Advanced Algorithms March 1, 2022

Lecture 3: Matchings, Intro to Linear Programming

Notes by Ola Svensson[1]

In this lecture we do the following:

- We give the augmenting path algorithm for maximum cardinality (unweighted) bipartite matching.
- We introduce and define a powerful algorithmic method: linear programming.

For further reading about linear programming, I recommend the excellent text book *Understanding and Using Linear Programming* by Matousek and Gärtner available here:

https://link.springer.com/book/10.1007%2F978-3-540-30717-4

1 Algorithm for Maximum Cardinality Bipartite Matching

This section is largely taken from the lecture notes by Michel Goemans:

http://math.mit.edu/~goemans/18433S09/matching-notes.pdf

Recall the (unweighted) bipartite matching problem:

Definition 1 *Given a bipartite graph $G = (V, E)$, find a matching of maximum size.*

Recall that a matching $M \subseteq E$ is a subset of edges so that every vertex is incident to at most one edge of M, i.e., $|\{e \in M : v \in e\}| \leq 1$ for all $v \in V$.

Also recall that a path is a collection of edges $(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)$ where the v_i's are distinct vertices. We can simply represent a path as $v_0 - v_1 - v_2 - \ldots - v_k$.

Definition 2 (Alternating path) *An* alternating path *with respect to M is a path that alternates between edges in M and edges in $E \setminus M$.*

Definition 3 (Augmenting path) *An* augmenting path *with respect to M is an alternating path in which the first and last vertices are unmatched.*

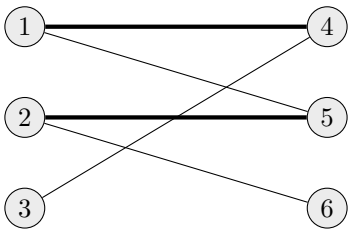


Figure 1: The edges $(3, 4), (4, 1), (1, 5), (5, 2), (2, 6)$ form an augmenting path

The definition of an augmenting path motivates the following algorithm:

[1] **Disclaimer:** These notes were written as notes for the lecturer. They have not been peer-reviewed and may contain inconsistent notation, typos, and omit citations of relevant works.

AugmentingPathAlgorithm(G):

Input: A bipartite graph $G = (V, E)$.
Output: A matching M of maximum cardinality.

1. Initialize $M = \emptyset$.
2. **while** exists an augmenting path P
3. update $M = M \Delta P \equiv (M \setminus P) \cup (P \setminus M)$.
4. **return** M.

Exercise 1 *Devise an efficient algorithm for finding an augmenting path P (if one exists). What is the total running time of the* AugmentingPathAlgorithm*?*

We now prove that AugmentingPathAlgorithm indeed finds a maximum matching.

Theorem 4 *A matching M is maximum if and only if there are no augmenting paths with respect to M.*

Proof (By contradiction)
($\Rightarrow$) Let P be some augmenting path with respect to M. Then $M' = M \Delta P$ is matching of greater cardinality than M. This contradicts the optimality of M.
($\Leftarrow$) If M is not maximum, let M^* be a maximum matching so that $|M^*| > |M|$. Let $Q = M \Delta M^*$. Then

- Q has more edges from M^* than from M (since $|M^*| > |M|$ implies that $|M^* \setminus M| > |M \setminus M^*|$).
- Each vertex is incident to at most one edge in $M \cap Q$ and one edge in $M^* \cap Q$.
- Thus Q is composed of cycles and paths that alternate between edges from M and M^*.
- Therefore there must be some path with more edges from M^* in it than from M. This path is an augmenting path with respect to M.

Hence, there must exist an augmenting path P with respect to M, which is a contradiction. ■

The above finishes a rather simple algorithm for *unweighted* bipartite matching. To devise an algorithm for the *weighted* version, it will be convenient to introduce a powerful algorithmic tool: linear programming. The concept of duality (that we will see in upcoming lectures) will then allow us to use (an adapted) version of the above algorithm for weighted graphs!

