不含钱机制：稳定匹配

2024-2025 学年春夏学期计算经济学讨论班

郑涵文

zhwen@zju.edu.cn

2025 年 4 月 6 日

Introduction

之前我们学习过的很多内容都和钱有关，比如说拍卖的相关机制设计、物品分配等等．但也有一些情况下我们不需要引入金钱，一个很好的例子是肾脏交换．

肾脏交换的背景

一些人肾脏衰竭急需肾脏移植，通常病人的家庭成员会愿意提供自己的一个肾脏．但是事实是病人和供体不一定兼容，比如 O 型血的病人只能从相同血型的人那里获取肾脏．假设病人 P_1 和供体 D_1 不兼容，病人 P_2 和供体 D_2 也不兼容，但是 P_1 和 D_2 兼容，P_2 和 D_1 也兼容．我们可以通过交换供体来解决这个问题．即 P_1 和 D_2 进行肾脏移植，P_2 和 D_1 进行肾脏移植．

问题建模

我们很容易能想到把这个问题抽象成为一个图论的问题．我们把一对病人和供体看成一个点．当点 (P_1, D_1) 和 (P_2, D_2) 之间有一条边时，表示 P_1 和 D_2 兼容，P_2 和 D_1 也兼容．我们可以把这个问题看成一个图的匹配问题．我们如果想要让最多的病人获救，那么就需要找到一个最大匹配．

图: 肾脏交换的图论模型

DSIC?

如果单纯只是求一个最大匹配，这在算法实现上很简单，但是事实是这个匹配不一定唯一．我们可以看下图的第二个例子：(P_1, D_1) 可以选择和剩下三个顶点的任意一个进行匹配．在这种情况下我们应该如何选择？

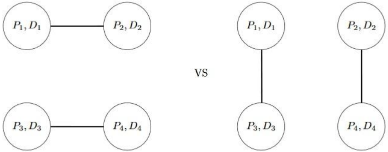


Figure 5: Different matchings can match the same set of vertices.

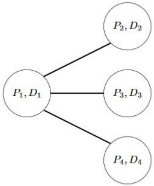


Figure 6: Different maximum matchings can match different subsets of vertices.

DSIC?

一个解决方案是在一开始就给所有的病人-供体对排出一个优先级．这个优先级由一些现实因素决定，如已经等待的时间、找到一个兼容肾脏的难易程度等等．具体而言，我们按照优先级从高到低给顶点编号排序为 $1, 2, \cdots, n$．然后使用以下的算法求匹配：

成对肾脏交换中的优先级机制

初始化 M_0 为 G 的最大匹配集合

for $i=1, 2, \cdots, n$ **do**

  用 Z_i 表示匹配集 M_{i-1} 中包含顶点 i 的所有匹配

  if $Z_i \neq \varnothing$ **then**

    设置 $M_i = Z_i$

  else if $Z_i = \varnothing$ **then**

    设置 $M_i = M_{i-1}$

返回任意一个匹配 M_n

图: 肾脏交换的优先级算法

肾脏交换的优先级算法

这个算法实际上在说，在第 i 轮的时候，检查一个最大匹配能否在匹配顶点 i 的时候不影响 i 前面的顶点的匹配．如果 i 阻碍了前面的顶点的匹配，那就跳过对 i 的匹配．从某一个角度来说，如果用二进制给每个最大匹配编号（优先级最高的在最高位），那么这个算法就是求解二进制最大的匹配．

定理

成对肾脏匹配的优先级机制算法是 DSIC 的．

问题定义

肾脏交换的例子是一个很好的引入．接下来我们研究稳定匹配问题，它和肾脏交换问题很相似，都可以从图论匹配角度进行建模．

稳定匹配的问题定义

W 表示一个由 n 个工人组成的集合，F 表示由 m 个公厂组成的集合．$c: F \to \mathbb{Z}_+$ 是容量函数，表示一个工厂 i 最多接受 $c(i)$ 数量的工人．二分图 $G = (W, F, E)$，其中 E 表示工人和工厂之间的边，表示这一对工人-工厂可能会进行匹配．对于在 G 中的节点 v，用 $N(v)$ 表示它的邻居（显然工人的邻居全是工厂，工厂的邻居全是工人）．每个工人 w 对于它的邻居 $N(w)$ 有一个偏好列表 $l(W)$，表示它对工厂的偏好排序．相应地每个工厂 f 对于它的邻居 $N(f)$ 也有一个偏好列表 $l(F)$，表示它对工人的偏好排序．工人和工厂偏向于选择自己的邻居进行匹配，并且对于和非邻居匹配这件事，它们更宁愿不进行匹配．如果一个工人或者工厂最后没有进行匹配，那么记它和 $\perp$ 匹配．

