Chapter 17

Network Flow VI - Min-Cost Flow

Applications

CS 573: Algorithms, Fall 2013
October 24, 2013

17.0.0.1 Lemma – decomposing flow into cycles

Lemma 17.0.1. \( f \) a circulation in \( G \). Then, \( f \) can be decomposed into \( m \) cycles, \( C_1, \ldots, C_m \), such that, \( \forall e \in E(G) \):

\[
f(e) = \sum_{i=1}^{t} \lambda_i \cdot \chi_{C_i}(e),
\]

\( \lambda_1, \ldots, \lambda_t > 0 \) and \( t \leq m, \ m = |E(G)| \).

Proof

(A) By conservation of flow, if \( f \) is not zero, then there is a cycle in \( f \).
   (i) ... \( v \): vertex non-zero flow into it.
   (ii) walk on an adjacent edge that has positive flow on it.
   (iii) Repeat, till visit a vertex that was already visited.
   (iv) Extract cycle contained in this walk.

17.0.0.2 Proof continued...

Proof continued:

(A) \( C_1 \): such a cycle.
(B) every edge of \( C_1 \) has positive flow.
(C) \( \lambda_1 \): smallest amount of flow on any edge of \( C_1 \).
   \( e_1 \) this edge.
(D) \( g = f - \lambda_1 \cdot \chi_{C_1} \).
   \( g \) has zero flow on \( e_1 \). It is a (weak) circulation.
(E) Removed \( e_1 \) from \( G \). \( H \): new graph.
(F) By induction on \( g \) on \( H \).
(G) \( \Rightarrow \) \( g \) can be decomposed into \( m - 1 \) cycles with positive coefficients.
(H) These cycles + \( C_1 \) (with \( \lambda_1 \)) implies the claim.

17.0.0.3 Result

Theorem 17.0.2. A flow \( f \) is a minimum cost feasible circulation \( \iff \) each directed cycle of \( G_f \) has nonnegative cost.
Proof
(A) \( C \): a negative cost cycle in \( G_f \).
(B) Circulate more flow on \( C \) and reduce price.
(C) \( \varepsilon > 0 \): sufficiently small constant:
\[
g = f + \varepsilon \chi_C \text{ is feasible circulation}
\]
(D) \( \text{cost}(g) = \text{cost}(f) + \sum_{e \in C} \kappa(e) \varepsilon = \text{cost}(f) + \varepsilon \sum_{e \in C} \kappa(e) = \text{cost}(f) + \varepsilon \kappa(C) < \text{cost}(f), \) since
\( \kappa(C) < 0 \). A contradiction.

17.0.0.4 Proof of other direction...

Proof of other direction
(A) Assume all cycles in \( G_f \) have non-negative cost.
(B) \( g \) any feasible circulation.
(C) Weak circulation: \( h = g - f \).
(D) By lemma: all edges used by \( h \) are in \( G_f \).
(E) By other lemma: \( \exists t \leq |E(G_f)| \) cycles \( C_1, \ldots, C_t \) in \( G_f \), and coefficients \( \lambda_1, \ldots, \lambda_t \): \( h(e) = \sum_{i=1}^t \lambda_i \chi_{C_i}(e) \).
(F) \( \text{cost}(g) - \text{cost}(f) = \text{cost}(h) = \text{cost}\left( \sum_{i=1}^t \lambda_i \chi_{C_i} \right) = \sum_{i=1}^t \lambda_i \text{cost}(\chi_{C_i}) = \sum_{i=1}^t \lambda_i \kappa(C_i) \geq 0, \) as \( \kappa(C_i) \geq 0 \), since there are no negative cycles in \( G_f \).
(G) \( \implies \text{cost}(g) \geq \text{cost}(f). \implies f \text{ minimum-cost circulation}. \)

17.1 A Strongly Polynomial Time Algorithm for Min-Cost Flow

17.1.0.5 A Strongly Polynomial Time Algorithm for Min-Cost Flow

(A) Start from a feasible circulation \( f \).
(B) Computed using unweighted algorithm (uses max-flow algorithm).
(C) Repeat:
   (A) find cycle \( C \) of minimum average cost in \( G_f \).
   (B) If cost of \( C \) is non-negative. Done.
   (C) Circulate as much flow as possible along \( C \)
       (without violating the lower-bound constraints and capacity constraints),
   (D) Reduces cost.
   (E) Compute such a cycle in \( O(mn) \) time.

