Discrete Optimization - ADM2

Lecturer
Prof. Dr. Tom McCormick

Teaching Assistants
Svenja Griesbach and Sarah Morell

Student Assistant
Christos Pavlidis

Notes
Simon (Morpheus) Cyrani

Version
git: a354169-*
compiled: Wednesday 20th July, 2022 20:06

Abstract

The following lecture notes are my personal (and therefore unofficial) write-up for 'Discrete Optimization' aka 'ADM II', which took place in summer semester 2022 at Technische Universität Berlin. I do not guarantee correctness, completeness, or anything else. Importantly, note that I willfully changed some specific notations, reordered some material, and left out parts that I didn't found worth typing down.

If you miss something, feel free to contribute in the repository!

Contents

Summary of lectures

Part I

Lecture notes

Lecture 1
Tu 19 Apr 2022

1 Introduction

In ADM1 we often already worked with Integer Programming and just assumed everything is fine. In this course however, we want to find out how Integer Programming actually works, why it is generally "hard", and under which circumstances it is "easy".

Definition 1.1 (Flavors of IP)**.** First of all, we want to define different variants of **Integer Programming**:

- **Pure Integer Programming** assumes *all* variables are integer.
- **Mixed Integer Programming** also allows some variables to be real.
- **Binary Integer Programming**, also called 0-1-Integer-Programming, restricts the integer variables to $\mathbb{B} \coloneqq \{0, 1\}$. Mixed variants are also possible.

Question 1.2. Why is IP harder than LP? Naively, one would assume this should *not* be the case, because our search space is smaller (at most countably infinite)!

Let's solve the IP in ADM1-style - suppose

$$\begin{aligned} \max_x \quad & w^T x \\ \text{s.t.} \quad & x \in Q = \{x \in \mathbb{Z}^n : Ax \leq b\} \end{aligned}$$

For simplicity, assume Q is bounded. Then the set of feasible points in Q is finite, and therefore we can consider the polytope

$$\text{conv}(Q) = \{x \in \mathbb{R}^n \mid A'x \leq b'\}$$

for suitable A' and b'. Notice all vertices must be in Q and thus are integral. As a consequence, it is sufficient to solve the LP

$$\begin{aligned} \max_x \quad & w^T x \\ \text{s.t.} \quad & A'x \leq b'. \end{aligned}$$

Warning. Computing A', b' is non-trivial! In fact, computing the **integer hull** $\text{conv}(Q)$ is what makes IP hard.

1.1 IP is "hard"

We can gather more evidence that IP must be hard.

Theorem 1.3. Every logical statement can be expressed with integer programming.

Proof. It suffices to show that for variables $x_1, x_2, y \in \mathbb{B}$ we can find IPs that model $\wedge, \vee, \neg$.

- $y = x_1 \wedge x_2$ can be modeled as

$$\begin{aligned} y &\leq x_1 \\ y &\leq x_2 \\ y &\geq x_1 + x_2 - 1 \end{aligned}$$

- $y \coloneqq x_1 \vee x_2$ can be modeled as

$$\begin{aligned} y &\geq x_1 \\ y &\geq x_2 \\ y &\leq x_1 + x_2 \end{aligned}$$

- $y \coloneqq \neg x$ can be modeled as

$$y = 1 - x$$

□

Theorem 1.4. IP can model (finite) unions of polyhedra, e.g. non-convex problems.

Proof. Consider

$$P \coloneqq \bigcup_{i=1}^{k} \underbrace{\{x \mid A_i x \leq b_i\}}_{P_i}$$

and introduce auxiliary binary variables

$$y_i \coloneqq \begin{cases} 1, & \text{if } x \in P_i, \\ 0, & \text{if we don't care.} \end{cases}$$

Now, assume $M \in \mathbb{R}$ large enough (**Big-M method**), such that

$$P_i \subseteq \overline{P} \coloneqq \{x \mid A_i x \leq b_i + M \cdot \mathbf{1}\}$$

for all i. Thus, we can construct following IP:

$$\begin{aligned} \min_x \quad & w^T x \\ \text{s.t.} \quad & A_i x \leq b_i + (1 - y_i) M \cdot \mathbf{1}, \quad i = 1, \ldots, k, \\ & \sum_i y_i = 1 \end{aligned}$$

This forces exactly one y_i to 1, resulting that $x \in P_i$ and $x \in \overline{P}$ is sufficient. We call this method **righthandside Big-M**, short RHS. Further information can be found in [NW99, Ch. 1]. □

Note. It's also possible to handle M as a symbolic value, but this makes other things more complicated.

Problem. Finding M big enough can make LP hard to solve, because of matrix-inversions getting numerically unstable.

Alternative proof of Theorem 1.4. Assume that we can bound each $x \in P_i$ by u_i, i.e. $x \leq u_i$ (note that this is basically a hidden big-M!). Now we can disaggregate x for each P_i as its own private $x_i \in \mathbb{R}^k$, and analogously introduce $y_i \in \mathbb{B}$ to restrict ourselves to one polyhedron:

$$\begin{aligned} \min_x \quad & w^T x \\ \text{s.t.} \quad & A_i x_i \leq y_i b_i, \quad i = 1, \ldots, k, \\ & x_i \leq y_i u_i, \quad i = 1, \ldots, k, \\ & \sum_{i=1}^{n} y_i = 1, \\ & \sum_{i=1}^{n} x_i = x, \\ & x, x_i \geq 0 \end{aligned}$$

Again, exactly one y_i is equal to 1, forcing the other x_j to be equal to 0, and thus setting x to x_i. We call this method **upper bound on x Big-M**, short UBX. □

Remark 1.5. In general, it cannot be said if RHS or UBX is better. Even though RHS only introduces n new variables as opposed to UBX's nk variables, UBX's disaggregated formulation often is *tighter* in the sense that the LP relaxation is closer to the integer hull.

Lecture 2
Th 21 Apr 2022

Theorem 1.6. IP can approximate any objective function infinitely good.

Proof. First, consider a **piecewise linear** objective function f with (not necessarily equidistant) breakpoints $a_1, \dots, a_k$.

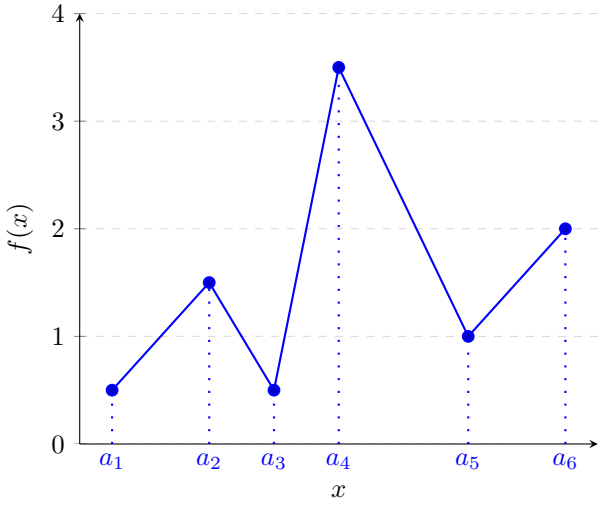


Figure 1: A piecewise linear function

Defining the intervals $I_i := [a_i, a_{i+1}]$ for each segment of f, we can introduce binary variables y_i such that

$$y_i = \begin{cases} 1, & \text{if } x \in I_i, \\ 0, & \text{otherwise} \end{cases}$$

Note that for $x \in I_i$, x is a convex combination of a_i, a_{i+1}. Therefore, we can express x as a linear combination of all breakpoints a_i, with the additional constraint that all except 2 scalars must be 0. By linearity, this also holds for $f(x)$. Translating into IP:

$$\begin{aligned} \min_{\lambda} \quad & \sum_i \lambda_i f(a_i) \\ \text{s.t.} \quad & \lambda_1 \le y_1, \\ & \lambda_k \le y_{k-1}, \\ & \lambda_i \le y_{i-1} + y_i, \quad i = 2, \dots, k-1, \\ & \sum_i y_i = 1 \end{aligned}$$

One can show this already suffices to model any cost function: For suitable choices of breakpoints we can approximate any function by piecewise linear functions. Details can be found in [AMO93, Ch. 14] or [NW99, Ch. 1]. □

Conclusion 1.7. Summarizing, following facts that hold for IP, but not LP, deliver an intuition why IP should be hard:

1. Consider a polyhedron $P = \{x \in \mathbb{R}^n \mid Ax = b\}$ and its integer hull Q. Even

though P can be "smooth" (e.g. cube), its integer hull can look more like a "disco ball" with many facets.

2. In real life, many problems can be modeled with decision variables. IP can handle this, see Theorem 1.3.
3. Additionally, IP also can handle non-convex problems, see Theorem 1.4 and Theorem 1.6.

1.2 Representation of IPs

For a given IP, a set Q of feasible points can be formulated by many polyhedra P.

Example 1.8. Consider

$$Q \coloneqq \{0000, 1000, 0100, 0010, 0110, 0101, 0011\} \subseteq \mathbb{Z}^4.$$

Then we can give following representations P_i such that P_i is an integer hull of Q:

$$\begin{aligned}
P_1 &= \{x \in \mathbb{R}^4 \mid 93x_1 + 49x_2 + 37x_3 + 29x_4 \leq 111\} \\
P_2 &= \{x \in \mathbb{R}^4 \mid 2x_1 + x_2 + x_3 + x_4 \leq 2\} \\
P_3 &= \{x \in \mathbb{R}^4 \mid 2x_1 + x_2 + x_3 + x_4 \leq 2, \\
&\qquad\qquad x_1 + x_2 \leq 1, \\
&\qquad\qquad x_1 + x_3 \leq 1, \\
&\qquad\qquad x_1 + x_4 \leq 1\}
\end{aligned}$$

One can show that $P_3 \subsetneq P_2 \subsetneq P_1$.

Example 1.9. A real-life example is the problem of placing facilities. Given n stores and m warehouses, decide which warehouse should be build at all, and which should deliver which store (for some cost function). Let $y_i \in \mathbb{B}$ denote if warehouse i should be opened, and x_{ij} if warehouse i should serve store j. Then

$$\begin{aligned}
P_1 &\coloneqq \{x \mid \forall i : \sum_j x_{ij} \leq m y_i\}, \\
P_2 &\coloneqq \{x \mid \quad \forall i, j : x_{ij} \leq y_i\}
\end{aligned}$$

both represent the condition to only serve stores from warehouses that are opened. Notice that $P_2 \subsetneq P_1$ is tighter, but has $n \cdot m$ instead of n constraints.

1.3 Complexity

In order to define hardness, it is useful to define complexity first. We can use **big-$\mathcal{O}$** for this. During the rest of the lecture, we had a recap on this. For details, refer to canonical sources.

Lecture 3
Tu 26 Apr 2022

2 Hardness

Prior, we only gave intuition why IP is "harder" than LP. In order to analyze IP more thoroughly, we now want to work towards a formal definition of hardness of problems. Basically, there are two types of problems for now:

Definition 2.1 (Problem types)**.** We differentiate between

- **decision problems**, which can be answered by *Yes* or *No* only, and
- **optimization problems**, which seeks for a numerical value minimizing a certain (cost) function.

For the beginning, we can cheat and restrict ourselves to decision problems.

Example 2.2. Possible decision problems could be:

1. Does there exist a Hamiltonian cycle?
2. Is the LP feasible?

Question 2.3. How do we model an optimization problem as an decision problem?

Answer. We can simply introduce a parameter z which we use as a bound for the value we want to optimize.

Example 2.4. Possible reformulations of optimization problems are therefore:

1. Does there exist a feasible x with $c^T x \leq z$?
2. Does there exist a spanning tree with cost less than z?
3. Is there a clique with size less than z?

Definition 2.5. A **clique** C is a subset of nodes V of a graph $G = (V, E)$ s.t. for all $i, j \in C$ it must hold true that $(i, j) \in E$.

Theorem 2.6. When we model an optimization problem as a decision problem, then there exist a oracle-polynomial way to solve the optimization problem using the decision problem as an oracle.

Algorithm 1: Oracle-polynomial algorithm for max-clique

Use binary search to find z^* in $\mathcal{O}(\log n)$
$G' \leftarrow G = (V, E)$
for $i = 1, ..., n$ **do**
  $G'' \leftarrow G'$, but remove all edges incident to node i
  if *Call of decision oracle on* G'', z^* *is true* **then**
    $G' \leftarrow G''$
  end
end

Theorem 2.7. Final $\overline{G} \coloneqq G'$ is a max-clique.

Proof. The size of a max clique in G' never goes below z^*. Therefore, there exists a clique $C \subseteq \overline{G}$ with $|C| = z^*$. Suppose $\overline{G}$ has more than z^* nodes. Then $i \in \overline{G} \setminus C$. But then the algorithm would have deleted this node! □

Corollary 2.8. If we have an optimal oracle, then one can solve decision version in oracle-polynomial time using the optimal oracle.

Conclusion 2.9. Optimization and decision version differ only by a polynomial factor of complexity. Therefore, either both are easy or both are hard.

Definition 2.10 (Certificate)**.** Given an instance of any problem with size n, a **certificate** is a binary-encoded string that is generated by some algorithm specific to the problem, taking the instance as input. We say the certificate is a **succinct certificate**, if its *length* is polynomial in the input size n.

Definition 2.11 (**NP**)**.** We say a (decision) problem P lies in **NP**, if for all Yes-instances I there exists a succinct certificate C and a certificate checking algorithm A that confirms $A(I, C)$ in polynomial time.

Theorem 2.12. Max-clique lies in **NP**

Proof. We use our clique C directly as the certificate.

Algorithm 2: Certificate checking for max-clique

```
if |C| < z then
    return NO
end
else
    for i, j ∈ C do
        if (i, j) ∉ E then
            return NO
        end
    end
end
return YES
```

□

Remark 2.13. Note that we don't care for No-instances! In order to verify them we would need to list all $\binom{n}{z}$ subsets (for max-clique), which is *not* polynomial.

Theorem 2.14. P ⊆ NP

Proof. Let $P \in \mathbf{P}$. Then there exists a polynomial algorithm A. Record the steps of A on an instance I and use this as a polynomial certificate. □

Theorem 2.15. LP ∈ **NP**, using decision variant if there is any feasible x.

Proof. If feasible, there exists a basic feasible solution x^*. We verify by checking $Bx^* = b$. One can show that x^* has polynomial bits. □

Let's also have a look at the canonical **NP** problem:

Definition 2.16 (Satisfiability problem, SAT)**.** Consider n logical variables $v_1, ..., v_n$, allowing also the negated literals $\overline{v_i}$. Additionally, we have m clauses $C_1, ..., C_m$, which are subsets of the literals. Determining if there is an assignment such that the overall clause is true (i.e. each subclause has at least one true literal) is known as the **satisfiability problem**, for short SAT.

Example 2.17. A few examples:

1. $(v_1 \vee v_2 \vee v_3) \wedge (\overline{v_1} \vee \overline{v_2} \vee \overline{v_3})$
   This instance is true for $v = (110)$.

2. $(v_1 \vee v_2) \wedge (\overline{v_1} \vee v_2) \wedge (\overline{v_2} \vee v_3) \wedge (\overline{v_3} \vee \overline{v_4})$
   One can check that this instance is always false.

Theorem 2.18. SAT $\in$ **NP**

Proof. The satisfiability truth assignment is a succinct certificate. □

Theorem 2.19 (Cook)**.** If $P \in$ **NP**, then P has an oracle-polynomial algorithm with SAT as an oracle.

