Lecture notes for Feb 15, 2023 Minimum weighted bases and shortest paths in nonnegative weighted digraphs

Chun-Hung Liu

February 15, 2023

Here are some simple properties of matroids.

Proposition 1 *Let $(E, \mathcal{I})$ be a matroid. Then every base has the same size.*

Proof. It immediately follows from (M3). ∎

Proposition 2 *Let $(E, \mathcal{I})$ be a matroid. Let C_1, C_2 be distinct circuits. If $e \in C_1 \cap C_2$, then $(C_1 \cup C_2) - \{e\}$ is not in $\mathcal{I}$ and hence contains a circuit.*

Proof. Suppose to the contrary that $(C_1 \cup C_2) - \{e\} \in \mathcal{I}$. Since C_1, C_2 are distinct circuits, $C_1 \not\subseteq C_2$ and $C_2 \not\subseteq C_1$. So there exists $f \in C_1 - C_2$, and $|(C_1 \cup C_2) - \{e\}| > |C_1 - \{e\}| = |C_1| - 1 = |C_1 - \{f\}|$. Let $I_0 = C_1 - \{f\}$. Since C_1 is a circuit, $I_0 \in \mathcal{I}$ with $|I_0| < |(C_1 \cup C_2) - \{e\}|$. So by (M3), for every i with $1 \leq i \leq |(C_1 \cup C_2) - \{e\}| - |I_0|$, there exists $e_i \in ((C_1 \cup C_2) - \{e\}) - I_{i-1}$ such that $I_{i-1} \cup \{e_i\} \in \mathcal{I}$, and we define $I_i = I_{i-1} \cup \{e_i\}$. Note that $|(C_1 \cup C_2) - \{e\}| - |I_0| = (|C_1| + |C_2 - C_1| - 1) - (|C_1| - 1) = |C_2 - C_1|$. So $C_2 \subseteq (C_1 - \{f\}) \cup (C_2 - C_1) = I_{|(C_1 \cup C_2) - \{e\}| - |I_0|} \in \mathcal{I}$, a contradiction. ∎

Proposition 3 *Let $(E, \mathcal{I})$ be a matroid. Let $I \in \mathcal{I}$. Let $e \in E$ with $I \cup \{e\} \notin \mathcal{I}$. Then there exists a unique circuit C contained in $I \cup \{e\}$, and for every $f \in C$, $(I \cup \{e\}) - \{f\} \in \mathcal{I}$.*

Proof. Since $I \cup \{e\} \notin \mathcal{I}$, $I \cup \{e\}$ contains circuits. Note that every such circuit must contain e, for otherwise it is a subset of the independent set I.

If there exist distinct circuits C_1, C_2 contained in $I \cup \{e\}$, then $e \in C_1 \cap C_2$, so Proposition 2 implies that $(C_1 \cup C_2) - \{e\}$ contains a circuit, but $(C_1 \cup C_2) - \{e\} \subseteq I \in \mathcal{I}$, a contradiction. Let C be the unique circuit contained in $I \cup \{e\}$. For every $f \in C$, if $(I \cup \{e\}) - \{f\} \notin \mathcal{I}$, then there exists a circuit C' contained in $(I \cup \{e\}) - \{f\}$, so C' is a circuit contained in $I \cup \{e\}$ distinct from C, a contradiction. ■

1 Minimum weighted base

Note that $|\mathcal{I}|$ can be exponential in $|E|$. So we cannot afford to list all members of $\mathcal{I}$ when we work on algorithmic problems on matroids. Instead, the input matroid $(E, \mathcal{I})$ is given by listing all the elements of E, and we assume that we are given an oracle to test whether a given subset of E is in $\mathcal{I}$ or not.

