Graphs and Network Flows
IE411

Lecture 21

Dr. Ted Ralphs

Combinatorial Optimization and Network Flows

- In general, most combinatorial optimization and integer programming problems are difficult to solve.
- Some class of combinatorial optimization programs have direct, efficient *combinatorial algorithms*.
- Many of these are somehow related to network flows.
- For example, we will see the connections between all of these problems.
  - Shortest Path Problem
  - Maximum Flow Problem
  - Matching Problem
  - Minimum Spanning Tree Problem
  - Minimum Cut Problem
  - Assignment Problem
  - Postman Problem

IP Formulation of MST

Let $A(S)$ be the set of arcs contained in the subgraph of $G = (N, A)$ induced by the node set S. Let x_{ij} be a 0-1 variable that indicates whether we select arc (i, j) to be in the spanning tree.

$$\text{Minimize} \quad \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{(i,j)\in A} x_{ij} = n - 1 \tag{2}$$

$$\sum_{(i,j)\in A(S)} x_{ij} \leq |S| - 1 \quad \forall S \subseteq N \tag{3}$$

LP Relaxation

- For any LP, we can use reduced costs and complementary slackness optimality conditions to assess whether a given feasible solution if optimal.
- Notice that when $S = N$, constraint (3) is redundant.
- We associate a potential μ_S with every $S \subset N$.
- From the dual, we find that μ_N is free but $\mu_S \geq 0$.
- Then, the reduced cost of arc (i, j) is $c^{\mu}_{ij} = c_{ij} + \sum_{A(S):(i,j)\in A(S)} \mu_S$.

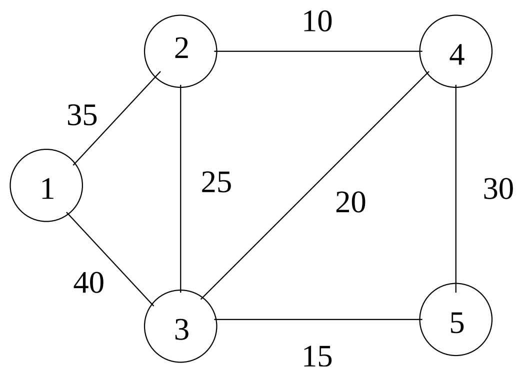
Results

Lemma 1. *A solution x of the MST problem is an optimal solution to the LP relaxation of the IP formulation if and only if we can find potentials μ_S defined on node sets S so that*

$$\begin{array}{ll} c^{\mu}_{ij} = 0 & \text{if } x_{ij} > 0 \\ c^{\mu}_{ij} \geq 0 & \text{if } x_{ij} = 0 \end{array}$$

Theorem 1. [13.9] *If x is the solution generated by Kruskal's Algorithm, then x solves both the integer program and its LP relaxation.*

Defining Potentials



- Set μ_N to the negative cost of the last arc added to the tree.
- Let $S(i,j)$ be the node component created by adding arc (i,j) to the tree.
- As the algorithm progresses, when it adds arc (p,q) to the tree, it combines node component $S(i,j)$ with one or more other nodes to define a larger component.
- Set $\mu_{S(i,j)} = c_{pq} - c_{ij}$.

Proving Optimality

- Check reduced cost of every arc. What do we find?

Theorem 2. [13.10] *The polyhedron defined by the LP relaxation of the packing formulation of the MST problem has integer extreme points.*

Matroids

- Notice the algorithms for finding minimum weight spanning trees depend on two properties:
  - Any acyclic subgraph with fewer than $n - 1$ edges can always be extended to a spanning tree.
  - If we have two acyclic subgraphs, one of which includes more edges, the smaller can be extended with an edge from the larger.
- We can generalize these properties to other combinatorial problems.

Submodular Functions

Definition 1. *A set function $f : 2^N \to \mathbb{R}$ is* submodular *if*

$$f(A) + f(B) \geq f(A \cap B) + f(A \cup B) \text{ for all } A, B \subseteq N.$$

Definition 2. *A set function f is* nondecreasing *if*

$$f(A) \leq f(B) \text{ for all } A, B \text{ with } A \subset B \subseteq N.$$

Proposition 1. *A set function f is nondecreasing and submodular if and only if*

$$f(A) \leq f(B) + \sum_{j \in A \setminus B} \left[f(B \cup \{j\}) - f(B) \right].$$

Submodular Polyhedra

- We now consider a *submodular polyhedron* defined by

$$\mathcal{P}(f) = \{x \in \mathbb{R}^n_+ \mid \sum_{j \in S} x_j \leq f(S) \text{ for } S \subseteq N\}.$$