Proof. Suppose $P \in$ **NP**, then P has a non-deterministic Turing Machine with polynomial size. Encode the Turing Machine as a logical formula such that it is true iff P is a Yes-instance. □

Lecture 4
Th 28 Apr 2022

We remind ourselves that IP can formulate logic, and therefore can encode SAT formulas.

Example 2.20. Translating from Example 2.17:

1.
$$\begin{aligned} x_1 + x_2 + x_3 &\geq 1 \\ (1 - x_1) + (1 - x_2) + (1 - x_3) &\geq 1 \\ x &\in \mathbb{B}^3 \end{aligned}$$

2.
$$\begin{aligned} x_1 + x_2 &\geq 1 \\ (1 - x_1) + x_2 &\geq 1 \\ (1 - x_2) + x_3 &\geq 1 \\ (1 - x_2) + (1 - x_3) &\geq 1 \\ x &\in \mathbb{B}^3 \end{aligned}$$

Definition 2.21 (Reduction)**.** We say $P \propto Q$ ("P **reduces to** Q") if there exists a polynomial algorithm A such that

1. for all instances $I \in P$, $A(I)$ is element of Q,
2. I is **Yes-preserving**, e.g. I is Yes-instance of P iff $A(I)$ is Yes-instance of Q.

Definition 2.22 (**NP**-complete)**.** A problem P is **NP-complete**, if

1. $P \in \mathbf{NP}$, and
2. for all $Q \in \mathbf{NP}$ it holds that $Q \propto P$.

We call the set of all **NP**-complete problems **NPC**.

Corollary 2.23. Using our new definition, it follows immediately from Theorem 2.19 that SAT $\in$ **NPC**.

Proof Strategy. In order to show a problem P is **NP**-complete we first describe a way to construct a succinct certificate, and state an algorithm that describes how we use the certificate to verify a Yes-instance is indeed a Yes-instance.

After that, we find a suitable problem Q, which is known to be **NP**-complete, and try to proof $Q \propto P$. We do this by converting each instance of Q into an instance of P in polynomial time, and verify that the conversion is Yes-preserving.

Theorem 2.24. SAT is as hard as 0-1-IP

Proof. We know 0-1-IP $\in$ **NP**, and therefore 0-1-IP $\propto$ SAT. It remains to show SAT $\propto$ 0-1-IP: Let $I \in$ SAT with clauses $c_j = l_1, ..., l_k$. We convert each clause to the inequality $l_1 + ... + l_k \geq 1$ for binary l. As shown in Theorem 1.3, this encodes exactly the logic formula. □

Definition 2.25 (3SAT)**.** We define 3SAT as a variant of SAT where we only allow clauses with exactly 3 literals, e.g. $|C_j| = 3$.

Theorem 2.26. 3SAT $\in$ **NPC**

Proof. 3SAT $\in$ **NP** follows directly from SAT $\in$ **NP**. It remains to show SAT $\propto$ 3SAT. Consider clause $C_j = (l_1 \vee ... \vee l_k)$ for $k > 3$. Add $k - 3$ new variables $y_{2,j}, ..., y_{k-2,j}$ and replace C_j with

$$(l_1 \vee l_2 \vee y_{2,j}) \wedge (\overline{y}_{2,j} \vee l_3 \vee y_{3,j}) \wedge ... \wedge (\overline{y}_{k-2,j} \vee l_{k-1} \vee l_k)$$

One can figure out via proof tables and induction that this is indeed Yes-preserving. □

Definition 2.27 (Node cover, NC)**.** Given graph $G = (N, E)$, we say $C \subseteq N$ is a **node cover** if for every edge in E at least one of the nodes is in N. We define NC as the decision problem if there is a node cover of at most size z.

Theorem 2.28. NC $\in$ **NP**C

Proof. We can easily check if for a given C, it is indeed a node cover in polynomial time. Therefore NC $\in$ **NP**C. We want to reduce from 3SAT:

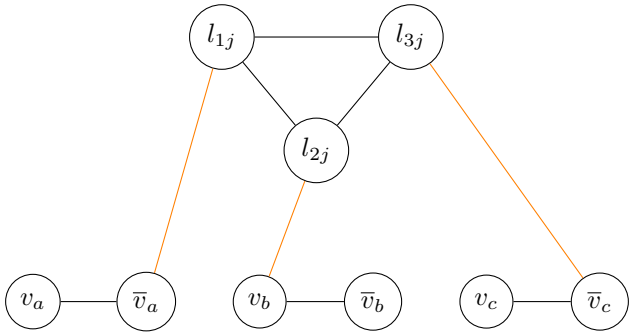


Figure 2: Schema of how to use triangle "gadgets" for a single clause C_j

Consider an instance of 3SAT and construct a graph as shown in *Figure 2*, e.g. for each variable v_i construct an edge between nodes v_i and $\overline{v}_i$, and for each clause C_j construct a triangle l_{1j}, l_{2j}, l_{3j}. Now, connect each node of the triangle with the corresponding literal in the clause (the orange edges). Using this construction, we want to proove that there is a node cover of size $n + 2m$ iff the 3SAT instance is valid.

Suppose the 3SAT instance is feasible. We use the n nodes of the feasible labeling corresponding to the literals. Now, because the labelling is valid, at least one orange edge per triangle must be covered, by construction. Therefore, we can choose 2 additional nodes per triangle that cover the triangle and the remaining orange edges.

On the other side, suppose there is a node cover of size (at most) $n+2m$. Analoguous, each triangle must have at least 2 chosen nodes to cover each edge, and each literal-pair at least 1 node, meaning our bounds must actually be exact to not overshoot $n + 2m$. Therefore, the node cover represents a valid truth assignment, which is also a valid labelling, because each clause has a remaining orange edge, which is covered by one of the literals.

Therefore, our reduction is Yes-preserving. □

Remark 2.29. NC in bipartite graphs is in **P**.

Definition 2.30 (Independent set, IS)**.** For a graph $G = (N, E)$ we call $S \subseteq N$ a **independent set** (or **stable set**) if no edge has both nodes in S. The decision problem, called IS, if there is a independent set of size at least z.

Theorem 2.31. IS $\in$ **NP**C

Proof. IS $\in$ **NP** trivial. We can also easily show that C is a node cover iff $N \setminus C$ is stable. □

Theorem 2.32. CLIQUE $\in$ **NP**C

Proof. CLIQUE $\in$ **NP** trivial. We can also easily show that C is a clique in $G = (N, E)$ iff C is stable in $(N, \overline{E})$. □

Definition 2.33 (Partition, PART)**.** Given $a_1, ..., a_n \in \mathbb{Z}^+$. The decision problem if there is a set $S \subseteq \{1, ..., n\}$ such that

$$\sum_{i \in S} a_i = \sum_{i \notin S} a_i$$

is called **partition problem**, PART.

Theorem 2.34. PART $\in$ **NP**C

Proof Sketch. We can show [KV18, Ch. 15.5]:

$$\text{SAT} \propto \text{3-dim match} \propto \text{subset sum} \propto \text{PART}$$

Remark 2.35. Still, PART has a pseudopolynomial algorithm using dynamic programming.

Definition 2.36. If a (numerical) problem is only **NP**-complete if it depends on the size of the numbers (e.g. polynomial in the unary bit model), we call it **weakly NP-complete**. Otherwise, we call it **strongly NP-complete**.

Definition 2.37 (3-partition, 3PART)**.** Given the numbers $a_1, ..., a_{3k} \in \mathbb{Z}$. The problem, if we can partition these numbers in sets of 3 such that every set has the same value, is called **3-Partition**, or 3PART.

Theorem 2.38. 3PART is strongly **NP**-complete.

Remark 2.39. Only weakly **NP**-complete problems could have pseudopolynomial algorithms (except $\mathbf{P} = \mathbf{NP}$).

Lecture 5
Tu 03 May 2022

Definition 2.40 (**NP**-hard)**.** Consider an optimization problem P. Formally, we can't have $P \in \mathbf{NP}$, but because of Theorem 2.6 we can introduce the notion to call P **NP-hard**, if its decision variant is **NP**-complete.

Definition 2.41 (co−**NP**)**.** We say $P \in$ co−**NP**, if we have a succinct certificate for verifying No-instances.

Example 2.42. Given a matrix A. We call it totally unimodular, if every square submatrix has determinant 0 or 1. Deciding if A is totally unimodular is in co−**NP**, because giving a failing submatrix as a succinct certificate is easy.

Theorem 2.43. The decision version of LP is in co−**NP**.

Proof. The answer to the decision problem is No iff

1. the system is infeasible, or
2. the system is feasible, but the optimal cost is larger than the z we want.

Because both can be decided with the tools we have in polynomial time, LP is indeed in co−**NP**. □

Definition 2.44 (co−**NP**-complete)**.** Analoguous to Definition 2.22, we can also define co−**NP-completeness**, or co−**NPC**, for the "most difficult" problem in co−**NP**.

Remark 2.45. It holds that co−$\mathbf{NPC} \cap \mathbf{NPC} = \emptyset$, except $\mathbf{P} = \mathbf{NP}$. See [KV18].

Open Question 2.46. Is $\mathbf{P} = \mathbf{NP}$?

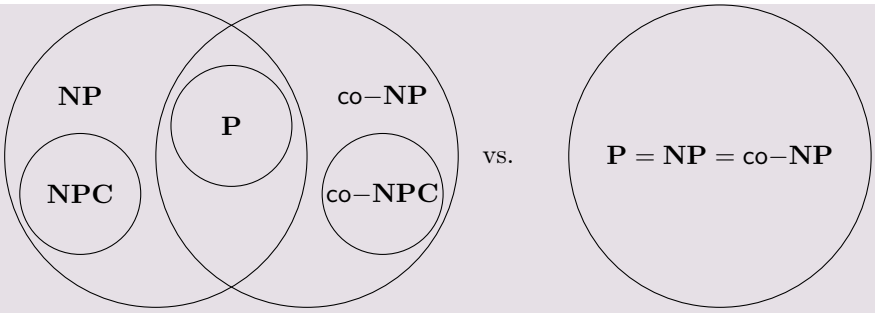


Figure 3: Landscape of problems depending if **P** = **NP**

3 Complexity differences between IP and LP

Some 0-1-IPs are easy, such as bipartite matching, or the assignment problem, even though IP $\in$ **NPC**. In the following section, we want to give intuition, why there are different complexities in IP.

3.1 Optimization vs. Separation

Recall (Ellipsoid Algorithm)**.** As discussed in ADM1, the **ellipsoid method** can be used to determine feasibility of LPs in polynomial time. One could also call the ellipsoid method a fancy "n-dimensional binary search". A rough draft how the algorithm worked:

1. Reduce optimization version to decision version and introduce bound $L = mn \cdot \log(\text{max abs. data})$
2. Volume-based argument: If the LP is feasible, there is a solution within the centered cube with length 2^L.
3. Volume is zero: Perturb the problem to $Ax \leq b + 2^{-L}$, which maintains feasibility, but now has positive volume.

For details, refer to the slides from ADM1.

Definition 3.1. Given a polyhedron P with an associated cost function. We want to introduce two distinct problem types:

- The **optimization problem** OPT denotes the problem of finding an optimal $x^* \in P$.
- The **separation problem** SEP denotes the problem of deciding if $x \in P$, or else stating a separating hyperplane.

Remark 3.2. The key step of the ellipsoid method is to find a hyperplane that

separates the current x from the considered polyhedron. Especially, if $\mathsf{SEP} \in \mathbf{P}$, then $\mathsf{OPT} \in \mathbf{P}$. The converse can also be proven.

Definition 3.3. Let $\mathcal{Q}$ be the class of full-dimensional polytopes with 0 inside. We define the **polar** Q^* for $Q \in \mathcal{Q}$ as

$$Q^* := \{y \in \mathbb{R}^n \mid \forall x \in Q \colon y^T x \leq 1\}.$$

Theorem 3.4. Considering this class of polytopes, one can proove [KV18, Ch. 4, Thm. 4.22]:

1. Q^* is also a full-dimensional polytope with 0.
2. $(Q^*)^* = Q$
3. v is a vertex of Q iff $v^T y \leq 1$ is a facet of Q^*

Theorem 3.5. Suppose we can solve OPT on $\mathcal{Q} \in \mathbf{P}$ with algorithm A. Then we can use A as an oracle to solve SEP on $\mathcal{Q}^*$ in polynomial time.

Proof. Suppose $Q^* \in \mathcal{Q}^*$, and we want to separate y^0. Use A to solve OPT on Q with objective function $\max (y^0)^T x$ to get $x^* \in Q$. This yields two cases:

- $(y^0)^T x^* \leq 1$: Then this holds for all $x \in Q$, and thus $y^0 \in Q^*$ by definition.
- $(y^0)^T x^* > 1$: Consider hyperplane $(x^*)^T y$. From $x^* \in Q$ it follows that for all $y \in Q^*$, that $(x^*)^T y \leq 1$, but $(x^*)^T y^0 > 1$. Thus, we found a separating hyperplane.

□

Theorem 3.6. $\mathsf{SEP} \in \mathbf{P}$ for $\mathcal{Q}$ iff $\mathsf{OPT} \in \mathbf{P}$ for $\mathcal{Q}$

Proof. Using what we proven so far:

$$\begin{aligned} \mathsf{OPT} \in \mathbf{P} \text{ for } \mathcal{Q} &\overset{3.5}{\Longrightarrow} \mathsf{SEP} \in \mathbf{P} \text{ for } \mathcal{Q}^* \\ \overset{\text{Ellips.}}{\Longrightarrow} \mathsf{OPT} \in \mathbf{P} \text{ for } \mathcal{Q}^* &\overset{3.5}{\Longrightarrow} \mathsf{SEP} \in \mathbf{P} \text{ for } (\mathcal{Q}^*)^* = \mathcal{Q} \\ \overset{\text{Ellips.}}{\Longrightarrow} \mathsf{OPT} \in \mathbf{P} \text{ for } \mathcal{Q} & \end{aligned}$$

□

Conclusion 3.7. There is a close relationship between OPT and SEP:

$$\underset{\text{OPT}}{\boxed{\text{Complexity}}} \iff \underset{\text{SEP}}{\boxed{\text{Integrality}}}$$

Lecture 6
Th 05 May 2022

Theorem 3.8 (Minkowski)**.** For a polyhedron P it holds $x \in P$ iff there exist vertices $v_1, ..., v_k$ and rays $r_1, ..., r_l$, such that

$$\sum_i \lambda v_i + \sum_j \mu_j r_j = x$$
$$\sum_i \lambda_i = 1$$
$$\lambda, \mu \geq 0.$$

Minkowski's Theorem is also known as **Resolution Theorem**.

Proof. See ADM1. □

Definition 3.9. We have different variants of representing a polyhedron P:

- The **H-representation** (from "hyperplane") is given by $P = \{x \mid Ax \leq b\}$.
- The **V-representation** (from "vertex") is given by Theorem 3.8.

Conclusion 3.10. Depending on the representation we have, we have different ways to solve OPT and SEP:

	H-representation	V-representation
OPT	LP Simplex/Ellipsoid	Brute Force
SEP	Brute Force	LP (Exercise 4.1)

Example 3.11. Consider the n-cube $C^n := \{x \in \mathbb{R}^n \mid -1 \leq x_i \leq 1\}$. It has $2n$ facets, but 2^n vertices.

Now, consider the polar of C^n, which can be shown to be the n-octahedron O^n. Remember the intuition, that the polar exchanges vertices with facets. Indeed it holds that now, we have 2^n facets, but only $2n$ vertices.