2 Linear Programming

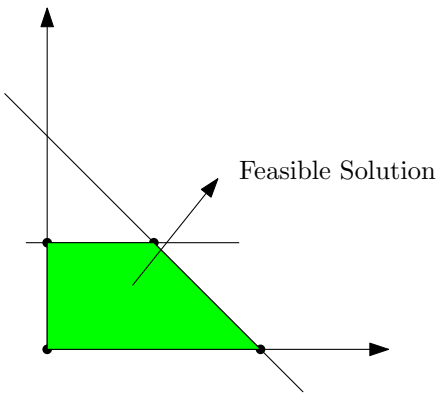
(Large parts of the following is taken from [1].)

A linear programing problem is the problem of finding values for variables that optimize a given linear objective function, subject to linear constraints.

Let us state an example of a linear program:

$$\begin{aligned} \text{Maximize} \quad & x + y \\ \text{Subject to} \quad & x + y \leq 2 \\ & y \leq 1 \\ & x, y \geq 0 \end{aligned}$$

The following figure shows the feasible area.



Definition 5 *A linear program (LP) is the problem of finding values for n variables $x_1, x_2, \ldots, x_n \in \mathbb{R}$ that minimize (or equivalently, maximize) a given linear objective function, subject to m linear constraints*

$$\begin{aligned}
\textbf{minimize:} \quad & \sum_{i=1}^{n} c_i x_i & \\
\textbf{Subject to:} \quad & \sum_{i} e_{i,j} x_i = b_j & \text{for } j = 1, \ldots, m_1 \\
& \sum_{i} d_{i,k} x_i \geq g_k & \text{for } k = 1, \ldots, m_2 \\
& \sum_{i} f_{i,p} x_i \leq l_p & \text{for } p = 1, \ldots, m_3
\end{aligned}$$

where $m_1 + m_2 + m_3 = m$.

2.1 Motivation

Linear Programming is a very powerful tool - it can be used in many applications, both industrial and theoretical such as:

- obtaining an optimal production plan to maximize a profit of a factory
- modeling network flow problems (what is maximum possible flow that doesn't exceed capacity of each connection)
- theory - many theoretical problems can be introduced as Integer Programs(LP in which $x_i \in \mathbb{Z}$) that can be relaxed to LP obtaining a good approximation of optimal result- we'll see such applications later in the course.

2.2 Some history

It was first formalized and applied to problems in economics in the 1930s by Leonid Kantorovich. Kantorivich's work was hidden behind the Iron Curtain (where it was largely ignored) and therefore unknown in the West. Linear programming was rediscovered and applied to shipping problems in the early 1940s by Tjalling Koopmans.
Simplex Method was published by George Dantzig in 1947: it is the first complete algorithm to solve linear programming problems.
The principle is the following: we start from an extreme point and then we look at its neighbors. If one of these is better we move to it and continue in the same way, else we stop. Once we stop, we can be sure that we have an optimal solution, since we're in a convex polytope. Even if it's usually extremely

fast, we know some bad examples where this method visits an exponential number of extreme points before to reach a solution, so this method does not always run in polynomial time.
Ellipsoid Method was studied by Leonid Khachiyan in the seventies.
This method is guaranteed to run in poly time (exactly $poly(n, m, \log(u))$ where u is the largest constant) but it is slow in practice.
We do a binary search for the optimal value of the objective function, so we can add the objective function as a constraint, and just need to decide if there exists a feasible point. We start by taking an ellipsoid surrounding the feasible area. We then check the center of the ellipsoid, if it's inside the area then we have our solution, else we identify a violated constraint, and cut our ellipsoid in two parts using this constraint, and construct a new ellipsoid around the part of the old ellipsoid in which the constraint is satisfied. We repeat this process until we find a feasible point or can be sure that the feasible area is empty.
Interior point Method developed by Narendra Karmarkar in 1984. In this method we move in the feasible region to find an OPT solution.

References

[1] Ashkan Norouzi-Fard, Christos Kalaitzis: *Scribes of Lecture 9 in Topics in TCS 2014. http://theory.epfl.ch/courses/topicstcs/Lecture9.pdf*