CSE 594: Combinatorial and Graph Algorithms
SUNY at Buffalo, Fall 2006

Lecturer: Hung Q. Ngo
Scribe: Hung Q. Ngo

Introduction to Network Flow Problems

1 Basic definitions and properties

Definition 1.1. A *flow network* is a directed graph $D = (V, E)$ with two distinguished vertices s and t called the *source* and the *sink*, respectively. Moreover, each arc $(u, v) \in E$ has a certain *capacity* $c(u, v) \geq 0$ assigned to it.

If $(u, v) \notin E$ (including pairs of the form (u, u)), we assume $c(u, v) = 0$. In this note, we shall restrict ourselves to the case where **capacities are all rational numbers**. Some of the capacities might be ∞.

Definition 1.2. Let $G = (V, E)$ be a graph or a digraph. Let X be a proper non-empty subset of V. Let $\bar{X} := V - X$, then the pair $(X, \bar{X})$ forms a partition of V, called a *cut* of G. The set of edges of G with one end point in each of X and $\bar{X}$ is called an *edge cut* of G, denoted by $[X, \bar{X}]$.

Definition 1.3. A *source/sink cut* of a network D is a cut (S, T) with $s \in S$ and $t \in T$. (Note that, implicitly $T = \bar{S}$.)

Definition 1.4. A *flow* for a network $D = (V, E)$ is a function $f : V \times V \to \mathbb{R}$, which assigns a real number to each pair (u, v) of vertices. A flow f is called a *feasible flow* if it satisfies the following conditions:

(i) $0 \leq f(u, v) \leq c(u, v), \forall (u, v) \in E$. These are the *capacity constraints*. (If a capacity is ∞, then there is no upper bound on the flow value on that edge.)

(ii) For all $v \in V - \{s, t\}$, the total flow into v is the same as the total flow out of v:

$$\sum_{u:(u,v)\in E} f(u, v) = \sum_{w:(v,w)\in E} f(v, w). \tag{1}$$

These are called the *flow conservation law*.

Definition 1.5. The *value* of a flow f for D, denoted by val(f), is the net flow out of the source:

$$\text{val}(f) := \sum_{u:(s,u)\in E} f(s, u) - \sum_{v:(v,s)\in E} f(v, s).$$

For notational conveniences, for every two subsets $X, Y \subseteq V$, define

$$f(X, Y) := \sum_{x\in X} \sum_{y\in Y} f(x, y).$$

For every proper and non-empty subset $S \subseteq V$ we define $f^+(S)$ to be the net flow leaving S and $f^-(S)$ to be the net flow entering S, namely

$$\begin{aligned} f^+(S) &:= f(S, \bar{S}), && (2) \\ f^-(S) &:= f(\bar{S}, S). && (3) \end{aligned}$$

If $S = \{w\}$ for some vertex $w \in V$, we write $f^+(w)$ and $f^-(w)$ instead of $f^+(\{w\})$ and $f^-(\{w\})$, respectively. The flow conservation law (1) now reads $f^+(v) = f^-(v), \forall v \in V - \{s,t\}$. And, the value of f is nothing but $f^+(s) - f^-(s)$.

With the conservation law held at all vertices other than the source and the sink, it is intuitively clear that the net flow into the sink is also val(f).

Proposition 1.6. *The value of a flow f is equal to the net flow into the sink:*

$$\text{val}(f) = f^-(t) - f^+(t).$$

Proof. The proof is intuitively trivial:

$$0 = \sum_{(u,v)\in E} (f(u,v) - f(u,v)) = \sum_{v\in V} \left(f^+(v) - f^-(v)\right) = (f^+(s) - f^-(s)) + (f^+(t) - f^-(t)).$$

□

Definition 1.7 (The Maximum Flow problem). The *maximum flow problem* is the problem of finding a feasible flow with maximum value, given a network D (and the capacities on the edges).

Exercise 1. Formulate the maximum flow problem as a linear program.

Definition 1.8. Given a source/sink cut (S,T), the *capacity* of the cut, denoted by cap(S,T) is the total capacity of edges leaving S:

$$\text{cap}(S,T) := \sum_{\substack{u\in S, v\in T,\\ (u,v)\in E}} c(u,v).$$

A cut with minimum capacity is called a *minimum cut*.

The following exercise generalizes the previous proposition. It should be intuitively obvious.

Exercise 2. Given a source/sink cut (S,T) and a feasible flow f for a network D, show that

$$\text{val}(f) = f^+(S) - f^-(S) = f^-(T) - f^+(T).$$

Theorem 1.9 (Weak duality). *For every source/sink cut $[S,T]$ and any feasible flow f for a network $D = (V,E)$, we have*

$$\text{val}(f) \le \text{cap}(S,T).$$

Proof. You should be able to guess the reason we call this theorem the *weak duality* property. The proof is more than trivial, given the result of Exercise 2

$$\text{val}(f) = f^+(S) - f^-(S) \le f^+(S) \le \text{cap}(S,T).$$

□

Due to the weak duality property, a feasible flow f with value equal to the capacity of some cut $[S,T]$ is a *maximum flow*. The cut is then a *minimum cut*. (Why?)

In the next sections, we develop the Max-Flow Min-Cut theorem, which basically says that the maximum flow value is always the same as the minimum cut capacity, which could both be infinite. (This is the analog of the strong duality property for linear programming.)

2 The augmenting path method

Since the 0-flow is always feasible, one might attempt to gradually increase the flow until the flow gets its maximum value. By definition, in order to increase the flow we have to either increase $f(s,v)$ of an out-edge (s,v) from s or decrease $f(v,s)$ of an in-edge (v,s) to s, as long as the capacity constraint for that edge is still valid. Doing so, however, requires adjusting the flows at edges incident to v so that the flow conservation constraint at v is still valid. In fact, this kind of update shall propagate down to t, which is the place where the conservation constraint does not have to hold.

The propagation, as described above, can be done via a "path" from s to t. We make this notion mathematically precise by introducing the notion of a residual network as follows.

Definition 2.1 (Residual capacity). Let $D = (V,E)$ be a network and f be a feasible flow for D. For each pair (u,v), the residual capacity $c_f(u,v)$ is defined to be

$$c_f(u,v) := c(u,v) - f(u,v) + f(v,u).$$

Consider an edge $(u,v) \in E$. How much more flow can we push from u to v? Clearly an amount of $c(u,v) - f(u,v)$ can be added. Moreover, if we reduce $f(v,u)$ to 0, then an amount $f(v,u)$ is also added. Even when $(u,v) \notin E$, the above analysis is still valid, since $c(u,v) = f(u,v) = 0$. Thus, the residual capacity $c_f(u,v)$ represents the additional flow which can be pushed from u to v.