问题定义

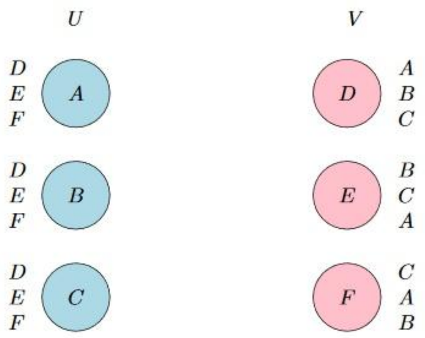


Figure 9: An instance of stable matching

问题定义

事实上这个问题可以从一些特殊情况入手，逐步过渡到我们上面的定义的一般情况．

Setting 1

这里我们让工人和工厂数保持一致，即 $n = m$；并且让所有工厂的容量均为 1，即 $c(i) = 1$；并且图是完全二分图，即每个工人和每个工厂都有边相连．这个特殊情况是最简单的情况，并且也是我们之后主要讨论的情况．

Setting 2

这里 n 和 m 不一定相等，并且图不一定是完全二分图，可以是任意的图．但是仍然保持工厂的容量均为 1．

Setting 3

这里就是上面定义所述的最一般的情况．

完美匹配/稳定匹配

一些约定

1. 如果一个工人 w 在工厂 f, f' 之中更喜欢 f，那么我们记 $f \succ_w f'$；对于工厂同理.
2. 完美匹配：如果一个匹配 $\mu \subset E$ 使得 G 中每个顶点都有一条 μ 中的边相连，那么这个匹配就是完美匹配.
3. 如果边 (w, f) 在匹配 μ 中，那么我们记 $\mu(w) = f$ 以及 $\mu(f) = w$.

阻塞对

在一个完美匹配 μ 中，工人 w 和工厂 f 如果满足 $w \succ_f \mu(f)$ 且 $f \succ_w \mu(w)$，那么我们称 (w, f) 是一个阻塞对. 也就是说 (w, f) 是一个阻塞对当且仅当 w 和 f 都更喜欢对方而不是各自的匹配.

稳定匹配

没有阻塞对的完美匹配被称为稳定匹配.

延迟接受算法

我们发现完美匹配能够让每个工人都有班上（每个工厂都有人干活），这是很好的．近一步地，在稳定匹配下，顾名思义这个匹配就稳定了下来，没有人有动机去改变自己的选择．怎样得到稳定匹配呢？我们接下来要提出一个算法：延迟接受算法．它具有非常好的性质，并且算法流程很简单．我们将从 Setting 1 入手．

Algorithm 1.8. Deferred acceptance algorithm

Until all firms receive a proposal, do:

1. $\forall w \in W$: w proposes to its best uncrossed firm.
2. $\forall f \in F$: f tentatively accepts its best proposal and rejects the rest.
3. $\forall w \in W$: If w got rejected by firm f, it crosses f off its list.

Output the perfect matching, and call it μ.

图: 延迟接受算法

延迟接受算法

这个算法是在说：每个工人都有一个目标工厂列表（当然在 Setting 1 中每个工人的初始目标工厂列表肯定是所有工厂）．总共进行若干轮选择，对于每一轮：首先，每个工人 w 都会向自己目标工厂列表中最喜欢的工厂提交申请；然后每个工厂 f 对其接到的若干个申请进行评估：**暂时**接受申请人中自己最喜欢的工人，并拒绝其他工人；最后，如果工人 w 被工厂 f 拒绝了，那么它就会从目标工厂列表中删除 f. 重复这样的轮次直到在某一轮中所有的工厂都收到了申请（这在 Setting 1 中表明，每个工厂收到且仅收到一个工人的申请，这样的结果一定是一个完美匹配），并输出这个完美匹配 μ.

一些观察

- 如果工厂 f 在某一轮收到了至少一个申请，那在接下来的每一轮它都会收到至少一个申请．
- 如果工厂 f 在某一轮收到了某个 w 的申请，那工厂 f 最终匹配到的工人 w' 一定满足 $w \succ_f w'$.

延迟接受算法

定理

延迟接受算法输出的完美匹配是稳定匹配.