==

Finding a minimum weighted base of a matroid
Input: A matroid $(E, \mathcal{I})$ and a weight function $w : E \to \mathbb{R}$.
Output: A base T of $(E, \mathcal{I})$ with minimum $\sum_{x \in T} w(x)$.
Procedure:

Step 1: Sort the elements of E to obtain an ordering of the elements $e_1, e_2, ..., e_{|E|}$ of E so that $w(e_1) \leq w(e_2) \leq ... \leq w(e_{|E|})$. Set $T = \emptyset$.

Step 2: For $i = 1, 2, ..., |E|$, if $T \cup \{e_i\} \in \mathcal{I}$, then add e_i into T.

==

Lemma 4 *The set T output from the above algorithm is a base.*

Proof. If the final output T is not a base, then there exists $e_i \in E - T$ such that $T \cup \{e_i\} \in \mathcal{I}$. But then e_i must be added into T by the algorithm and (M2). ■

Theorem 5 *The above algorithm outputs a base with minimum weight in time $O(|E| \log |E| + |E|\theta)$, where θ is the running time of the oracle for checking the membership of $\mathcal{I}$.*

Proof. Let T be the set output from the algorithm. Note that T is a base by Lemma 4. Let $e_1, e_2, ..., e_{|E|}$ be the ordering of E such that $w(e_1) \leq w(e_2) \leq ... \leq w(e_{|E|})$. Suppose to the contrary that the weight of T is larger than another base. Let T^* be a base with minimum weight, and subject to this, $\min\{i : e_i \in (T - T^*) \cup (T^* - T)\}$, denoted by i^*, is maximum. Note that $\min\{i : e_i \in (T - T^*) \cup (T^* - T)\}$ exists since the weight of T is greater than the weight of T^*. So for every $i \in [i^* - 1]$, either $e_i \in T \cap T^*$ or $e_i \notin T \cup T^*$.

We first assume $e_{i^*} \in T^* - T$. By the algorithm, $(T \cap \{e_i : i \in [i^* - 1]\}) \cup \{e_{i^*}\} \notin \mathcal{I}$. So $T^* \cap \{e_i : i \in [i^*]\} = (T \cap \{e_i : i \in [i^* - 1]\}) \cup \{e_{i^*}\} \notin \mathcal{I}$, a contradiction.

So $e_{i^*} \in E(T) - E(T^*)$. Since T^* is a base, $T^* \cup \{e_{i^*}\} \notin \mathcal{I}$. By Proposition 3, there exists a unique circuit C in $T^* \cup \{e_{i^*}\}$ such that $(T^* \cup \{e_{i^*}\}) - \{f\} \in \mathcal{I}$ for every $f \in C - \{e_{i^*}\}$. Since $(T^* \cap \{e_i : i \in [i^* - 1]\}) \cup \{e_{i^*}\} = T \cap \{e_i : i \in [i^*]\} \in \mathcal{I}$, $C - ((T^* \cap \{e_i : i \in [i^* - 1]\}) \cup \{e_{i^*}\}) \neq \emptyset$, so there exists $e_j \in C - ((T^* \cap \{e_i : i \in [i^* - 1]\}) \cup \{e_{i^*}\})$ for some $j \geq i^* + 1$. So $(T^* \cup \{e_{i^*}\}) - \{f\} \in \mathcal{I}$. Let $T' = (T^* \cup \{e_{i^*}\}) - \{f\}$. Since $|T'| = |T^*|$ and $T' \in \mathcal{I}$, T' is a base. Since $\min\{i : e_i \in (T - T') \cup (T' - T)\} \geq i^* + 1$, $\sum_{e \in T'} w(e) > \sum_{e \in T^*} w(e)$ by the choice of T^*. But it implies that $w(e_{i^*}) > w(e_j)$, contradicting $i^* < j$.

