CS 120: Intro to Algorithms and their Limitations

Problem Set 6

Due: November 1, 2023 11:59pm **Denny Cao**

Collaborators:
No. of late days used on previous psets: 2
No. of late days used after including this pset: 2

§1 Matching Algorithms

(a)

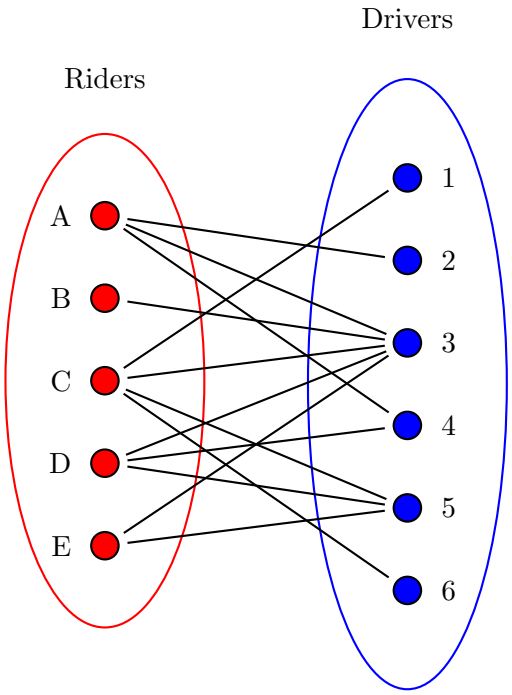

Figure 1: Bipartite graph of riders and drivers.

(b) Let purple edges denote the current matching, green denote the augmented path, and orange denote an edge that is in both the current matching and the augmented path. Let the image on the left depict the current matching and the image on the right depict the augmented path.

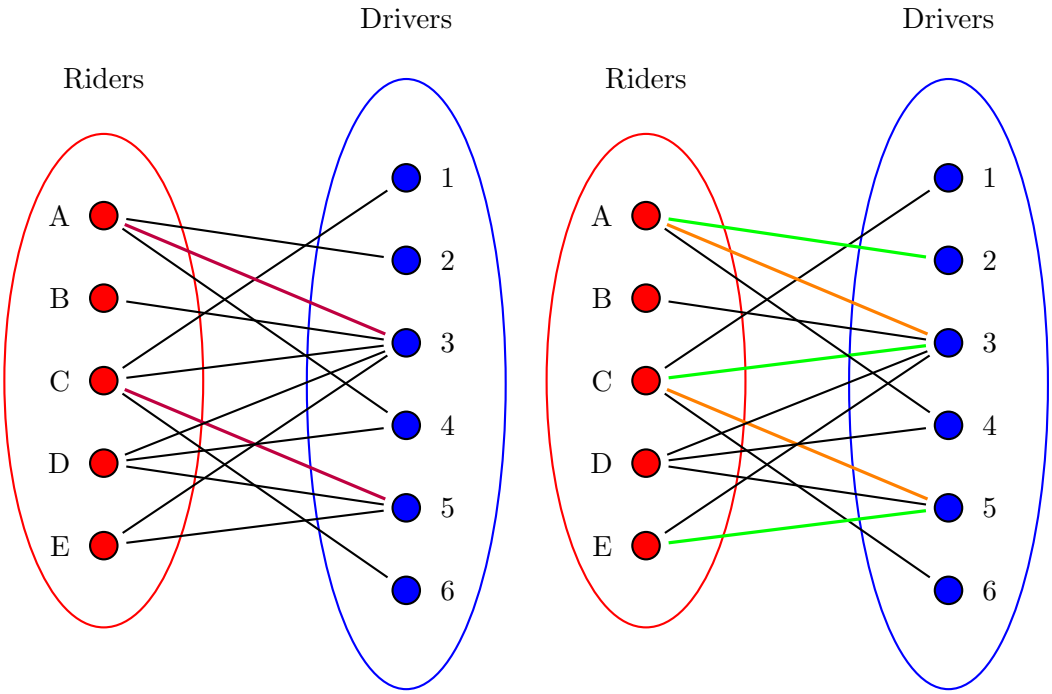

Figure 2: Step 1.

