底
- 启发式方法：回溯与分支限界、局部搜索(随机游动、模拟退火)
- 分析问题平均情形下的计算难度
- 找到难解算例，以测试算法性能