- We are interested in solving the associated submodular optimization problem

$$\min\{cx : x \in \mathcal{P}(f)\}$$

- Consider the following greedy algorithm.
  - Order the variables so that $c_1 \leq c_2 \leq \cdots \leq c_r > 0 \leq c_{r+1} \leq \cdots \leq c_n$.
  - Set $x_i = f(S^i) - f(S^{i-1})$ for $i = 1, \ldots, r$ and $x_j = 0$ for $j > r$, where $S^i = \{1, \ldots, i\}$ for $i = 1, \ldots, r$ and $S^0 = \emptyset$.

The Greedy Algorithm and Matroids

- Surprisingly, the greedy algorithm solves all submodular optimization problems!
- Furthermore, when f is integer-valued, the greedy algorithm provides an integral solution.
- In the special case when $f(S \cup \{j\}) - f(S) \in \{0, 1\}$, we call f a *submodular rank function*.

Definition 3. *Given a submodular rank function r, a set $A \subseteq N$ is* independent *if $r(A) = |A|$. The pair $(N, \mathcal{F})$, where $\mathcal{F}$ is the set of independent sets is called a* matroid*.*

Properties of Matroids

- Given a matroid $(N, \mathcal{F})$.

  1. If A is an independent set and $B \subseteq A$, then B is an independent set.
  2. If A and B are independent sets with $|A| > |B|$, then there exists some $j \in A \setminus B$ such that $A \cup \{j\}$ is independent.
  3. Every maximal independent set has the same cardinality.

- Pairs $(N, \mathcal{F})$ with property 1 are *independence systems*.

- In fact, properties 1 and 2 are equivalent to our original definition and properties 2 and 3 are equivalent.

Common Matroids

- Matric Matroids
  - Ground set is the set of columns/rows of a matrix.
  - Independent sets are the sets of linearly independent rows/columns.
- Graphic Matroid
  - The ground set is the set of edges of a graph.
  - Independent sets are the sets of edges of the graph that do not form a cycle.
- Partition Matroid
  - Ground set is the union of m finite disjoint sets E_i for $i = 1, \ldots, r$.
  - Independent sets are sets formed by taking at most one element from each set E_i.

Generalizing from Spanning Trees

- Everything we learned from spanning trees can be generalized.
- All maximal independent sets have the same cardinality and are called *bases*.
- A spanning tree is a basis of the graphic matroid.
- A fundamental property of matroids is that it is always possible to find a basis of minimum weight using a *greedy algorithm*.
- In fact, an independence system is a matroid if and only if the greedy algorithm always finds a basis of minimum weight.

Red-Blue Algorithm for the Minimum Spanning Tree Problem

- Start with all edges uncolored.
- The Blue Rule
  - Find a cut with no BLUE edges.
  - Pick an edge of minimum weight and color it BLUE.
- The Red Rule
  - Find a cycle containing no RED edges.
  - Pick an uncolored edge of maximum weight and color it RED.
- Arbitrary application of the RED and BLUE rules result in a minimum weight spanning tree.

Generalizing to Matroids

- A *cycle* is a setwise minimal dependent set.
- A *cut* is a setwise maximal subset that intersects all maximal independent sets.
- The Red-Blue Algorithm can be applied to any matroid to find a basis of minimum weight.
- Matroids arise naturally in many contexts.
- We will see them later in the assignment problem context.

Matching Problems

- MST and Matching Problems are two combinatorial optimization problems that are defined over graphs with a weight associated with each arc.
- A *matching* in a graph is a set of edges with the property that no two share a common endpoint.
- Two well-known matching problems
  - Find a matching that has as many edges as possible.
  - Given weights for each edge, find a matching with the largest total weight.
- Matching algorithms use the concept of *augmentations*, but detecting and performing augmentations efficiently is more complicated here.

Definitions

- Given a graph $G = (N, A)$, the objective of the matching problem is to find a maximum matching M of G.
- We say that the matching is *complete* or *perfect* when the cardinality of M is $\lfloor \frac{|N|}{2} \rfloor$.
- Given a matching M in G, edges in M are called *matched* edges; others are *free* edges.
- Nodes that are not incident upon any matched edge are called *exposed*; remaining are *matched*.

Definitions (con't)

- A path $p = [u_1, u_2, \cdots, u_k]$ is called *alternating* if edges (u_1, u_2), (u_3, u_4) $\cdots$ are free and (u_2, u_3), (u_4, u_5) are matched.
- An alternating path p is called *augmenting* if both u_1 and u_k are exposed.