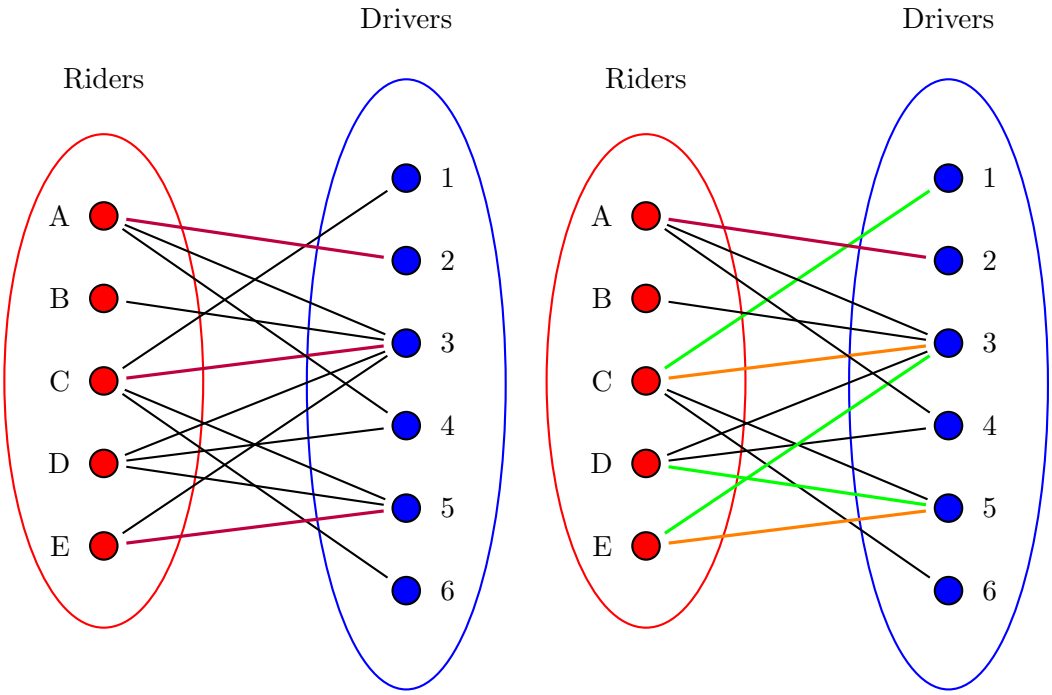

Figure 3: Step 2.