Hence T is a base of $(E, \mathcal{I})$ with minimum weight. And Step 1 of the algorithm takes time $O(|E| \log |E|)$, and Step 2 takes time $O(|E|\theta)$. ■

2 Shortest paths in weighted (di)graphs with nonnegative weights

Let (D, w) be a weighted directed graphs. The length of a directed path P in D is defined to be $\sum_{e \in E(P)} w(e)$. For any two vertices u, v of D, the *distance in* (D, w) *from* u *to* v, denoted by $d_{(D,w)}(u, v)$, is defined to be the length of a shortest path in D from u to v (or defined to be ∞ if no path in D is from u to v).

Let (G, w) be a weighted graph. The length of a path P in G is defined to be $\sum_{e \in E(P)} w(e)$. For any two vertices u, v of G, the *distance in* (D, w) *between* u *to* v is defined to be the length of a shortest path in G between u and v (or defined to be ∞ if no path in G is between u and v). In fact, the distance between any two vertices in (G, w) equals the distance from u to v in (D_G, w_G), where (D_G, w_G) is the digraph obtained from G by replacing each edge e by two arc with different directions e_1, e_2 and assigning $w_G(e_1) = w_G(e_2) = w(e)$. So if we want to find the distance between two vertices in (G, w), it suffices to find the distance in (D_G, w).

==

Dijkstra's algorithm
Input: A weighted digraph (D, w), where w is a nonnegative function, and a vertex $r \in V(D)$.
Output: A forest T rooted at r such that T is a subgraph of D with $V(T) = V(D)$ and $d_{(D,w)}(r, v) = d_{(T,w|_{E(T)})}(r, v)$ for every $v \in V(D)$.
Procedure:

Step 1: Set $R = \emptyset$. Set $f(r) = 0$ and $f(v) = \infty$ for every $v \in V(D) - \{r\}$. Set $p(r) = r$. Set $T = (V(D), \emptyset)$.

Step 2: For $i = 1, 2, ..., |V(D)|$, do the following:

- Find a vertex v_i with $f(v_i) = \min_{x \in V(D)-R} f(x)$.
- If $f(v_i) = \infty$, then stop; otherwise, do the following:

Step 2-1: If $v_i \neq r$, then add an edge $(p(v_i), v_i)$ into T.

Step 2-2: Add v_i into R, and for every edge $(v_i, x) \in E(D)$, if $f(v_i) + w((v_i, x)) < f(x)$, then define $p(x) = v_i$ and redefine $f(x) = f(v_i) + w((v_i, x))$.

==

Lemma 6 *During the entire process, the following properties are preserved:*

1. *For every vertex $v \in R$, $d_{(D,w)}(r, v) = d_{(T,w)}(r, v) = d_{(D[R],w)}(r, v)$.*

2. *For every vertex $v \in V(D) - \{r\}$, if $p(v)$ is defined, then $p(v) \in R$ and $(p(v), v) \in E(D)$.*

3. *T is a forest and a subgraph of D with $V(T) = V(D)$, and $R \cup \{r\}$ is the vertex-set of the component of T containing r, and the direction of every edge of $T[R \cup \{r\}]$ is from the side having r.*

4. *For every vertex $v \in V(D) - R$, if there exists an edge of D from R to v, then $p(v)$ is defined, and $f(v) = d_{(D[R]+(p(v),v),w)}(r, v) = d_{(D[R\cup\{v\}],w)}(r, v)$.*

Proof. All properties are clearly preserved at the end of Step 1. Assume we just find a new v_i for some i, and all the properties are preserved at this moment. If $f(v_i) = \infty$, then the algorithm stops and we are done. So we may assume $f(v_i) \neq \infty$.

Since Properties 2 and 3 are preserved until this moment, Property 3 is preserved at the end of Step 2-1. And Properties 1, 2 and 4 are preserved at the end of Step 2-1 since R and p were not changed.

Now we do Step 2-2 by adding v_i into R and updating f. If $v_i = r$, then all properties are clearly preserved. So we may assume $v_i \neq r$. Properties 2 and 3 are clearly preserved at the end of Step 2-2 by our definition of p and T. For clarity, let R_0 be the set R at the end of Step 2-1, and let $R_1 = R_0 \cup \{v_i\}$.