Definition 2.2 (Residual network). Let $D = (V,E)$ be a network and f be a feasible flow for D. Let $D_f = (V, E_f)$ be the directed graph whose edges are the pairs (u,v) with $c_f(u,v) > 0$. This graph D_f is called the *residual network* of D with respect to f.

Definition 2.3 (Augmenting path). Let $D = (V,E)$ be a network and f be a feasible flow for D. A path P from s to t in D_f is called an *augmenting path* for D with respect to f. Let $\delta(P) := \min\{c_f(u,v) \mid (u,v) \in P\}$. Note that $\delta(P) > 0$, by definition.

Lemma 2.4. *Let P be an augmenting path for D with respect to a flow f. Let f' be a flow which is the same as f except in the cases as follows. For each $(u,v) \in P$, let*

$$f'(u,v) = f(u,v) + \min\{\delta(P), c(u,v) - f(u,v)\},$$

and

$$f'(v,u) = \begin{cases} f(v,u) - (\delta(P) - (c(u,v) - f(u,v))) & \textit{if } \delta(P) > (c(u,v) - f(u,v)) \\ f(v,u) & \textit{otherwise}. \end{cases}$$

Then, f' is feasible and $\text{val}(f') = \text{val}(f) + \delta(P)$.

Proof. Trivial. □

Theorem 2.5 (Max-Flow Min-Cut). *Let f be a feasible flow of a network D, then the following statements are equivalent:*

(i) f is a maximum flow

(ii) there is no augmenting path on D with respect to f

(iii) there some source/sink cut $[S,T]$ with $\text{val}(f) = \text{cap}(S,T)$

Proof. If f is maximum, then there cannot be any augmenting path; this is a consequence of the previous lemma. Thus, $(i) \implies (ii)$.

Suppose there is no augmenting path for f. In the residual network D_f, let S be the set of all vertices reachable from s by some path P. Note that if $v \in S$ via an s, v-path P, then all vertices of P are in S. As there is no augmenting path, t is not in S.

Consider a forward edge $(u, v) \in E$ with $u \in S, v \in \bar{S}$. Clearly $f(u, v) = \text{cap}(u, v)$, otherwise v would have been in S. Similarly, for a "backward" edge $(v, u) \in E$, with $u \in S, v \notin S$, it must be the case that $f(u, v) = 0$. Thus, $\text{val}(f) = f^+(S) - f^-(S) = \text{cap}(S, \bar{S})$. Let $T = \bar{S}$ and we have just shown that $(ii) \implies (iii)$.

Lastly, if there was a cut whose capacity is the same as the value of f, then f is maximal by the weak duality property. □

The previous theorem suggests a few strategies to find a maximum flow.

One could repeatedly try to find an augmenting path P for some feasible flow f, staring from the 0-flow. If P is found, we can augment f by an amount of $\delta(P)$ and repeats the search. If P is not found, then f is maximum. This general strategy is called the *augmenting path method*. Depending on how one find augmenting paths, we have algorithms of different time complexities; some may not even terminate.

The second idea which comes from the proof is to find augmenting paths by starting from $S = \{s\}$ and keep adding to S the set of vertices reachable from s by some path P with $\delta(P) > 0$. If we reach t, then an augmenting path is found. If we do not reach t, then we found a cut whose capacity is the same as f's value. This is the content of the Ford-Fulkerson algorithm:

Ford-Fulkerson

```
1: $f(u, v) \leftarrow 0, \forall (u, v) \in V \times V$
2: while there is an $s, t$-path in $D_f$ do
3:     update $f \leftarrow f'$, where $f'$ is defined in Lemma 2.4
4:     update $D_f$
5: end while
6: return val($f$)
```

If there is an augmenting path P with $\delta(P) = \infty$, then the copy of P in D has all edges with infinite capacity. This is the case where there is no maximum flow. Thus, for useful conversations we assume that there is no s, t-path with infinite capacity.

Recall the assumption that non-infinite capacities are rationals. In this case each update $\delta(P)$ of the algorithm increases the flow value by at least one over the largest common denominator.

If all capacities are integers, then each iteration increases the flow by at least 1. Let $\text{val}(f^*)$ be the value of a maximum flow, then the algorithm takes time at most $O(|E|\text{val}(f^*))$. In fact, it is easy to see that all $f(u, v)$ are always integers in this case. Thus, we have the following theorem.

Theorem 2.6 (Integrality theorem). *If the finite capacities are all integers, and the maximum flow is bounded, then there is a maximum flow f in which $f(u, v)$ and* $\text{val}(f)$ *are all integers.*

Exercise 3. Show that a maximum flow in $D = (V, E)$ can always be found by a sequence of at most $|E|$ augmenting paths.

3 The Edmonds-Karp algorithm

Recall the augmenting path method from the last lecture. The method may not terminate if capacities are not all rational numbers. Even when the capacities are integers, the method may take a long time to terminate: $O(|E|\text{val}(f^*))$. This kind of running time is said to be *pseudo-polynomial time*. The problem is that we did not specify a sensible method to pick an augmenting path in D_f to augment up on.

The Edmonds-Karp algorithm simply says that we should pick an s, t-path in D_f with shortest length. It turns out that this method gives a polynomial time algorithm which does not depend on val(f^*) or any numeric input. We thus have a *strongly polynomial time* algorithm.

Lemma 3.1. *Let $d_f(u, v)$ denote the distance from u to v in D_f. Then, under the Edmonds-Karp algorithm, $d_f(s, u)$ is weakly increasing as f is augmented, for all $u \in V$.*

Proof. We show that $d_f(s, u) \le d_{f'}(s, u)$ whenever f' is obtained from f by an augmentation. Suppose the shortest path in D_f chosen to be augmented was $s = u_0, u_1, \dots, u_k = t$.

We first show that $d_f(s, u) \le d_{f'}(s, u), \forall u \in \{u_0, \dots, u_k\}$. Note that after the augmentation, some of the (u_i, u_{i+1}) edges are removed due to saturation, and a few (or all) of the edges (u_{i+1}, u_i) were introduced.