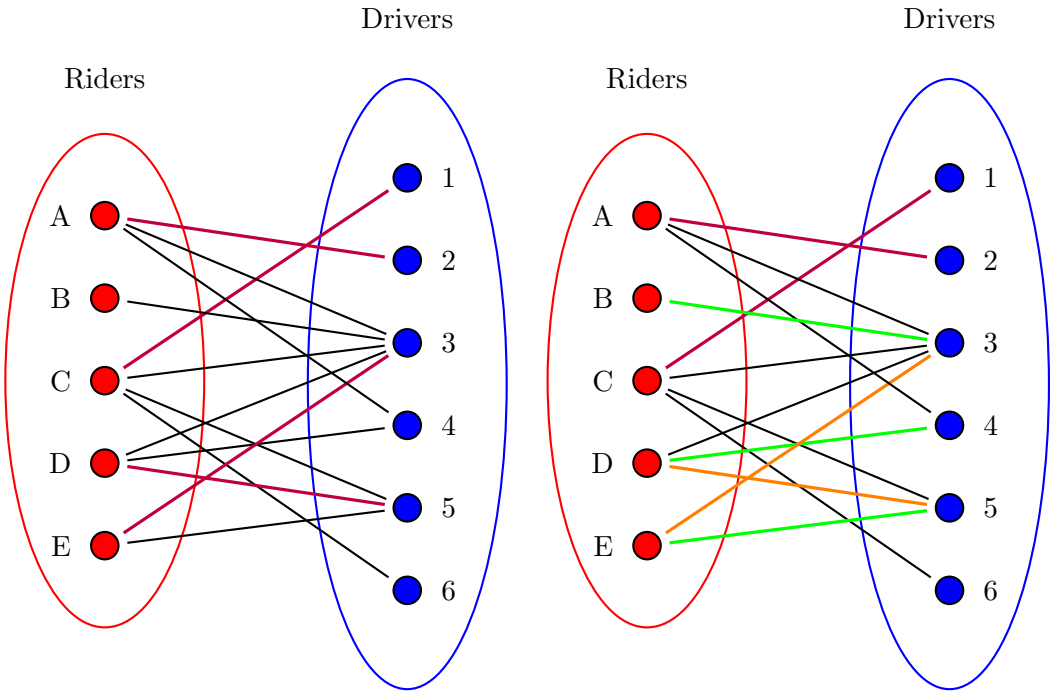

Figure 4: Step 3.

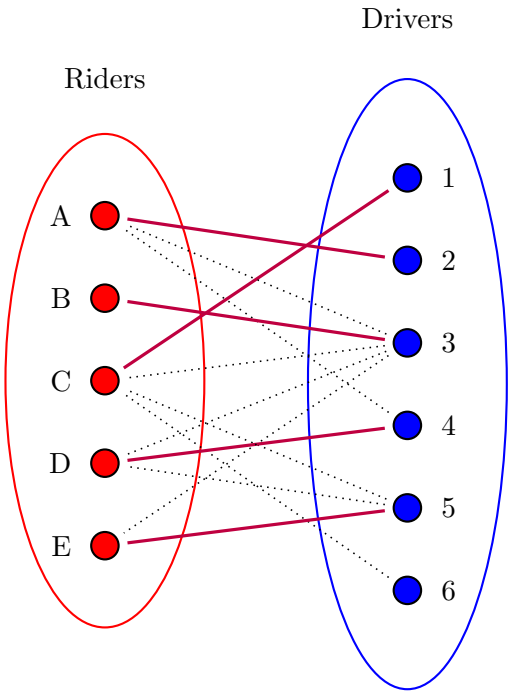

Figure 5: Maximal Matching of G.

(c) We use `GreedyIntervalScheduling` to solve this problem by converting each trip as an interval $[a, b]$, where a is the pick up time and b is the arrival time and converting the time to the sum of the hour and the minutes divided by 60 in order

for $a, b \in \mathbb{Q}$. We then sort the trips in order of increasing order of end time b. We now run the loop:

i	S
0	$\emptyset$
1	$\{[10, 10:29]\}$
2	$\{[10, 10:29]\}$
3	$\{[10, 10:29], [11:15, 11:29]\}$
4	$\{[10, 10:29], [11:15, 11:29]\}$
5	$\{[10, 10:29], [11:15, 11:29]\}$
6	$\{[10, 10:29], [11:15, 11:29], [11:30, 12:14]\}$
7	$\{[10, 10:29], [11:15, 11:29], [11:30, 12:14], [12:15, 12:29]\}$
8	$\{[10, 10:29], [11:15, 11:29], [11:30, 12:14], [12:15, 12:29]\}$
9	$\{[10, 10:29], [11:15, 11:29], [11:30, 12:14], [12:15, 12:29], [12:30, 12:59]\}$
10	$\{[10, 10:29], [11:15, 11:29], [11:30, 12:14], [12:15, 12:29], [12:30, 12:59]\}$
11	$\{[10, 10:29], [11:15, 11:29], [11:30, 12:14], [12:15, 12:29], [12:30, 12:59], [13:00, 13:29]\}$
12	$\{[10, 10:29], [11:15, 11:29], [11:30, 12:14], [12:15, 12:29], [12:30, 12:59], [13:00, 13:29]\}$
13	$\{[10, 10:29], [11:15, 11:29], [11:30, 12:14], [12:15, 12:29], [12:30, 12:59], [13:00, 13:29], [13:30, 13:59]\}$