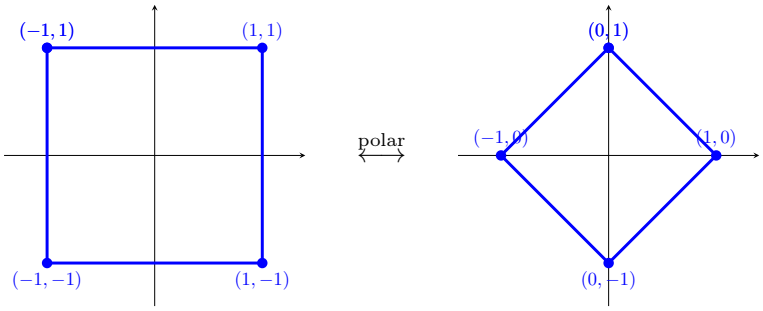


Figure 4: 2-cube vs. 2-octahedron

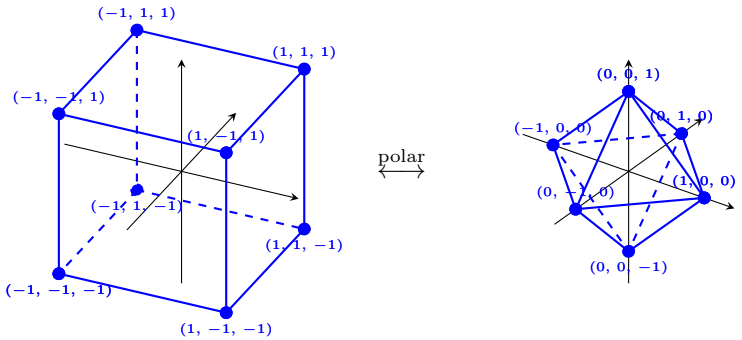


Figure 5: 3-cube vs. 3-octahedron

Information. Polymake is a tool for converting programmatically between H-representation and V-representation.

3.2 Certificate construction

Question 3.12. Consider the problem of finding a solution x to a linear or integer system. How do we construct succinct certificates of feasibility and infeasibility?

In order to answer this question, we will have to look at different kinds of systems each on their own. For the following theorems, we will consider two different systems every time, and show that the solutions can be used as the certificate we seek for.

Theorem 3.13. Exactly one of the following systems is feasible:

$$Ax = b \qquad \text{vs.} \qquad \begin{aligned} y^T A &= 0 \\ y^T b &= 1 \end{aligned}$$

In particular, the left system delivers a certificate of feasibility, whereas the right a certificate of infeasibility.

Proof. Suppose both are feasible. Then we have solutions y^0, x^0, and can construct following contradiction:

$$\begin{aligned} & Ax^0 = b \\ \Leftrightarrow \quad & \underbrace{(y^0)^T A}_{0} x^0 = (y^0)^T b = 1 \end{aligned}$$

It remains to prove at least one system is feasible. We can use **Gaussian Elimination** for that: Gaussian Elimination either yields a solution x^0 we can use as a succinct certificate for feasibility, or determine it is infeasible by yielding the row multiplier y^0 in order to generate $0^T x = 1$ as a succinct certificate of infeasibility. □

Warning. Gaussian Elimination is *not* polynomial in its natural variant because of numbers generated during the algorithm. Nonetheless, if careful and using certain tricks, Gaussian Elimination is polynomial.

Remark 3.14. Theorems stating that exactly one of two systems have a solution are called **Theorem of the Alternative**.

Theorem 3.15 (**Farkas' Lemma**)**.** Exactly one of the following systems is feasible:

$$Ax \leq b \qquad \text{vs.} \qquad \begin{aligned} y^T A &= 0 \\ y^T b &< 0 \\ y &\geq 0 \end{aligned}$$

In particular, the left system delivers a certificate of feasibility, whereas the right a certificate of infeasibility.

Proof. Suppose both are feasible. Analoguous to previous proof we can see the contradiction:

$$\begin{aligned} & Ax^0 \leq b \\ \Leftrightarrow \quad & \underbrace{(y^0)^T A}_{0} x^0 \leq (y^0)^T b < 0 \end{aligned}$$

At least one system is feasible, which we can see by using the Ellipsoid Method to get a feasible x as a feasibility certificate for $Ax \leq b$. Otherwise, $Ax \leq b$ is infeasible. In that case we can use Phase 1 of the Simplex Method to generate $0^T x \leq z$ (for some $z \in \mathbb{Z}^-$) and use the extracted row multipliers y as an infeasibility certificate. Note that this y solves the right system. □

Conclusion 3.16. We already knew from Theorem 2.15 and Theorem 2.43, that previous feasibility problems both lie in $\mathbf{NP} \cap \mathsf{co}-\mathbf{NP}$. Thus, we found that Gaussian Elimination is the suspected polynomial algorithm.

Definition 3.17 (Diophantine equations)**.** An equation of the form $Ax = b$, for $x \in \mathbb{Z}^n$, is called **diophantine equation**.

Theorem 3.18. Exactly one of following systems is feasible:

$$\begin{aligned} Ax &= b & \text{vs.} \qquad & y^T A \in \mathbb{Z}^n \\ x &\in \mathbb{Z}^n & & y^T b \notin \mathbb{Z} \end{aligned}$$

In particular, the left system delivers a certificate of feasibility, whereas the right a certificate of infeasibility.

Proof. Suppose both are feasible. Then

$$\begin{aligned} & Ax^0 = b \\ \Leftrightarrow \quad & \underbrace{(y^0)^T A}_{\mathbb{Z}^n} \underbrace{x^0}_{\mathbb{Z}^n} = \underbrace{(y^0)^T b}_{\notin \mathbb{Z}} \end{aligned}$$

We can use the **Hermite Normal Form** algorithm to show that at least one system is feasible. Note that the Hermite Normal Form can be calculated in polynomial time. □

Problem. IP is defined as finding $x \in \mathbb{Z}^n$ for $Ax \leq b$. We have already shown that $\mathsf{IP} \in \mathbf{NPC}$ (see Theorem 2.24), meaning that certificates cannot be calculated in polynomial time (unless $\mathbf{P} = \mathbf{NP}$).

Conclusion 3.19. Summing everything up, we can summarize our findings for calculation of feasibility certificates in following table:

	continuous	integer
$=$	Gaussian Elim./Phase 1	Hermite Normal Form
$\leq$	Linear Programming	not possible

The problem with integer inequality systems is its missing duality, e.g. there is no way of generating succinct certificates for verifying infeasibility, making it impossible to use the Theorem of the Alternative.

Remark 3.20. If we have an LP in standard form, we can also formulate a Theorem

of the Alternative using **Farkas' Lemma**:

$$\begin{array}{rcr} Ax = b & & Ax \leq b \\ x \geq 0 & \iff & -Ax \leq -b \\ & & -x \leq 0 \end{array}$$

$$\overset{3.15}{\text{vs.}}$$

$$\begin{array}{rcr} (y^1)^T A - (y^2)^T A - (y^3)^T = 0 & & y^T A \geq 0 \\ (y^1)^T b - (y^2)^T b < 0 & \iff & y^T b < 0 \\ y^1, y^2, y^3 \geq 0 & & y \text{ free} \end{array}$$

Note for the last equivalence that we used $y = y^1 - y^2$ with $y^1, y^2 \geq 0$, and interpreted y^3 as slack.

Theorem 3.21 (**Gourdan's Theorem**)**.** Exactly one of following systems is feasible:

$$\begin{array}{ccc} Ax < 0 & \text{vs.} & y^T A = 0 \\ & & y \geq 0 \\ & & y \neq 0 \end{array}$$

Proof. Consider a feasible x such that $Ax^0 < 0$. Then we can scale and get x^0 such that $Ax \leq -\mathbf{1}$, and, again, using Theorem 3.15, yields our other system. Note that $y^T(-\mathbf{1}) < 0$ is equivalent to $\sum y_i > 0$, implying with $y \geq 0$ that $y \neq 0$. □

Note. We can also check that both systems cannot be feasible:

$$\begin{array}{rc} & Ax^0 < 0 \\ \Leftrightarrow & \underbrace{(y^0)^T A}_{0} x^0 < \underbrace{(y^0)^T 0}_{0}, \end{array}$$

which is a contradiction.

3.3 Non-linearity in LP

Consider an LP with lower and upper bounds and its dual:

$$\begin{array}{ll}\min\limits_{x} & c^T x \\ \text{s.t.} & Ax = b, \\ & l \le x \le u\end{array} \iff \begin{array}{ll}\min\limits_{x} & c^T x \\ \text{s.t.} & Ax = b, \\ & x \ge l, \\ & -x \ge -u, \\ & x \text{ free}\end{array} \overset{\text{Dual}}{\iff} \begin{array}{ll}\max\limits_{y,\lambda,\mu} & b^T y + l^T\lambda - u^T\mu \\ \text{s.t.} & y^T A + \lambda^T - \mu^T = c^T, \\ & \lambda, \mu \ge 0, \\ & y \text{ free}\end{array}$$

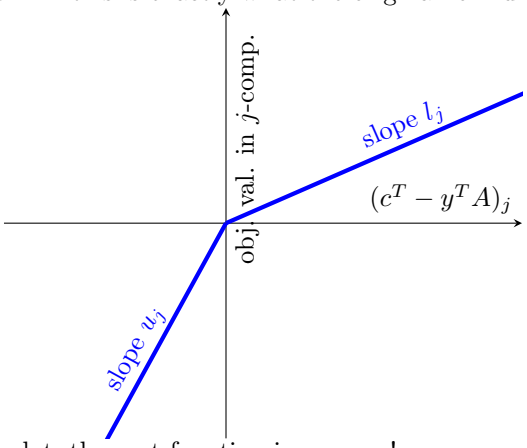
Note that we can write the dual constraint as

$$\lambda^T - \mu^T = c^T - y^T A,$$

allowing an interpretation of λ^T as the positive part of $c^T - y^T A$, and μ^T as the negative part. Reformulating thus yields

$$\begin{array}{ll}\max\limits_{y,\lambda,\mu} & b^T y + l^T (c^T - y^T A)^+ - u^T (c^T - y^T A)^- \\ \text{s.t.} & y \text{ free}\end{array}$$

Note that now, the system doesn't seem to have any constraints left, but clearly this cannot be true. In fact, $c^T - y^T A)^+$ and $c^T - y^T A)^-$ are piecewise linear only! Therefore, in order to get a valid LP - this is exactly what the original formulation did.



As can be seen in the plot, the cost function is concave!

Lecture 7
Tu 10 May 2022

Note that the function isn't smooth, but still we can define a notion for its gradient:

Definition 3.22. We call the set of possible "derivatives" in a point of a function the **subdifferential**.

Let's visualize using previous example. Let's also consider the implications for optimal solutions from complementary slackness:

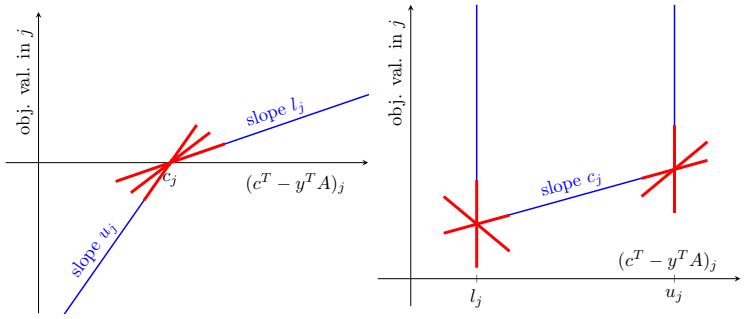
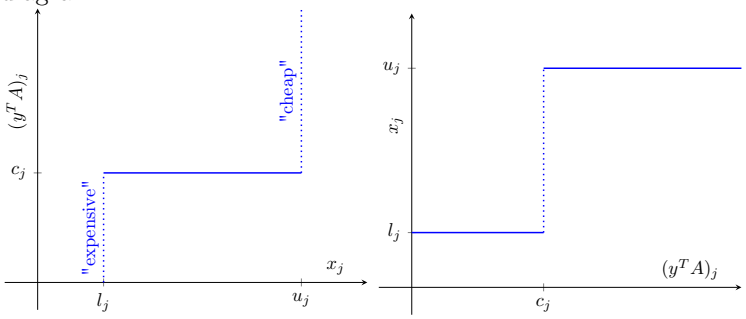


Figure 6: Exemplary dual and primal cost function in jth component with subset of subdifferentials in red

- For $\lambda_j > 0$ follows that $c_j > (y^T A)_j$ ("expensive"), and by CS $x_j = l_j$.
- For $\mu_j > 0$ follows that $c_j < (y^T A)_j$ ("cheap"), and by CS $x_j = u_j$.
- For $c_j = (y^T A)_j$ ("neutral") follows by CS that $l_j \leq x_j \leq u_j$.

Definition 3.23 (Kilter diagram)**.** We can visualize variable-constraint pairs of complementary slackness with a **Kilter diagram**.

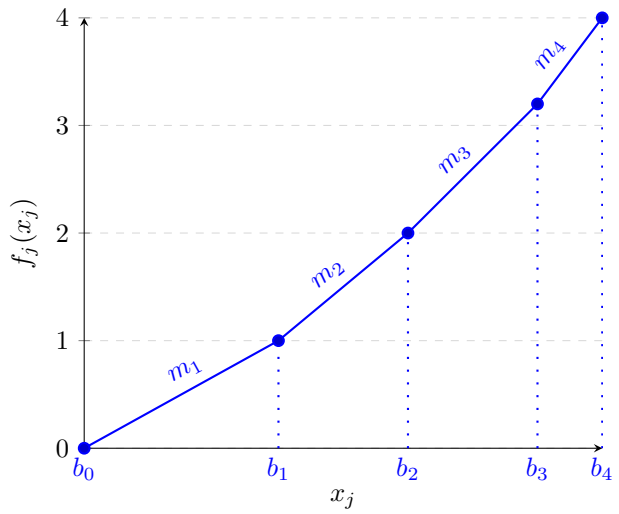
Example 3.24. Again, using our system, we can draw the primal and dual Kilter diagram:



Fact 3.25 (Convex duality)**.** If term j of the primal objective is $f_j(x_j)$ for convex f_j, then term j of the dual objective is $-f_j^\bullet(y_j)$, such that y_j is the dual of x_j.

We're using $f^\bullet$ as the **convex conjugate** of f, which has the property that $(f^\bullet)^\bullet = f$.

Now, consider some piecewise linear convex function with slopes m_i and breakpoints b_i.



We can express values on this function with following LP:

$$\begin{aligned} \min_{x} \quad & m_1 x_{j,1} + m_2 x_{j,2} + m_3 x_{j,3} + \dots \\ \text{s.t.} \quad & 0 \le x_{j,i} \le b_i - b_{i-1}, \quad i = 1, \dots, \\ & x_j = x_{j,1} + x_{j,2} + x_{j,3} \end{aligned}$$

Note that this forces the LP to use as much of $x_{j,i}$ as possible before moving to the next component.

Conclusion 3.26. Piecewise linear convex problems don't need IP.

Note. In fact, IP is only needed for non-convex/non-concave functions, see [AMO93, Ch. 14].