This assertion is trivial when $u = s$. Consider a shortest path P' from s to $u \ne s$ in $D_{f'}$. Let $u_{i_1}, \dots, u_{i_j} = u$ be the vertices of $\{u_1, \dots, u_k\}$ which appear in P' in that order. Thus, P' consists of several segments s to u_{i_1}, ..., $u_{i_{j-1}}$ to u_{i_j}. If we can show that the part of P' from the beginning to u_{i_x} has length at least i_x, then we are done.

Clearly, the part of P' from s to u_{i_1} is at least i_1, because this part consists only of edges of $E_f \cap E_{f'}$. Suppose our assertion holds up to u_{i_x}, for some $x \in \{1, \dots, j-1\}$. Consider the part of P' from u_{i_x} to $u_{i_{x+1}}$. If $i_{x+1} < i_x$, then we are done, since the part of P' from s to u_{i_x} already has length at least i_x. If $i_{x+1} > i_x$, then the part of P' from u_{i_x} to $u_{i_{x+1}}$, which consists solely of edges in $E_f \cap E_{f'}$, has length at least $d_f(u_{i_x}, u_{i_{x+1}}) = i_{x+1} - i_x$. This completes the proof of our assertion.

Secondly, consider $u \in V - \{u_0, \dots, u_k\}$. Let P' be a shortest path in $D_{f'}$ from s to u. Suppose u_i is the last vertex on P' belonging to $\{u_0, \dots, u_k\}$. Then, concatenating the path $u_0, \dots, u_i$ and the part of P' from u_i to u would form an s, u-path in D_f of length at most the length of P'. □

Theorem 3.2. *The Edmonds-Karp algorithm terminates after at most $|E|(|V| - 2)/2$ augmentations.*

Proof. We use the previous lemma to determine at most how many augmentations are possible. An edge $(u, v) \in E_f$ is *critical* if after augmentation on f, turning f into f', (u, v) is no longer in $E_{f'}$. Each augmentation has at least one critical edge. We want to know, for each $(u, v) \in V \times V$, how many times (u, v) can be critical. Let f be the flow right before the augmentation with a critical edge (u, v), then $d_f(s, u) + 1 = d_f(s, v)$. In order for (u, v) to become critical the second time, there must be some later flow f' at which we augment along (v, u). Thus

$$d_{f'}(s, u) = d_{f'}(s, v) + 1 \ge d_f(s, v) + 1 = d_f(s, u) + 2.$$

Hence, right before the second time (u, v) becomes critical, $d_f(s, u)$ is increased by at least 2 compared to the last time. Thus, an edge (u, v) is critical at most $(|V| - 2)/2$ times, implying the number of iterations of Edmonds-Karp algorithm is $O(|E|(|V| - 2)/2)$. □

Corollary 3.3. *The running time of Edmonds-Karp algorithm is $O(|V||E|^2)$.*

Proof. This follows since finding a shortest s, t-path, using breadth first search, takes $O(|E|)$. □

Exercise 4. Show that a maximum flow f^* for a flow network $D = (V, E)$ can always be found by a sequence of $|E|$ augmenting paths.

Exercise 5. Formulate the maximum flow problem as a linear programming problem.

Exercise 6. Let $f : V \times V \to \mathbb{R}^+$ be a flow for $D = (V, E)$. For each real number α, define a flow $\alpha f : V \times V \to \mathbb{R}^+$ by

$$(\alpha f)(u, v) = \alpha \cdot f(u, v).$$

We also define the flow sum $f_1 + f_2$ of any two flows f_1 and f_2 in the obvious way.

Show that the set of *feasible* flows for D is convex, namely if f_1 and f_2 are feasible, then $\alpha f_1 + (1-\alpha)f_2$ is also feasible, for any $\alpha \in [0,1]$.

Exercise 7. Suppose that a flow network $D = (V, E)$ is symmetric, i.e. $(u,v) \in E$ iff $(v,u) \in E$. Show that the Edmonds-Karp algorithm terminates after at most $|V||E|/4$ augmentations.

4 The push-relabel method

A *preflow* f is just like a feasible flow, except that the flow conservation constraint at each $v \in V - \{s,t\}$ is replaced by

$$e(v) := f^-(v) - f^+(v) \geq 0.$$

The quantity $e(v)$ is the *excess flow* into v. A node $v \in V - \{s,t\}$ is *overflowing* if $e(v) > 0$.

Given a flow network $D = (V, E)$ and a preflow f for D. A function $h : V \to \mathbb{N}$ is called a *height function* if $h(s) = |V|$, $h(t) = 0$, and

$$h(u) \leq h(v) + 1$$

whenever $(u,v) \in E_f$. In other words, if some more flow can be pushed from u to v, then v cannot be more than 1 level lower than u.

In the push-relabel method, two basic operations are performed:

- **PUSH**(u,v) pushes some positive amount of excess flow from u to v, not to exceed $c_f(u,v)$. More precisely, this procedure pushes exactly $\min\{e(u), c_f(u,v)\}$ units of flows from u to v.

  We shall maintain an invariant that PUSHing can only be done from an overflowing vertex to a vertex at exactly one level lower ($h(u) = h(v) + 1$).

  A PUSH is *saturating* if the pushed flow amount is $c_f(u,v)$, otherwise it is *nonsaturating*. After a saturating push from u to v, clearly (u,v) is no longer an edge of the residual network.

- **RELABEL**(u) applies to an overflowing vertex u at which no PUSHing can be done. This procedure simply assigns

  $$h(u) \leftarrow 1 + \min\{h(v) : (u,v) \in E_f\}.$$

  This is done so that a PUSH could be done at u later on. Note also that since u is overflowing, the set $\{h(v) : (u,v) \in E_f\}$ is not empty, hence the operation is well-defined.

  Moving u up to this particular height would allow pouring as much water (or flow) as possible into nodes which u could still share the burden. If we move u up too high too quickly, we might be pouring water back to s without making full use of out pipes' capacities.

GENERIC-PUSH-RELABEL

```
1: $h[s] \leftarrow |V|$; $h[u] \leftarrow 0, \forall u \in V - \{s\}$
2: $e[u] \leftarrow 0, \forall u \in V$; $f(u,v) \leftarrow 0, \forall (u,v) \in V \times V$
3: for each $u$ with $(s,u) \in E$ do
4:     $f[s,u] \leftarrow c[s,u]$; $e[u] \leftarrow c[s,u]$; $e[s] \leftarrow e[s] - c[s,u]$
5: end for
6: // End of initialization, start the algorithm
7: while it's possible to PUSH or RELABEL do
8:     if a PUSH$(u,v)$ is possible then
9:         PUSH$(u,v)$
10:    else
```

```
11:         **RELABEL**$(u)$ for some overflowing vertex $u$
12:     **end if**
13: **end while**
```

Lemma 4.1. *If there is some overflowing vertex, then either a PUSH or a RELABEL is possible.*

Proof. Suppose u is overflowing, then there is some v such that $f(v,u) > 0$, which means $(u,v) \in E_f$. If a PUSH is not possible, then clearly RELABEL is as E_f is not empty. □

Also note that PUSHing or RELABELing can only be done at an overflowing vertex. Thus, the algorithm terminates, if at all, when there is no more overflowing vertex. We shall show that the flow at termination is a maximum flow.

