Introduction to Modern Combinatorial Optimization **POSTECH, Winter 2025**
Lecture #5: extensions of bipartite matching January 15, 2025
Lecturer: Dabeen Lee

Outline

In this lecture, we discuss some applications and extensions of bipartite matching. First, we consider the problem of matching markets based on the Vickrey–Clarke–Groves pricing mechanism. Second, we discuss stable matching that provides a common framework for matching medical students to hospitals. Lastly, we study online bipartite matching, which is a dynamic variant of bipartite matching.

1 Matching markets

Suppose that we have a nework of sellers and buyers for certain items in a market place. To simplify our discussion, let us assume that there are three sellers labeled u, v, and w and that we have a set of three buyers labeled x, y, and z. Each seller offers an item, and each buyer has certain valuations of the items as shown in Figure 5.1. The sellers, or the market, are supposed to set the prices of

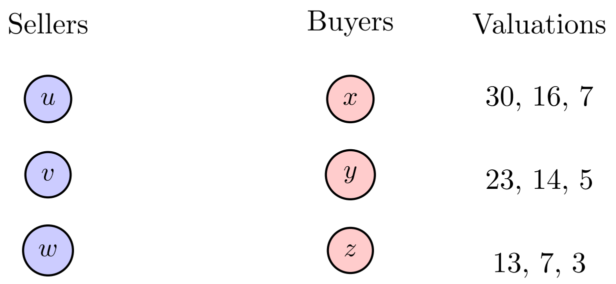

Figure 5.1: matching market example

items. For the item offered by seller $i \in \{u, v, w\}$, we use notation p_i for its price. We use notation v_{ij} to denote the valuation of buyer $j \in \{x, y, z\}$ for the item offered by seller $i \in \{u, v, w\}$. Suppose that buyer j is assigned to seller i and gets to buy its item. Then the **utility** of buyer j buying the item of seller i is given by

$$u_{ij} := v_{ij} - p_i.$$

We assume that the **rational behavior** of buyer j, which means that the buyer would decide to buy the item from seller i only if u_{ij} is nonnegative. It is natural that the assignment of buyers to sellers can be represented as a bipartite matching. Let $M \subseteq \{u, v, w\} \times \{x, y, z\}$ denote a matching or an assignment of buyers and sellers. Then the **social welfare** is defined as

the social welfare = the total profit of sellers + the total profit of buyers.

Then it follows that

$$\begin{aligned}\textbf{the social welfare} &= \sum_{ij \in M} (\textbf{the profit of buyer } i + \textbf{the profit of seller } j) \\ &= \sum_{ij \in M} (p_i + v_{ij} - p_i) \\ &= \sum_{ij \in M} v_{ij}.\end{aligned}$$

Therefore, the social welfare equals the valuation sum of items that are matched with buyers. Then the social welfare can be viewed as the weight of a matching M where each assignment between seller i and buyer j is given by the item valuation v_{ij}. In turn, this implies that the social welfare is maximized if the corresponding matching is a maximum weight matching.

We have just argued that finding a maximum weight matching leads to the maximum social welfare. However, individual buyers would behave rationally, so they will always target an item with the highest utility. It is quite likely to have conflicts between buyers. To respond to such scenarios, a market moderator would set a high price for a popular item. We call the set of prices are **market clearing** when a perfect matching is available under the prices. In this section, we will explain the **Vickrey–Clarke–Groves (VCG) mechanism** that is proven to be market clearing.

Let us explain how the VCG mechanism works. The basic idea is that whenever there is a conflict which forbids a perfect matching, we increase the price of some item. Here, a conflict can be captured by the notion of **preferred-seller graph**. For each buyer j, we draw an edge between buyer j and seller u for every $u \in \operatorname{argmax} \{u_{ij} = v_{ij} - p_i : i \in \{u, v, w\}\}$. To elaborate, let us set the prices of items to 0 initially, and the corresponding preferred-seller graph is given in Figure 5.2. By Hall's marriage theorem, the current preferred-seller graph does not have a perfect matching

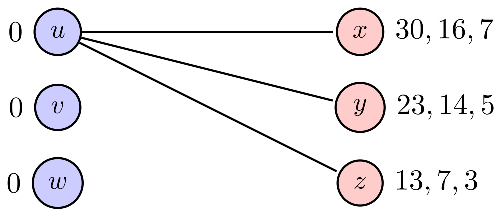

Figure 5.2: initial preferred-seller graph

because $N(\{x, y, z\}) = \{u\}$. We have just identified a conflict $S_1 = \{x, y, z\}$, which means that $|N(S_1)| < |S_1|$. Then we increase the price of the item of seller $u \in N(S_1)$ until we get a change in the preferred-seller graph. Let us set the price p_u to 6. Then we get the following new preferred-seller graph in Figure 5.3. The preferred-seller graph in Figure 5.3 also has a conflict $S_2 = \{x, y, z\}$

