作业 6

1）

1. C=0

2. E’=E[G];

3. While (E’!=0) DO

4.任取(u, v)属于 E’;

5. c += u + v（将 u，v 两个顶点加入 c）

6.从 E’中删除所有与 u 或 v 相连的边;

7.return C

证明：

A 集合为所取边的集合，C 为所取顶点的集合，因为每次取完边后，会去掉所有临接的边，所以每次 C 增加的两个顶点一定不相同。故 2|A| = |C|。又，根据要求可知，设 C* 为优化解，每条边的一个顶点一定会在 C*中，A 是所有边集合的子集,所以 A 集合的优化解一定在 C*中，而 A 集合优化解的数量为|A|，故|A| <= |C*|。所以 |C| = 2|A| <= 2|C*|。 |C|/|C*| <= 2

2)

按处理时间递减顺序。近似解中第 j 台机器的处理时间最长,最后处理任务 t h，凡是在 th 任务后的任务时间都小于 t h

记第 h 个任务开始执行的时间为 T start

则 T=T start +t h

T start <= sum(i = 1 to h-1)t i / m

其余 n – h 个任务分配到了 m – 1 个机器，并且分配的并行时间都不大于 t h

故 th >= sum(i = h+1 to n)ti / m – 1

(m – 1)/m * th >= sum(i = h+1 to n)ti

T start + th <= sum(i = 1 to h)t i / m + (m – 1)/m * th <= sum(i = 1 to n)ti / m + (m – 1)/m * th – sum(i = h+1 to n) / m <= T * + [(m – 1)* th – sum(i = h+1 to n)] / m <= T* + (m – 1) / m * th

值最大的情况是 t h 是最后一个任务，即 t h = t min

t min * n / m <= T*

(m – 1) / m * t min <= (m – 1) * T* / n

所以　T <= (1 + (m – 1) / n) * T

是个 1 + (m – 1) / n 近似算法

3)

解:

修改贪心策略:寻找能覆盖最多未覆盖点的集合,且它的代价最小。代价计算如下：

设每个集合 S 的覆盖代价为 c(S),则它的成本效益为 α=c (S)/ |S−C| ,我们要找 α 最小的。

C 为已经覆盖点集合。∀e∈S−C, 规定 price(e)=α。

按照算法覆盖元素的顺序对 U 中元素进行编号,设 e ,e 2 ,...,e k 。

在任意一次迭代中,最优解中剩余的集合能以至多 OPT 的费用覆盖剩余元素,因此在这些集合中,必定存在一个成本效益至多为 OPT/|U−C|的集合,即未被覆盖元素集合。在覆盖元素 e k 的那次迭代中, U−C 至少包含 n-k+1 个元素,因为这次迭代中用成本效益最小的集合覆盖 e k ,所以有: price (e k) ≤ OPT| U−C | ≤OPT/n−k+1 总费用 sun(i = 1 to k)price(e i)≤ OPT ∗ sum(i = 1 to k)1/n−i+1=OPT ∗ (1/n+ 1/n−1 +⋯+ 1)= H(n) ∗OPT

因此算法的近似比为 H(n)。

4)

先求出 G 的最小生成树 T,取 T 中度数为奇数的点构成子集 V 1 ,令 G 1 为只包含 V 1 所有顶点的 G 的最大子图,求出 G 1 的权值最小的最大匹配 M,将 M 加入 T 中,则所有顶点度数为偶数,构造一个欧拉回路 G 2 ,删去非第一次出现的顶点,得到哈密顿环近似解 C。

计算近似比：设优化解为 C * ,显然 |C *| <= |C| ; |C| <= |T| + |M|, |T| <= |C*|;

删去 C * 中不在 V 1 中的点,构造回路 G3 , |G3| <= |C*| ,而 G3 可以看做由 G1 的两个最大匹配构成,即 G3 = M1 + M2 ,而 |M1| >= |M|, |M2| >= |M| ,所以|G3| = |M1| + |M2| >=2|M| ,即 |M| <= 1/2|G3| <= 1/2|C*|;于是|C| <= |T| + |M| <= 3/2|C*|;

近似比为 3/2。

5)

(1)整数规划问题：

{ki | 1,如果 i 顶点在 C 中;0，i 顶点不在 C 中}

问题变成：

min sum(ki * w(i)) 1 式

Opt ki + kj >= 1 如果 以顶点 i，j 为端点的边在边集合中。

(2)令 ki >= 0，求出线性规划最优解，对于所有 ki >= 1/2 的顶点 i 划入 C 中。

显然|C(opt)| <= |C*| <= |C|,而 1 式 >= sum(1/2 * w(i))如果 ki >= ½ = 1/2|C|;

所以|C| <= 2|C(opt)| <= 2|C*|，近似比为 2。

6)

(1)证明：

按照最优解中的边将两个顶点分为一组,$z_*=\Sigma w_{ij}$, $\in E_*$,且每个顶点只出现一次。因此 z_{LP} 是 z_*的上界。

(2) 证明:

对于我们按贪心算法选出的边(i,j),我们对 xi 和 xj 均赋值,代价为 z_G。这个赋值满足线性规划问题的所有约束。$2z_G \le z_{LP} \le z_*$因此这个贪心算法的近似比是 2。

作业 7

1）

算法：

d = max{m, n, l};

a = RandomGet(1, 2, 3, ..., f ∗ d)//从 1,2,3,...到 f ∗ d 中随机选取一个数 a

if p(a) ∗ q(a) = r(a) then

  return yes;

else

  return no;

分析：

分为以下三种情况说明:

1. 如果 p(x) ∗ q(x) = r(x)恒成立,那么算法必定正确;

2. 如果 p(x) ∗ q(x) = r(x)恒不成立,当算法输出 no 时,算法正确;

3. 如果 p(x) ∗ q(x) = r(x)恒不成立,当算法输出 yes 时,算法错误;

下面只需要考虑算法错误的情况。 此时, $p(a) * q(a) - r(a) = 0$, 该方程的阶至多为 d, 其中 $d = \max\{m, n, l\}$. 由代数基本定理可知, 方程至多有 d 个根, 也即在 1, 2, 3, ..., $f * d$ 的 $f * d$ 个数中至多有 d 个数使得方程为 0, 也就是算法出错。

那么算法出错的概率 $P \le d / f * d = 1 / f$. f 是人工输入的参数,用来控制随机算法出错概率的上界。

时间复杂度: $O(m) + O(n) + O(l) = O(d)$.

由于算法一定会输出解,但可能给出错误解,所以是 Monte Carlo 算法。

2)

构造一个随机的 $r \times 1$ 的 01 矩阵 r 。 同时设置一个记号 P :

算法:

随机生成 r（r 是一个 $r * 1$ 阶的随机 01 矩阵）

令 $P = A \times (B r) - C r$;

```
if P = 0 then
    return yes;
else
    return no;
```

分析:

先考虑 $A \times B = C$:

$P = A \times (Br) - Cr = 0$

此时，算法正确。

再考虑 $A \times B \mathrel{!=} C$: 此时,将 P 写为:

$P = Dr = (p_1, p_2, ..., p_n)^T$.

$D = A \times B - C = (d_{ij})$.

$P_i = \text{sum}(k = 1 \text{ to } n) d_{ik} * r_k$

又全概率公式可以推得

$p_{max} = ½$

所以算法出错的概率最大为 1/2。

对于判定 $A \times B = C$, 精确算法需要的时间复杂度是 $O(pqr)$. 该算法判定的时间复杂度是 $O(pr)$.

由于算法一定会输出解,但可能给出错误解,所以是 Monte Carlo 算法。

3)

考虑使用 Kruskal 方法生成 MST: 算法先将所有的边按边权升序排序,先放入最小权的边,对剩下的每条边依此考虑边的两个端点是否在一个集合中,如果是则忽略,否则就加入生成树。 对照课本的 Min-cut 算法,我们只需要证明二者等价。

Min-cut 算法每次收缩一条边(u, v),将 u 和 v 之间的边删除,然后将连接 u,v 的边连接至一个新节点 z,z 是 u,v 合并后的点; 对应于 Kruskal, 选择一条边加入生成树的同时,将边的两个端点合并,放在一个点集合中,这一步操作和上述操作等价;

最后 Min-cut 算法输出剩余两个点之间的边作为割集; Random-Mincut 最后去掉 MST 中最大权的边,将一个点集分成两部分,这两个点集等价于 Min-cut 收缩后剩余的两个点,可以将这两个点 看成 Random-Mincut 输出的两个点集的任意一个祖先。

由 Min-cut 易得最小割的概率为 ω(1/1^2)。

4)

(1) 对于图 G(V,E), 反证:假设 I 不是独立集,那么∃u, v ∈ I, u,v 相连。对于 u,在算法第 6 步中删除了所有和 u 相邻的顶点,而 v ∈ I 和 u 相连产生矛盾,原命题成立,即 I 是 G(V, E)的一个独立集。

(2)证明:

∀u ∈ V，度为 du，意味着如果 u 的 du 个相邻节点被选中则，u 不会进入 I，反之谁都没选中则有可能成为被选中进入 I 的。则不会进入 I 与进入 I 的比值是 du / 1，所以选入 I 的概率是(1 / du + 1)。

5)

(1)算法:

在 0, 1, ..., n − 1 内随机选一个数 x,让 y = z − x mod n;

将 F (x) + F (y) mod m 的结果作为 F (z)的结果输出。

又 P (F (x) is wrong) = 1/5 ,P (F (y) is wrong) = 1/5 ,x 和 y 并不是相互独立,所以 P (F (x) is wrong ∪ F (y)is wrong) ≤ 2/5 ;

则 P (F (z) is true) ≥ 3/5 > 1/2 ;

(2)如果算法运行 3 次,我们使用投票原则,即两次或两次以上 F (z)为真就认定 F (z)为真,设一次试验成功的概率为 p,则 p ≥ 3/5 ,那么有:

P (F (z) is true) =C(3,2) * p * p * (1 – p) + C(3,3) * p * p * p >= 0.648