Lemma 4.2. *During the execution of GENERIC-PUSH-RELABEL, the following hold*

(i) The function h remains a height function.

(ii) The height of each vertex is weakly increasing. Moreover, each RELABEL(u) increases $h[u]$ by at least 1.

Proof. (i) We show this by induction on the number of operations. Suppose h is a height function before a RELABEL(u) at a preflow f. After the move, $h(u) \leq h(v)+1, \forall (u,v) \in E_f$ is clear. Consider a $(w,u) \in E_f$, the move only increases $h(u)$, hence $h(w) \leq h(u)+1$ still holds.

(ii) We only move vertices up, hence the heights are weakly increasing. If a RELABEL(u) does not increase the height of u, then a PUSH is possible and we would have done the PUSH instead.

□

Theorem 4.3 (Correctness of the algorithm). *If GENERIC-PUSH-RELABEL terminates, then the preflow f at termination is a maximum flow*

Proof. Firstly, we show that during the execution, there is never a path from s to t in D_f. This assertion obviously holds after the initialization. As vertices are moving up, s and t stays put. In order to get to t, we have to move down. Due to the condition on the height function, each edge in E_f moves down at most one step. There are only $n-2$ vertices other than s and t, yet s and t are separated by n height levels.

Secondly, the algorithm terminates only when there is no overflowing vertex, i.e. the preflow becomes a feasible flow. Since there is no s,t-path, this feasible flow cannot be augmented; it is thus a maximum flow. □

We next show that the algorithm terminates after a bounded number of steps.

Lemma 4.4. *During the execution of GENERIC-PUSH-RELABEL, for any overflowing node u, there is always a path from u to s in D_f.*

Proof. Intuitively, the only way for u to be overflowing is that there are PUSHes along some path from s to u. Thus, there must be a path from u to s in the residual network. However, we have to turn this into a rigorous argument.

Note that the algorithm ensures $e(v) \geq 0$ for all $v \neq s$. Let U be the set of vertices reachable from u in E_f. Then, $s \notin U$ and $f(\bar{U}, U) = 0$ (why?). However,

$$\begin{aligned} 0 &< \sum_{v \in U} (f^-(v) - f^+(v)) \\ &= f(V, U) - f(U, V) \\ &= (f(\bar{U}, U) + f(U, U)) - (f(U, \bar{U}) + f(U, U)) \\ &= -f(U, \bar{U}) \\ &\leq 0, \end{aligned}$$

which is a contradiction. □

Corollary 4.5. *(i) It is always the case that $h[u] \leq 2|V| - 1$.*

(ii) The number of calls to RELABEL is at most $(2|V| - 1)(|V| - 2) < 2|V|^2$.

Proof. (i) As there's always a path from u to s, and each edge moves down at most 1 level, u cannot be two high from s.

(ii) Each RELABEL increases some vertex' height at least 1. There are $|V| - 2$ such vertices each of whom has height at most $2|V| - 1$.

□

Lemma 4.6. *The number of saturating PUSH operations called is at most $2|V||E|$.*

Proof. If a saturating (u, v)-push was done, then (u, v) is no longer an edge. In order for another saturating (u, v)-push to occur, there must be a (v, u)-push. It is easy, then, to see that $h[u]$ from one saturating (u, v) push to the next is increased by at least 2. This observation completes the proof. □

Lemma 4.7. *The number of nonsaturating PUSH operations called is at most $4|V|^2(|V| + |E|)$.*

Proof. Define a potential function

$$\Phi = \sum_{v : e(v) > 0} h[v].$$

It is easy to see that

- relabeling a vertex increases Φ by at most $2|V|$,
- a saturating push increases Φ by at most $2|V|$,
- a nonsaturating push decreases Φ by at least 1.

As the total increase is at most $2|V|(2|V|^2) + 2|V|(2|V||E|) = 4|V|^2(|V| + |E|)$, this number is also an upper bound for the number of nonsaturating PUSHes. □

Corollary 4.8. *With appropriate data structures, the GENERIC-PUSH-RELABEL algorithm can be implemented in $O(|V|^2|E|)$-time.*

5 Applications of the Max-Flow Min-Cut theorem

Before proceeding to more formal discussions on algorithms to find maximum flows, let us discuss some applications of the Max-Flow Min-Cut theorem to connectivity problems in graph theory.

5.1 Matchings, covers, and systems of distinct representatives

Definition 5.1. A *matching* G is a subset M of edges no two of which share an end point. A *maximal matching* is a matching where no more edge can be added to it to form another matching. A *maximum matching* is a matching with maximum size. A *perfect matching* is a matching which covers all vertices. The size of a maximum matching, called the *matching number* of G, is denoted by $\nu(G)$.

For any matching M of G, an M*-alternating path* is a path of G which alternates between edges in M and not in M; an M*-augmenting path* is an M-alternating paths which starts and ends at edges not in M.

Exercise 8. Prove that a matching M of a graph G is maximum iff there is no M-augmenting path.

Definition 5.2. A subset $U \subseteq V(G)$ is called a *vertex cover* of G iff every edge of G is incident to at least one vertex in U. The size of any smallest vertex cover of G is called the *vertex covering number* of G, and is denoted by $\tau(G)$.

Definition 5.3. An *edge-cover* of G is a set of edges whose set of end points is $V(G)$. The size of any smallest edge cover of G is denoted by $\rho(G)$, and is called the *edge covering number* of G.

Definition 5.4. A set of vertices is *independent* if there's no edge between any two of them. The size of any maximum independent set is called the *independent number* of G, and is denoted by $\alpha(G)$.

Exercise 9 (Gallai Identities, 1959 [10]). For any graph G, let $n = V(G)$, prove that

(i) $\alpha(G) + \tau(G) = n$.

(ii) $\nu(G) + \rho(G) = n$ if G has no isolated vertex.

Exercise 10. Prove the following statements

(i) A minimal edge-cover is minimum iff it contains a maximum matching.