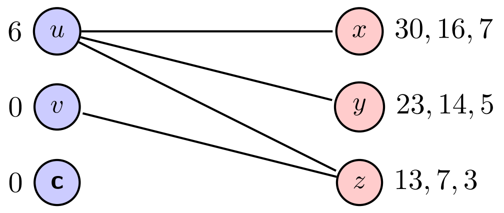

Figure 5.3: after increasing the price of the item in $N(S_1)$

as $N(S_2) = \{u, v\}$ and $|N(S_2)| < |S_2|$. Again, we increase the prices of items in $N(S_2) = \{u, v\}$ until we deduce a change in the preferred-seller graph. Let us increase p_u and p_v by 4. As a result, we deduce our new preferred-seller graph given in Figure 5.4. Note that Figure 5.4 still has a

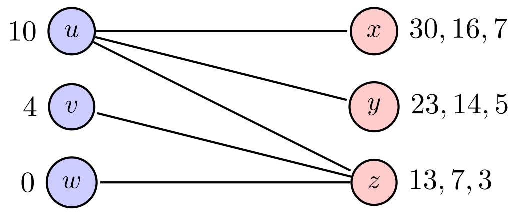

Figure 5.4: after increasing the prices of the items in $N(S_2)$

conflict, $S_3 = \{x, y\}$. As before, we increase the price p_u by 3, which gives us the new graph given in Figure 5.5. Finally, the preferred-seller graph admits a perfect matching without a conflict. The

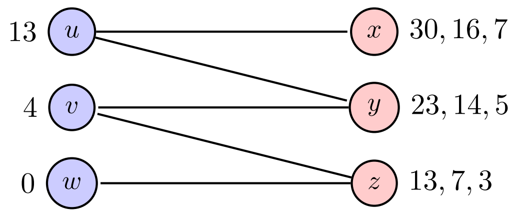

Figure 5.5: after increasingthe prices of the items in $N(S_3)$

perfect matching generates the assigned pair of (u, x), (v, y), and (w, z).

Theorem 5.1. *The Vickrey–Clarke–Groves (VCG) mechanism always finds a market clearing price that maximizes the social welfare in finite time.*

Proof. We prove finite termination first. Let I denote the set of sellers, and let J denote the set of buyers. We take the potential defined by

$$\Phi = \sum_{i \in I} p_i + \sum_{j \in J} \max\{v_{ij} - p_i : i \in I\}.$$

Initially, the price of each item is set to 0. Hence, the initial potential is given

$$\Phi_{\text{initial}} = \sum_{j \in J} \max\{v_{ij} : i \in I\}.$$

We next argue that the potential strictly decreses until we find a perfect matching. If there is no perfect matching in the current preferred-seller graph, there is a conflict set $S \subseteq J$ such that $|N(S)| < |S|$. In this case, we increase the price of items in $N(S)$ by some δ. After the update, the total profit of sellers increases by $|N(S)|\delta$ while the total profit of buyers decreases by $|S|\delta$. As $|N(S)|$ is strictly less than $|S|$, it follows that the potential strictly decreases. Therefore, the algorithm terminates in finite time.

When the preferred-seller graph contains a perfect matching, the VCG mechanism selects a perfect matching M that maximizes

$$\sum_{ij \in M} (v_{ij} - p_i) = \sum_{ij \in M} v_{ij} - \sum_{i \in I} p_i,$$

which is equivalent to maximizing the social welfare. $\square$

2 Stable matching

Let us recall the doctor-hospital assignment scenario for the US medical system. One may associate

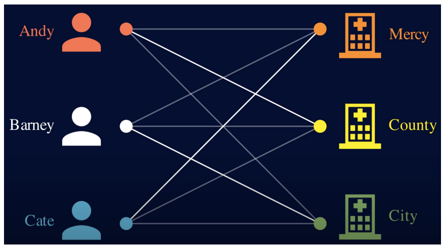

Figure 5.6: doctor-hospital assigment

it with a bipartite network between a list of medical doctors and a list of hospitals. To simplify our discussion, we assume that a hospital has at most one position available. Then we can imagine that the assignment problem can be solved by bipartite matching. In real world scenarios, however, doctors have their preferences over certain hospitals, and at the same time, it is common for hospitals to set priorities over candidates with certain specialties.

