在线算法

什么是在线算法

- 考虑以下现实场景中
  - 输入随时间逐步到达
  - 算法需要做出一些"即时"决策。
- 例子
  - 买股票
  - 滴滴上匹配司机

举例：租还是买？

打篮球

- 坤坤没有篮球，但他经常会去打篮球
- 打篮球需要篮球，价目表如下
  - 租一个篮球：10元
  - 买一个篮球：200元
- 作为ikun，你需要帮坤坤做决定，现在是坤坤租篮球的第x天了，请问他该**继续租**，还是**买下**一个篮球呢？

IKUN讨论时间

如何评价我们的在线算法（策略）

- 上帝的策略是什么？
  - 假设我们知道坤坤会打几天篮球，我们应该怎么办？
- 我们能达到最优策略吗？
- 回顾：近似算法中的近似比
- 你会怎么**定义**算法的好坏？

竞争比

- 如果一个算法在任何情况下都满足
  - $ALG \leq \Gamma \cdot OPT$
  - OPT： Offline Optimal （上帝的最优解）
- 我们称
  - 这个算法达到了 Γ **竞争比** (Competitive Ratio)
  - 这个算法是一个 Γ −**竞争** 的算法 (Γ-Competitive)
- 你的算法能达到什么竞争比?

2竞争的算法

- 如果输入的价格
  - 租的价格是 1
  - 买的价格是 b （为了方便假设b为整数）
- 算法：
  - 如果$b \leq 1$，买。
  - 如果$b > 1$，持续租 $b-1$ 天后，如果坤坤又去打篮球，就买。
- 分析：
  - 如果坤坤最终玩的天数 $< b$ 天 ，那么最优解就是一直租，我们是最优解。
  - 如果坤坤最终玩的天数 $\geq b$ 天，那么最优解应该是一开始就买
    - 最优解：b
    - 我们花的钱：$(b-1)+b=2b-1$

我们能做的更好吗?

- 你们怎么认为?
- 你们打算怎么证明?

证明2竞争就是最好的算法

- 如果算法在 $t \le b-1$ 天以内就选择买篮球，坤坤可能会在第 t 天后**再也不打**篮球了，此时：
  - 算法花费: $t-1+b$
  - 最优解花费: $t-1$
  - 比例： $\frac{t-1+b}{t-1} \ge 2$
- 如果算法租了$b-1$ 天仍然没有买，坤坤可能打**一辈子**篮球，此时：
  - 算法花费: $\ge b+(b-1)$
  - 最优解花费: b
  - 比例： $\frac{2b-1}{b} \to 2$

我们真的没有办法了吗？

ikun从不放弃

- 能不能利用随机性?
- 如果算法没有随机性
  - 任何一个确定型的策略都有**一个对应地未来**使他比较差。
- 如果算法具有随机性
  - 任何一个未来我可能**都有机会**是好的。
- 举例:
  - 买股票：A或B，第二天会有一个涨。
  - 如果我固定买一个，最坏情况我就是不会涨。
  - 如果我随机挑一个，无论哪个涨，我都有一定的收益。

随机算法的评价方法

- 随机算法的竞争比定义
- 对任何输入情况
  - $E[ALG] \le \Gamma \cdot OPT$
- 所以我们需要保证
  - 如果坤坤最后玩了d天，
  - $\max_d \frac{E[ALG(d)]}{OPT(d)} \le \Gamma$

怎么随机？

随机想法

- 为了简单，我们以一个例子来说明
  - 租：1元
  - 买：100元
- 原确定型算法：在b天买。
- 是不是可以留一定的概率早一点买?

算法1

- 为了简单，我们缩放一下价格
  - 租：1元
  - 买：100元
- 随机算法
  - 以50%的概率在第100天买
  - 以50%的概率在第80天买
- 请大家计算现在的期望收益的近似比。

如何进一步

- 算法定义
  - 随机选择一个t, 让坤坤在第t天打篮球的时候买下篮球。
  - $\Pr[t = i] = p_i$
  - $\forall i > b,\ p_i = 0$
- 算法表现
  - 如果最终坤坤打了d天篮球
  - $OPT(d) = \min\{d, b\}$
  - 算法表现如何呢?

$$\boxed{E[ALG(d)] = p_1 \cdot b + p_2 \cdot (1+b) + p_3 \cdot (2+b) + \cdots p_d \cdot (d-1+b) + \sum_{i>d} p_i \cdot d}$$

随机算法

- 随机算法的竞争比定义
  - $\max_{d} \frac{E[ALG(d)]}{OPT(d)} \leq \Gamma$

$$\boxed{E[ALG(d)] = p_1 \cdot b + p_2 \cdot (1+b) + p_3 \cdot (2+b) + \cdots p_d \cdot (d-1+b) + \sum_{i>d} p_i \cdot d}$$

- 我们需要设置恰当的p_i使得

$$\boxed{\max_{d \geq 1} \frac{p_1 \cdot b + p_2 \cdot (1+b) + p_3 \cdot (2+b) + \cdots p_d \cdot (d-1+b) + \sum_{i>d} p_i \cdot d}{\min\{d, b\}} \leq \Gamma}$$

放缩小技巧

- 我们需要考虑 $d > b$ 的情况吗?
- $d > b$相比$d = b$
  - OPT 总是等于b，无影响。
  - 对于算法呢？由于$\forall i > b, p_i = 0$，无影响。
- 所以我们仅需要考虑去证明

$$\boxed{\max_{1\le d\le b} \frac{p_1 + p_2 \cdot (1+b) + p_3 \cdot (2+b) + \cdots p_d \cdot (d-1+b) + \sum_{d<i\le b} p_i \cdot d}{d} \le \Gamma}$$

数学时间！

转换成连续问题

- 我们把$p_1 \dots p_b$的选择看成一个函数的选择。
- $p(i) = p_i$
- 离散版本

$$\boxed{\max_{1\le d\le b} \frac{p_1 \cdot b + p_2 \cdot (1+b) + p_3 \cdot (2+b) + \cdots p_d \cdot (d-1+b) + \sum_{d<i\le b} p_i \cdot d}{d} \le \Gamma}$$

- 连续版本

$$\boxed{\max_{d\in[0,b]} \frac{\int_0^d p(t)(b+t)dt + d \cdot \int_d^b p(t)\,dt}{d} \le \Gamma}$$

平衡的思维（非严格）

$$\max_{d\in[0,b]} \frac{\int_0^d p(t)(b+t)dt + d\cdot\int_d^b p(t)\,dt}{d} \le \Gamma$$

- 我们想要找到最好的$p(t)$使得最差的d最好，我们应该尽可能让不同d之间表现相同。
- 所以，我们尝试求解：

$$\forall d \in [0,b],\quad \frac{\int_0^d p(t)(b+t)dt + d\cdot\int_d^b p(t)\,dt}{d} = \Gamma$$

求解

$$\forall d \in [0,b], \qquad \frac{\int_0^d p(t)(b+t)dt + d\cdot\int_d^b p(t)\,dt}{d} = \Gamma$$

$$\int_0^d p(t)(b+t)dt + d\cdot\int_d^b p(t)dt = \Gamma d$$

$$p(d)(b+d) + \int_d^b p(t)dt - dp(d) = \Gamma$$

$$bp'(d) - p(d) = 0$$

$$p(d) = \alpha\cdot e^{\frac{d}{b}}$$

怎么继续确定$p(d)$?

