Lecture 7: nonbipartite matching and polyhedral theory basics

Dabeen Lee

Industrial and Systems Engineering, KAIST

2025 Winter Lecture Series on Combinatorial Optimization

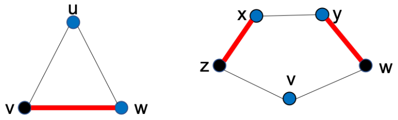
January 16, 2025

Outline

- Nonbipartite matching
- Edmonds' blossom algorithm
- Polyhedral theory basics
- From integer programming to linear programming

Nonbipartite matching

- In this lecture, we study the matching problem for general graphs that are not necessarily bipartite.
- In particular, a **nonbipartite** graph contains an odd cycle.
- Let us consider odd cycles.



- A triangle can take at most one edge for a matching while it requires at least two vertices for a vertex cover.
- An odd cycle of length five can take at most two edges for a matching but any vertex cover of it needs at least three vertices.

Nonbipartite matching

- In general, for an odd cycle of length $2k + 1$ for $k \geq 1$, the maximum size of a matching is k while the minimum size of a vertex cover is $k + 1$.
- This means that König's theorem does not hold for odd cycles.

Nonbipartite matching

- In general, for an odd cycle of length $2k + 1$ for $k \geq 1$, the maximum size of a matching is k while the minimum size of a vertex cover is $k + 1$.
- This means that König's theorem does not hold for odd cycles.
- For a bipartite graph, the augmenting path algorithm with the alternating tree procedure gives a maximum matching and a minimum vertex cover.
- **Can we just apply the augmenting path to a nonbipartite graph?**

Nonbipartite matching

Issue 1

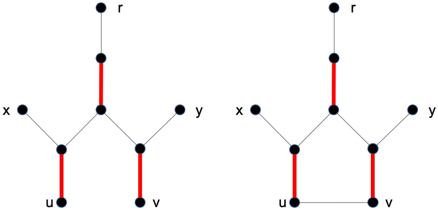


Figure: alternating trees for a bipartite graph (left) and a nonbipartite graph (right)

- Adding an edge between vertices u and v, we create an odd cycle and thus a nonbipartite graph.
- The alternating tree procedure that starts with r would return the same alternating tree, which consists of the ru-path and the rv-path.
- Deleting the alternating tree, we are left with vertices x and y.

Nonbipartite matching

Issue 1

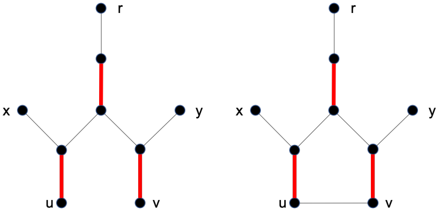


Figure: alternating trees for a bipartite graph (left) and a nonbipartite graph (right)

- Then the alternating tree procedure would conclude that there is no M-augmenting path.
- This is true for the bipartite graph on the left. What about the graph with the additional edge uv?
- In fact, uv gives rise to an M-augmenting path.
- This suggests that the alternating tree procedure is incomplete for the case of nonbipartite graphs.

Nonbipartite matching

Issue 2

- What goes wrong with the alternating tree procedure when applied to a nonbipartite graph?
- Suppose that the alternating tree procedure returns an alternating tree from an M-exposed vertex r, without finding an M-augmenting path.

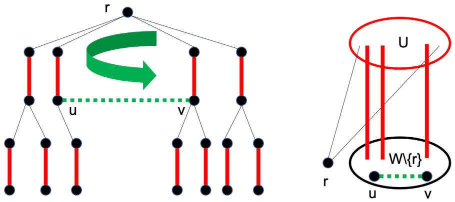
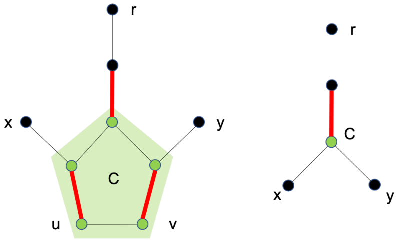


Figure: illustrating the issue with the alternating tree procedure for a nonbipartite graph

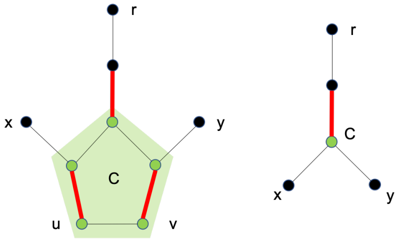
Edmonds' blossom algorithm

- Fortunately, there is a simple remedy for the alternating tree procedure due to Jack Edmonds.
- We saw that the analysis of the alternating tree procedure may fail when two vertices from an even level are adjacent.