§2 EthiCS Reflection

The kidney should go to Patient B in a Utilitarian approach in order to maximize total welfare in society—we must make the most of the scarce resources we are provided with. The overall utility gained from giving the kidney to Patient B will 4 QALYs more than if the kidney was given to Patient A. In doing so, we maximize the benefit to society as a whole by achieving the greatest amount of welfare. However, we must recognize that this situation could further perpetuate inequalities, as if all decisions were made in this way, we forgo the possibility that Patient B will gain more utility due to socioeconomic reasons affecting healthcare, as well as the fact that, potentially, Patient A was waiting substantially longer than Patient B. With the given information though, we hold all else the same for Patient A and Patient B, and thus giving the kidney to Patient B would be my decision.

§3 Vertex-Weighted Matching

(a) *Proof.* If M is the maximum-size matching, then $V(M) = V(M')$, and thus $V(M) \subseteq V(M')$. If M is not the maximum-size matching, then by Berge's Theorem, there exists an augmenting path in G with respect to M. We can thus run `MaxMatchingAugPaths(`G`)` starting with $M_0 = M$ rather than $\emptyset$. In every loop until we reach M', we augment M using P. Let this matching be M_{i+1} which represents the matching produced after the i'th loop. Let $P = (v_0, \ldots, v_l)$, and $\{v_0, v_1\} \in E \setminus M$ and $\{v_{l-1}, v_l\} \in E \setminus M$ since both v_0 and v_l are unmatched. By the proof of Lemma 5.2:

$$M_{i+1} = (M_i - \{\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{l-2}, v_{l-1}\}\}) \cup \{\{v_0, v_1\}, \{v_2, v_3\}, \ldots, \{v_{l-1}, v_l\}\}$$

We see that, although we remove the edges that connect v_j and v_{j+1}, $j \in \{1, \ldots, l-2\}$ and thus remove all v_j and v_{j+1} from the matching, we "add back" both, as we include edges that connect v_k and v_k, $k \in \{0, l-1\}$ to the matching, and thus the vertices $v_j, v_{j+1}, j \in \{1, \ldots, l-2\}$ remain in the matching, and we add an additional vertex to the matching, v_l. All other edges in the matching remain unchanged, and thus the vertices that are not in P are unchanged and remain in the matching. Thus, we have a loop invariant that that all vertices of $V(M_i)$ remain in $V(M_{i+1})$ and there is an additional vertex in $V(M_{i+1})$. It follows

that, at the end of the loop, we will reach M' where all elements of $V(M)$ are in $V(M')$. Thus, $V(M) \subseteq V(M')$ and the proof is complete. □

(b) *Proof.* If M is the vertex-weighted maximum matching, then $M = M^*$, and thus $w(M) \leq w(M^*)$ and $|M| \leq |M^*|$. If M is not the vertex-weighted maximum matching, then by Berge's Theorem, there exists an augmenting path in G with respect to M. We can thus run `MaxMatchingAugPaths(G)` starting with $M_0 = M$ rather than $\emptyset$. In every loop until we reach M', we augment M using P. Let this matching be M_{i+1} which represents the matching produced after the i'th loop. Let $P = (v_0, \ldots, v_l)$, and $\{v_0, v_1\} \in E \setminus M$ and $\{v_{l-1}, v_l\} \in E \setminus M$ since both v_0 and v_l are unmatched. By the proof of Lemma 5.2:

$$M_{i+1} = (M_i - \{\{v_1, v_2\}, \{v_3, v_4\}, \ldots, \{v_{l-2}, v_{l-1}\}\}) \cup \{\{v_0, v_1\}, \{v_2, v_3\}, \ldots, \{v_{l-1}, v_l\}\}$$