很简单！

- 我们需要的p函数是概率分布，所以
- $\int_0^b p(t)dt = 1$
- 结合 $p(x) = \alpha \cdot e^{\frac{x}{b}}$
- 得到$p(x) = \frac{1}{b(e-1)} e^{\frac{x}{b}}$

$$\boxed{\forall d \in [0, b], \qquad \frac{\int_0^d p(t)(b+t)dt + d \cdot \int_d^b p(t)\,dt}{d} = \Gamma = \frac{e}{e-1} \approx 1.58}$$

这个随机算法是最好的吗？

- 我们在课上不做介绍，但是我们可以证明：
- 没有随机算法可以达到比 $\frac{e}{e-1}$ 更好的竞争比。

在线匹配

广告投放问题

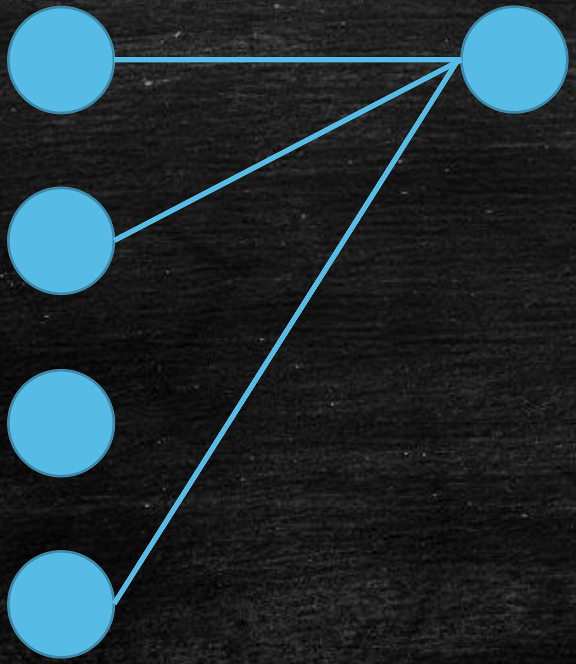
- 平台有很多条广告需要投放，但是用户的访问是在线到达的，每个用户适合的广告也不同，我们该怎么成功投放尽可能多合适的广告给用户？