4 Approaches to IP

Even though IP is **NP**-complete, we still want to solve them as they model many real-world problems. For certain cases, though, we can use tricks to make calculation easier:

1. If the IP only has integer vertices for all b, we can just use LP.
2. If the IP only has integer vertices for a single useful b, we can at least use LP for this b, and might derive useful information anyway.
3. We could get a direct combinatorial algorithm that doesn't use LP, using OPT-SEP-duality.
4. For a *fixed* (small) dimension, we can solve IP in polynomial time.
5. If we are only interested in *good* solutions, we could use approximation algorithms and heuristics.

6. Alternatively, solve the relaxed LP and round to an IP solution.
7. We can relax "bad" constraints and variables.
8. Just use Cutting Planes.

4.1 Integer-optimal solutions in LP for special coefficient matrices

Recall. In ADM1 we proved that there are combinatorial problems that can be solved using LP nonetheless, e.g. Max-Flow, Min-Cut, Bipartite Matching, Min-Cost-Flow etc.

Question 4.1. Why do exactly these problems have the property of integer vertices?

Given an optimal vertex solution $x^* = (x_B^*, 0) = (B^{-1}b, 0)$ with basis B to an LP. By **Cramer's Rule**, it holds for all $j \in B$, that

$$x_j^* = \frac{\overbrace{\det(B_1, B_2, ..., b_j, ..., B_n)}^{\text{integer}}}{\det(B)}.$$

Thus, if $|\det(B)| = 1$, then x^* is integer.

Definition 4.2 (Totally unimodular)**.** A matrix A is **totally unimodular**, if for all square submatrices B of A it holds that $\det(B) \in \{-1, 0, 1\}$.

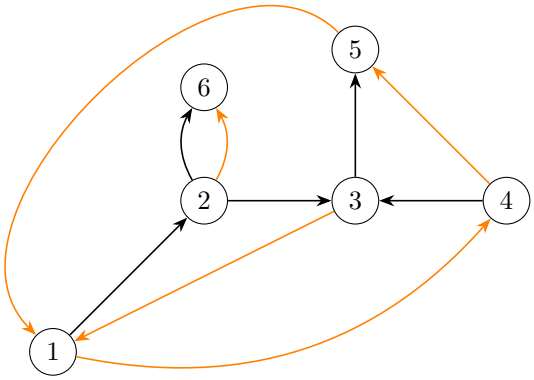
Note. Obiviously, A itself must consist only of $\{-1, 0, 1\}$ entries in order to be totally unimodular.

Theorem 4.3. Given A is totally unimodular. Then all optimal vertices x^* are integer for all righthandside b's. Conversely, if all vertices of $\{x \mid Ax \leq b, x \geq 0\}$ are integer for all righthandside b, then A is totally unimodular.

Proof. See [NW99, Thm 2.5 III 1.2]. □

Definition 4.4. Let $G = (N, A)$ be a directed graph, T a spanning tree of G, and $S \subseteq A$. We construct a matrix $D \in \mathbb{R}^{|N-1|,|S|}$, such that every column corresponds to an arc $(u, v) \in S$, and every row to an arc in T. Consider the undirected (unique) path from u to v in T. We set in each column every entry to 1, where we used the arc as supposed, to -1, if we used the arc backwards, and 0 otherwise. Then D is called a **tree-path**, or **network matrix**.

Example 4.5. Given following graph:



Then a network matrix of this graph is given by

$$\begin{array}{c} \\ 1\to 2 \\ 2\to 3 \\ 3\to 6 \\ 3\to 5 \\ 4\to 3 \end{array} \begin{array}{c} \begin{array}{ccccc} 1\to 4 & 2\to 6 & 3\to 1 & 4\to 5 & 5\to 1 \end{array} \\ \begin{pmatrix} 1 & 0 & -1 & 0 & -1 \\ 1 & 0 & -1 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ -1 & 0 & 0 & 1 & 0 \end{pmatrix} \end{array}$$

Theorem 4.6. Any network matrix M is totally unimodular.

Proof. Every submatrix of a network matrix M is again a network matrix (deleting a row contract the arc of T). Thus it suffices to show that every square network matrix M_s has $\det(M_s) \in \{-1, 0, 1\}$. We prove by induction over dimension d of M_s.

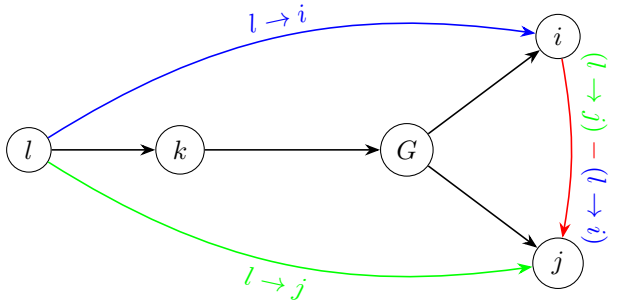
For $d = 1$ this is clear. Thus, consider the statement true for some d. Let node l be a leaf of the spanning tree T, and consider row $l \to k$. Using a case distinction:

- 0 arcs in S hit l. Then row $l \to k$ is 0, and thus $\det(M_s) = 0$.
- Exactly 1 arc in S hits l. Then row $l \to k$ is a unit vector, and block decomposition yields

$$\det(M_s) = \det \left(\begin{array}{c|c} \pm 1 & 0 \cdots\cdots 0 \\ \hline * & \\ \vdots & M_s' \\ * & \end{array}\right) = \pm \det(M_s')$$

- Otherwise, there are at least 2 arcs in S that hit l. Let $l \to i$ and $l \to j$ be two of them. Then we can subtract column $l \to i$ from $l \to j$ and zero out the first entry of $l \to j$. Additionally, the column now is the incidence vector of

$i \to j$, by uniqueness of tree paths. As a consequence, our matrix is still a network matrix. Therefore, iteratively applying this step until a single 1 remains let us use previous case, noting that column subtraction only negates the determinant.



□

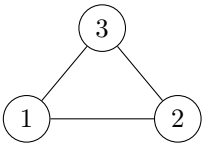
Corollary 4.7. A node-arc incidence matrix of a directed graph is totally unimodular.

Proof. Add root r and for every node i arcs $r \to i$. Because every column has form $[0, \ldots, -1, \ldots, -1, \ldots, 0]$, this corresponds to $i \to r \to j$. □

Corollary 4.8. A node-edge incidence matrix of a bipartite graph is totally unimodular.

Proof. Add root r, for every $i \in L$ arcs $i \to r$, and for every $j \in R$ arcs $r \to j$. Note that columns refer to $i \to r \to j$ □

Remark 4.9. Not all node-edge incidence matrices are totally unimodular! For example



with $\det \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} = 2$

Definition 4.10. A 0-1-matrix A has the **consecutive ones-property** if the 1's in each row do not have any 0's between them, i.e. 0001111100.

Corollary 4.11. A matrix A with consecutive ones-property is totally unimodular.

Proof. Suppose T is a line of connected nodes. For each row, construct an arc from the first 1 to the last 1. Then A is a network matrix for this graph. □

Note that there are totally unimodular matrices that aren't network matrices. Furthermore, it's also possible to combine totally unimodular matrices to get new ones.

Theorem 4.12. Let A_1, A_2 be two totally unimodular matrices. Then

$$\left(\begin{array}{c|c} A_1 & 0 \\ \hline 0 & A_2 \end{array}\right)$$

is also totally unimodular.

Theorem 4.13 (Seymour)**.** The set of totally unimodular matrices is fully defined by

- all network matrices,
- two additional 5×5 matrices, and
- three different composition operations, e.g. Theorem 4.12.

Conclusion 4.14. Network problems are the easiest IPs.

Lecture 8
Th 12 May 2022

4.2 Integer-optimal solutions in LP for special RHSs

Definition 4.15. Given a graph $G = (N, E)$ and cost vector $c \in \mathbb{R}^E$. Finding a spanning tree T such that $c(T)$ is minimal is called **Minimal Spanning Tree** problem, short MST.

Recall. In ADM1 we learned that we can use **Kruskal's algorithm** to find a MST in polynomial time. Kruskal sorted the edges and added edges in this order if it wouldn't create a cycle.

On the other hand, there is also an LP formulation:

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & x(\gamma(K)) \leq |K| - 1, \quad K \subsetneq N, \\ & x(E) = n - 1, \\ & x \geq 0 \end{aligned}$$

Tom doesn't like the LP formulation, though, because

1. "min" and "$\leq$" just feel wrong,
2. rather than $x(\gamma(K))$, we should use $x(S) \leq ?$ for $S \subseteq E$.

We can circumvent this problems with following LP, for some $M >> \max_i w_i$ and $w = M - c$, and some magic function r:

$$\begin{aligned} \max_{w} \quad & w^T x \\ \text{s.t.} \quad & x(S) \leq r(S), \quad S \subseteq E, \\ & x \geq 0 \end{aligned} \tag{1}$$

Let's have a closer look at $r(S)$, and define it as the maximum number of edges we can choose in S without creating a cycle.

Let's also define $cc(S)$ as the number of **connected components** in (N, S).

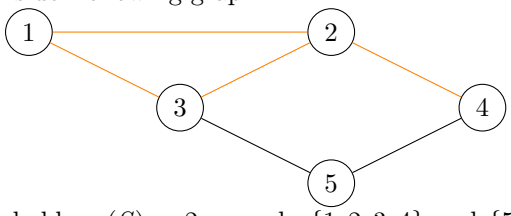
Theorem 4.16. For a graph $G = (N, E)$ it holds that $r(S) = n - cc(S)$.

Proof. Let $C_1, ..., C_k$ be node sets of connected components of (N, S), meaning $cc(S) = k$. Note that $\sum_i |C_i| = n$. Furthermore, we can choose at most $|C_i| - 1$ acyclic edges per component C_i by choosing any spanning tree. Thus,

$$\begin{aligned} r(S) = & \text{ maximum acyclic edges in } (N, S) \\ = & \sum_i (\text{maximum acyclic edges in } C_i) \\ = & \sum_i |C_i| - 1 = n - k = n - cc(S). \end{aligned}$$

□

Example 4.17. Consider following graph:



For the drawn S, it holds $cc(S) = 2$, namely $\{1, 2, 3, 4\}$ and $\{5\}$, and thus $r(S) = 5 - 2 = 3$.

We can construct the dual of (1):

$$\begin{aligned} \min_{y_S} \quad & r(S) y_S \\ \text{s.t.} \quad & \sum_{S: e \in S} y_S \geq w_e, \quad \forall e \in E, \\ & y \geq 0 \end{aligned} \tag{2}$$

Warning. Our original LP had 2^n constraints, which means our dual has 2^n variables. Even the data $r(S)$ cannot be written down in polynomial time!

This means we need to think about $r(S)$ a little bit more. Consider a set R and edge $e \in E$ with $R \subseteq S \subseteq S + e$. The so-called **marginal cost** of adding e to R is $r(R+e) - r(R) \in \{0, 1\}$. Same goes for S.

Definition 4.18. For a set M, let $r : \mathcal{P}(M) \to \mathbb{R}_0^+$ be a function. If the marginal costs are non-decreasing for $R \subseteq S$ for the same edge, i.e.

$$r(S + e) - r(S) \leq r(R + e) - r(R),$$

then we call r **submodular**.

Theorem 4.19. Our acyclic-$r(S)$ is submodular.

Proof. It suffices to show that $r(S + e) - r(S) = 1$ and $r(R + e) - r(R) = 0$ cannot happen. Note that the connected components of R are subsets of connected components of S. Thus, if we add an edge e that connects two connected components, it also connects two components of R. By Theorem 4.16, this is the only way $r(S)$ increases by one, but also forces $r(R)$ to increase. □

There is also an equivalent definition of submodularity:

Theorem 4.20. A function $r : \mathcal{P}(M) \to \mathbb{R}$ is **submodular** iff for all $S, R \subseteq E$

$$r(S) + r(R) \geq r(S \cap R) + r(S \cup R).$$

Proof. See Exercise 5.2. □

Now let's try to derive Kruskal from our LPs (1) and (2):

Lemma 4.21. Suppose x^*, y^* are optimal in their corresponding LPs (1) and (2). Among the dual optimal solution let y^* be the one with $y^* = \sum_{S \subseteq E} (y_S)^2$ (notice the square!).

Then for $R, S \subseteq E$, if $y_R^*, y_S^* > 0$, it follows that either $R \subseteq S$ or $S \subseteq R$. We call this property **nested**.

Proof. Assume $y_R^*, y_S^* > 0$, but are not nested. Let $\varepsilon = \min(y_R^*, y_S^* 0) > 0$ and

$$y'_Q = \begin{cases} y_Q^* - \varepsilon, & \text{if } Q = R \vee Q = S, \\ y_Q^* + \varepsilon, & \text{if } Q = R \cup S \vee Q = R \cap S, \\ y_Q^*, & \text{otherwise.} \end{cases}$$

Then y' is still feasible, because the ε cancel out, since $R \cup S$ and $R \cap S$ must be new sets, and every edge is either in all four sets, no set, or exactly one of R, S and $R \cup S, R \cap S$ each. The choice of ε ensures $y' \geq 0$.

Now have a look how the objective function changes and consider the difference given by:

$$\varepsilon(\underbrace{r(R \cap S) + r(R \cup S) - r(R) - r(S)}_{\leq 0 \text{ by submodularity}})$$

We cannot get cheaper though, because we were already optimal. Therefore, y' is also an optimal dual solution, but

$$\sum_S (y'_S)^2 < \sum_S (y^*_S)^2$$

is a contradiction! (One can check this by tedious calculations.) □

Theorem 4.22. Optimal feasible basic solutions to (1) and (2) are integer, and thus solve MST.

Proof. Consider $\mathcal{I} \coloneqq \{S \subseteq E \mid y^*_S > 0\}$. Because of previous lemma we can build a chain of elements of $\mathcal{I}$:

$$S_1 \subseteq \ldots \subseteq S_m, \quad |S_i| = i.$$

If length m is not possible, add some $y^*_S = 0$ sets. Thus, using this S as a basis yields following system:

$$\begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_m \end{array} \begin{array}{c} \begin{array}{cccccc} S_1 & S_2 & S_3 & \cdots & \cdots & S_m \end{array} \\ \begin{pmatrix} 1 & \cdots & \cdots & \cdots & \cdots & 1 \\ 0 & \ddots & & & & \vdots \\ \vdots & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & \cdots & \cdots & 0 & 1 & 1 \end{pmatrix} \end{array} \begin{pmatrix} y^*_{S_1} \\ y^*_{S_2} \\ y^*_{S_3} \\ \vdots \\ y^*_{S_m} \end{pmatrix} = \begin{pmatrix} w_{e_1} \\ w_{e_2} \\ w_{e_3} \\ \vdots \\ w_{e_m} \end{pmatrix}$$

Note the basis matrix has continuous ones property (and thus being totally unimodular by Corollary 4.11), and is an upper triangular matrix, meaning the resulting equation system is easy to solve. In fact, substituting yields

$$\begin{aligned} y^*_{S_m} &= w_{e_m} \\ y^*_{S_{m-1}} &= w_{e_m - 1} - w_{e_m} \\ &\vdots \\ y^*_{S_1} &= w_{e_1} - w_{e_2} \end{aligned}$$

So, for this y^* to be feasible (i.e. non-negative), the edges need to be ordered by weight, just like in Kruskal! Furthermore, for every edge $e_k \in E$ it holds

$$\begin{aligned}\sum_{S:e_k \in S} y^*_S &= \sum_{i=k}^{m} y^*_{S_i} \\ &= \sum_{i=k}^{m} w_i - w_{i+1} \\ &= w_{e_k},\end{aligned}$$

concluding feasibilty according to (2).