证明

假设不是稳定匹配，那么存在一个阻塞对 (w, f'). 不妨设 $\mu(w) = f, \mu(f') = w'$. 既然 (w, f') 是阻塞对，那么 w 更喜欢 f' 而不是 f，说明在算法过程中 w 肯定先向 f' 提出申请，并被拒绝了. 根据我们先前的观察，f' 最后一定匹配到了一个比 w 更好的工人，也就是说明 $w' \succ_{f'} w$. 但是这和我们根据阻塞对的定义得到的结论，即 w 更喜欢 f' 相矛盾了.

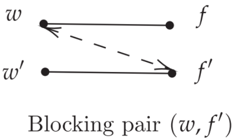


Figure 1.3 Blocking pair (w, f').

延迟接受算法

我们接下来要讨论延迟接受算法的另一个特点，即它会使得提出申请的那一方获得最好的结果．也就是说，对于工人来说，使用我们上面所讲的延迟接受算法，能够得到一个稳定匹配，使得这个稳定匹配对于每个工人得到的结果，在所有可能的稳定匹配中是最好的．

定义

记 S 表示 (W, F) 的所有稳定匹配的集合．对于每个工人 w，它可能的匹配集就是所有稳定匹配中可能和它匹配的工厂的集合：定义为 $R(w) = \{f \in F | \exists \mu \in S, \mu(w) = f\}$．定义**最优工厂**为 $R(w)$ 中 w 最喜欢的那一个，我们使用 $optimal(w)$ 表示．对于**最劣工厂**我们则使用 $pessimal(w)$ 表示．对于工厂而言它们也有匹配集、最优工人和最劣工人，定义也是类似的．

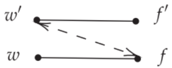
延迟接受算法

引理

两个工人不可能有同样的最优工厂，即工人和工厂之间通过最优工厂存在一个一一对应关系.

证明

假设不是这样，那么 w, w' 有同样的最优工厂 f. 不失一般性假设 $w' \succ_f w$. 取某个匹配 μ 使得 $\mu(w) = f$ (一定存在这样的匹配)，然后假设在 μ 中 $\mu(w') = f'$. 既然 f 是 w' 的最优工厂，那么一定有 $f \succ_{w'} f'$. 根据阻塞对的定义，(w', f) 是一个阻塞对，这和 μ 是稳定匹配矛盾了.



Blocking pair (w', f) with respect to μ.

Figure 1.4 Blocking pair (w', f) with respect to μ.

延迟接受算法

根据上面的引理我们很容易得到推论，就是让每个工人和它的最优工厂进行匹配，得到的是一个完美匹配，我们记其为 μ_W. 并且我们还有更好的结论.

引理

μ_W 是一个稳定匹配.

证明

假设不是如此，那么匹配 μ_W 存在一个阻塞对 (w,f')，并且我们记 $\mu_W(w)=f, \mu_W(w')=f'$. 那么根据阻塞对的定义有 $w \succ_{f'} w'$，$f' \succ_w f$. 因为 $optimal(w')=f'$，所以存在一个稳定匹配 μ' 使得 $\mu'(w')=f'$. 假设 $\mu'(w)=f''$. 既然 $optimal(w)=f$，那么 $f \succ_w f''$. 结合前面得到的 $f' \succ_w f$，说明 $f' \succ_w f''$，结合 $w \succ_{f'} w'$ 能够得到 (w,f') 是 μ' 的一个阻塞对，这和 μ' 是稳定匹配矛盾了.

(a) Blocking pair (w,f') with respect to μ_W (b) Blocking pair (w,f') with respect to μ'

Figure 1.5

最优匹配/最劣匹配

同理我们能够证明每个工人的最劣工厂都是唯一的，并且每个工人和其最劣工厂进行匹配也能得到一个稳定匹配．我们称每个工人和最优工厂匹配得到的这个匹配为**工人最优的稳定匹配**，和最劣工厂匹配得到的这个匹配为**工人最劣的稳定匹配**．对于工厂来说他们也有工厂最优的稳定匹配和工厂最劣的稳定匹配，定义是类似的．我们约定工人最优的稳定匹配为 μ_W，工厂最优的稳定匹配为 μ_F．

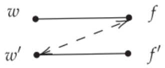
延迟接受算法 -> 工人最优匹配

定理

由工人提出申请的延迟接受算法输出的匹配是工人最优稳定匹配 μ_W.