在线二分图匹配问题

广告（离线顶点）

在线二分图匹配问题

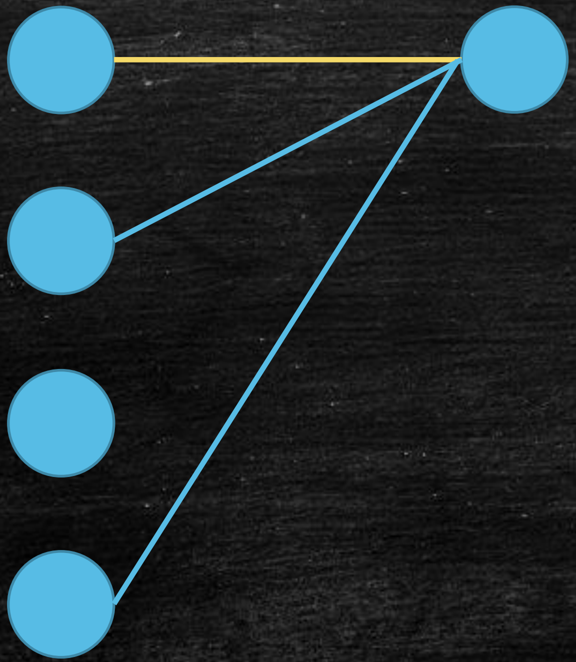
广告（离线顶点）　　用户（在线顶点）



选择匹配点

在线二分图匹配问题

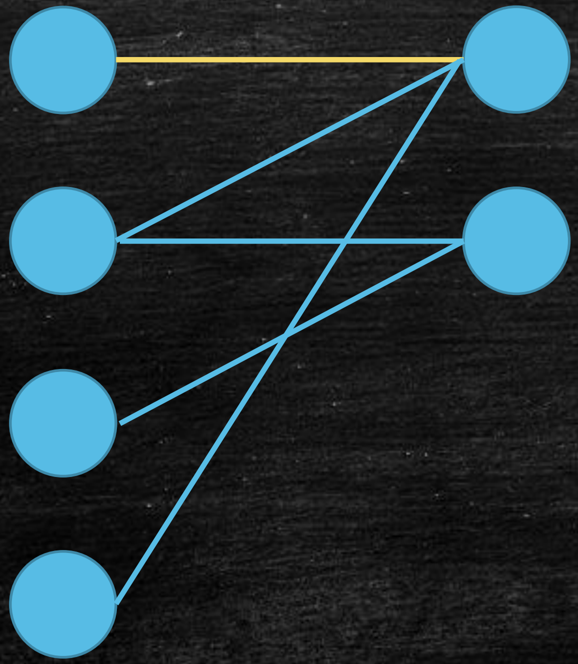
广告（离线顶点）　　用户（在线顶点）



选择匹配点

在线二分图匹配问题

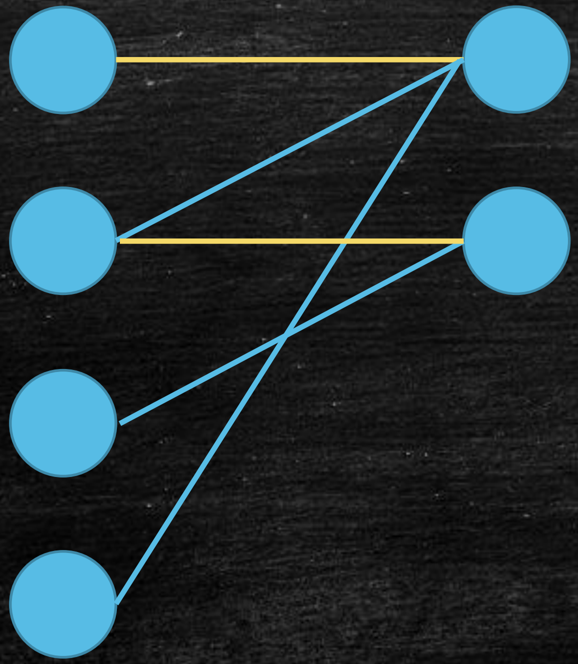
广告（离线顶点）　　用户（在线顶点）



用户到达

在线二分图匹配问题

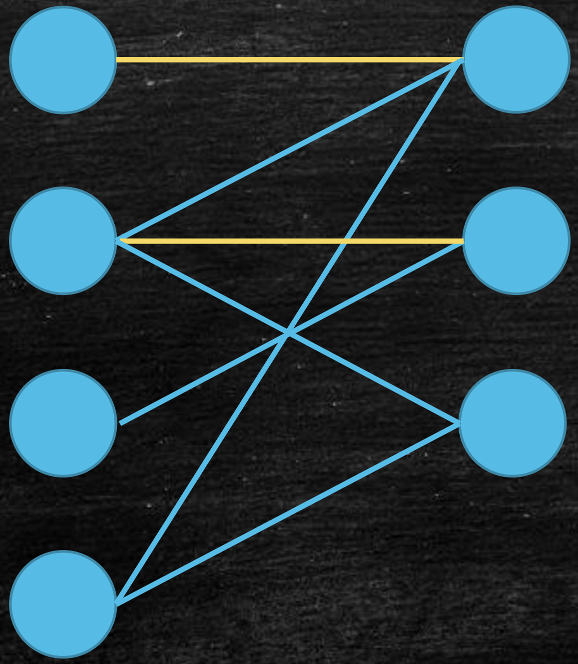
广告（离线顶点）　　用户（在线顶点）



选择匹配点

在线二分图匹配问题

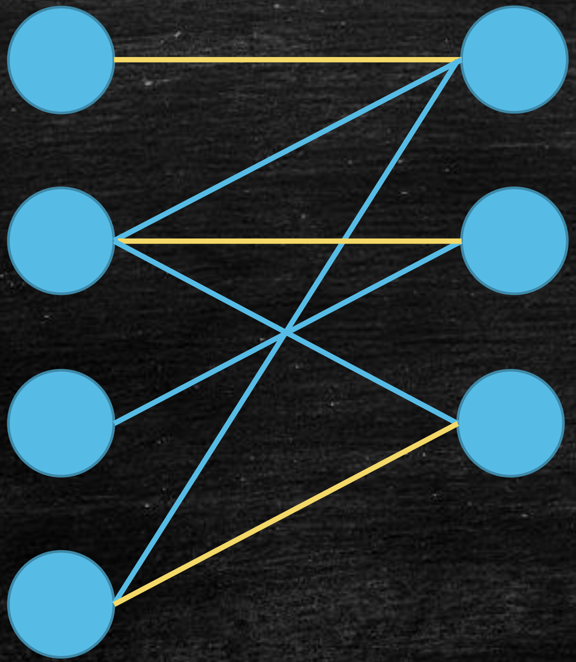
广告（离线顶点）　　用户（在线顶点）



用户到达

在线二分图匹配问题

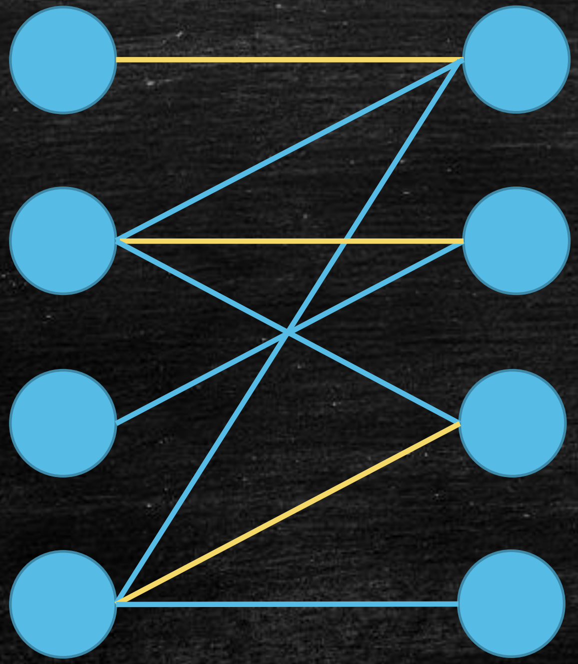
广告（离线顶点）　　用户（在线顶点）



选择匹配点

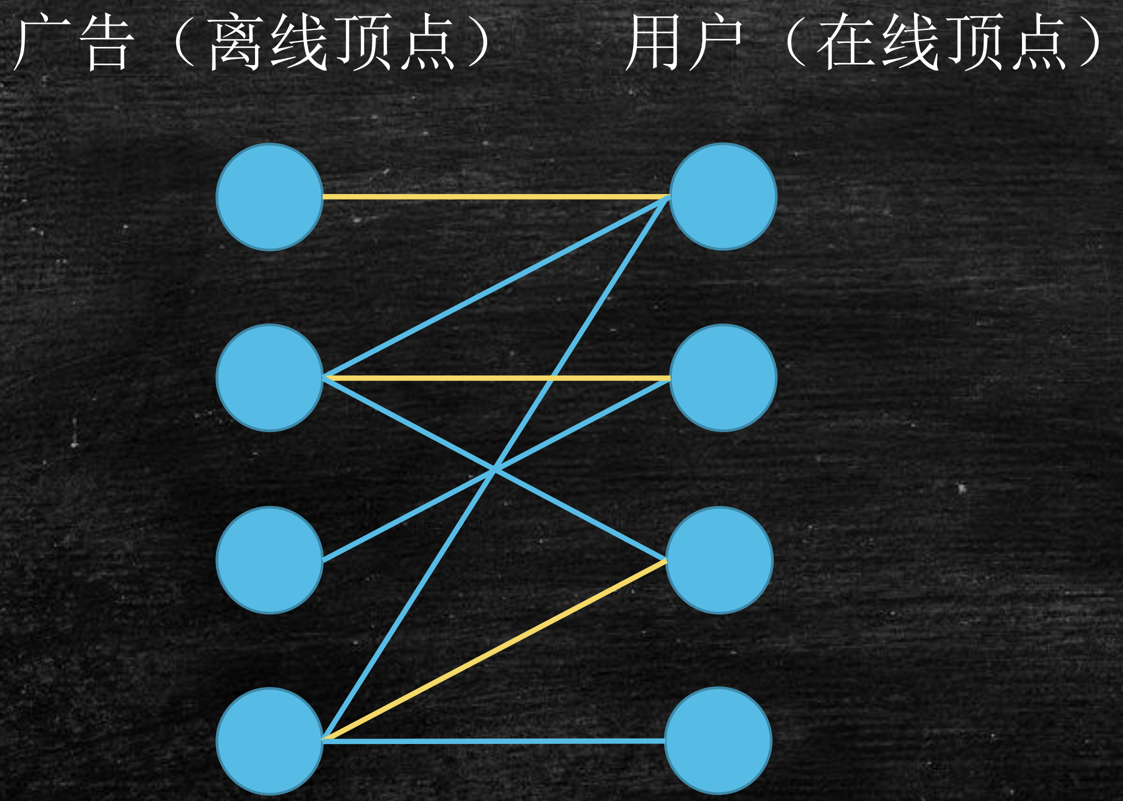
在线二分图匹配问题

广告（离线顶点）　　用户（在线顶点）



用户达到

在线二分图匹配问题



选择匹配点

最终目标

- 目标：获得尽可能多的匹配点
- 竞争比：Γ
  - 算法获得的匹配：ALG
  - 最终图中的最大匹配：OPT
  - $ALG \geq \Gamma \cdot OPT$
- 你能证明多好的竞争比?

一些讨论

- 贪心算法
  - 策略：能匹配就匹配
  - 可以达到多少竞争比?
- 如果不允许随机性，最好的竞争比是多少?