On the other hand, we can simply derive a primal optimal basis:

$$\begin{array}{c} \\ S_1 \\ S_2 \\ S_3 \\ \vdots \\ S_m \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} e_1 & e_2 & e_3 & \cdots\cdots & e_m \end{array} \\ \begin{pmatrix} 1 & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & 0 \\ 1 & \cdots & \cdots & \cdots & 1 \end{pmatrix} \end{array} \begin{pmatrix} x^*_{e_1} \\ x^*_{e_2} \\ x^*_{e_3} \\ \vdots \\ x^*_{e_m} \end{pmatrix} = \begin{pmatrix} r(S_1) \\ r(S_2) \\ r(S_3) \\ \vdots \\ r(S_m) \end{pmatrix}$$

Analoguous, we get

$$\begin{aligned} x^*_{e_i} &= r(S_i) - r(S_{i-1}) \\ &= r(S_{i-1} + e_i) - r(S_i) \in \{0, 1\} \end{aligned}$$

which represents the edge addition step, and assures feasibilty of (1)!

It remains to be shown that our constructed solutions are indeed optimal. Using strong duality, we see

$$\begin{aligned}\sum_{i=1}^{m} w_i x^*_{e_i} &= \sum_{i=1}^{m} w_i (r(S_i) - r(S_{i-1})) \\ &= \sum_{i=1}^{m} r(S_i)(w_i - w_{i+1}) \\ &= \sum_{i=1}^{m} r(S_i) y^*_{S_i}\end{aligned}$$

□

Conclusion 4.23. Using only a submodular function and LPs, we constructed an algorithm that solves MST. In particular, no special facts about trees etc. were used.

Lecture 9
Tu 17 May 2022

Definition 4.24 (Greedy)**.** Given a set $\mathcal{I} \subseteq \mathcal{P}(E)$. We call following class of algorithms **greedy**:

Algorithm 3: General greedy algorithm

```
sort w_1 ≥ ··· ≥ w_m
T ← ∅
for i = 1, ..., m do
    if T ∪ e_i ∈ I then
        T ← T ∪ e_i
    end
end
```

Theorem 4.25. The LPs (1) and (2) can be solved greedily with *any* submodular function r.

Proof. We show that it suffices to choose for $i = 1, \dots, m$ greedily x_{e_i} as big as possible, i.e.

$$\begin{aligned} x_{e_1} &:= r(S_1) \\ x_{e_2} &:= r(S_2) - r(S_1) \\ &\vdots \end{aligned}$$

It is clear that $x \geq 0$. Thus, consider $S \subseteq E$ in order to show $x(S) \leq r(S)$. Let $k := \max\{i \mid e_i \in S\}$. Using induction over k with the trivial base case $k = 0$, we define $S' := S \setminus e_k$. Observe, by choice of k, that $S \cap S_{k-1} = S'$ and $S \cup S_{k-1} = S_k$. Therefore, by submodularity 4.20:

$$r(S) \geq r(S') + r(S_k) - r(S_{k-1})$$

Induction on k yields $x(S') \leq r(S')$, and thus

$$\begin{aligned} x(S) = x(S') + x_{e_k} &= x(S') + r(S_k) - r(S_{k-1}) \\ &\leq r(S') + r(S_k) - r(S_{k-1}) \\ &\leq r(S) \end{aligned}$$

□

Note. Assuming non-degeneracy, then the vertices of the submodular polyhedron correspond to permutations of E. In particular, there are $m!$ vertices.

Now, let's concentrate on submodular functions where $r(\{e_i\}) \in \{0, 1\}$. We can derive this notion with matroids - a common generalisation of (combinatorial) graphs/matrices that define "independence".

Definition 4.26 (Matroid)**.** Given a set $\mathcal{I} \subseteq \mathcal{P}(E)$. If also

1. $\emptyset \in \mathcal{I}$,
2. $S \in \mathcal{I}, R \subseteq S \Rightarrow R \in \mathcal{I}$, and
3. $R, S \in \mathcal{I}, |R| < |S| \Rightarrow \exists e \in S \setminus R : R + e \in \mathcal{I}$,

then $\mathcal{I}$ is a **matroid**. If only properties (1) and (2) hold, we call it a **independence system**. Property (3) is also called **extensibility**.

Definition 4.27 (Rank function)**.** Given a matroid $\mathcal{I}$. Define the **rank function** $r : \mathcal{P}(E) \to \mathbb{Z}$ as

$$r(S) = \max_{I \in \mathcal{I}, I \subseteq S} |I|.$$

Theorem 4.28. For all matroids, the rank function is submodular.

Proof. It suffices to show for $R \subsetneq S \subsetneq S + e$ that

$$\underbrace{r(S+e) - r(S)}_{0,1} \leq \underbrace{r(R+e) - r(R)}_{0,1}.$$

The only bad case is $r(S+e) - r(S) = 1$ and $r(R+e) - r(R) = 0$. Suppose

$$r(R) = |I_1|, I_1 \in \mathcal{I}, I_1 \subseteq R$$
$$r(S) = |I_2|, I_2 \in \mathcal{I}, I_2 \subseteq S$$

Set $k = r(S) - r(R) \geq 0$. From $R \subsetneq S$ and extensibility follows that there is a $K \subseteq S \setminus R$ with $|K| = k$ such that

$$I_2 = I_1 \cup K$$

Also, from $r(S+e) > r(S)$ it follows from extensibility, that there is $f \in S + e$ such that $I_2 + f \in \mathcal{I}$. We see that $f = e$, otherwise it follows from $I_2 + f \subseteq S$ that $r(S) > |I_2|$, which is a contradiction.

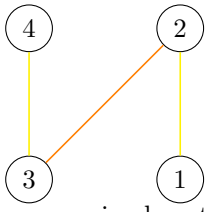
Summarizing, we see $I_2 + e \in \mathcal{I}$. From property (2) follows that $I_1 + e \in \mathcal{I}$. Also $I_1 + e \subseteq R + e$, which implies $r(R+e) = r(R) + 1 = |I_1| + 1$. This contradicts our initial assumption! □

Corollary 4.29. Greedy is optimal for *any* matroid.

Theorem 4.30. If Greedy is optimal for all cost vectors w on an independence system, then it is a matroid.

Let $(E, \mathcal{I})$ be an independence system. Then extensibility states that additionally all maximal independent subsets of S have the same size, and are thus *maximum*.

Example 4.31. Note the difference of *maximal* and *maximum*! Given following graph:



Then the yellow edges form only a maximal matching, while the orange edges a maximum (and maximal) matching.

Define

$$\rho(S) = \min\{|I| \mid I \subseteq S, I \in \mathcal{I}, I \text{ maximal}\}.$$

Especially, for matroids $\rho(S) = r(S)$.

We want to prove a stronger theorem, though:

Theorem 4.32. If we apply Greedy to an independence system $(E, \mathcal{I})$, then it holds

$$q \coloneqq \min_{S \subseteq E} \frac{\rho(S)}{r(S)} \leq \frac{\text{greedy obj. value}}{\text{optimal obj. value}} \leq 1$$

Additionally, the worst case is attainable.

Proof. Consider $w_{e_1} \geq ... \geq w_{e_n}$ and $S_i \coloneqq \{e_1, ..., e_i\}$. Let $G \subseteq E$ be a greedy solution and $G_k \coloneqq G \cap S_k$. Let $O \subseteq E$ be an optimal solution and $O_k \coloneqq G \cap O_k$.

Consider two cases:

- $e_i \in G$: Then $|G_i| = |G_{i-1}| + 1$
- $e_i \notin G$: Then $|G_i| = |G_{i-1}|$

Therefore, the greedy objective value is

$$\begin{aligned}\sum_i (|G_i| - |G_{i-1}|) w_{e_i} &= \sum_i |G_i|(w_{e_i} - w_{e_{i+1}}) \\ &\geq \sum_i \rho(S_i)(w_{e_i} - w_{e_{i+1}}) \\ &\geq q \sum_i r(S_i)(w_{e_i} - w_{e_{i+1}}) \\ &\geq q \sum_i |O_i|(w_{e_i} - w_{e_{i+1}}) \\ &= q \sum_i (|O_i| - |O_{i-1}|) w_{e_i} \\ &= q \cdot \text{ optimal obj. value}\end{aligned}$$

Suppose our q is attained at S,

$$q = \frac{\rho(S)}{r(S)}.$$

By definition there are $I_1, I_2 \subseteq S$ with $I_1, I_2 \subseteq \mathcal{I}$ such that $r(S) = |I_2|$ and $\rho(S) = |I_1|$. Choose

$$w_e = \begin{cases} 1, & e \in S \\ 0, & e \notin S \end{cases}.$$

and consider following ordering:

I_1	$S \setminus I_1$	S^c

$\underbrace{\qquad\qquad\qquad}_{w=1} \quad \underbrace{\qquad}_{w=0}$

Greedy solution will be I_1 because of maximality, but optimal solution is I_2. Therefore $q = \frac{|I_1|}{|I_2|}$ as proposed. □

Proof for Theorem 4.30. For matroids, $q = 1$. Otherwise, $q < 1$. □

Corollary 4.33. Greedy is a 2-approximation for matching.

Example 4.34. Other matroids are given by:

1. Uniform: $\mathcal{I} = \{S \mid |S| \leq k\}$
2. Partition: E is partitioned into $P_1, ..., P_k$. Then $I \in \mathcal{I}$ iff for all i, $|I \cap P_i| \leq 1$.
   (a) $G = (N, A)$, $P_i = \delta^+(\{i\})$

(b) $G = (S \cup T, E),\ P_i = \delta(\{i\}), i \in S$

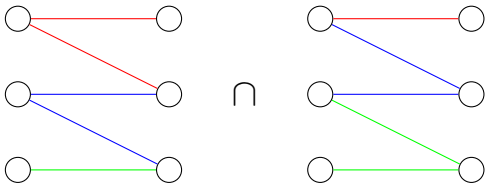
Example 4.35. Independence systems, that are *not* matroids, are for example:

1. Bipartite matching
2. Branching: $G = (N, A)$, $B \subseteq A$ is a branching iff acyclic, i.e. for all i, $\delta^-(\{i\}) \leq 1$.

Remark 4.36. Intersections of matroids seem interesting: Previous examples can be generated via intersections of partition matroids, i.e. for a bipartite graph $G = (S \cup T, E)$, the set of bipartite matchings M is given by

$$\mathcal{I}_1 \coloneqq \{I \subseteq E \mid \forall i \in S : |I \cap \delta^+(\{i\})| \leq 1\}$$
$$\mathcal{I}_2 \coloneqq \{I \subseteq E \mid \forall i \in T : |I \cap \delta^+(\{i\})| \leq 1\}$$
$$M = \mathcal{I}_1 \cap \mathcal{I}_2.$$

We can visualize this by coloring every partition set and defining the matroids as the subsets where we take at most one edge per color:



By taking the intersection we prohibit to take multiple edges from the same node, leaving us at bipartite matchings.

Lecture 10
Th 19 May 2022

We now want to combine submodular functions with the geometrical interpretation of LPs.

Definition 4.37 (Polymatroid)**.** Given a monotonically increasing submodular function r with $r(\emptyset) = 0$. We call the polytope

$$\{x \in \mathbb{R}^E_{\geq 0} \mid \forall S \subseteq E : x(S) \leq r(S)\}$$

a **polymatroid**, and r a **polymatroid rank function**.

Consider following exemplary use of rank functions:

Theorem 4.38. Let $r(S)$ be the max-flow value when we set the capacity for all incident $j \notin S$ to 0, i.e. $u_{sj} = 0$. Then r is submodular.

Proof. See homework . □ homework

Theorem 4.39. Greedy works for polymatroids.

Proof. Polymatroids just use special submodular functions. □

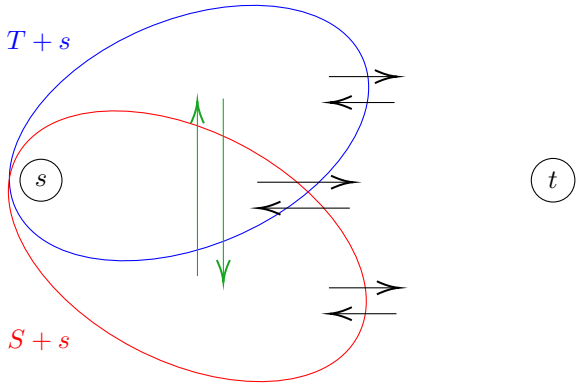
In fact, Greedy still works for non-monotone submodular functions, for example:

Theorem 4.40. Consider max-flow/min-cut on network $N = (s, t, E)$, and define $r(S)$ for all $S \subseteq E$ as the capacity of $s + S$. Then this r is submodular.

Proof. Considering $S+s, T+s, S\cap T+s, S\cup T+s$, then every edge occurs on both sides of the inequality, except the ones from $S + s$ to $T \setminus S$, which immediately leads to

$$r(S) + r(T) \geq r(S \cap T) + r(S \cup T).$$

Notice though that $r(\emptyset) > 0$ and r not monotone.



□

Let's compare Greedy vs. Monotonicity. If r is monotone, then $x_{e_i} = r(S_i) - r(S_{i-1}) \geq 0$. But if we don't care for $x \geq 0$, then we can apply Greedy.

$$x \text{ free} \implies \sum_{S: e \in S} y_S = w_e$$

Notice that "=" was "≥", but our previous proof showed that we get equality anyway for the dual constraint.

Conclusion 4.41. Greedy works for polymatroids, matroids, and general submodular functions, even with negative values.

Remark 4.42. Submodular LPs are *not* totally unimodular. Consider e.g. following submatrix of a submodular LP:

$$\det \begin{array}{c} \\ e_1e_2 \\ e_1e_3 \\ e_2e_3 \end{array} \begin{array}{c} \begin{array}{ccc} e_1 & e_2 & e_3 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \end{array} = -2$$

Still, we get only integer solutions, meaning submodular RHS's are special

Definition 4.43. We call a polyhedron **integral** if $P = P_I$, with P_I being the integer hull of P. Equivalently, all vertices of P are integral.

We first want to derive a tool to prove that $P = P_I$.

Theorem 4.44. If for all c such that an optimum exist holds that

$$z := \max\{c^T x \mid x \in P\} \in \mathbb{Z},$$

then P is integral.

Proof. Let v be any vertex of P. We know there is an outer cone of possible vectors c such that $z_c = c^T v \in \mathbb{Z}$. For c long enough, we can add unit vectors and still stay in the cone, i.e. for any index i also $z_{c_i} = (c + e_i)^T v \in \mathbb{Z}$. Therefore, $(c + e_i)^T v - c^T v = v_i \in \mathbb{Z}$, and in particular $v \in \mathbb{Z}^n$ □

In order to apply this result following definition is useful:

Definition 4.45. We call a system $Ax \leq b$ **totally dual integral** if for all $c \in \mathbb{Z}^n$ with an optimum to $\{c^T x \mid x \in P\}$ the corresponding y^* is integral.

Corollary 4.46. If $Ax \leq b$ is totally dual integral, and $b \in \mathbb{Z}^m$, then P is integral.

Proof. Recall that our z is also the optimal objective value of the dual. We know for all c with an optimum, that $y^* \in \mathbb{Z}^m$ for $b^T y^* = z \in \mathbb{Z}$. By Theorem 4.44, all primary vertices are integral. □

Theorem 4.47. For submodular r, the polyhedron $\{x \mid x(S) \leq r(S)\}$ is totally dual integral.