证明

假设不是这样，那么在算法过程中一定存在一个工人被他的最优工厂拒绝．不妨设**第一次**出现这种情况时，工人 w 向他的最优工厂 f 提交申请并被拒绝了．那说明在那一轮申请中，工厂 f 肯定暂时接受了一个更好的工人 w'，显然有 $w' \succ_f w$. 由于两个工人不会有相同的最优工厂，所以 $optimal(w') = f' \neq f$. 由于 w 是第一个被最优工厂拒绝的工人，说明 w' 肯定还没有被他的最优工厂拒绝，那这显然说明他肯定还没有给最优工厂提交过申请，这说明 $f \succ_{w'} f'$. 我们现在考虑工人最优稳定匹配 μ_W，显然有 $(w, f), (w', f') \in \mu_W$，那么根据前面得到的结果，可以发现 (w', f) 是一个阻塞对，与 μ_W 是稳定匹配矛盾了．



Blocking pair (w', f) with respect to μ

对称性：工人最优——工厂最劣

引理

工人最优稳定匹配一定是工厂最劣稳定匹配.

证明

记 μ 是一个工人最优稳定匹配，并假设它和工厂最劣稳定匹配不一样，记工厂最劣稳定匹配为 μ'. 那么一定存在某个 $(w, f) \in \mu$，使得 $pessimal(f) \neq w$. 不妨设 $pessimal(f) = w'$，显然有 $w \succ_f w'$；我们再令 $w = pessimal(f')$，那么有 $(w, f'), (w', f) \in \mu'$. 既然 $optimal(w) = f$，并且 w 和 f' 在一个稳定匹配中匹配上了，那说明 $f \succ_w f'$. 结合 $w \succ_f w'$，我们可以得到 (w, f) 是 μ' 中的一个阻塞对，这和 μ' 是稳定匹配矛盾了.

总结一下，我们证明了延迟接受算法能够输出一个稳定匹配，并且这个稳定匹配对于申请方是最好的，同时对于被申请方是最差的. 如果我们把上面延迟接受算法改成工厂向工人申请(发 offer)，工人选择一个工厂接受，那么结果是得到工厂最优稳定匹配/工人最差稳定匹配. 接下来我们扩展到 Setting 2 进行讨论.

Setting 2 下的延迟接受算法

回忆一下两种设置的区别，Setting 2 中工人和工厂的数量不一定相等，并且图不是完全图，也就是说最终可能会有工人或者工厂没有匹配上（或者说和 $\perp$ 匹配）．在 Setting 2 下显然稳定匹配不一定是一个完美匹配，但它一定是一个最大匹配．匹配 μ 达到最大是在说，它不能再通过加入 $E-\mu$ 中的边进行扩展．
我们对阻塞对的定义也要有一些改变．

Definition 1.19. Let μ be any maximal matching in $G=(W,F,E)$. Then the pair (w,f) forms a *blocking pair* with respect to μ if $(w,f)\in E$ and one of the following holds:

- **Type 1.** w,f are both matched in μ and prefer each other to their partners in μ.
- **Type 2a.** w is matched to f', f is unmatched, and $f \succ_w f'$.
- **Type 2b.** w is unmatched, f is matched to w', and $w \succ_f w'$.

但事实上我们可以把 $\perp$ 看成一个虚拟的工厂或者工人，工人/工厂对于它的偏好是大于所有非邻居但小于所有邻居．这样就能够仍然沿用之前的简单定义．

Setting 2 下的延迟接受算法

我们还需要修改延迟接受算法的细节，唯一要修改的就是它的终止条件．这时候的终止条件为：每个工人都要么被一个工厂暂时接受，要么目标工厂列表已经为空（已经被所有的邻居工厂拒绝了）．当这个条件满足时，我们就输出匹配，其中满足前者条件的和工厂进行匹配，满足后者条件的和 $\perp$ 进行匹配．

定理

上述延迟接受算法输出一个最大匹配，并且是稳定匹配．

定理

对于所有稳定匹配，成功匹配的工人（工厂）数量不变．进一步地，对于所有稳定匹配，成功匹配的工人（工厂）集合也不变，也就是不同的稳定匹配只有它们之间的匹配关系不同．

接下来我们讨论 Setting 3，并说明它能够被规约到 Setting 2．

Setting 3

考虑它和 Setting 2 区别，我们发现多了一个容量函数．很容易想到，我们可以把一个具有 $c(i)$ 的工厂 f 变成 $c(i)$ 个一模一样的、容量为 1 的工厂 $f_1, f_2, \cdots, f_{c(i)}$．我们不需要改变这些工厂的偏好列表，因为工人并没有发生变化；需要改变的只是工人的偏好列表，我们只需要把之前 f 所在的位置替换成 $f_1, f_2, \cdots, f_{c(i)}$ 即可，这使得工人之前比 f 更喜欢的工厂仍然比这些复制工厂更喜欢，之前比 f 更不喜欢的工厂仍然比这些复制工厂更不喜欢．通过把具有大于 1 容量的工厂变为一堆容量为 1 的工厂，我们就完成了从 Setting 3 向 Setting 2 的转换．具体而言，形式化的定义如下：