贪心算法竞争比

- 贪心算法
  - 策略：能匹配就匹配
  - 可以达到竞争比0.5
- 分析思路
  - 对于任何一条最优解选择的边，其左右两个端点至少有一个被贪心算法所选择。
  - 贪心算法匹配的点数>最优解匹配的边数> 0.5倍最优解匹配的点数

确定型算法竞争比上界

- 没有确定型算法可以突破0.5竞争比

你该选谁？

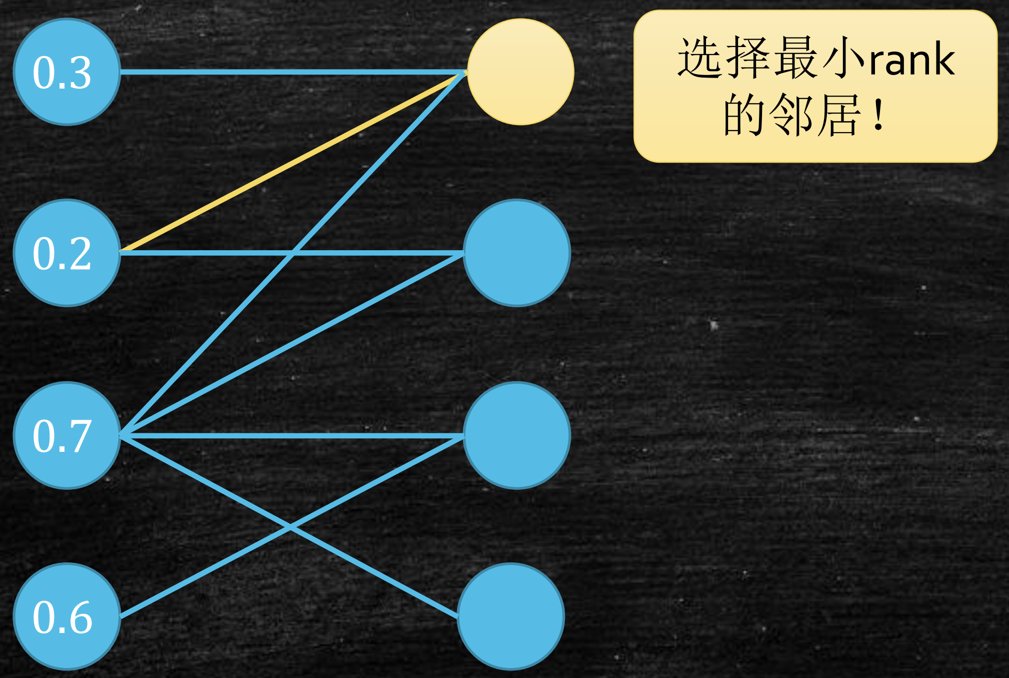
是否可以使用随机算法?

Ranking 算法

- 由 Karp, Vazirani, and Vazirani 在 1990 年提出.
- $E(ALG) \geq \left(1 - \frac{1}{e}\right) OPT.$
- 1990年时，这个算法的分析非常的复杂.
- 在2013 年 Devanur 等人把他变得简单且可拓展。

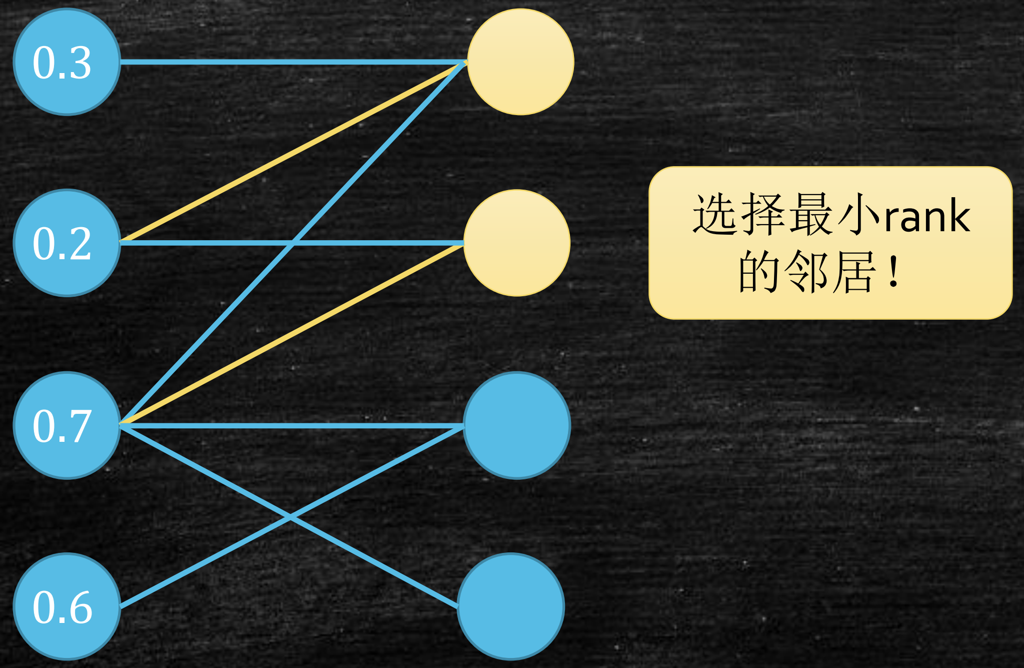
Ranking Algorithm

- Ranking: 给每个离线顶点均匀随机一个 rank ∈ [0,1).



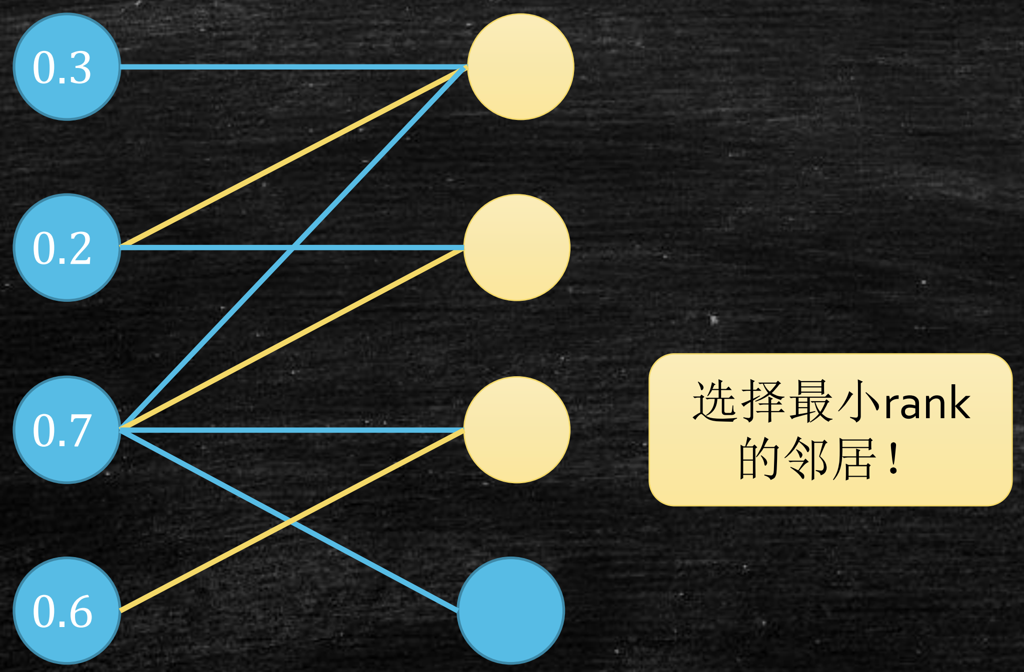
Ranking Algorithm

- Ranking: 给每个离线顶点均匀随机一个 rank ∈ [0,1).