Proof. Greedy says (note w is synonym to c):

$$y_S^* = \begin{cases} w_i - w_{i+1}, & \text{if } S = S_i, \\ 0, & \text{else} \end{cases} \in \mathbb{Z}^{2^E}$$

□

Conclusion 4.48. Note the difference:

"Totally unimodular" corresponds to integral polyhedra for *all* integer-RHS.

"Totally dual integral" corresponds to integral polyhedra for *special* RHS. Submodular RHS are an example.

Consider again intersections of matroids, i.e. bipartite matchings as in Remark 4.36.

Theorem 4.49. For submodular r_1, r_2, given the primal LP

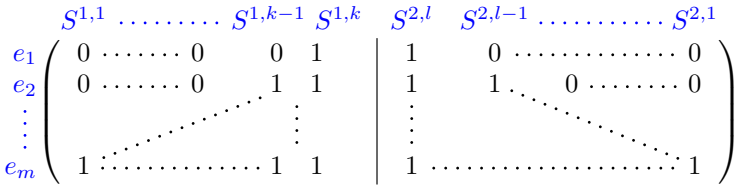
$$\begin{aligned} \max_x \quad & w^T x \\ \text{s.t.} \quad & x(S) \le r_1(S), \\ & x(S) \le r_2(S), \\ & x \ge 0 \end{aligned}$$

with its dual

$$\begin{aligned} \min_{y^1, y^2} \quad & r_1^T y^1 + r_2^T y^2 \\ \text{s.t.} \quad & \sum_{S: e \in S} y_S^1 + y_S^2 \ge w_e, \\ & y^1, y^2 \ge 0. \end{aligned}$$

Then the primal is totally dual integral.

Proof. Analoguous to the proof of Theorem 4.22 we can show there exist optimal $(y^1)^*, (y^2)^*$ whose tight sets are nested. Then, we can construct following basis:



Notice we obtained continuous-one property, and thus totally unimodularity. Therefore, $y^* \in \mathbb{Z}^m$ as the solution to the induced linear system, resulting for the primal in totally dual integrality by definition. □

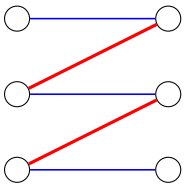
We remind ourselves of the duality of complexity and integrality: It remains to develop an efficient optimization algorithm. We restrict us to a cardinality (i.e. $c = \mathbf{1}$) matroid intersection algorithm:

Theorem 4.50. For two matroids $\mathcal{I}_1, \mathcal{I}_2$, we can solve

$$\max_{I \in \mathcal{I}_1 \cap \mathcal{I}_2} |I|$$

efficiently.

Observation 4.51. A (naive) greedy approach to bipartite matchings doesn't work:



The red edges form a maximum matching which isn't maximal like the blue set of edges.

Idea. In order to solve the issue of wrong greedy choices we want to imitate an augmenting path algorithm on a bipartite graph defined by the current maximal solution I and I^c as partition and some edges connecting I and I^c.

We want to use following definition to describe the edges of our matroid-intersection-graph:

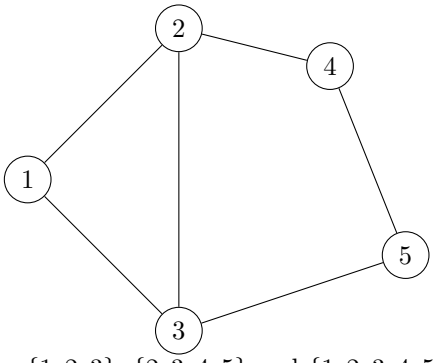
Definition 4.52 (Circuit)**.** A **circuit** C is a minimal dependent subset in a matroid.

Note that we can visualize a matroid $\mathcal{I}$ as a graph such that every acyclic subset is element of $\mathcal{I}$. A circuit is therefore always a cycle.

Example 4.53. The matroid over $E = \{1, 2, 3, 4, 5\}$,

$$\begin{aligned}\mathcal{I} = \{&\emptyset, \{1\}, \{2\}, \{3\}, \{4\}, \{5\},\\ &\{1,2\}, \{1,3\}, \{1,4\}, \{1,5\}, \{2,3\}, \{2,4\}, \{2,5\}, \{3,4\}, \{3,5\}, \{4,5\},\\ &\{1,2,4\}, \{1,2,5\}, \{1,3,4\}, \{1,3,5\}, \{2,3,4\}, \{2,3,5\}, \{2,4,5\}, \{3,4,5\},\\ &\{1,2,4,5\}, \{1,3,4,5\}\},\end{aligned}$$

can be visualized as



The circuits are given as $\{1,2,3\}$, $\{2,3,4,5\}$ and $\{1,2,3,4,5\}$.

Lemma 4.54. If $I \in \mathcal{I}$, but $I + e \notin \mathcal{I}$, then $I + e$ has a unique circuit.

Proof. We prove by contradiction. Suppose $I + e$ contains two distinct circuits C_1, C_2. Suppose w.l.o.g. that $|I|$ is minimal and $I+e = C_1 \cup C_2$. Then there must exist $a \in C_1 \setminus C_2$ and $b \in C_2 \setminus C_1$.

To construct our contradiction, it suffices to show now that

$$I' \coloneqq (C_1 \cup C_2) \setminus \{a, b\} \in \mathcal{I}.$$

Suppose this doesn't hold. Then there exist a circuit $C \subseteq \mathcal{I}'$. Then $(I-a)+e$ contains C_2 (since $a \notin C_2$) and C (since $I' \subseteq (I-a)+e$). But $|I-a| = |I|-1$ contradicts minimality of I. □

This lemma motivates following definition:

Definition 4.55. For every $I \in \mathcal{I}$ and $e \in E$ we define

$$C(I,e) = \begin{cases} \emptyset, & \text{if } I + e \in \mathcal{I} \\ \text{unique circuit in } I + e, & \text{else} \end{cases}$$

as the **fundamental circuit**.

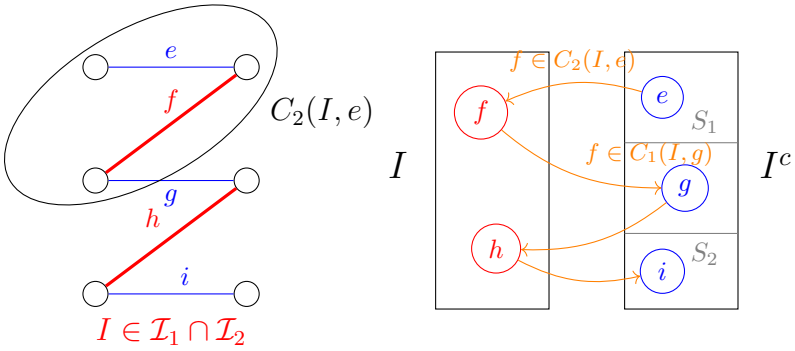
In particular, we want to consider fundamental circuits C_1 of $\mathcal{I}_1$ and C_2 of $\mathcal{I}_2$: Going back to our initial idea, for any $f \in I$ and $e \in I^c$, we now draw an edge iff $f \in C_1(I,e)$ or $f \in C_2(I,e)$.

Lecture 11
Tu 24 May 2022

Further consider

$$S_1 \coloneqq \{e \in I^c \mid I + e \in \mathcal{I}_1\},$$
$$S_2 \coloneqq \{e \in I^c \mid I + e \in \mathcal{I}_2\}.$$

Example 4.56. Consider a bipartite matching and its matroid-intersection graph for some independent set $I \subseteq \mathcal{I}_1 \cap \mathcal{I}_2$:



Assumption. *From now on assume $S_1 \cap S_2 = \emptyset$. Otherwise, if there exists an $e \in S_1 \cap S_2$, then we could choose $I + e \in \mathcal{I}_1 \cap \mathcal{I}_2$ greedily.*

Observation 4.57. Given $I \in \mathcal{I}_1 \cap \mathcal{I}_2$. Using circuits, we can swap edges such that we stay at least in $\mathcal{I}_1$: Suppose $e \in S_1$ (and $e \notin S_2$). Therefore, $C_2(I,e) - e \neq \emptyset$, meaning there is a $f \in C_2(I,e) - e$ which is also in I. In order to add e to I, we must remove f. Consider $I - f$. We add $g \in I^c$ such that $f \in C_1(I,g) - g$, because deleting f will remove the only possible circuit that can appear when adding g, therefore keeping $(I - f) + g \in \mathcal{I}_1$.

We want to use previous observation along an augmenting path from S_1 to S_2 in order to generate an I' that is also in $\mathcal{I}_2$.

Theorem 4.58. We prove two parts:

1. As long as there is an augmenting path from S_1 to S_2 (i.e. a path of edges in a bipartite matching), we can augment our solution.
2. If there is no augmenting path, we are optimal.

In order to prove the first part, we need a little bit more work first.

Definition 4.59. Given a bipartite graph $(\{e_0, \dots, e_k\} \cup \{f_1, \dots, f_l\}, E)$. A path

$$P = e_{i_0} \to f_{i_1} \to e_{i_1} \to \cdots \to f_{i_s} \to e_{i_s}$$

is **shortcut-free** if for every i_j there is no larger index $i' > i_j$ such that $(e_{i_j}, f_{i'}) \in E$ or $(f_{i_j}, e_{i'}) \in E$.

Lemma 4.60. Given a shortcut-free path, for all i holds

$$C_1(I + e_0, e_i) \subseteq I \setminus \{f_1, \dots f_{i-1}\} + e_0 + e_i$$

Proof. Suppose the opposite. Then for some i there exists a j with $1 \leq j < i$ such that $f_j \in C_1(I, e_i)$, therefore $(f_j, e_i) \in E$. But $f_j \to e_i$ is a shortcut. □

Definition 4.61. We define

$$I_i \coloneqq I \cup \{e_0, \dots, e_i\} \setminus \{f_1, \dots, f_i\}$$

as the **partial augmentation**.

Lemma 4.62. Given a shortcut-free full augmentation. For all i holds $I_i \in \mathcal{I}_1$.

Proof. We use induction over i. For $i = 0$ this follows by $e_0 \in S_1$. Assume now $I_{i-1} \in \mathcal{I}_1$ and consider two cases:

- $I_{i-1} + e_i \in \mathcal{I}_1$: Then $I_i = I_{i-1} + e_i - f_i \in \mathcal{I}_1$ by subset stability.
- $I_{i-1} + e_i \notin \mathcal{I}_1$: Then $C_1(I_{i-1}, e_i) \neq \emptyset$. Observe by Lemma 4.60 that

  $$C_1(I + e_0, e_i) \subseteq I \setminus \{f_1, \dots, f_{i-1}\} + e_0 + e_i \subseteq I_{i-1} + e_i.$$

  By uniqueness of circuits, $C_1(I + e_0, e_i) = C_1(I_{i-1}, e_i)$. As a result, $f_i \to e_i \in E$ iff $f_i \in C_1(I_{i-1}, e_i)$, concluding $C_1(I_{i-1}, e_i) - f_i \in \mathcal{I}_1$, and finally $I_i \in \mathcal{I}_1$.

□

Corollary 4.63. If we augment on a shortcut-free path P, then $I' \in \mathcal{I}_1 \cap \mathcal{I}_2$.

Proof. For $I' \in \mathcal{I}_1$ this follows directly from Lemma 4.62. For $I' \in \mathcal{I}_2$ we can derive an analoguous proof. □

This concludes part 1 of Theorem 4.58.

For the second part, we want to prove optimality after termination of part 1. Note that for $I \in \infty \cap \in$, the matroid rank function r_i and any $S \subseteq E$

$$|I^*| = |I^* \cap S| + |I^* \cap S^c| \leq r_1(S) + r_2(S^c),$$

thus, tightness induces optimality of I. We want to find such S for an maximal I:

Lemma 4.64. Suppose $I^* \in \mathcal{I}_1 \cap \mathcal{I}_2$ does not have any augmenting path. Let S be the set of nodes reachable via partial augmentation. Then

1. $|I^* \cap S^c| = r_2(S^c)$, and
2. $|I^* \cap S| = r_1(S)$.

Proof. Observe $S_1 \subseteq S^c$ and $S_2 \subseteq S$ by assumption of I^*. We prove both parts by contradiction.

1. Suppose strict inequality. By extensibility, we can add $e \in S^c \setminus I^*$ to $I^* \cap S^c$ such that $(I^* \cap S^c) + e \in \mathcal{I}_2$.

   Notice $e \notin S_2$, therefore $C_2(I^*, e) \neq \emptyset$ by definition, meaning there is $f \in C_2(I^*, e) - e \subseteq I^*$, but $f \notin I^* \cap S^c$. Otherwise, $C_2(I^*, e) \subseteq (I^* \cap S^c) + e$, and by subset stability $C_2(I^*, e) \in \mathcal{I}_2$, contradicting the circuit definition.

   As a consequence, $f \in I^* \cap S$, meaning f is not reachable by definition of S, but a path $e \to f$ exists by definition of our matroid intersection graph - contradiction!

2. Suppose strict inequality. By extensibility, we can add $e \in S \setminus I^*$ to $I^* \cap S^c$ such that $(I^* \cap S) + e \in \mathcal{I}_1$. Analoguous, we can derive there is an $f \in I^* \cap S^c$ with a path $f \to e$, meaning f is reachable, but e is not by definition - contradiction!

□

Corollary 4.65. If $I^* \in \mathcal{I}_1 \cap \mathcal{I}_2$ does not have an augmenting path, then it is optimal.

Remark 4.66. We can extend the cardinality case to arbitrary weighted intersection matroids. As a general idea, we can extend on the augmenting paths idea and additionally use successive shortest paths or the Hungarian Algorithm for a weighted bipartite matching. See [Coo+97, Ch. 8] for details.

Remark 4.67. We can use the same ideas for polymatroid and submodular intersection.

Unfortunately, matroid-intersection cannot be efficiently extended to arbitrary numbers of matroids. Intuitively, this can be explained because we cannot construct a basis with Continuous One Property by glueing the parts together as in the case with 2 matroids. This prohibits us from using Totally Dual Integrality.

Theorem 4.68. Triple-matroid intersection is in **NPC**, therefore cannot be solved efficiently.

Proof. We can reduce from HAMPATH. See the tutorial for the proof. □

As a closing thought, we want to have a last look at Totally Dual Integrality:

Theorem 4.69. If $Ax \leq b$ is rational, then there is $q \in \mathbb{Z}$ such that $q^{-1}Ax \leq q^{-1}b$ is Totally Dual Integral.

Nonetheless, this fact is useless since $q^{-1}b$ would have to be integral to get integer optimal solutions.

Theorem 4.70. Given a rational polyhedron. Iff it can be written as a Totally Dual Integral system where b is integral, then the polyhedron is integral.

Proof idea. We can add redundant constraints to "help" the dual. □

Conclusion 4.71. Totally Dual Integrality is a property of the linear system, not the polyhedron.

Lecture 12
Th 02 June 2022

4.3 Combinatorial algorithms for IP

We already considered polynomial algorithms for bipartite matchings, but can we extend this to general graphs?

Recall. We defined a (weighted) matching via IP as

$$\begin{aligned} \max_x \quad & w^T x \\ \text{s.t.} \quad & \sum_{e \in \delta(v)} x_e \leq 1, \quad \forall v \in V, \\ & x \in \mathbb{B}^E \end{aligned}$$

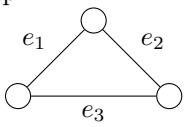
For the linear relaxation, we can find non-bipartite graphs with non-integral optimal solutions.