(ii) A maximal matching is maximum iff it is contained in a minimum edge-cover.

Exercise 11. Show that for any graph G, $\nu(G) \le \tau(G) \le 2\nu(G)$.

Exercise 12 (König, 1916 [14]). Show that $\alpha(G) = \rho(G)$ if G is a bipartite graph.

Definition 5.5. A graph $G = (A \cup B, E)$ where A and B are non-empty independent sets, is called a *bipartite graph*. A *complete matching* from A into B is a matching of G which covers all vertices of A.

The following theorem is another duality-type of theorem.

Theorem 5.6 (König-Egerváry Theorem). *Let G be a bipartite graph, then $\tau(G) = \nu(G)$.*

Proof. Suppose $G = (A \cup B, E)$. Construct a flow network $D = (V, A)$ from G as follows. Let $V = \{s\} \cup A \cup B \cup \{t\}$, where s and t are two new vertices. The edges of D consists of:

- all edges of the form (s, a), for each $a \in A$
- edges of the form (a, b), whenever $(a, b) \in E$
- all edges of the form (b, t), for each $b \in B$.

Set the capacities of all edges in D to be 1.

The theorem asserts that in a bipartite graph, the size $\nu(G)$ of a maximum matching is equal to the size $\tau(G)$ of a minimum vertex cover of G. We shall show that $\nu(G)$ is equal to the maximum flow value val(f^*), and $\tau(G)$ is the same as the capacity of a minimum cut of D.

By the integrality theorem, there is a maximum flow f^* with integer flow values on each edge. It is easy to see that the set $\{(a, b) \mid f(a, b) = 1\}$ forms a matching of G. Hence, val$(f^*) \leq \nu(G)$. Conversely, given a matching M of G we can construct a feasible flow f with val$(f) = |M|$ by assigning $f(a, b) = 1, \forall(a, b) \in M$, $f(s, a) = 1$ if there is some b such that $(a, b) \in M$, and $f(b, t) = 1$ if there is some a such that $(a, b) \in M$. Thus, $\nu(G) \geq$ val(f^*). Consequently, $\nu(G) =$ val(f^*).

Let (S, T) be a source/sink cut of D with minimum capacity. Let $X = S \cap A$ and $Y = S \cap B$. It is easy to see that

$$\text{cap}(S, T) = (|A| - |X|) + |[X, Y]| + (|B| - |Y|).$$

(Drawing a little picture would help understanding this.) Suppose $[X, Y]$ is not empty. Let (x, y) be some edge in $[X, Y]$. Let $S' = S - \{x\}$ and $T' = T \cup \{x\}$, then

$$\text{cap}(S', T') \leq (|A| - |X| + 1) + (|[X, Y]| - 1) + (|B| - |Y|) = \text{cap}(S, T).$$

As (S, T) is a minimum cut, it must be the case that (S', T') is also a minimum cut. Keep doing this until we get a minimum cut (S, T) for which $[X, Y] = \emptyset$. Then, it is clear that $(A - X) \cup (B - Y)$ is a vertex cover of G. Moreover, cap(S, T) is precisely the size of this vertex cover as $|[X, Y]| = 0$. This shows that cap$(S, T) \geq \tau(G)$. Conversely, suppose $C \subseteq A \cup B$ is a vertex cover of G of size $|C| = \tau(G)$. We construct a min-cut with capacity $|C|$ by "reversing" the previous argument. Let $S = \{s\} \cup (A - C)$ and $T = (B - C) \cup \{t\}$. Then (S, T) is a source/sink cut. Since C is a vertex cover, there cannot be any edge of G with one end point in $A - C$ and the other in $B - C$. Hence, this cut (S, T) has capacity exactly $|C \cap A| + |C \cap B| = |C|$. Consequently, $\tau(G)$ is at least the size of a minimum cut. This completes the proof that min-cut size is the same as $\tau(G)$.

The max-flow min-cut theorem then finishes the proof of the theorem. $\square$

Exercise 13. Given a bipartite graph $G = (A \cup B, E)$. Suppose G is k-regular, namely each vertex $v \in A \cup B$ has degree precisely k.

(i) Show that $|A| = |B|$.

(ii) Use network flow to show that G has a perfect matching.

Exercise 14 (Hall's Theorem). For each subset S of vertices of a graph $G = (V, E)$, let $\Gamma(S) := \{v \mid \exists u \in S, uv \in E\}$.

Use network flow to show that a bipartite graph $G = (A \cup B, E)$ has a complete matching from A into B if and only if $|S| \leq |\Gamma(S)|, \forall S \subseteq A$.

Exercise 15 (König's Line Coloring Theorem (1916, [14])). Show that for every bipartite graph G, $\chi_e(G) = \Delta(G)$. Here $\chi_e(G)$ is the chromatic index of G, i.e. $\chi_e(G)$ is the minimum integer so that a $\chi_e(G)$-edge-coloring of G exists, and $\Delta(G)$ is the maximum degree of all vertices in G.

Exercise 16. Two network routers R and S are connected by f fibers. The jth fiber can accommodate up to n_j **different** wavelengths, $1 \leq j \leq f$.

A set C of connections are routed through (R, S). Each connection in C is to be carried on a pre-assigned wavelength. There are w different wavelengths. In C, there are m_i connections on the ith wavelength, $1 \leq i \leq w$.

We are to route the connections in C through (R, S), namely each connection in C is assigned to one of the f fibers such that no two connections with the same wavelength are assigned on the same fiber, and that the jth fiber does not get assigned to more than n_j connections.

(i) Show how to use network flows to test whether the routing can be done.

(ii) Suppose $m_1 \geq \dots \geq m_w$, and $n_1 \leq \dots \leq n_f$. Show that the routing can be done if and only if, for all k, and l, where $0 \leq k \leq w$, $0 \leq l \leq f$, it holds that $k(f-l) + \sum_{j=1}^{l} n_j \geq \sum_{i=1}^{k} m_i$.

Exercise 17 (Common System of Distinct Representatives). Let $\mathcal{X} = \{X_1, \dots, X_m\}$ be a collection of sets. A set of distinct elements $X = \{x_1, \dots, x_m\}$ is called a *system of distinct representatives* of $\mathcal{X}$ if there exists a one-to-one mapping $\phi : X \to \mathcal{X}$ such that $x_i \in \phi(x_i), \forall i = 1 \dots m$.