Let I be given by (W, F, E, c) together with preference lists $l(w), \forall w \in W$ and $l(f), \forall f \in F$. Instance I' will be given by (W', F', E') together with preference lists $l'(w), \forall w \in W'$, and $l'(f), \forall f \in F'$, where:

- $W' = W$.
- $F' = \cup_{f \in F} \{f^{(1)}, \ldots, f^{(c(f))}\}$; i.e., corresponding to firm $f \in I$, I' will have $c(f)$ firms, namely $f^{(1)}, \ldots, f^{(c(f))}$.
- Corresponding to each edge $(w, f) \in E$, E' has edges $(w, f^{(i)})$ for each $i \in [1 \ldots c(f)]$.
- $\forall w \in W'$, $l'(w)$ is obtained by replacing each firm, say f, in $l(w)$ by the ordered list $f^{(1)}, \ldots, f^{(c(f))}$. More formally, if $f \succ_w f'$ then for all $1 \le i \le c(i)$ and $1 \le j \le c(j)$ we have $f^{(i)} \succ_w f'^{(j)}$ and for all $1 \le i < j \le c(i)$ we have $f^{(i)} \succ_w f^{(j)}$.
- $\forall f \in F$ and $i \in [1 \ldots c(f)]$, $l'(f^{(i)})$ is the same as $l(f)$.

Setting 3

定理

从 Setting 3 到 Setting 2 的转换实际上是一个双射.

所以，我们在求解 Setting 3 问题时，可以消耗多项式时间就把它转换成一个 Setting 2 的问题，并使用延迟接受等算法进行求解.
接下来我们讨论延迟接受算法的另一个重要性质：DSIC.

DSIC?

我们仍然从 Setting 1 入手.

引理：Blocking Lemma

设 μ_W 为偏好关系 $\succ$ 下的工人最优稳定匹配，μ 为任意完美匹配（不必稳定）. 令 W_0 为在 μ 下匹配结果优于 μ_W 的工人集合，即

$$W_0 = \{w \in W \mid \mu(w) \succ_w \mu_W(w)\},$$

并假设 $W_0 \neq \emptyset$. 则 $W_0 \neq W$，且存在 $w \in (W \setminus W_0)$ 和 $f \in \mu(W_0)$，使得 (w, f) 构成 μ 的阻塞对.

DSIC?

证明

显然，对于 $w \in (W \setminus W_0)$，有 $\mu_W(w) \succeq_w \mu(w)$. 分为两种情况：1. **工人通过交换匹配对象改善结果**：即 $\mu(W_0) = \mu_W(W_0)$ 成立. 2. **工人通过其他方式改善结果**：即 $\mu(W_0) \neq \mu_W(W_0)$.

我们分别研究这两种情况. 情况 1：$\mu(W_0) \neq \mu_W(W_0)$. 由于 $|\mu(W_0)| = |\mu_W(W_0)| = |W_0|$ 且 $(\mu(W_0) \setminus \mu_W(W_0)) \neq \emptyset$，因此 $W_0 \neq W$. 选取任意 $f \in (\mu(W_0) \setminus \mu_W(W_0))$，并令 $w = \mu_W(f)$. 此时 $w \in (W \setminus W_0)$，因为若 $f \notin \mu_W(W_0)$，则 $\mu_W(f) \notin W_0$.

我们将证明 (w, f) 是 μ 的阻塞对. 为此，定义以下工人与企业：令 $w' = \mu(f)$（因 $f \in \mu(W_0)$，故 $w' \in W_0$）；令 $f' = \mu_W(w')$（显然 $f' \in \mu_W(W_0)$）；令 $f' = \mu(w)$（因 $w \in (W \setminus W_0)$，故 $f' \in (F \setminus \mu(W_0))$.

需证明两点：1. $f \succ_w f'$：因 $w \in (W \setminus W_0)$ 且 $f \neq f'$，由 $\mu_W(w) \succeq_w \mu(w)$ 可知 $f = \mu_W(w) \succeq_w \mu(w) = f'$. 若 $f \sim_w f'$，则 w 对 μ 和 μ_W 无差异，矛盾于 $w \in (W \setminus W_0)$. 因此 $f \succ_w f'$.