- In such a case, the paths from an M-exposed vertex r to the two vertices and the edge between them would create an odd cycle.
- We refer to the odd cycle C as a **blossom** and the path from the M-exposed vertex to the blossom as a **stem**.
- Just for a reference, the structure with a blossom attached to a stem is called a **flower**.

Edmonds' blossom algorithm



- Edmonds' idea is that every time the alternating tree procedure detects a blossom, we may just **contract** it.
- We replace the vertices of the blossom C by a single vertex.
- Then connect the new vertex to the vertices that are adjacent to a vertex in the blossom.
- We denote by G/C the graph obtained from G after contracting the blossom C.

Edmonds' blossom algorithm

Lemma

Let $G = (V, E)$ be a graph, and let M be a matching. Let C be a blossom. Then there is an M-augmenting path in G if and only if there is an $(M \setminus C)$-augmenting path in G/C.

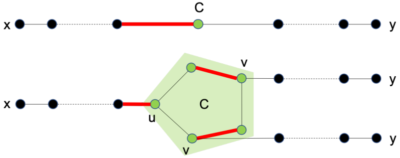


Figure: reconstructing an M-augmenting path from an $(M \setminus C)$-augmenting path

Edmonds' blossom algorithm

Edmonds' blossom algorithm

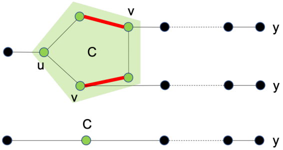


Figure: constructing an $(M' \setminus C)$-augmenting path from an M'-augmenting path

Edmonds' blossom algorithm

Modification of the augmenting path algorithm

1. Given a matching M in G, take an M-exposed vertex r and start growing an M-alternating tree from r.
2. If the alternating tree procedure detects two adjacents vertices that have an even distance from r, we detect a blossom C.
3. Contract the blossom C and continue the algorithm with G/C.

Edmonds' blossom algorithm

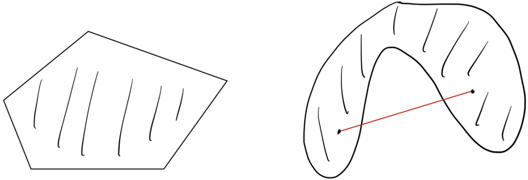
- This algorithm is referred to as the **blossom algorithm** for nonbipartite matching.
- Note that one application of the contraction operation reduces the number of vertices.
- Therefore, we may contract the graph at most $O(|V|)$ times.
- Recall that the alternating tree procedure takes $O(|E|)$ iterations as it is equivalent to enumerating the edges.
- Moreover, we know that we may augment the matching at most $O(|V|)$ times.
- Therefore, the time complexity of Edmonds' blossom algorithm is $O(|V|^2|E|)$.

Convex set

A set $X \subseteq \mathbb{R}^d$ is **convex** if it holds for any $u, v \in X$ and any $\lambda \in [0,1]$ that

$$\lambda u + (1-\lambda)v \in X.$$

In words, the line segment joining any two points is entirely contained the set.



Convex combination, convex hull

Given $v^1, \ldots, v^k \in \mathbb{R}^d$, a **convex combination** of $v^1, \ldots, v^k$ is

$$\lambda_1 v^1 + \cdots + \lambda_k v^k$$

where

$$\sum_{i=1}^{k} \lambda_i = 1 \quad \text{and} \quad \lambda_i \geq 0 \quad \text{for } i = 1, \ldots, k.$$

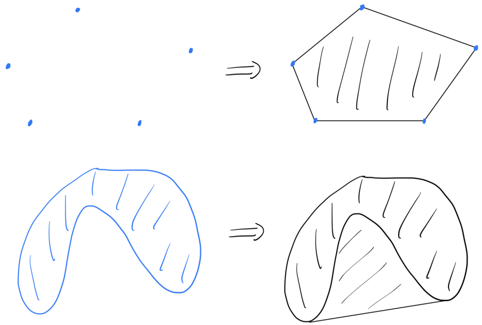
The **convex hull** of a set X, denoted $\text{conv}(X)$, is the set of all convex combinations of points in S.

Convex hull example

By definition,

$$\operatorname{conv}(X) = \left\{ \sum_{i=1}^{n} \lambda_i v^i \quad : \quad \begin{array}{l} n \in \mathbb{N},\ v^1, \ldots, v^n \in X, \\ \sum_{i=1}^{n} \lambda_i = 1,\ \lambda_1, \ldots, \lambda_n \geq 0 \end{array} \right\}.$$

Exercise: $\operatorname{conv}(X)$ is always convex, regardless of X.