Let $\mathcal{A} = \{A_1, \dots, A_m\}$ and $\mathcal{B} = \{B_1, \dots, B_m\}$ be two collections of subsets of $[n] = \{1, \dots, n\}$, $m \leq n$. A common system of distinct representatives (CSDR) is a set $S = \{s_1, \dots, s_m\}$ of m (different) elements such that S represents both $\mathcal{A}$ and $\mathcal{B}$. (Note that the one-to-one mappings from S to $\mathcal{A}$ and $\mathcal{B}$ do not need to be the same.)

Use network flows to show that $\mathcal{A}$ and $\mathcal{B}$ have a CSDR if and only if

$$\left| \left(\bigcup_{i \in I} A_i \right) \cap \left(\bigcup_{j \in J} B_j \right) \right| \geq |I| + |J| - m, \text{ for all } I, J \subseteq [m].$$

5.2 Connectivity concepts for graphs

Definition 5.7. A *separating set* (also called *vertex cut*) of a graph $G = (V, E)$ is a subset S of vertices such that $G - S$ has more than one components or $G - S$ is an isolated vertex. A graph G is *k-connected* iff every separating set has size at least k. The *connectivity* $\kappa(G)$ of G is the maximum k such that G is k-connected.

Note that if G is k-connected, then is is also j-connected for all $j \leq k$. The definition basically means that if G is k-connected, then G would still be connected if we remove less than k vertices. For example, a connected graph is certainly 0-connected and 1-connected. A path P is 1-connected but not 2-connected. A cycle C of large length is 0-, 1- and 2-connected but not 3-connected. Clearly $\kappa(P) = 1$ and $\kappa(C) = 2$.

Definition 5.8. Given $u, v \in V$, then an *u, v-separating set* (or *u, v-vertex cut*) is a subset $S \subseteq V - \{u, v\}$ such that $G - S$ has no u, v-path. Naturally, we use $\kappa(u, v)$ to denote the minimum size of a u, v-separating set.

Definition 5.9. Two u, v-paths are *internally disjoint* if they have no vertex in common except u and v. Let $\lambda(u, v)$ denote the maximum number of internally disjoint u, v-paths.

Definition 5.10. A *disconnecting set* of a graph $G = (V, E)$ is a subset $F \subseteq E$ such that $G - F$ has more than one connected component. A graph G is *k-edge-connected* iff every disconnecting set has size at least k. The *edge-connectivity* $\kappa'(G)$ of G is the minimum size of a disconnecting set of G.

We shall adopt Douglas West's convention [21] to append a "prime" after a graph parameter (like κ') to specify that it is the edge version of the parameter.

Definition 5.11. Given $u, v \in V$, then an *u, v-disconnecting set* (or *u, v-cut*) is a subset $S \subseteq E$ such that $G - S$ has no u, v-path. (Thus, $G - S$ has at least two components.) Naturally, we use $\kappa'(u, v)$ to denote the minimum size of a u, v-disconnecting set.

Definition 5.12. Two u, v-paths are *edge disjoint* if they have no edge in common. Let $\lambda'(u, v)$ denote the maximum number of edge disjoint u, v-paths.

Exercise 18. Let G be an undirected graph, and u, v be two vertices of G. Show that:

(i) $\kappa'(u,v) \geq \lambda'(u,v)$,

(ii) $\kappa(u,v) \geq \lambda(u,v)$ if $(u,v) \notin E(G)$.

Exercise 19. Let G be a graph, and u, v be two vertices of G. Show that

(i) $\kappa'(G) = \min\{\kappa'(u,v) \mid u, v \in V(G), u \neq v\}$.

(ii) $\kappa(G) = \min\{\kappa(u,v) \mid u, v \in V(G), u \neq v\}$.

5.3 Connectivity concepts for digraphs

Definition 5.13. A digraph $D = (V, E)$ is *strongly connected* iff there is a directed path from u to v and a directed path from v to u for each pair $u, v \in V$.

Definition 5.14. A *separating set* (or *vertex cut*) of a digraph $D = (V, E)$ is a set S of vertices such that $D - S$ is not strongly connected or $G - S$ is an isolated vertex. The *connectivity* $\kappa(D)$ of D is the minimum size of separating sets. The graph is k-connected if $\kappa(D) \geq k$.

The concepts of $\kappa(u,v)$ and $\lambda(u,v)$ are similar to the undirected case, hence we shall be brief.

Definition 5.15. Let $\kappa(u,v)$ denote the minimum number of vertices in $V - \{u, v\}$ whose removal leaves a graph with no directed u, v-path. Let $\lambda(u,v)$ denote the maximum number of internally disjoint directed u, v-paths.

Definition 5.16. Let $D = (V, E)$ be a digraph. Let $\emptyset \neq S \subset V$, and $[S, \bar{S}]$ denote the set of edges going from S to $\bar{S}$. Then $[S, \bar{S}]$ is called an *edge-cut* of D. The *edge-connectivity* $\kappa'(D)$ is the size of a minimum edge-cut. The graph is *k-edge-connected* if $\kappa'(D) \geq k$.

Definition 5.17. A cut $[S, T]$ with $u \in S$, $v \in T$ is called a *u, v-cut* (or *u, v-disconnecting set*). Let $\kappa'(u,v)$ denote the minimum size of a u, v-cut. Let $\lambda'(u,v)$ denote the maximum number of edge disjoint u, v-paths.

Exercise 20. Let D be a directed graph, and u, v be two vertices of D. Show that

(i) $\kappa'(u,v) \geq \lambda'(u,v)$,

(ii) $\kappa(u,v) \geq \lambda(u,v)$ if $(u,v) \notin E(G)$.

Exercise 21. Let D be a directed graph, and u, v be two vertices of D. Show that

(i) $\kappa'(D) = \min\{\kappa'(u,v) \mid u, v \in V(D), u \neq v\}$.

(ii) $\kappa(D) = \min\{\kappa(u,v) \mid u, v \in V(D), u \neq v\}$.

5.4 Menger theorems for digraphs

Theorem 5.18 (Local Menger Theorem for digraph, edge version). *Let $s \neq t$ be vertices of a digraph $D = (V, E)$. Then, $\kappa'(s,t) = \lambda'(s,t)$.*

Proof. Think of D as a flow network all whose edges have capacity 1, and s as the source and t as the sink. We shall show that

$$\text{max-flow value } \leq \lambda'(s,t) \leq \kappa'(s,t) \leq \text{min-cut capacity},$$

which would complete the proof.

The fact that $\lambda'(s,t) \le \kappa'(s,t)$ is obvious.