2. $w \succ_f w'$：假设反设 $w' \succeq_f w$. 由于 $w' \in W_0$，$f = \mu(w')$ 且 $f' = \mu_W(w')$，而 $f \succ_{w'} f'$（因 w' 偏好 μ 的匹配）. 此时 (w', f) 将成为 μ_W 的阻塞对，与 μ_W 的稳定性矛盾. 因此 $w \succ_f w'$.

综上，(w, f) 是 μ 的阻塞对.

DSIC?

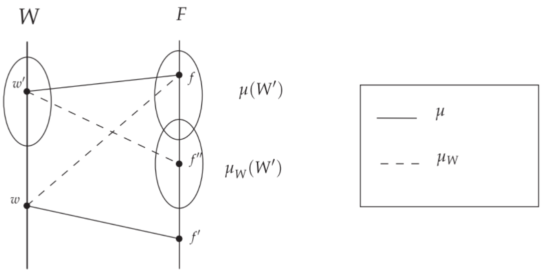


Figure 1.8 Blocking pair $\left(w, f^{(i)}\right)$.

ZHEJIANG UNIVERSITY

DSIC?

证明（续）

情况 2：$\mu(W_0) = \mu_W(W_0)$.

此时需利用 μ_W 是由工人提案的延迟接受算法（DA）生成的匹配．设 i 为 DA 算法的最后一次迭代，其中某个工人 $w' \in W_0$ 首次向其最终匹配对象 $f \in \mu_W(W_0)$ 提案．令 $w'' = \mu(f)$，因 $f \in \mu(W_0)$，故 $w'' \in W_0$.

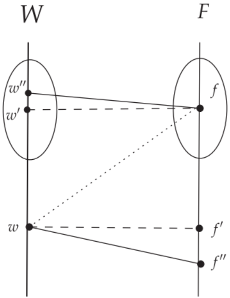
由 W_0 的定义，$f = \mu(w'') \succ_{w''} \mu_W(w'')$，因此 w'' 必在迭代 i 之前向 f 提案，并在迭代 i 或更早时转向 μ_W 的最终匹配．根据先前的观察，f 在 w'' 提案后仍需在每次迭代中接收提案．假设迭代 $i-1$ 结束时 f 暂接受工人 w 的提案．在迭代 i 中，f 将拒绝 w，而 w 将在迭代 $i+1$ 或之后提案其最终匹配对象．由于 w' 是 W_0 中最后一个提案的工人，故 $w \notin W_0$，因此 $W_0 \neq W$.

证明 (w, f) 是 μ 的阻塞对：

1. $w \succ_f w''$：在 DA 算法中，f 在暂接受 w 前已拒绝 w''，故 $w \succ_f w''$.
2. $f \succ_w f'$：令 $f = \mu_W(w)$，$f' = \mu(w)$．在 DA 中，w 曾向 f 提案后才匹配到 f，故 $f \succ_w f$．因 $w \in (W \setminus W_0)$，有 $f \succeq_w f'$，从而 $f \succ_w f'$.

结合 $w \succ_f w''$ 与 $f \succ_w f'$，可知 (w, f) 是 μ 的阻塞对（图 1.9）.

DSIC?



$$\mu(W') = \mu_W(W')$$

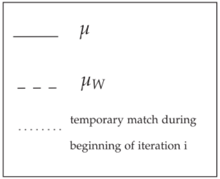


Figure 1.9

DSIC?

定理

记 $\succ$ 为原始偏好关系，$\succ'$ 为工人 w 在 $\succ$ 基础上进行谎报的偏好关系.（只有 w 相对于 $\succ$ 的偏好关系发生了变化）. 记 μ_W 为偏好关系 $\succ$ 下的工人最优稳定匹配，μ'_W 为偏好关系 $\succ'$ 下的工人最优稳定匹配. 则有 $\mu_W(w) \succeq_w \mu'_W(w)$.

证明

我们使用 Blocking Lemma. 在这里我们让 μ'_W 作为 Blocking Lemma 中的 μ. 假设定理的结论不对，即存在 w 使得 $mu'_W(w) \succ_w \mu_W(w)$，则 $w \in W'$，可以运用 Blocking Lemma，得到 μ 关于偏序 $\succ$ 存在一个阻塞对 (w', f). 我们又知道 $\succ'$ 只是 w 的偏好关系发生了变化，不影响 w' 和 f 的，因此 (w', f) 也是 μ 关于偏序 $\succ'$ 的一个阻塞对，与 μ 是在 $\succ'$ 求得的稳定匹配相矛盾了.