Idea. To fix the issue of fractional vertices, we want to add cutting planes that cut only these fractional vertices off. More general, we want to cut off odd cycles by introducing constraints

$$\sum_{e \in E(S)} x_e \leq \left\lfloor \frac{|S|}{2} \right\rfloor$$

for every $S \subseteq V$ with odd cardinality. We will call these **blossom constraints**.

Example 4.72. For following graph



is $[0.5, 0.5, 0.5]^T$ an optimal vertex for the LP relaxation, but introducing

$$x_{e_1} + x_{e_2} + x_{e_3} \leq 1$$

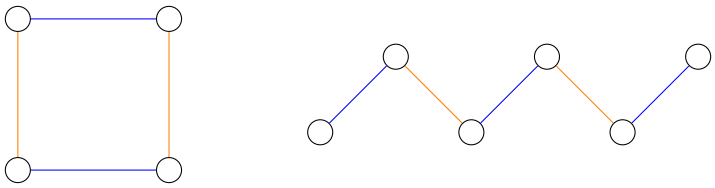
would cut this solution.

Definition 4.73. Given a graph $G = (N, E)$ and matching M.

1. A path is **alternating** w.r.t. M if its edges alternate between M and $E - M$.
2. A node is **exposed** if no edge in M hits the node.
3. A path is **augmenting** if it is alternating and both endpoints are exposed.

Theorem 4.74. A matching M is not maximum iff there exist an augmenting path w.r.t. M.

Proof. The reverse is clear. Thus, consider a non-maximum matching M. Then there is a matching M' with $|M'| = |M| + 1$. Let $D = M \Delta M'$, then D is composed from paths or even-sized cycles only, e.g.



We see that

$$|D| = |M| + |M'| - 2|M \cap M'| = 2(|M| - |M \cap M'|) + 1$$

is odd. This implies that there is an alternating path of odd length. Note $|M' - M| > |M - M'|$, so there is an M-augmenting path. □

Using this theorem might enable us to create an algorithm!

Recall. The cardinality of a matching is at most the cardinality of a node cover. For the special case of bipartite graphs, König's theorem state that they are even equal.

Theorem 4.75 (**Tutte-Berge**)**.** For $G = (N, E)$,

$$\max_{\text{matching } M} |M| = \min_{X \subseteq V} \frac{1}{2}(|V| - \text{oc}(X) + |X|)$$

where $\text{oc}(X)$ is the number of odd components in $G(V - X)$.

Proof of $\leq$. Let $X \subseteq V$ be such that $G(V - X)$ has k odd components H_i. Let M be a matching. For each H_i, either there exist a M-exposed node, or there exist an edge in M from H_i to X.

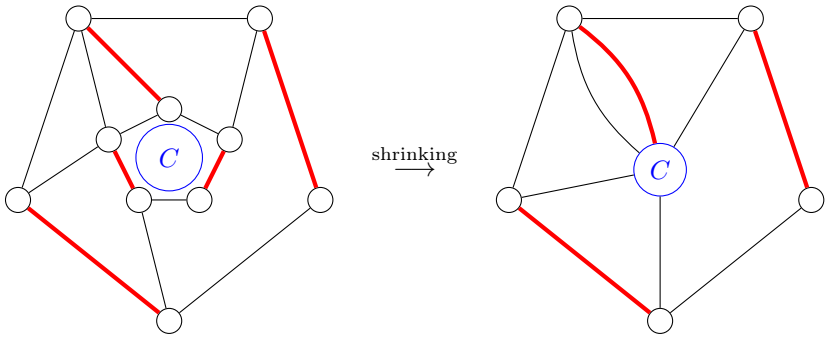
The number of such edges is at most $|X|$, therefore there must be at least $k - |X|$ exposed nodes. Because these nodes are by definition not in the matching, there remain only at most the right side of our equation many matched edges. □

Proof of $\geq$. See tutorial or Corollary 4.85. □

Definition 4.76 (Cycle-shrinking)**.** For an odd cycle C in G, we define $G' := G \times C$ to be the graph that shrinks C to a single node, but maintains all external (possibly multi-)edges.

Furthermore, if we have a matching in G', we define its **extended matching** in G to be the original edges, plus all possible edges from C.

Example 4.77. A possible shrinking operation can be:



Additionally, a matching M' in the shrinked graph and its corresponding extended matching M in the original graph is shown.

Observation 4.78. The number of M'-exposed nodes in G' is retained by the extended matching M.

Warning. By this procedure, it is generally *not* true that M is maximum iff M' is maximum. Therefore, we need to circumvent this issue in order to use this fact.

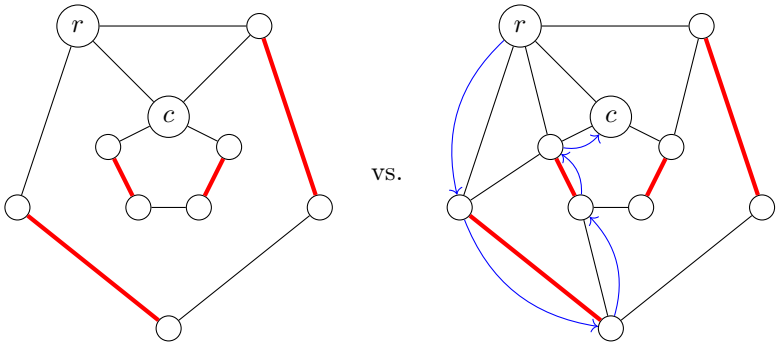
Theorem 4.79. Given G, M, and an odd cycle C such that

$$|M \cap C| = \frac{|C| - 1}{2}.$$

Let $c \in C$ be the one not covered within C. Suppose that for any M-exposed node r, all alternating $r - c$-paths are disjoint from $V(C) - c$.

Then, M is maximum in G iff M' is maximum in G'.

Example 4.80. Following graphs illustrate when the conditions of the theorem hold and do not hold for some matching M, with a bad $r - c$-path.



Proof. First, suppose M is maximum, but M' is not. Then $|M'| = |M| - k$ with

$$k := |M \cap C| = \frac{|C| - 1}{2}.$$

If there exists a matching N' in G' with $|N'| > |M'|$, then there exists a matching N in G with

$$|N| = |N'| + k > |M|$$

Contradiction!

For the other direction, suppose M' is maximum, but M is not. Then there exists an augmenting path P in G. Let v be an endnode of P that is not in $V(C)$, and consider two cases:

- $P \cap C = \emptyset$: Let w be the other endnode.
- $P \cap C \neq \emptyset$: Then $P \cap C$ consists only of the unmatched node c inside of C, otherwise we could construct an alternating $r - c$-path starting from one of the end nodes of P. Again, let w be the other endnode.

As a consequence, the $v - w$-path is M'-augmenting - contradiction! □

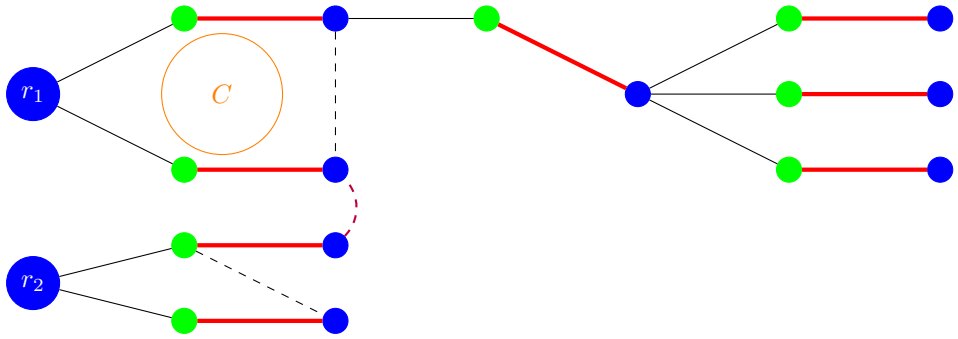
ask why script seems wrong

Algorithm 4: Alternating tree-algorithm

Graph G, matching M, exposed node r
$A \leftarrow \emptyset$
$B \leftarrow \{r\}$
while *for any* $w \in V, u \in B, v \notin A \cup B$*:* $(u,v) \in E, (v,w) \in M$ **do**
  $A \leftarrow A + v$
  $B \leftarrow B + w$
end

Definition 4.81. The tree we get by this algorithm is called M-**alternating tree**.

Example 4.82. An exemplary run of the algorithm with two roots r_1, r_2, a matching M, and one odd cycle C could yield following alternating forest,



with A and B colored correspondingly. Non-tree edges are left dashed. The purple edge could be used to connect the two roots. (Actually, going from r_1, the node colors for the part after this edge should be swapped. Same for r_2 the other way round)

Observation 4.83. Using this algorithm we see

- A (B) contains all nodes that are endnodes of an odd (even)-length M-alternating path on T,
- every node other than r is covered by an edge in $M \cap E(T)$,
- every path from r to any $v \in V(T)$ in T is alternating,
- $|B| = |A| + 1$

Additionally, we can conclude that any $(u,v) \in E$ with $u, v \in B$ forms an odd cycle C - shrinking it to node c (and adding c to B) does not affect our structure.

Finally, if M-alternating trees from two roots r_1, r_2 share an $(u, v) \in E$ with $u \in B_1, v \in B_2$, then there exists an M-augmenting path from r_1 to r_2.

Lecture 13
Tu 07 June 2022

Now all preliminaries are set to construct the algorithm we long for.

Algorithm 5: Blossom algorithm for maximum matching

Graph $G = (V, E)$, matching M
Maintain M-alternating forests for all M-exposed nodes as roots. **while** *any following case can still occur* **do**
  if $v \in V$ *can extend tree* **then**
    extend this tree with corresponding edge (u, v)
  end
  if *blossom exists* **then**
    shrink blossom and adjust M
  end
  if *M-augmenting path exists between two roots* **then**
    augment M along this path
  end
end
Deshrink blossoms and readjust M
return M

Theorem 4.84. If algorithm 5 stops, its output is a maximal matching.

Proof. Let M be the matching after termination and $T_1, \dots, T_k$ its M-alernating trees. Let $A_1, \dots, A_k$ ($B_1, \dots, B_k$) be the set of nodes with odd (even) distance from the corresponding tree's root. Consider $A \coloneqq \dot{\bigcup} A_i$ and $B \coloneqq \dot{\bigcup} B_i$. Then all nodes in $V \setminus (A \cup B)$ must be already matched. Therefore, all its components are of even size (and form a perefect matching by itself). Since the algorithm stopped, there are no edges between nodes in B (otherwise there is an augmenting path or blossom). So, B has $|B|$ odd components. Therefore $\operatorname{oc}(A) = |B|$. Recall $|B_i| = |A_i| + 1$, and being disjoint unions $\operatorname{oc}(A) - |A| = k$. The current matching has exactly its k roots as M-exposed nodes. By Theorem 4.75, the cardinality of any matching is at most

$$\begin{aligned}\min_{X \subseteq V} \frac{1}{2}(|V| - (\operatorname{oc}(X) - |X|)) &= \frac{1}{2}(|V| - \max_{X \subseteq V}(\operatorname{oc}(X) - |X|)) \\ &\leq \frac{1}{2}(|N| - (\operatorname{oc}(A) - |A|)) \\ &= \frac{1}{2}(|N| - k) = |M|\end{aligned}$$

which proves maximality of $|M|$ for the shrunk graph. By Theorem 4.79, the reextended matching is also maximal in the original graph. □

Corollary 4.85. This proves the $\geq$-direction of Theorem 4.75.

Remark 4.86. The naive running time is given by $O(n)$ augmenting steps which extends $O(n)$ trees and shrinks $O(n)$ blossoms each. Since every blossom shrinking needs $O(n)$ our running time is $O(n^3)$.

However, using more efficient tree data structures can improve the running time to $O(mn \log n)$. Furthermore, ensuring augmentations use only shortest augmenting paths further reduces the time to $O(\sqrt{n} \cdot m)$. See [Coo+97, Thm. 5.11].

Now let us try to generalize this method to weighted matchings.

Theorem 4.87. Following problems are equally hard to solve:

1. min-cost perfect matching (i.e. $c^T x$), and
2. max-weight general matching (i.e. $w^T x$).

Proof for 2. $\implies$ 1. Set $w^0 := -c$ and let $w_{\min}$ be the smallest entry of w^0. If $w_{\min} < 0$, set $w^1 := w^0 - w_{\min} \cdot \mathbf{1}$ to ensure $w^1 \geq 0$ (otherwise $w^1 := w^0$). Let $w_{\max}$ be n times the largest entry of w^1 and define $w^2 := w^1 + w_{\max} \cdot \mathbf{1}$.

We show: Finding a max-weight general matching on $(w^2)^T x$ is also a min-cost perfect matching. Suppose there is an augmenting path P. Then the cost change by augmenting is given by

$$\begin{aligned}\sum_{e \in P \setminus M} w_e^2 - \sum_{e \in P \cap M} w_e^2 &= \sum_{e \in P \setminus M} w_e^1 + w_{\max} - \sum_{e \in P \cap M} w_e^1 + w_{\max} \\ &= w_{\max} + \underbrace{\sum_{e \in P \setminus M} w_e^1}_{\geq 0} - \underbrace{\sum_{e \in P \cap M} w_e^1}_{< w_{\max}} > 0.\end{aligned}$$

As a consequence, we will always choose a perfect matching if one exists. By definition and because the number of chosen edges is fixed, it must have minimal cost. □

Proof for 1. $\implies$ 2. Set $c = -w$. Define the graph $G' = (V', E')$ as two merged copies G^1, G^2 of the original G such that G' also includes self-crossing-edges, i.e.

$$\begin{aligned}V' &= V^1 \dot\cup V^2, \\ E' &= E^1 \dot\cup E^2 \dot\cup \{(i^1, i^2) \mid i \in V\}.\end{aligned}$$

Then we can always choose a perfect matching by mirroring a possibly non-perfect matching of G in G^1 and G^2, and fill unmatched nodes with these zero-cost crossing edges. □

Consider the LP relaxations of the two variants of weighted matchings

$$\begin{aligned} \max_x \quad & w^T x \\ \text{s.t.} \quad & x(\delta(\{i\})) \le 1, && \forall i \in V, \\ & x(\gamma(S)) \le \frac{|S|-1}{2}, && \forall S \subseteq V,\ |S| \text{ odd}, \\ & x \ge 0 \end{aligned} \tag{3}$$

and

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & x(\delta(\{i\})) = 1, && \forall i \in V, \\ & x(\delta(S)) \ge 1, && \forall S \subseteq V,\ |S| \text{ odd}, \\ & x \ge 0. \end{aligned} \tag{4}$$

We call the second set of constraints **blossom constraints**. These are supposed to make $1/2$-integral solutions (and therefore weird things happening in the blossoms) infeasible. Now also consider their duals

$$\begin{aligned} \min_{y,z} \quad & \sum_i y_i + \sum_S \frac{|S|-1}{2} z_S \\ \text{s.t.} \quad & y_i + y_j + \sum_{S: i,j \in S} z_s \ge w_{i,j}, \quad \forall (i,j) \in E, \\ & y, z \ge 0 \end{aligned} \tag{5}$$

and

$$\begin{aligned} \max_x \quad & \sum_i y_i + \sum_S z_S \\ \text{s.t.} \quad & y_i + y_j + \sum_{S: i \in S \text{ or } j \in S} z_S \le c_{i,j}, \quad \forall (i,j) \in E, \\ & z \ge 0, \\ & y \text{ free}. \end{aligned} \tag{6}$$

Using these formulation we want to find to reuse the blossom shrinking idea to construct the desired algorithm. Note that we can assign z_s to blossoms and set $y_S := z_S$ for the pseudonode S after shrinking. Consider the reduced cost for an edge $e = \{i, j\}$:

$$(5): \qquad y_i + y_j + \sum_{S: i,j \in S} z_S - w_{i,j} \ge 0$$

$$(6): \qquad c_{i,j} - y_i - y_j - \sum_{S: i \in S \text{ or } j \in S} z_S - w_{i,j} \ge 0$$

Either this term or $x_{i,j}$ must be 0 by complementary slackness. So, $e \in M$ only if it has reduced cost 0. Thus, forest groth is blocked by lack of such edges. The idea now is to update the dual variables in the blossom algorithm, but the details are omitted.