Ranking Algorithm

- Ranking: 给每个离线顶点均匀随机一个 rank ∈ [0,1).



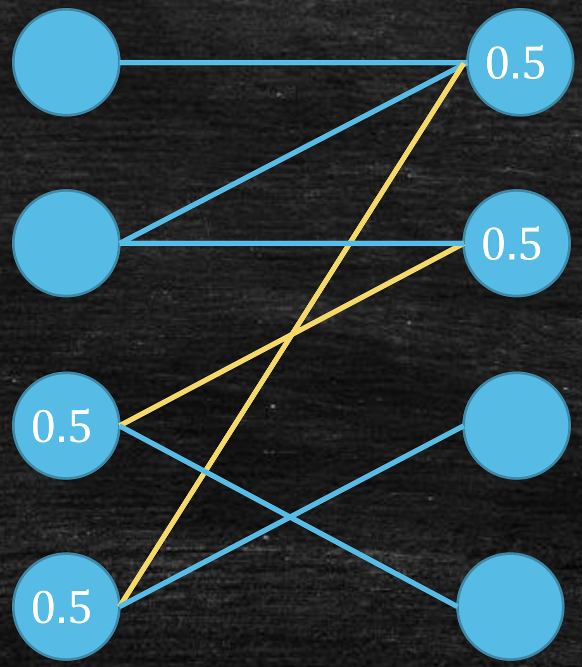
分析方法：分饼

- 在匹配时分饼
  - 每次匹配一条边
  - ALG会增加1
  - 相当于我们获得了1份饼
  - 我们把这一份饼分给这条边的两个端点。

- 如何分析竞争比？
  - 如果对于图中的任意一条边，两端点分到的饼数量都至少为Γ。
  - $\forall (u,v) \in E,\ y_u + y_v \geq \Gamma$
  - 说明算法的竞争比至少为Γ。
  - 为什么？

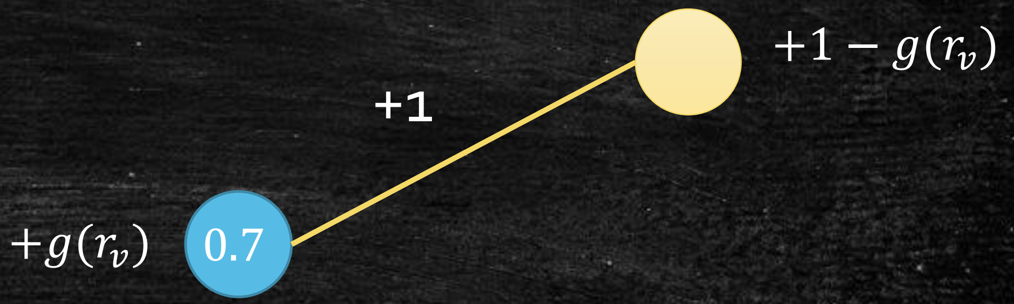
贪心算法分析再现

- 对半分饼策略
- 容易证明
- $\forall (u,v) \in E,\ y_u + y_v \geq 0.5$



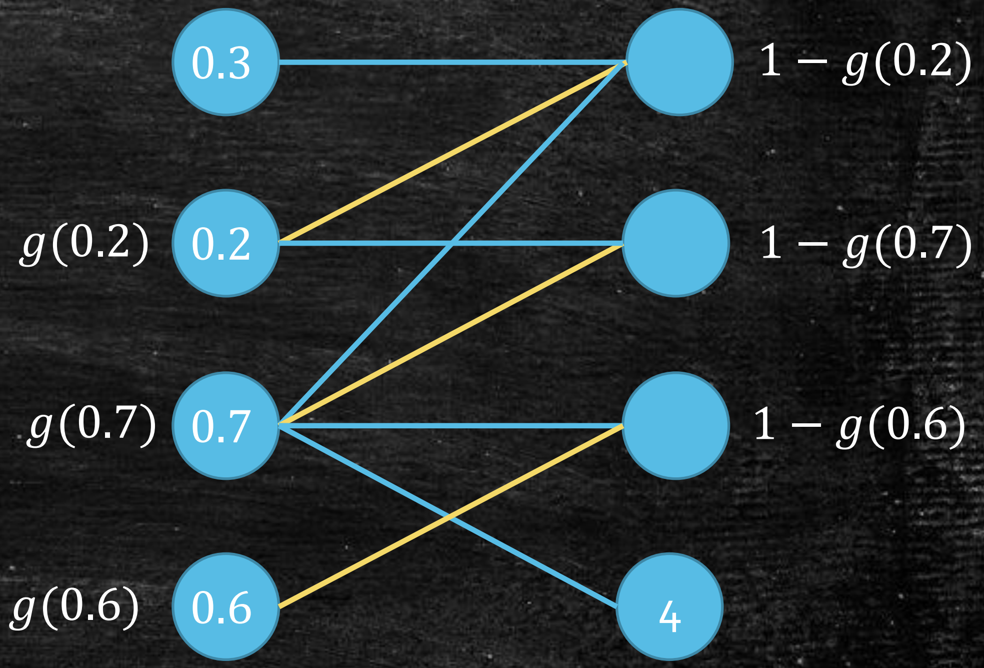
贪心算法的分饼策略

- 当 Ranking 算法匹配一条边后
  - ~~u get $0.5 \rightarrow x_u = 0.5$~~
  - ~~v get $0.5 \rightarrow x_v = 0.5$~~
  - 提前固定一个分饼函数 $g(r) = e^{r-1}$.
  - v 获得 $g(r_v) \rightarrow y_v = g(r_v)$.
  - u 获得 $1 - g(r_v) \rightarrow y_u = 1 - g(r_v)$
- 简单理解
  - $rank$越小，分的饼越少