Polyhedra

A set $P \subseteq \mathbb{R}^d$ is a **polyhedron** if it is defined by a **finite** number of linear inequalities, i.e.

$$P = \{x \in \mathbb{R}^d : \; Ax \leq b\}.$$

Hence, a polyhedron is a finite intersection of half-spaces.

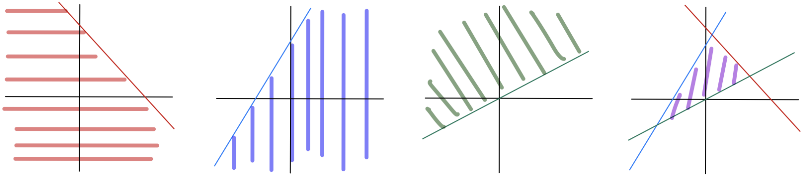


Figure: Polyhedron defined by three inequalities

Polyhedra

- A polyhedron is **rational** if it is defined by a system of linear inequalities where all coefficients and right-hand sides are rational.

- A set $P \subseteq \mathbb{R}^d$ is a **polytope** if it is a polyhedron and bounded, i.e., $P \subseteq [-M, M]^d$ for some sufficiently large $M > 0$.

Theorem (Minkowski-Weyl theorem for polytopes)

A set $P \subseteq \mathbb{R}^d$ is a polytope if and only if

$$P = \mathrm{conv}(v^1, \dots, v^p)$$

for some vectors $v^1, \dots, v^p$.

Integer programming formulation

A **mixed integer linear program (MIP or MILP)** has the form

$$\begin{aligned} \max \quad & c^\top x + h^\top y \\ \text{s.t.} \quad & Ax + Gy \leq b, \\ & x \in \mathbb{Z}^d, \ y \in \mathbb{R}^p \end{aligned} \tag{MILP}$$

where A, b, c, G, h are vectors/matrices of appropriate dimension with rational entries.

We refer to (MILP) simplly as an **integer program**.

Replacing $x \in \mathbb{Z}^d$ by $x \in \mathbb{R}^d$, we obtain the **linear programming (LP) relaxation**.

Feasible region

The **feasible region** or the **solution set** of (MILP) is the set of solutions satisfying the linear constraints and the integrality constraints:

$$S = \left\{ (x, y) \in \mathbb{Z}^d \times \mathbb{R}^p : \; Ax + Gy \leq b \right\}.$$

A set of the form S is often referred to as a **mixed integer linear set**.

The feasible region of the LP relaxation is given by

$$P = \left\{ (x, y) \in \mathbb{R}^d \times \mathbb{R}^p : \; Ax + Gy \leq b \right\},$$

which gives rise to a **relaxation** of S.

Convex hull of a mixed integer linear set

Take the feasible region of (MILP)

$$S = \left\{(x, y) \in \mathbb{Z}^d \times \mathbb{R}^p : \; Ax + Gy \leq b\right\},$$

whose convex hull is given by

$$\operatorname{conv}(S) = \operatorname{conv}\left(\left\{(x, y) \in \mathbb{Z}^d \times \mathbb{R}^p : \; Ax + Gy \leq b\right\}\right).$$

Meyer's theorem: there exists a system of rational linear inequalities $A'x + G'y \leq b'$ such that

$$\operatorname{conv}(S) = \left\{(x, y) \in \mathbb{R}^d \times \mathbb{R}^p : \; A'x + G'y \leq b'\right\}.$$

Reduction to linear programming

Lemma

(MILP) *whose feasible region is given by* $S \subseteq \mathbb{Z}^d \times \mathbb{R}^p$ *satisfies*

$$\max\left\{c^\top x + h^\top y : \ (\boldsymbol{x}, \boldsymbol{y}) \in \boldsymbol{S}\right\} = \max\left\{c^\top x + h^\top y : \ (\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{conv}(\boldsymbol{S})\right\}.$$

Moreover, the supremum of $c^\top x + h^\top y$ *is attained over* S *if and only if it is attained over* $\mathrm{conv}(S)$*.*

Reduction to linear programming

Reduction to linear programming

Consequently, (MILP) is equivalent to the linear program

$$\max\left\{c^\top x + h^\top y : \ A'x + G'y \le b'\right\}$$

for some rational A', G', b'.

Does this contradict our earlier discusstion that integer programming is NP-hard while linear programming is in class P?

The answer is NO.

The reason is that Meyer's theorem shows the **existence** of such a linear system.

In fact, computing a linear system that gives us the convex hull of S is in general hard.