Now we show that Property 1 is preserved at the end of Step 2-2. Since Property 4 is preserved at the end of Step 2-1, $f(v_i) = d_{(D[R_0]+(p(v_i),v_i),w)}(r, v_i) = d_{(D[R_0\cup\{v_i\}],w)}(r, v_i)$. Since $(p(v_i), v_i)$ is the unique edge in $D[R_0] + (p(v_i), v_i)$ between R_0 and v_i by Properties 2 and 3 preserved so far, $d_{(D[R_0],w)}(r, p(v_i)) + w((p(v_i), v_i)) = d_{(D[R_0]+(p(v_i),v_i),w)}(r, v_i) = d_{(D[R_0\cup\{v_i\}],w)}(r, v_i)$. Since Property 1 is preserved at the end of Step 2-1, $d_{(D,w)}(r, p(v_i)) = d_{(T,w)}(r, p(v_i)) = d_{(D[R_0],w)}(r, p(v_i))$. So $d_{(T,w)}(r, v_i) = d_{(T,w)}(r, p(v)) + w((p(v_i), v_i)) = d_{(D[R_0],w)}(r, p(v_i)) + w((p(v_i), v_i)) = d_{(D[R_0\cup\{v_i\},w)}(r, v_i) = d_{(D[R_1],w)}(r, v_i)$. Since $D[R_1]$ is a subgraph of D, $d_{(D[R_1],w)}(r, v_i) \geq d_{(D,w)}(r, v_i)$.

Suppose to the contrary that $d_{(D[R_1],w)}(r, v_i) > d_{(D,w)}(r, v_i)$. Then there exists a path P in D from r to v_i with length $d_{(D,w)}(r, v_i)$. Let P' be the maximal subpath from r contained in $D[R_0]$. Let y be the sink of P'. So $(y, z) \in E(P)$ for some $z \in V(D) - R$. Since $P' + (y, z)$ is contained in $D[R_0 \cup \{z\}]$, $d_{(D[R_0\cup\{z\}],w)}(r, z)$ equals the length of $P' + (y, z)$, which is at most the length of P and hence is strict smaller than $d_{(D[R_1],w)}(r, v_i) = f(v_i)$. Then at the beginning of Step 2, $f(z) = d_{(D[R_0\cup\{z\}],w)}(r, v) < f(v_i)$, contradicting the choice of v_i. So $d_{(D[R_1],w)}(r, v_i) \leq d_{(D,w)}(r, v_i)$

Hence $d_{(D[R_1],w)}(r, v_i) = d_{(D,w)}(r, v_i)$. So Property 1 is preserved at the end of Step 2-2.

Now show that Property 4 is preserved at the end of Step 2-2. Let f_0 be the f at the end of Step 2-1. Let f_1 be the f at the end of Step 2-2. Let $v \in V(D) - R_1$ such that there exists an edge of D from R_1 to v. Let Q be a shortest path in $(D[R_1 \cup \{v\}], w)$ from r to v such that $v_i \notin V(Q)$ if possible.

We first assume $v_i \notin V(Q)$. Then $d_{(D[R_1\cup\{v\}],w)}(r, v) = d_{(D[R_0\cup\{v\}],w)}(r, v) = f_0(v)$. If $(v_i, v) \notin E(D)$, then $f_1(v) = f_0(v)$ and we are done. If $(v_i, v) \in E(D)$, then since $f_0(v_i) + w((v_i, v)) = d_{(D[R_0\cup\{v_i\}],w)}(r, v_i) + w((v_i, v)) = d_{(D[R_1],w)}(r, v_i) + w((v_i, v)) \geq d_{(D[R_1\cup\{v\}],w)}(r, v) = f_0(v)$, we know $f_1(v) =$

$f_0(v)$, so we are done.