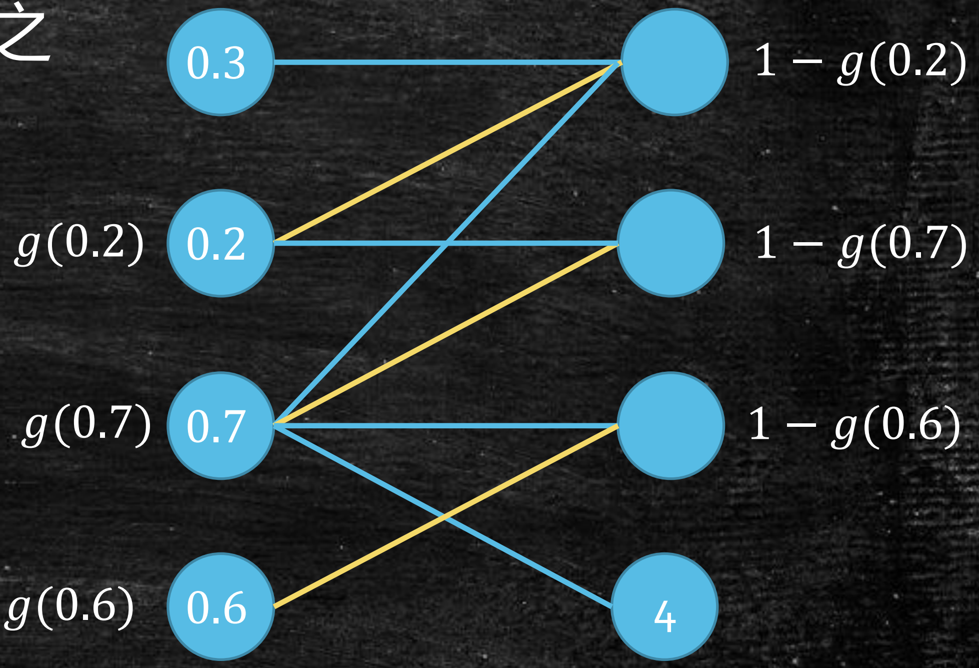
证明目标

- 我们希望证明
- $E(y_u + y_v) \geq 1 - \frac{1}{e}, \forall (u, v) \in E.$
- 这可以推出
  - $E(Ranking) \geq \left(1 - \frac{1}{e}\right) OPT$



任选一条边 (u, v) 开始证明

- 我们固定任意一种v之外的点的rank，称之为$\vec{r}(-v)$。
- 然后我们证明
- $E_{r_v}[y_u + y_v \mid \vec{r}(-v)] \geq 1 - \frac{1}{e}$
- 这可以推出
- $E[y_u + y_v] \geq E_{r_v}[y_u + y_v \mid \vec{r}(-v)] \geq 1 - \frac{1}{e}$



讨论两种情况

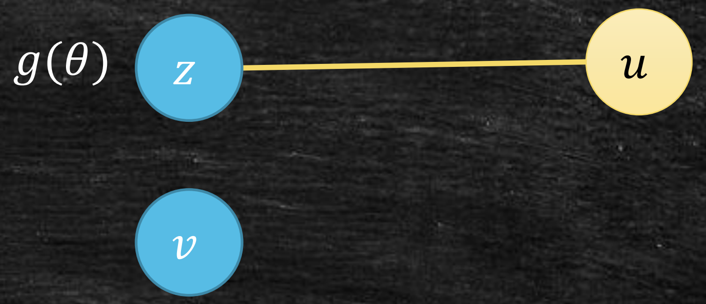
- 当我们固定某一种 $\vec{r}(-v)$之后，让我们分情况讨论
- 如果v从来不存在，u会匹配谁?
  - 固定 $\vec{r}(-v)$ 之后，这个情况是固定的。
  - 情况 1: u和某个z匹配
  - 情况 2: u不和任何人匹配

情况 1: u 没有匹配任何人

- 如果v回来了，会发生什么事?
- 发生的事情和v的rank有关吗?
- v 一定被u匹配吗?
- $E_1[y_u + y_v] \geq \int_0^1 g(r)dr$

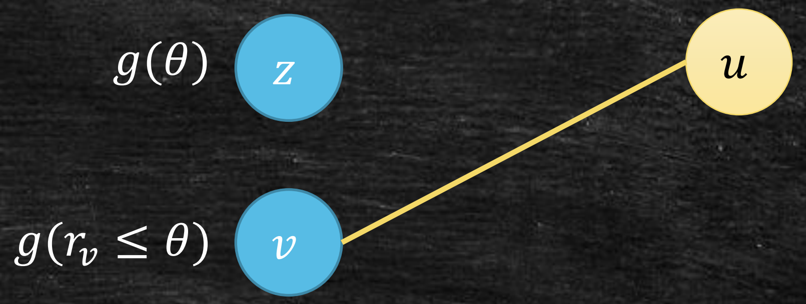
情况2: u 和某个z匹配

- 不妨定义 $r_z = \theta$.
- 如果v回来了，会放生什么事?
- 如果 $r_v < \theta$?
  - v 一定会被匹配吗?
  - v 一定会被u匹配吗?
  - v分到的饼：$y_v \geq g(r_v)$
- u会获得多少饼?
  - 情况 1: v的加入没有影响 u 匹配的点： $y_u = 1 - g(\theta)$.
  - 情况 2: v的加入影响了u 匹配的点，u的饼会发生什么变化?



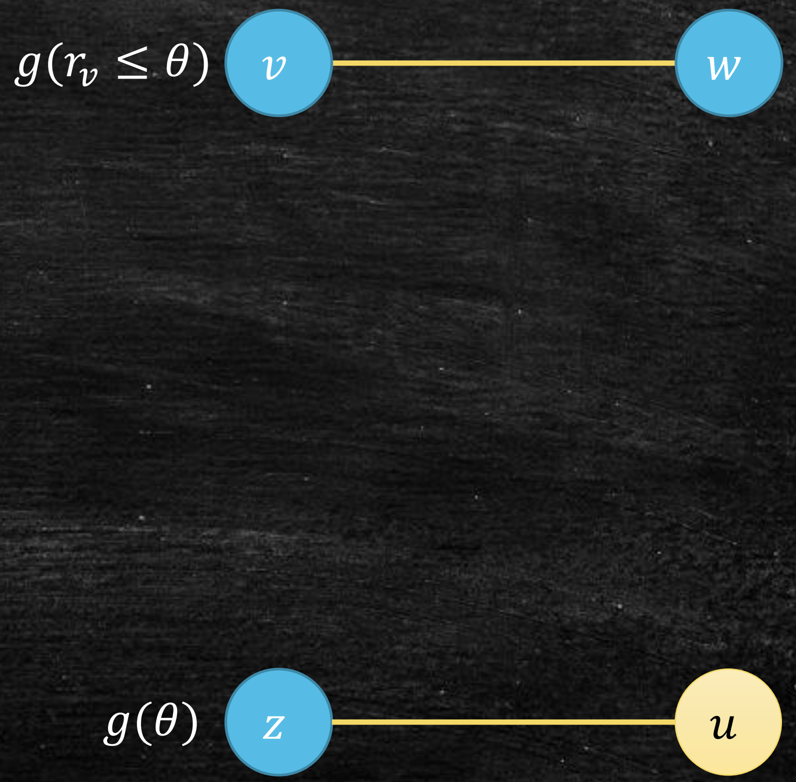
v 影响了 u

- 简单的影响
  - u直接抛弃了原选项z选择了v。
  - $y_u = 1 - g(r_v) > 1 - g(\theta)$
  - 回顾：$g(r) = e^{r-1}$