上述定理证明了工人提出申请的延迟接受算法对于工人来说是 DSIC 的. 我们容易将其推广到 Setting 2 中，因为只需要略微修改条件，Blocking Lemma 依然适用. 同时，由于 Setting 3 可以规约到 Setting 2，因此延迟接受算法在 Setting 3 中也是工人 DSIC 的.

DSIC?

如果我们把延迟接受算法改成工厂向工人提出申请，在容量为 1 的情况下这显然和之前是对称的，所以这个时候对于工厂是 DSIC 的．但遗憾的是，在容量大于 1 的情况下，并没有对称性，这时候也无法使用工厂向工人提出申请的延迟接受算法．而且，我们有这样的结论：

定理

对于 Setting 3，没有能够使得工厂满足 DSIC 的机制．

接下来我们不仅仅局限于以图的形式去研究单个匹配，而是研究匹配之间的关系，这就引出了**格**．

定义

Definition 1.36. Let S be a finite set, $\geq$ be a reflexive, anti-symmetric, transitive relation on S and $\pi = (S, \geq)$ be the corresponding partially ordered set. For any two elements $a, b \in S$, $u \in S$ is said to be an *upper bound* of a and b if $u \geq a$ and $u \geq b$. Further, u is said to be a *least upper bound* of a and b if $u' \geq u$ for any upper bound u' of a and b. The notion of the *(greatest) lower bound* of two elements is analogous. The partial order π is said to be a *lattice* if any two elements $a, b \in S$ have a unique least upper bound and a unique greatest lower bound. If so, these will be called the *join* and *meet* of a and b and will be denoted by $a \vee b$ and $a \wedge b$, respectively, and the partial order will typically be denoted by $\mathcal{L}$. Finally, $\mathcal{L}$ is said to be a *finite distributive lattice*, abbreviated *FDL*, if for any three elements $a, b, c \in S$, the distributive property holds, i.e.,

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \quad \text{and} \quad a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c).$$

定义

定义

记 S_μ 为一个满足 Setting 1 的给定设置的实例 (W, F) 的所有稳定匹配的集合. 定义 S_μ 上的偏序关系 $\geq$: 对于 $\mu, \mu' \in S_\mu$, $\mu \geq \mu'$ 当且仅当 $\forall w \in W$, $\mu(w) \succ_w \mu'(w)$.

定理

$L_\mu = (S_\mu, \geq)$ 是一个 FDL.

定义

想要证明 L_μ 是一个 FDL，就要去定义匹配之间的交和并．我们首先在定义匹配之间的最大值和最小值．

定义

对于两个稳定匹配 μ, μ'，对于工人 w，$\max(\mu(w), \mu'(w))$ 为 w 在 $\mu(w)$ 和 $\mu'(w)$ 中更喜欢的那个工厂；$\min(\mu(w), \mu'(w))$ 为 w 在 $\mu(w)$ 和 $\mu'(w)$ 中更不喜欢的那个工厂．对于公司 f 的最大值和最小值是同理的．

我们很容易证明当 $f = max(\mu(w), \mu'(w))$ 时，一定有 $w = min(\mu(f), \mu'(f))$．也就是说构成了一个双射，因此很自然地能够定义一个完美匹配 μ_1 为 $\mu_1(w) = max(\mu(w), \mu'(w))$．同理我们也能够定义一个完美匹配 μ_2 为 $\mu_2(w) = min(\mu(w), \mu'(w))$．

定理

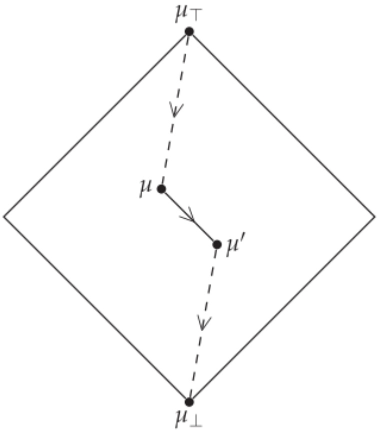
完美匹配 μ_1 和 μ_2 都是稳定匹配．

显然 μ_1 和 μ_2 分别是 μ, μ' 的最小上界和最大下界，因此可以定义交和并操作为：

$$\mu \vee \mu' = \mu_1 \text{ and } \mu \wedge \mu' = \mu_2$$

特殊匹配

FDL 表明 L_μ 是一个有限的格，我们容易发现格里面有两个特殊的节点，我们可以叫它们最顶匹配和最底匹配，分别对应着工人最优匹配和工厂最优匹配.