To model this situation, let us take a bipartite graph $G = (V, E)$ where the vertex set V is decomposed into D and H where D represents doctors and H is for hospitals. Individual doctors in D have a ranking of the hospitals of H based on their preferences. Similarly, individual hospitals in H have a ranking of the doctors in D based on their priorities. Essentially, we want to compute a matching between doctors and hospitals, taking into account the rankings. The goal of this section is to find a matching without an **unstable pair**, which is called a **stable matching**. What is an unstable match here? Suppose that a doctor u is matched to a hospital b and a doctor v is matched to a hospital a. Imagine a situation when doctor u prefers hospital a over hospital b and at the same time, hospital a also prefers doctor u over doctor v. Then doctor u and hospital a have an

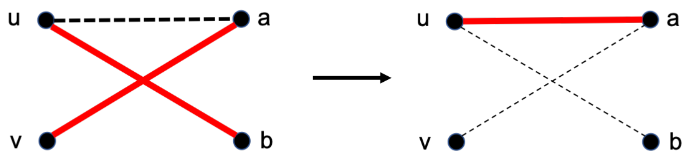

Figure 5.7: doctor-hospital assigment

incentive to break their current assignments and start a new contract between them. In this case, we call (u, a) an unstable pair.

In 1962, David Gale and Lloyd Shapley propsed an algorithm for finding a stable matching, which is now known as the Gald-Shapley algorithm or the propose-and-reject algorithm. The algorithm works as follows.

1. Each doctor applies to the hospital that is on the top of the preference ranking which has not previoulsy rejected the doctor.

2. Each hospital rejects all applicants except for the top candidate and keeps the candidate until a better one applies.

3. Repeat steps 1–3 until every doctor either has been linked to a hospital or has been rejected from all hospitals on the preference list.

Theorem 5.2. *The Gale-Shapley algorithm correctly finds a stable matching in $O(|V|^2)$ iterations.*

Proof. Let us first establish finite termination for the algorithm. Every time a doctor receives a rejection, the list of available hospital choices shrinks. Moreover, a hospital updates its candidate only with a better applicant. Note that the algorithm continues until when no hopsital rejects a doctor. As the total number of rejections that can be made is $O(|V|^2)$, the algorithm terminates in $O(|V|^2)$ iterations.

Next, we show that the algorithm is guaranteed to find a stable matching. Suppose for a contradiction that there is an unstable pair of a doctor u and a hospital x. This means that doctor u is matched to another hospital y and hospital x hires another doctor v while u prefers x over y and x prefers u over v as illustrated in Figure 5.7. However, if doctor u applied to hospital x, then it would reject doctor v and keep u instead. At the same time, doctor u would not apply to hospital y before getting rejected by hospital x. Therefore, such an unstable pair (u, x) should not exist under the Gale-Shapley algorithm. □

Next we consider the weighted case. In the remainder of this section, we explain a linear programming-based method for computing a maximum weight stable matching. Recall that the maximum weight bipartite matching problem without the stability condition can be formulated as

$$\begin{aligned} \text{maximize} \quad & \sum_{e\in E} w_e x_e \\ \text{subject to} \quad & \sum_{v\in V: uv\in E} x_{uv} \leq 1 \quad \text{for all } u \in V, \\ & x_e \geq 0 \quad \text{for all } e \in E. \end{aligned} \tag{5.1}$$

Then the question is, how do we exclude an unstable pair as illustrated in Figure 5.7? We need to write a constraint to avoid unstability between doctor u and hospital a. The idea is as follows. Given two edges $e, f \in E$, we say that f **precedes** e if they satisfy the following conditions.

- e and f share a common end point.
- If $e = uy$ and $f = ux$, then u prefers x over y.
- If $e = vx$ and $f = ux$, then x preferx u over v.

In other words, f precedes e if the connection f has a higher priority over the connection e. When f precedes e, we express it as $f \succeq e$. Then $e \succeq e$ trivially holds. Vande Vate in 1989 observed that for any $e \in E$, unstability for e can be avoided by imposing

$$\sum_{f\in E: f\succeq e} x_f \geq 1. \tag{5.2}$$

Let us consider the validity of the constraint. Suppose that $e = ux \in E$ ends being unstable. Then there exist $uy, vx \in E$ such that $uy \not\succeq ux$ and $vx \not\succeq ux$ while $x_{uy} = x_{vx} = 1$. This means that