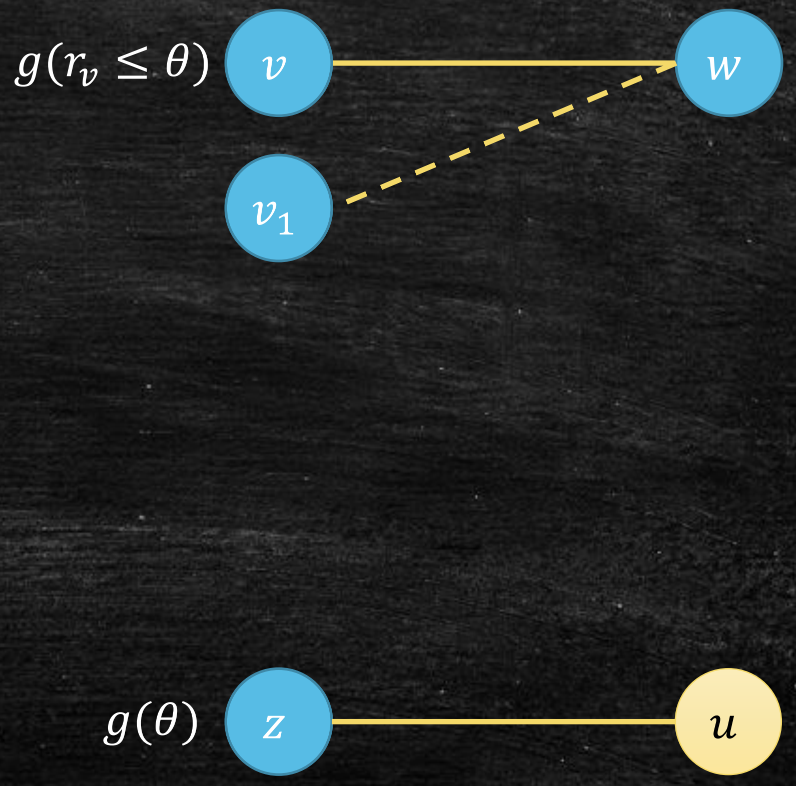
v 影响了 u

- 复杂的影响
  - w 选择了 v



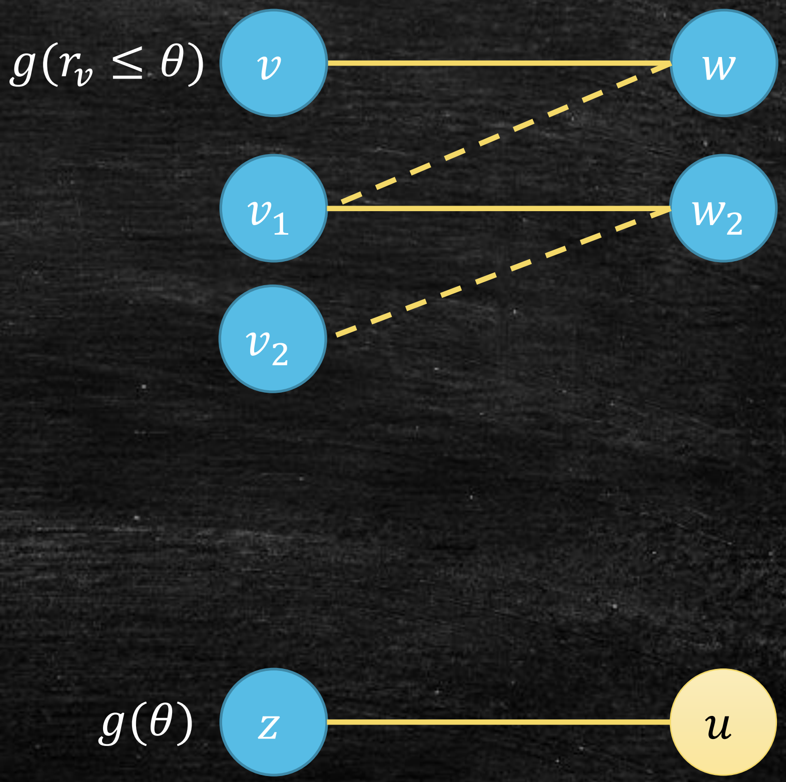
v 影响了 u

- 复杂的影响
  - w 选择了 v
  - v_1 是 w 原来的选择



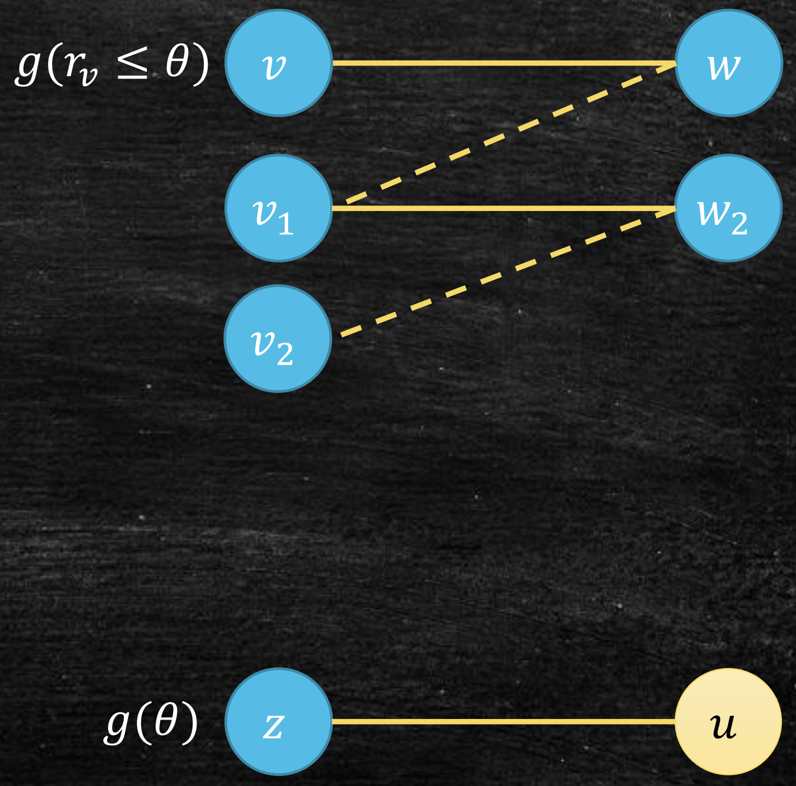
v 影响了 u

- 复杂的影响
  - w 选择了 v
  - v_1 是 w 原来的选择
  - w_2 抛弃了原来的选择，选了v_1。



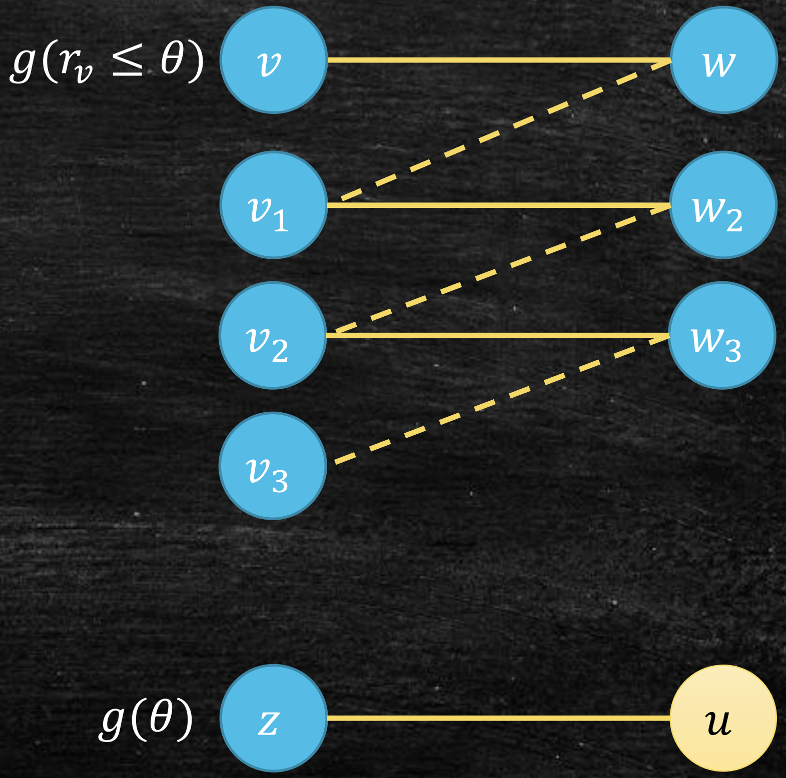
v 影响了 u

- 复杂的影响
  - w 选择了 v
  - v_1 是 w 原来的选择
  - w_2 抛弃了原来的选择，选了v_1。



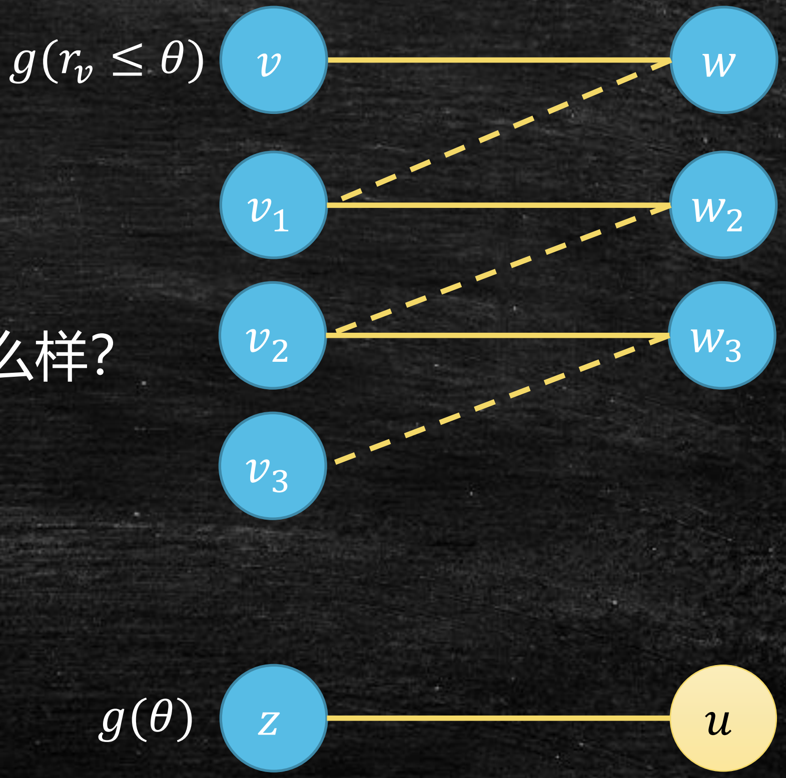
v 影响了 u

- 复杂的影响
  - w 选择了 v
  - v_1 是 w 原来的选择
  - w_2 抛弃了原来的选择，选了v_1。
  - w_3 抛弃了原来的选择，选择v_2。



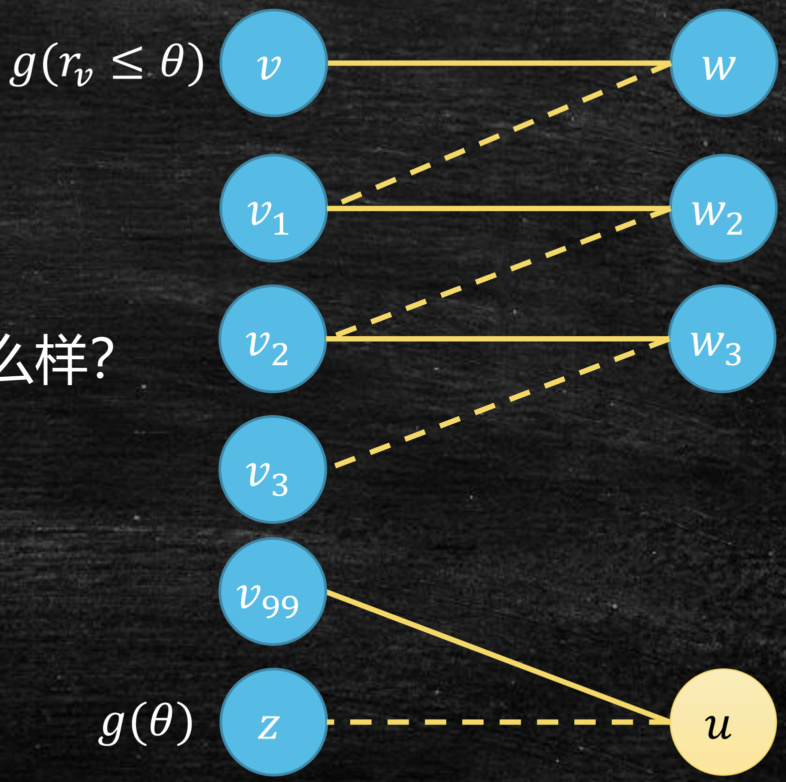
v 影响了 u

- 复杂的影响
  - w 选择了 v
  - v_1 是 w 原来的选择
  - w_2 抛弃了原来的选择，选了v_1。