By definition of total weight:

$$\begin{aligned} w(M_{i+1}) &= w(M_i) - (w(v_1) + w(v_2) + w(v_3) + w(v_4) + \cdots + w(v_{l-2}) + w(v_{l-1})) \\ &\quad + (w(v_0) + w(v_1) + w(v_2) + w(v_3) + \cdots + w(v_{l-1}) + w(v_l)) \end{aligned}$$

This simplifies to:

$$w(M_{i+1}) = w(M_i) + w(v_0) + w(v_l)$$

Thus, $w(M_i) \leq w(M_{i+1}) \leq \cdots \leq w(M^*)$, where M^* is the matching produced when the loop terminates. It follows that, for all M, $w(M) \leq w(M^*)$. From Part 3a, we also know that $V(M) \leq V(M^*)$, and thus $|M| \leq |M^*|$, and the proof is complete. □

(c) The same is true for the maximin objective as we can, instead of choosing an arbitrary augmented path P, we can instead choose the path P that preserves the edges that have the least weight; we choose a $P = (v_0, \ldots, v_l)$ such that the edges added to M_i are of least weight and the edges removed are the greatest possible weight. This will still maximize the size, as we continuously add the minimum edge that will lead to a larger matching.

§4 Edge-Weighted Bipartite Matching

(a)

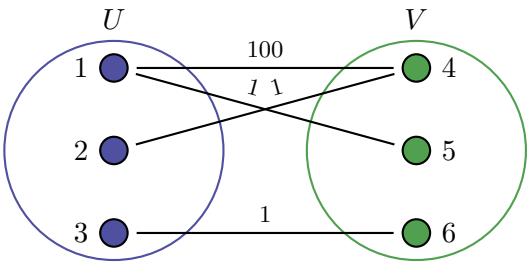

Figure 6: Edge-weighted bipartite graph where there is no matching in G that maximizes weight $w(M)$ and size $|M|$.

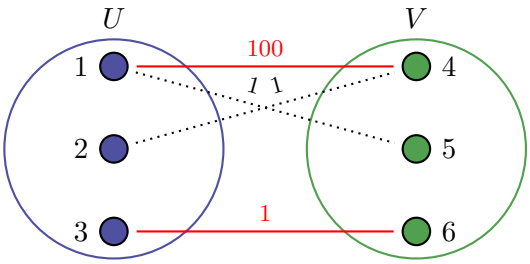

Figure 7: Matching that maximizes $w(M) = 101$, but $|M| = 2$.

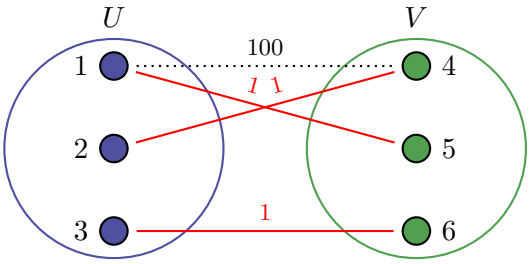

Figure 8: Matching that maximizes $|M| = 3$, but $w(M) = 3$.

There is an inherent tension between these two objectives, as in order to maximize size $|M|$, we may have to choose an edge that does not maximize the weight $w(M)$, and when maximizing $w(M)$, we may have to choose an edge that will lead to a decrease in total matchings, but increase $w(M)$.

(b) *Proof.* We can construct a weighed graph $G' = (V_0' \cup V_1', E', w)$, where $V_0' = V_0, V_1' = V_1$, and $E' = E \cup \{\{d_i, p_i\} \mid 0 \leq i \leq n-1, i \in \mathbb{Z}\}$ and w is an array specifying a nonnegative edge weight $w(e), \forall e \in E$. Let $w(\{d_i, p_i\}) = 0, \forall 0 \leq i \leq n-1, i \in \mathbb{Z}$, and for all other edges e, let $w(e) = 1$. We can then reduce to a maximum-weight perfect matching, which will prefer to pick an edge that is not in the form $\{d_i, p_i\}$, as the weight of all other edges are $1 > 0$. From the result, we can then count the amount of matchings there are and subtract by the amount of matchings from d_i to p_i. □