Augmenting a Matching

Lemma 2. *Let* P *be the set of edges on an augmenting path* $p = [u_1, u_2, \cdots, u_{2k}]$ *in a graph* G *with respect to the matching* M*. Then* $M' = M \oplus P$ *is a matching of cardinality* $|M| + 1$*.*

Proof:

Maximum Matching

Theorem 3. *A matching M in a graph G is maximum if and only if there is no augmenting path in G with respect to M.*

- Theorem characterizes maximum matchings in terms of augmenting paths.
- Like maximum flow, it suggests an algorithm: Start with any matching. Repeatedly discover augmenting paths.
- All known algorithms for matchings are based on this idea, but the details are quite involved...except for the case of bipartite graphs.

Bipartite Matching and Network Flow

- A graph $G = (N, A)$ is a bipartite graph if we can partition its node set into two subsets N_1 and N_2 so that for each arc $(i, j) \in A$ either (i) $i \in N_1$ and $j \in N_2$ or (ii) $i \in N_2$ and $j \in N_1$.
- We can reduce bipartite matching problem to maximum flow problem for simple networks and solve efficiently by making use of any algorithm for maximum flow.
- How can we convert the bipartite matching problem into an equivalent maximum flow problem?

Maximum Matching

Lemma 3. *The cardinality of the maximum matching in a bipartite graph equals the value of the maximum flow in the corresponding maximum flow network.*

Proof: 1. Given any matching M, we can construct a feasible flow in $N(G)$ with value $|M|$.

2. Given a maximum flow in $N(G)$, we can construct a matching with cardinality of the maximum flow value.

Notes on Maximum Cardinality Matching

- We can solve the bipartite matching problem in $O(\sqrt{n}\, m)$ time.
- Asymptotically fastest algorithm for bipartite matching.
- Non-Bipartite Matching
  - Reduction to maximum flow does not seem to carry over.
  - Augmenting path theorem holds for general graphs, so idea of repeatedly augmenting can be extended.
  - Finding augmenting paths is more difficult with non-bipartite structure.

Weighted Matching

- Given the graph $G = (N, A)$ with a corresponding weight w_{ij} for each arc (i, j), the objective of the weighted matching problem is to find a matching with the largest possible sum of weights.
- Assumptions
  - Underlying graph is complete.
  - Underlying graph has even number of nodes.
  - (For bipartite), underlying graph has node sets that are equal in size.
- Another name is *Assignment Problem*.

Assignment Problem

- Write the IP formulation for the Assignment Problem.
- Assignment Problem is a special case of which network flow problem?
- How can we solve the Assignment Problem?

Matching and the Postman Problem

- Given an undirected graph (G, A), the *postman problem* is to find the shortest *tour* that traverses each edge at least once.
- A graph for which it is possible to do this while traversing each edge exactly once is called *Eulerian*.
- An undirected graph is Eulerian if and only if every node has even degree.
- How can we use this fact to solve the postman problem?
- How can we extend this to directed graphs?

Back to Matroids: Matroid Intersection

- Consider two matroids M_1 and M_2 defined on the same ground set N and with the same rank k.
- M_1 and M_2 admit a common basis if and only if for every $S \subseteq N$, we have $r_1(A) + r_2(N \setminus S) \geq k$.
- A perfect matching in a bipartite graph is a common basis for two partition matroids, one associated with each set of nodes.
- From this, we can derive that G has a perfect matching if and only if it has a vertex cover of size less than k.
- Associated problems are that of finding the largest common independent set of M_1 and M_2 and that of finding the common independent set of minimum/maximum weight.
- These problems can be solved efficiently in general for two matroids (but not for three or more).

Back to Matroids: Max Flow and Min Cut Matroids

- The maximum flow problem can be viewed as a combinatorial problem as follows.

- Let us consider a network $G = (N, A)$ with associated cost vector c and capacities u, as usual.
  - We designate one edge e as a *special edge*.
  - We consider a collection F of (not necessarily distinct) cycles, each including the special edge.
  - Any collection in which no more than u_{ij} cycles include arc (i, j) for all $(i, j) \in A$ is called *feasible*.
  - Then the maximum flow problem is to find a feasible collection with the largest cardinality.
  - We can define an analog of the minimum cut problem similarly.

- This problem can be interpreted in terms of general matroids, but the max flow-min cut does not hold in this more general setting.