17.1.0.6 Analysis...

(A) Algorithm preserve integrality.
(B) cost of the flow is monotonically decreasing.
(C) algorithm would terminate if all the number involved are integers.
(D) This algorithm performs a polynomial number of iterations in \( n \) and \( m \).
(E) Simple algorithm!
17.2 Analysis of the Algorithm
17.2.0.7 Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>f, g, h, i</td>
<td>Flows or circulations</td>
</tr>
<tr>
<td>$G_f$</td>
<td>The residual graph for $f$</td>
</tr>
<tr>
<td>$c(e)$</td>
<td>The capacity of the flow on $e$</td>
</tr>
<tr>
<td>$\ell(e)$</td>
<td>The lower-bound (i.e., demand) on the flow on $e$</td>
</tr>
<tr>
<td>cost($f$)</td>
<td>The overall cost of the flow $f$</td>
</tr>
<tr>
<td>$\kappa(e)$</td>
<td>The cost of sending one unit of flow on $e$</td>
</tr>
<tr>
<td>$\psi(e)$</td>
<td>The reduced cost of $e$</td>
</tr>
</tbody>
</table>

17.2.0.8 Analysis

(A) $f_i$: flow in the beginning of the $i$th iteration.
(B) $C_i$: cycle used in the $i$th iteration.
(C) $C_f$: minimum average-length cycle of $G_f$.
(D) $\mu(f) = \kappa(C_f)/|C_f|$ average “cost” per edge of $C_f$.

**Lemma 17.2.1.** Let $f$ be a flow, and let $g$ the flow resulting from applying the cycle $C = C_f$ to it. Then, $\mu(g) \geq \mu(f)$.

*Proof:* See class notes.

17.2.1 Bounding the number of iterations
17.2.1.1 Lemma

**Lemma 17.2.2.** Let $f$ be a flow used in the $i$th iteration of the algorithm, let $g$ be the flow used in the $(i+m)$th iteration, where $m$ is the number of edges in $G$. Furthermore, assume that the algorithm performed at least one more iteration on $g$. Then, $\mu(g) \geq (1 - 1/n)\mu(f)$.

*Proof:* See class notes.

17.2.1.2 Edges disappear

**Lemma 17.2.3.** Let $f$ be the circulation maintained by the algorithm at iteration $\rho$. Then there exists an edge $e$ in the residual network $G_f$ such that it never appears in the residual networks of circulations maintained by the algorithm, for iterations larger than $\rho + t$, where $t = 2nm \lceil \ln n \rceil$.

*Proof:* See class notes.

17.2.1.3 The result

(A) at least one edge disappears forever every $O(mn \log n)$ iterations.
(B) it follows that after $O(m^2n \log n)$ iterations algorithm terminates.
(C) Every iteration takes $O(mn)$ time.

**Theorem 17.2.4.** Given a digraph $G$ with $n$ vertices and $m$ edges, lower bound and upper bound on the flow of each edge, and a cost associated with each edge, then one can compute a valid circulation of minimum-cost in $O(m^3n^2 \log n)$ time.
17.2.1.4 Bibliographical Notes

(A) minimum average cost cycle algorithm is due to Karp [Karp 1978].
(B) Our follows [Schrijver 2004].
(C) First strongly polynomial time algorithm for minimum-cost circulation is due to Éva Tardos [Tardos 1985].
(D) Algorithm show Andrew Goldberg and Robert Tarjan [Goldberg and Tarjan 1989].
(E) Initial research on this problem can be traced back to the 1940s.
(F) Took almost fifty years to find a satisfactory solution!

17.3 Applications of min cost flow

17.4 Efficient Flow

17.4.0.5 Efficient flow

(A) A flow \( f \) would be considered to be efficient if it contains no cycles in it.
(B) Ford-Fulkerson algorithm might generate flows with cycles.
(C) See figure...
(D) A in middle of edges: split the edge into multiple edges by introducing vertices/edges.
(E) All edges have capacity one.
(F) Demonstrate how to get cycle...

17.4.0.6 Efficient flow

(A) compute the max flow in \( G \).
(B) \( \alpha \) be value of this flow.
(C) compute min-cost flow in this network from \( s \) to \( t \) with flow \( \alpha \).
(D) Every edge has cost one.
(E) New flow will not contain any cycles!
(F) Can also be achieved directly by removing cycles directly in the flow. Might be less efficient than the min-cost flow computed.

Theorem 17.4.1. Computing an efficient (i.e., acyclic) max-flow can be done in polynomial time.

17.5 Efficient Flow with Lower Bounds

17.5.0.7 Lower bounds and efficient flows

(A) \textbf{AFWLB}: acyclic flow with lower-bounds.
(B) Compute efficient flow (acyclic) with lower bounds on the edges.
(C) Require that the returned flow is integral, if all the numbers involved are integers.
(D) This problem is \textbf{NP-Complete}.
(E) Because...
Hamiltonian Path

**Instance:** A directed graph $G$ and two vertices $s$ and $t$.

**Question:** Is there a Hamiltonian path (i.e., a path visiting every vertex exactly once) in $G$ starting at $s$ and ending at $t$?