$$\sum_{z\in H: uz\succeq ux} x_{uz} \leq 1 - x_{uy} = 0,$$

$$\sum_{w\in D: wx\succeq ux} x_{wx} \leq 1 - x_{vx} = 0.$$

This in turn implies that

$$\sum_{f\in E: f\succeq e} x_f = 0 \not\geq 1,$$

violating the constraint (5.2). Therefore, imposing (5.2) would let us avoide any unstable pair. In fact, Vande Vate in 1989 further proved that the linear program with (5.2) given by

$$\begin{aligned} \text{maximize} \quad & \sum_{e\in E} w_e x_e \\ \text{subject to} \quad & \sum_{v\in V: uv\in E} x_{uv} \leq 1 \quad \text{for all } u\in V, \\ & \sum_{f\in E: f\succeq e} x_f \geq 1 \quad \text{for all } e\in E, \\ & x_e \geq 0 \quad \text{for all } e\in E \end{aligned} \tag{5.3}$$

returns a maximum weight stable matching.

3 Online bipartite matching

So far, one of the inherent assumptions was that the entire structure of a given bipartite graph is available to the decision-maker. Hence, an algorithm receives the entire graph and computes a matching that is globally optimal. In many real world applictions, only some local structures of the graph is accessible while others are revealed gradually over time. For example, one may think of the **weapon-target assignment** problem where one side prepares a missile defense system while the other side launches fighter aircrafts. It is quite rare that all enemy jets arrive at the same time, while it is more common that they arrive in an unpredictable sequence. To defend against an enemy fighter, we would have to assign a missile to it in real time. Otherwise, it would incur a considerable damage.

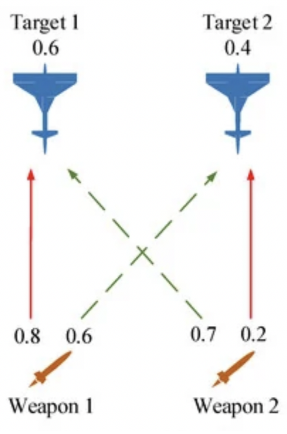

Figure 5.8: missile-fighter assignment

To model such scenarios, we consider the so-called **online bipartite matching** problem. Take a bipartite graph $G = (V, E)$ where the vertex set V is partitioned into V_1 and V_2. At the beginning, the vertex set V_1 is present. In contrast, the vertices in V_2 arrive **online**, which means that the vertices arrive one by one in a sequence while the sequence is not known. When a vertex v in V_2 arrives, we may take its neighbor u in V_1 to match with it or we may decide to just skip it.

An algorithm for online bipartite matching is evaluated by the size of the matching obtained after all vertices of V_2 arrive. Of course, as an algorithm makes decisions only with local information about the graph, the size of the final matching cannot be better than the maximum size of a matching in G. Nevertheless, our performance measure is the **competitive ratio** defined as

$$\frac{\text{The size of a matching constructed by algorithm } \mathcal{A}}{\text{The maximum size of a matching in } G}.$$

First, let us consider the simple greedy algorithm. The algorithm runs with the following simple rule:

- Every time a vertex v in V_2 arrives, match it to one of its available neighbors.

Proposition 5.3. *The simple greedy algorithm achieves a competitive ratio of* $1/2$ *for online bipartite matching.*

Proof. Note that a matching returned by the greedy algorithm is always maximal. In Lecture 1, we proved that the number of edges in any maximal matching is at least half of the maximum size of a matching in a bipartite graph. This proves that the competitive ratio is at least $1/2$. □

Although the greedy algorithm already achieves a constant approximation, we may achieve an improvement by **randomization**. A **randomized algorithm** would have some random aspects in selecting vertices for online bipartite matching. In this section, we cover the famous **ranking algorithm** due to Richard Karp, Umesh Vazirani, and Vijay Vazirani in 1990. The algorithm works as follows.

1. For each vertex $u \in V_1$, sample a weight $p_u \in [0, 1]$ uniformly at random.
2. Whenever a vertex $v \in V_2$ arrives, match v to its available neighbor that has the highest weight.

This simple algorithm achieves a better performance in expectation. To be more precise, we consider the notion of **expected competitive ration** defined as

$$\frac{\text{The } \textbf{expected} \text{ size of a matching constructed by algorithm } \mathcal{A}}{\text{The maximum size of a matching in } G}.$$

Theorem 5.4 (Karp, Vazirani, and Vazirani in 1990)**.** *The ranking algorithm achieves an expected competitive ratio of* $(1 - 1/e)$ *for online bipartite matching.*

Here, $1 - 1/e$ is roughly 0.6321. Although the ranking algorithm is simple, proving Theorem 5.4 is not as trivial.