By the integrality theorem, there is a maximum integral flow f. Let G be the subgraph of D consists of all edges with flow value 1. We use out-deg(v) and in-deg(v) to denote the out-degree and in-degree of a vertex $v \in V(G)$. Then, val(f) = out-deg(s) − in-deg(s) = in-deg(t) − out-deg(t). Moreover, out-deg(v) = in-deg$(v), \forall v \in V(G) - \{s,t\}$.

Walk along any directed walk starting from s, and we shall come back to s or end up at t, never get stuck in the middle since each vertex $v \in V(G) - \{s,t\}$ has the same in-degree and out-degree. If we come back to s, remove that closed walk from G and the values out-deg(v) − in-deg(v) is unchanged for all $v \in V(G)$. If we end up at t, then we get an s,t-path after removing cycles along the walk. Remove the path from G and we have reduced $\big($out-deg(s) − in-deg$(s)\big)$ and $\big($in-deg(t) − out-deg$(t)\big)$ each by 1, while $\big($out-deg(v) − in-deg$(v)\big)$ is still 0 for all other v. Repeating this procedure val(f) times and we obtain val(f) edge disjoint paths from s to t. This shows that val$(f) \le \lambda'(s,t)$.

To this end, let (S,T) be a minimum source/sink cut, then $[S,T]$ is an s,t-disconnecting set. Consequently, $\kappa'(s,t) \le |[S,T]| =$ cap(S,T). □

Theorem 5.19 (Local Menger Theorem for digraph, vertex version). *Let $s \neq t$ be vertices of a digraph $D = (V,E)$ for which $(s,t) \notin E$. Then, $\kappa(s,t) = \lambda(s,t)$.*

Proof. Construct a digraph D' from D as follows. Each vertex $v \in V(D) - \{s,t\}$ is separated into v^- and v^+ with all edges going into v now going into v^- in D', and all edges going out from v now going out of v^+ in D'. Moreover, we add the edge (v^-, v^+) to D'.

Each s,t-path in D' alternate between edges of D and edges of the form (v^-, v^+). Hence, edge disjoint s,t-paths in D' are internally disjoint in D and vice versa. This means

$$\kappa'_{D'}(s,t) = \lambda'_{D'}(s,t) = \lambda_D(s,t) \le \kappa_D(s,t).$$

To complete the proof, we show that $\kappa_D(s,t) \le \kappa'_{D'}(s,t)$.

Let C' be an s,t-disconnecting set of edges in D' of size $|C'| = \kappa'_{D'}(s,t)$. For each edge $(x,y) \in C'$, not both x and y are in $\{s,t\}$. (Note that x and y, if not s or t, have to be some v^+ or v^- for some $v \in V$.)

Let C be a subset of V obtained from C' by picking one end point of each edge $(x,y) \in C'$ which is neither s or t, and remove the $+$ or $-$ from that end point. For example, if $(x,y) = (s, v^-)$ then we put v in C; or if $(x,y) = (u^+, v^-)$ then we put either u or v in C; etc.

It is clear that $|C| \le |C'|$. Moreover, C must be an s,t-separating set of D, because any s,t-path which avoid all vertices in C corresponds to an s,t-path in D' which avoids all edges in C'. This shows that $\kappa_D(s,t) \le |C| \le |C'| = \kappa'_{D'}(s,t)$ as desired. □

Theorem 5.20 (Global Menger Theorem for digraphs). *Let $D = (V,E)$ be a digraph, then*

$$\begin{aligned}\kappa'(D) &= \min\{\lambda'(s,t) \mid s,t \in V, s \neq t\},\\ \kappa(D) &= \min\{\lambda(s,t) \mid s,t \in V, s \neq t\}.\end{aligned}$$

Proof. Since $\kappa'(D) = \min\{\kappa'(s,t) \mid s,t \in V, s \neq t\}$, and $\kappa'(s,t) = \lambda'(s,t)$, the first equality is trivial.

For the second equality, we also note that $\kappa(D) = \min\{\kappa(s,t) \mid s,t \in V, s \neq t\}$. However, $\kappa(s,t) = \lambda(s,t)$ only for cases when $(s,t) \notin E(D)$.

When $(s,t) \in E(D)$, we have $\kappa(s,t) = \infty$. Thus, we we can show that $\lambda(s,t) \ge \kappa(D)$ for $(s,t) \in E(D)$ then we are done. We have

$$\lambda(s,t) = 1 + \lambda_{D-(s,t)}(s,t) = 1 + \kappa_{D-(s,t)}(s,t) \ge 1 + \kappa(D - (s,t)) \ge \kappa(D).$$

All inequalities are straightforward, except possibly the last one, which asserts that deleting an edge reduces the (vertex) connectivity by at most 1.

Let S be a separating set of minimum size in $D - (s,t)$. If either s or t is in S, then S is also a separating set of D, which implies $\kappa(D - (s,t)) \geq \kappa(D)$. Suppose both s and t are not in S, and that there is no u,v-path in $D - (s,t) - S$. If $\{u,v\} \neq \{s,t\}$, then either $S \cup \{s\}$ or $S \cup \{t\}$ is a separating set of D, which means $\kappa(D) \leq |S| + 1 = \kappa(D - (s,t))$. The last case is when $\{u,v\} = \{s,t\}$. If $u = t, v = s$ then S is also a separating set of D.

Lastly, suppose $u = s$ and $v = t$. If $S = V - \{s,t\}$, then $S \cup \{s\}$ is a separating set of D by definition. On the other hand, if there is some $w \in V - S - \{s,t\}$, then either there is no s,w-path or no w,t path: we get back to a case we have seen before. □

5.5 Menger theorems for graphs

Theorem 5.21 (Local Menger Theorem for digraph, edge version). *Let $s \neq t$ be vertices of a graph $G = (V,E)$. Then, $\kappa'(s,t) = \lambda'(s,t)$.*

Proof. Construct a directed graph from G by turning each edge $(u,v) \in E(G)$ into a pair of edges (u,v) and (v,u) in D. The rest follows from the digraph version of the theorem. □

Theorem 5.22 (Local Menger Theorem for digraph, vertex version). *Let $s \neq t$ be vertices of a graph $G = (V,E)$ for which $(s,t) \notin E$. Then, $\kappa(s,t) = \lambda(s,t)$.*

Proof. Construct a directed graph from G by turning each edge $(u,v) \in E(G)$ into a pair of edges (u,v) and (v,u) in D. The rest follows from the digraph version of the theorem. □

Theorem 5.23 (Menger Theorem for graphs). *Let $G = (V,E)$ be a graph, then*

$$\begin{aligned} \kappa(G) &= \min\{\lambda(s,t) \mid s,t \in V, s \neq t\}, \\ \kappa'(G) &= \min\{\lambda'(s,t) \mid s,t \in V, s \neq t\}. \end{aligned}$$

Proof. Similar to the directed case. □

Historical Notes

The book by Ahuja, Magnanti and Orlin [1] contains extensive discussions on network flows, related problems and applications.