17.5.0.8 Lower bounds and efficient flows

(A) **Hamiltonian Path** is **NP-Complete**

(B) reduce **Hamiltonian Path** to **AFWLB**.

(C) Replace vertex of $G$ with two vertices and a direct edge in between them.

(D) Set the lower-bound and capacity of each such edge to 1.

(E) Consider an acyclic flow in $H$ of capacity 1 from $s$ to $t$ which is integral. Its 0/1-flow, and as such it defines a path that visits all the special edges we created.

(F) Corresponds to a path in the original graph that starts at $s$, visits all the vertices of $G$ and ends up at $t$.

17.5.0.9 Lower bounds and efficient flows

**Theorem 17.5.1.** Computing an efficient (i.e., acyclic) max-flow with lower-bounds is **NP-Hard** (where the flow must be integral). The related decision problem (of whether such a flow exist) is **NP-Complete**.

(A) Confusing...

(B) We can model an acyclic max-flow problem with lower bounds as min-cost flow, and solve it, no?

(C) Not quite. Solution returned from the min-cost flow might have cycles.

(D) Can not remove them by canceling the cycles.

(E) Removing cycles was only possible when there was no lower bounds.

(F) min-cost flow algorithm would return us a solution with cycles in it if there are lower bounds on the edges.

17.6 Shortest Edge-Disjoint Paths

17.6.0.10 Shortest Edge-Disjoint Paths

(A) $G$: a directed graph.

(B) Q: Compute $k$-edge disjoint paths from $s$ to $t$.

(C) Solve with network flow.

(D) Can compute shortest $k$-edge disjoint paths using min-cost flow.

(E) Assign cost 1 for every edge, and capacity 1 for every edge.

(F) min-cost flow in this graph with value $k$, corresponds to a set of $k$ edge disjoint paths, such that their total length is minimized.

17.7 Covering by Cycles

17.7.0.11 Covering by Cycles

(A) $G$: directed graph.

(B) cover all its vertices by a set of cycles which are vertex disjoint.
(C) can be done using min-cost flow.
(D) \( \forall u \in V(G) \) replace by an edge \((u' \rightarrow u'')\).
(E) \( u' \): incoming edges connected \( u' \). Outgoing edges start from \( u'' \).
(F) \( H \): resulting graph.
(G) All new edges have a lower bound/capacity = 1.
(H) All other edges: no lower bound, capacity 1.
(I) Compute minimum cost circulation in \( H \).
(J) corresponds to a collection of cycles in \( G \) covering all the vertices of minimum cost.

**Theorem 17.7.1.** \( G \) + costs on edges. Compute cover of \( G \) by a collection of vertex disjoint cycles, of min total cost of the cycle edges.

### 17.8 Minimum weight bipartite matching

**17.8.0.12 Minimum weight bipartite matching**

(A) \( G \): undirected bipartite graph \( G \),
(B) Task: Compute max cardinality matching in \( G \) t of minimum cost.
(C) Reduce to network flow as done before.
(D) Compute maximum flow.
(E) any edge has capacity 1. Compute \( \phi = \) size of matching.
(F) Compute min-cost flow in \( G \) with this value \( \phi \).
(G) Zero cost to new edges, original cost on other.
(H) min-cost flow corresponds to a maximum cardinality min-cost flow in the original graph.
(I) Using: flow computed is integral, and as such, it is a 0/1-flow.

**17.8.0.13 Minimum weight bipartite matching**

**17.8.0.14 Minimum weight bipartite matching**

**Theorem 17.8.1.** Given a bipartite graph \( G \) and costs on the edges, one can compute the maximum cardinality minimum cost matching in polynomial time.

### 17.9 The transportation problem

**17.9.0.15 The transportation problem**

(A) Input: \( m \) facilities \( f_1, \ldots, f_m \).
(B) \( f_i \): contains \( x_i \) units of some commodity, for \( i = 1, \ldots, m \).
(C) $u_1, \ldots, u_n$ customers want to buy.
(D) $u_i$ would like to buy $d_i$ units, for $i = 1, \ldots, n$.
(E) costs $c_{ij}$ to send one unit of commodity from facility $i$ to costumer $j$.
(F) Q: how to supply the demands while minimizing the total cost.

17.9.0.16 Solution

(A) Create a bipartite graph with $f_1, \ldots, f_m$ on one side, and $u_1, \ldots, u_n$ on the other side.
(B) edge from ($f_i \to u_j$): costs $c_{ij}$.
(C) $s$: a source connected to $f_i$ with capacity $x_i$.
(D) edges from $u_j$ to the sink $t$, with capacity $d_i$.
(E) Compute min-cost flow pushing $\phi = \sum_j d_k$ units.
(F) Solution encodes the required optimal solution to the transportation problem.

Theorem 17.9.1. The transportation problem can be solved in polynomial time.

Sometime called the Earth mover distance.
Bibliography