Hence we may assume $v_i \in V(Q)$. By the choice of Q, $\sum_{e\in E(Q)} w(e) = d_{(D[R_1\cup\{v\}],w)}(r,v) < d_{(D[R_0\cup\{v\}],w)}(r,v) = f_0(v)$. Let q be the neighbor of v in Q. If $q \neq v_i$, then $q \in R_0$, so by Property 1, there exists a path Q' in $D[R_0]$ with length at most the length of $Q - v$, and hence $Q' + (q,v)$ is not longer than Q but $Q' + (q,v)$ does not contain v_i, contradicting the choice of Q. So $q = v_i$. Hence $f_0(v_i) + w((v_i,v)) = d_{(D[R_0\cup\{v_i\}],w)}(r,v_i) + w((v_i,v)) = d_{(D[R_1],w)}(r,v_i) + w((v_i,v)) = \sum_{e\in E(Q)} w(e) < f_0(v)$. So $p(v)$ is redefined to be v_i, and $f_1(v) = f_0(v_i) + w((v_i,v)) = \sum_{e\in E(Q)} w(e) = d_{(D[R_1\cup\{v\}],w)}(r,v) = d_{(D[R_1]+(v_i,v)),w)}(r,v)$. Therefore Property 4 is preserved. ■

Theorem 7 *Dijkstra's algorithm works correctly and runs in time $O(|V(D)|^2 + |E(D)|)$.*

Proof. By Property 1 in Lemma 6, $d_{(D,w)}(r,v) = d_{(T,w)}(r,v)$ for all $v \in V(R)$. By Property 3 in Lemma 6, $d_{(T,w)}(r,v) = \infty$ for all $v \in V(D) - V(R)$, so it suffices to show that $d_{(D,w)}(r,v) = \infty$ for all $v \in V(D) - V(R)$. And it suffices to show that there exists no edge of D from $V(R)$ to $V(D) - V(R)$. Suppose to the contrary that there exists an edge of D from R to a vertex $v \in V(D) - V(R)$. In particular, $|R| < |V(D)|$. By Properties 1 and 3, $d_{(D[R\cup\{v\}],w)}(r,v) < \infty$. By Property 4, $f(v) = d_{(D[R\cup\{v\}],w)}(r,v) < \infty$. So when $i = |R| + 1$ at Step 2, the vertex v_i satisfies $f(v_i) < \infty$, so v_i should be added into R. It implies that $|R|$ should be bigger, a contradiction.

Note that Step 1 takes time $O(|V(D)|)$. In for each round in Step 2, it takes time $O(|V(D)|)$ to find v_i, and it takes time $O(1+\deg^+_D(v_i))$ to do Steps 2-1 and 2-2, where $\deg^+_D(v_i)$ is the out-degree of v_i in D. So the total running time for Step 2 is $\sum_{v\in V(D)} O(|V(D)| + \deg^+_D(v)) = O(|V(D)|^2 + |E(D)|)$. Hence the algorithm takes time $O(|V(D)|^2 + |E(D)|)$. ■

Note that Dijkstra's algorithm is best possible when $|E(D)| = \Omega(|V(D)|^2)$. Note that if D is simple, then $|E(D)| = O(|V(D)|^2)$, so $O(|V(D)|^2 + |E(D)|) = O(|V(D)|^2)$.

On the other hand, if we implement the algorithm by using heaps, then it only takes $O(\log |V(D)|)$ time to find each v_i in Step 2, so the total running time can be improved to $O(|E(D)| + |V(D)| \log |V(D)|)$.

Corollary 8 *Given a weighted digraph (D,w), where w is nonnegative, and a vertex $r \in V(D)$, in time $O(|V(D)|^2 + |E(D)|)$, we can compute $d_{(D,w)}(r,v)$ and a shortest path in D from r to v (if it exists) for all $v \in V(D)$.*

Proof. Apply Dijkstra's algorithm to get the forest T. Then the distance and path can be found in T. ∎