The Max-Flow Min-Cut theorem was obtained independently by Elias, Feinstein, and Shannon (1956, [8]), Ford and Fulkerson (1956, [9]). The special case with integral capacities was also discovered by Kotzig (1956, [16]).

The augmenting path method, sometime referred to as the Ford-Fulkerson algorithm, was devised by Ford and Fulkerson (1956, [9]) based on earlier ideas from Egerváry (1931, [7]) and Kuhn (1955, [17])

The augmenting path method could loop forever, or converge to a sub-optimal flow if some of the capacities were irrational. Examples were constructed by Ford and Fulkerson (1956, [9]), Papadimitriou and Steiglitz (1982, [20]). Edmonds and Karp (1970, [6]) later found a polynomial time algorithm which works on all real capacities, with at most $(n^3 - n)/4$ iterations. Actually, they devised two polynomial time algorithms. The first algorithm augments along paths with maximum residual capacities, which terminates after $O(n \log val f^*)$ iterations. The second algorithm, which we referred to as the Edmonds-Karp algorithm forces flow augmentation to be made along an augmentation path with shortest length. This algorithm works in $O(nm^2)$ time. Dinic (1970, [5]) independently proposed a different variation of the shortest path idea, which works in $O(n^2m)$-time.

Karzanov (1974, [13]) introduced the first push-relabel algorithm, which runs in $O(n^3)$. The generic push-relabel method and several improvements on implementation running times were worked out by Goldberg (1985, [11]), Goldberg and Tarjan (1986, [12]). Further improvements in running times can be found in Ahuja, Orlin, and Tarjan (1989, [2]), Alon (1990, [3]), and Cheriyan, Hagerup, and Mehlhorn (1990, [4]).

A superb text on matching theory is [18]. The book also contains many interesting topics, including discussions on linear programming, convex polytopes, and Pfaffian. The König-Egerváry theorem is from König (1931, [15]) and Egerváry (1931, [7]).

The local, edge version of Menger's theorem was discovered by Menger (1927, [19]). Other versions were later shown by Whitney (1932, [22]), Ford-Fulkerson (1956, [9]), and Elias-Feinstein-Shannon (1956, [8]).

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows*, Prentice Hall Inc., Englewood Cliffs, NJ, 1993. Theory, algorithms, and applications.

[2] R. K. Ahuja, J. B. Orlin, and R. E. Tarjan, *Improved time bounds for the maximum flow problem*, SIAM J. Comput., 18 (1989), pp. 939–954.

[3] N. Alon, *Generating pseudo-random permutations and maximum flow algorithms*, Inform. Process. Lett., 35 (1990), pp. 201–204.

[4] J. Cheriyan, T. Hagerup, and K. Mehlhorn, *Can a maximum flow be computed in $o(nm)$ time?*, in Automata, languages and programming (Coventry, 1990), vol. 443 of Lecture Notes in Comput. Sci., Springer, New York, 1990, pp. 235–248.

[5] E. A. Dinic, *An algorithm for the solution of the problem of maximal flow in a network with power estimation*, Dokl. Akad. Nauk SSSR, 194 (1970), pp. 754–757.

[6] J. Edmonds and R. M. Karp, *Theoretical improvements in algorithmic efficiency for network flow problems*, in Combinatorial Structures and their Applications (Proc. Calgary Internat. Conf., Calgary, Alta., 1969), Gordon and Breach, New York, 1970, pp. 93–96.

[7] J. Egerváry, *Matrixok kombinatorikus tulajdonságairól*, Mathematikai és Fizikai Lápok, 38 (1931), pp. 19–28.

[8] P. Elias, A. Feinstein, and C. E. Shannon, *Note on maximal flow through a network*, IRE Transactions on Information Theory IT-2, (1956), pp. 117–199.

[9] L. R. Ford, Jr. and D. R. Fulkerson, *Maximal flow through a network*, Canad. J. Math., 8 (1956), pp. 399–404.

[10] T. Gallai, *Über extreme Punkt- und Kantenmengen*, Ann. Univ. Sci. Budapest. Eötvös Sect. Math., 2 (1959), pp. 133–138.

[11] A. V. Goldberg, *A new max-flow algorithm*, Tech. Rep. MIT/LCS/TM-291, Laboratory for Computer Science, MIT, Cambridge, 1985.

[12] A. V. Goldberg and R. E. Tarjan, *A new approach to the maximum-flow problem*, J. Assoc. Comput. Mach., 35 (1988), pp. 921–940.

[13] A. V. Karzanov, *The problem of finding the maximal flow in a network by the method of preflows*, Dokl. Akad. Nauk SSSR, 215 (1974), pp. 49–52.

[14] D. König, *Über graphen und ihre anwendung auf determinantentheorie und mengenlehre*, Math. Ann., 77 (1916), pp. 453–465.

[15] ———, *Graphen und matrizen*, Mathematikai és Fizikai Lápok, 38 (1931), pp. 116–119.

[16] A. Kotzig, *Súvislost' a pravidelinâ súvislost' konečných grafov*, Bratislava: Vysoká Škola Ekonomická, (1956).

[17] H. W. Kuhn, *The Hungarian method for the assignment problem*, Naval Res. Logist. Quart., 2 (1955), pp. 83–97.

[18] L. Lovász and M. D. Plummer, *Matching theory*, North-Holland Publishing Co., Amsterdam, 1986. Annals of Discrete Mathematics, 29.

[19] K. Menger, *Zur allgemeinen kurventheorie*, Fund. Math., 10 (1927), pp. 95–115.

[20] C. H. Papadimitriou and K. Steiglitz, *Combinatorial optimization: algorithms and complexity*, Prentice-Hall Inc., Englewood Cliffs, N.J., 1982.

[21] D. B. West, *Introduction to graph theory*, Prentice Hall Inc., Upper Saddle River, NJ, 1996.

[22] H. Whitney, *Congruent graphs and the connectivity of graphs*, Amer. J. Math., 54 (1932), pp. 150–168.