  - w_3 抛弃了原来的选择，选择v_2。
  - 问题：如果这个过程最终影响了u，会怎么样？



v 影响了 u

- 复杂的影响
  - w 选择了 v
  - v_1 是 w 原来的选择
  - w_2 抛弃了原来的选择，选了v_1。
  - w_3 抛弃了原来的选择，选择v_2。
  - 问题：如果这个过程最终影响了u，会怎么样？
    - u一定是因为更想要v_{99}而不要z。
    - 所以$y_u = 1 - g\left(r_{v_{99}}\right) \geq 1 - g(\theta)$



总结

- 任意固定一条边(u, v)，对任意一种$\vec{r}(-v)$。
- 情况 1: u 没有匹配
  - $g(r) = e^{r-1}$
  - $E_{r_v}[y_u + y_v] \geq \int_0^1 g(r)dr = 1 - \frac{1}{e}$
- 情况 2: u 匹配了 $r_z = \theta$:
  - $y_v \geq g(r_v)$ if $y_v < \theta$
  - $y_u \geq 1 - g(\theta)$ for all $y_v \in [0,1)$
  - $E_{r_v}[y_u + y_v] \geq \int_0^\theta g(r)dr + 1 - g(\theta) = 1 - \frac{1}{e}$
- So, $E[y_u + y_v] \geq 1 - \frac{1}{\mathrm{e}}$