旋转

但其他的稳定匹配没有最顶匹配和最底匹配这么极端．我们如何去研究它们，或者说通过比较好获得的（通过 DA 算法即可）最顶匹配和最低匹配，去生成这些中间的匹配呢？我们引入**旋转**的概念．

定义

固定一个与最底匹配不相同的稳定匹配 μ，定义 $next$ 函数：$next: \ W \to F \cup \{\emptyset\}$，它表示对于一个工人 w，找到一个它最喜欢的、满足如下条件的 f：相比于 $\mu(f)$，f 更喜欢 w．如果这样的 f 存在那么 $next(w) = f$，否则 $next(w) = \emptyset$．
一个关于 μ 的旋转 ρ 是一个有序对集合 $\rho = \{(w_0, f_0), (w_1, f_1), \cdots, (w_{r-1}, f_{r-1})\}$ 使得：

1. $(w_i, f_i) \in \mu$
2. $next(w_i) = f_{i+1}$， $next(w_{r-1}) = f_0$

对 μ 应用旋转 ρ 是指让 w_i 全部和 $next(w_i)$ 匹配．即 $w_0, w_1, \cdots, w_{r-2}, w_{r-1}$ 和 $f_1, f_2, \cdots, f_{r-1}, f_0$ 匹配．我们如此得到了一个新的匹配 μ'，并记 $\mu' = \rho(\mu)$．

旋转

一个例子如下：

as $\mu' = \rho(\mu)$. For example, in Figure 1.14, the rotation $\{(1,1),(2,2),(3,4),(4,5)\}$ applied to μ yields μ'. Observe that the matched edge $(5,3)$ is not in the rotation and remains unchanged in going from μ to μ'.

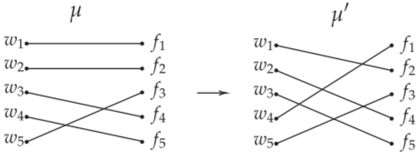


Figure 1.14

旋转

关于旋转有若干很好的性质，它能够帮助我们从一个对工人比较好的匹配，逐步变换到一个对工厂比较好的匹配.

定理

记 ρ 是 μ 的一个旋转，并且有 $\mu' = \rho(\mu)$. 那么:

1. 从 μ 到 μ'，工人们的匹配变得更差（或不变），工厂们的匹配变得更好（或不变）.
2. μ' 是一个稳定匹配.
3. $\mu > \rho(\mu) = \mu'$

定理

对于两个匹配 μ, μ' 满足 $\mu > \mu'$，那么一定存在一个 μ 的旋转 ρ 使得 $\mu > \rho(\mu) \geq \mu'$.

有了上面的结论，我们可以得到一些观察：比如说，从最顶匹配开始，永远能找到旋转使得匹配变得更接近最底匹配，直到最终旋转到最底匹配.

旋转

事实上，旋转还有更好的性质.

定理

记 ρ_1, ρ_2 是 μ 的两个旋转，$\rho_1 \neq \rho_2$. 并记 $\mu_1 = \rho_1(\mu)$，$\mu_2 = \rho_2(\mu)$. 那么:

1. ρ_1, ρ_2 没有相同的工人-工厂对.
2. ρ_1 是 μ_2 的旋转，并且同时 ρ_2 是 μ_1 的旋转.

关于这个定理我们可以这样理解：ρ_1, ρ_2 分别在 μ 上找到了一个环，显然这两个环不可能有交集，所以就没有相同的工人-工厂对. ρ_2 只是把 μ 中的一个环利用了，而另一个环完全没有变化，所以仍然能够使用 ρ_1 进行旋转.

旋转

定理

记 ρ_1, ρ_2 是 μ 的两个旋转，$\rho_1 \neq \rho_2$. 并记 $\mu_1 = \rho_1(\mu)$，$\mu_2 = \rho_2(\mu)$. 那么:

1. ρ_1, ρ_2 没有相同的工人-工厂对.
2. ρ_1 是 μ_2 的旋转，并且同时 ρ_2 是 μ_1 的旋转.

总结

1. 肾脏交换
2. 稳定匹配
   1. 三种 Setting
   2. DA 算法
3. 稳定匹配与格
4. 稳定匹配与线性规划