Theorem 4.88. The runtime of this algorithm is $O(n^3)$.

Corollary 4.89. Both primal formulations (3), (4) have integral vertices.

Theorem 4.90. It holds that (5) is totally dual integral, whereas (6) is not and has $1/2$-integral solutions.

Lecture 14
Th 09 June 2022

We know how to efficiently find optimal general matchings, implying we can also efficiently solve the separation problem with blossom contraints.

Theorem 4.91. There is an efficient non-Ellipsoid separation algorithm for blossom constraints.

Proof. The algorithm can be found in [Coo+97, Ch. 6.8]. The rough idea is to use submodularity. □

We want to have a look at further matching-like problems.

Definition 4.92. Given an undirected graph $G = (V, E)$. We want to find a min-cost walk traversing every edge at least once and finishing where we started. This problem is called **Postal Delivery Problem**, or archaically **Chinese Postman Problem**.

Remark 4.93. In the optimal case, every edge needs only to be traversed once, leading to the lower bound $\sum_{e \in E} c(e)$. The decision problem if this is possible is called **Euler Tour**, or historically **Königsberg Bridge Problem**.

Fact 4.94. Given graph G. An Euler Tour exists in G iff all nodes have even degree and G is connected.

Definition 4.95. We call G **Eulerian** if G has an Euler Tour.

What remains to be analyzed for the Postal Delivery Problem is the case if G is not Eulerian. For this, let T be the set of nodes with odd degree. We want to find a min-cost subset $J \subseteq E$ such that $E \uplus J$ is Eulerian, because this J can be interpreted as the edges we need to traverse twice (one can show there are optimal solutions with this property).

But in fact, nothing restricts us to let T be any even subset of nodes:

Definition 4.96. For a graph $G = (V, E)$, consider even-sized $T \subseteq V$ and $J \subseteq E$. If

$$T = \{v \in V \mid \deg_J(v) \text{ is odd}\},$$

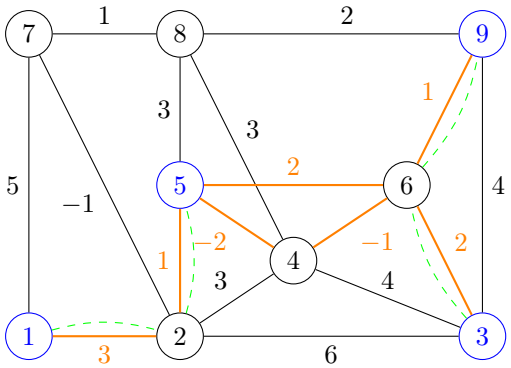
we call J a T-join. For an additional cost vector c, the corresponding optimization problem of finding a minimal T-join is given by

$$\begin{aligned} \min_J \quad & c(J) \\ \text{s.t.} \quad & J \text{ is } T\text{-join} \end{aligned}$$

Remark 4.97. Applications of this are

1. our postal delivery problem using all odd nodes,
2. for $T = \{s, t\}$, an optimal T-join is an $s - t$-path with negative costs allowed,
3. for even $|N|$ and $T = N$, a T-join is a min-cost general matching.

Example 4.98. Let's construct a T-join, for $T = \{1, 3, 5, 9\}$:



Then the orange edges J form a T-join, but while cheaper, this isn't minimal like the green edges J'.

Theorem 4.99. If J is a minimal (w.r.t. to its choice of edges) T-join, then J is a collection of $|T|/2$ edge-disjoint paths.

Proof. We prove by induction over even size $|J|$. Base case 0 is trivial. Consider $t \in T$ and let C be a connected component of (V, J) cont.t. Note $\delta_J(t)$ is odd, and $\sum_{i \in C} \delta_J(i)$ is even, therefore there exists another $s \in C$ with $\delta_J s$ odd. By removing $\{s, t\}$ from T and the $s - t$-path from J, we keep our T-join property, enabling us to use our induction step. □

Theorem 4.100. Let $s, t \in T$ connected by path P in a T-join J. Suppose there exists a cheaper $s-t$-path P'. Then, J is not an optimal T-join.

Proof. We show that

$$J' \coloneqq (J - P)\Delta P'$$

is a better T-join. Deleting P will result only in s, t changing parity. Adding P' then again changes parity of s, t, either by adding or deleting exactly one edge. For the other nodes in P', either $0, 1$, or 2 edges are also in Q, but every case doesn't change parity again. So, we maintain T-join-property.

Calculating everything further yields

$$\begin{aligned} c(J') &\leq c(J) + c(P' \setminus P) - c(P \setminus P') \\ &= c(J) + (c(P' \setminus P) - c(P' \cap P)) - (c(P \setminus P') - c(P' \cap P)) \\ &= c(J) + \underbrace{c(P') - c(P)}_{<0} < c(J) \end{aligned}$$

as wished. □

Corollary 4.101. If $c \geq 0$, then optimal J consists of $|T|/2$ shortests paths.

Consider now negative costs c.

Definition 4.102. Let $M \subseteq E$ be the set of negative-valued edges, and define

$$T^M \coloneqq \{i \in \mathbb{N} \mid \delta_M(i) \text{ is odd}\}.$$

Remark 4.103. M is a T^M-join.

Lemma 4.104. If J is a T-join and J' is a T'-join, then $J\Delta J'$ is a $(T\Delta T')$-join.

Proof. Let $v \in T\Delta T'$. Then w.l.o.g. $v \in T$ and $v \notin T'$, so $\deg_J(v)$ is odd, while $\deg_{J'}(v)$ is even. Because only edges that are both in J and J' cancel each other out in $J\Delta J'$, it holds $\deg_{J\Delta J'}(v) \equiv \deg_J(v) + \deg_{J'}(v) \equiv 1 \pmod 2$.

Otherwise, $v \notin T\Delta T'$, so v is either in both or none of J, J'. An analoguous argument shows $\deg_{J\Delta J'}(v) \equiv \deg_J(v) + \deg_{J'}(v) \equiv 0 \pmod 2$. □

Theorem 4.105. J is an optimal T-join w.r.t. cost vector c iff $J\Delta M$ is an optimal $(T\Delta T^M)$-join w.r.t. cost vector $|c|$.

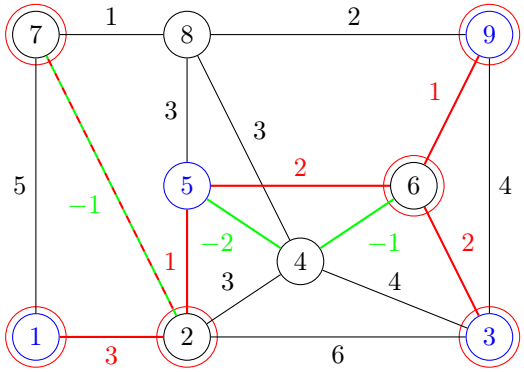
Proof. By Lemma 4.104 and Remark 4.103 we can form the symmetric difference of any side and get the other side as a corresponding join (note $A\Delta B\Delta B = A$). It remains to prove optimality. Calculations show:

$$\begin{aligned} c(J) &= c(J \setminus M) + c(J \cap M) + \underbrace{c(M \setminus J) - c(M \setminus J)}_{=0} \\ &= \underbrace{c(J \setminus M)}_{\text{only positive}} + c(M) - \underbrace{c(M \setminus J)}_{\text{only negative}} \\ &= |c|(J\Delta M) + c(M) \end{aligned}$$

So, minimizing one side also minimizes the other side, which immediately proves our equivalence (note $c(M)$ is constant). □

Corollary 4.106. Calculating an optimal T-join is in **P**.

Example 4.107. Building upon previous example, again consider $T = \{1, 3, 5, 9\}$, but note now $T^M = \{2, 5, 6, 7\}$ and $T\Delta T^M = \{1, 2, 3, 6, 7, 9\}$.



Note that the red edges form an optimal $T\Delta T^M$-join. Taking its symmetric difference with M directly yields our initial (optimal) T-join.

We still need an LP-formulation, though. We will utilize following definition for this:

Definition 4.108. If $S \subseteq E$ such that $|S \cap T|$ is odd (even), we call S **T-odd** (**T-even**).

Theorem 4.109. If $c \geq 0$, then solving

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & x(\delta(S)) \geq 1, \quad \forall\ T\text{-odd } S \subseteq E, \\ & x \geq 0 \end{aligned}$$

yields an optimal T-join.

Proof. See [Coo+97, Thm. 5.28]. □

However, there are two problems:

1. It is only $\frac{1}{2}$-integral and not totally dual integral.
2. If c arbitrary, then it could be unbounded.

We can circumvent these problems with a tighter LP. Consider following motivation: Let $S \subseteq N$ and $F \subseteq \delta(S)$. If S is T-odd and $|F|$ even, then $|\delta(S) \setminus F|$ is odd. This means, if J is a T-join and uses all edges in F, then it must also use at least one edge from $\delta(S) \setminus F$. Analoguous, if S is T-even, we conclude $|F|$, and get to the same result. We encode this property as follows:

why is $\delta(S)$ always odd?

Theorem 4.110. For any c, the LP

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & x(\delta(S) \setminus F) \geq 1 + x(F) - |F|, \quad \forall S, F, |S \cap T| + |F| \text{ odd}, \\ & 1 \geq x \geq 0 \end{aligned}$$

is Totally Dual Integral and yields an integral optimal T-join.

Proof. Note the right side is equal to 1 iff all edges of F are used. For the actual proof, see [Coo+97, Thm. 5.30]. □

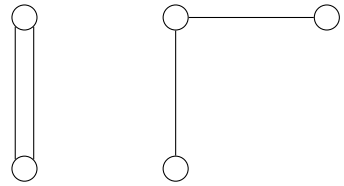
Lecture 15
Tu 14 June 2022

content lec15

Definition 4.111. We can generalize matchings as follows:

- Replacing $x(\delta(i)) \leq 1$ by $x(\delta(i)) \leq b$ yields b-matchings.
- Additionally introducing upper bounds $x_e \leq u_e$ is called (b, u)-matching.

Example 4.112. A 2-matching is given by



Notice how perfect $(2,1)$-matchings are close to TSP. Suppose $S \subseteq V, F \subseteq \delta(S)$ such that $b(S) + u(F)$ is odd. Further, suppose $b(S)$ is even. Then on one hand, $u(F)$ is odd, on the other hand $x(\delta(S))$ is even. Then $x(F) \neq x(\delta(S))$. Thus

$$x(\delta(S) \setminus F) \geq 1 + x(F) - u(F)$$

Analoguous, the same follows for $b(S)$ odd.

A perfect (b,u)-matching is given by the LP:

$$\begin{aligned} \min_x \quad & c^T x \\ \text{s.t.} \quad & x(\delta(i)) = b_i, && \forall i, \\ & 0 \leq x \leq u, \\ & x(\delta(S) - F) \geq 1 + x(F) - u(F), && \forall S, F \text{ s.t. } b(S) + u(F) \text{ odd} \end{aligned}$$

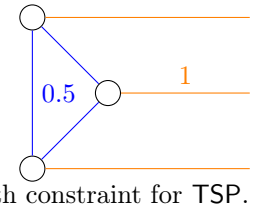
There is also an equivalent form: If b is even, then $b(H)$ is also even.

handle teeth stuff

$$x(E(H)) + x(T) \leq |H| + \left\lfloor \frac{|T|}{2} \right\rfloor$$

Note. For 2-matching, we have exponentially many constraints. These can be separated in polynomial time. This, however, does *not* mean we can optimize TSP in polynomial time with Ellipsoid, since there could be non-integer optimal solutions (or: we need more constraints, and these make TSP **NP**-complete).

Example 4.113. Consider following part of a 2-matching, where same colors have same edge values in the solution:



This violates our handle-teeth constraint for TSP.

4.4 Fixed dimension IPs

Theorem 4.114. For a fixed number of variables, or a fixed number of constraints, there exists a polynomial algorithm that solves IP.

Proof. The proof requires lattice theory. See [NW99, Ch. 2-6.5, Thm. 5.4+5] for details. □

4.5 Approximation algorithms

Recall. Greedy on independent systems, which are not matroids, are suboptimal, but can be used for approximated solutions, see Theorem 4.32.

Definition 4.115. For a maximisation problem and (polynomial) algorithm A, if

$$\max \frac{z^*}{\text{worst-case result of } A} \leq \alpha$$

we call A an **α-approximation** algorithm.

Example 4.116. Matching can be solved by Greedy as a 2-approximation.

Definition 4.117. There exist several classes of approximation algorithms:

- **FPTAS** (**Fully Polynomial Time Approximation Scheme**): There exists an algorithm polynomial in $|I|$ and $1/\varepsilon$ to get an $(1+\varepsilon)$-approximation.
- **PTAS** (**Polynomial Time Approximation Scheme**): There exists an algorithm polynomial in $|I|$ and potentially exponential in $1/\varepsilon$ to get an $(1+\varepsilon)$-approximation.
- **Fixed Factor**: There exists a threshold β such that there is a polynomial algorithm for all $\alpha \geq \beta$, but it is **NP**-hard to approximate for $1 \leq \alpha < \beta$.
- Inapproximate **NP**-hard to approximate for any $\alpha > 1$.

what does this mean? Inapproximate

4.6 Rounding non-integer solutions

Idea. Solve LP, get x^{LP}, and round up or down to get a solution x^{IP}.

This is a bad idea in general:

Theorem 4.118. Naive rounding of x^{LP} can introduce arbitrarily large errors (see [KV18, Thm. 5.7] for a more exact version).

Proof. We can construct following bad instance depending on some parameter α, such that the gap between the rounded optimal LP solution and the actual optimal IP solution

scales with α. □

lattice plot

Remark 4.119. Not all instances can be rounded out-of-the-box to a feasible solution.

sharp triangle plot non-roundable

4.7 Cutting Planes

Recall our initial idea was to compute the integer hull P^I of P, which is however NP-hard. But, instead of computing *all* of P^I, in most cases it suffices to calculate P^I "near" x^I. Thus, we want to find a cutting plane $\alpha^T x > \beta$ such that

1. x^{LP} violates the plane, i.e. $\alpha^T x^{\mathsf{LP}} > \beta$, and
2. all of P^I satisfies the cutting plane, i.e. for all $x \in P \cap \mathbb{Z}^n$ should hold $\alpha^T x^{\mathsf{LP}} \leq \beta$

Idea. Find Cutting Planes, add to LP (reoptimize via dual simplex), and repeat until we get x^{IP}.

Finding cutting planes can be done via SEP.

Remark 4.120. There is a technical issue with convergence. While we know, in general we cannot guarantee polynomial running time, it is also possible that P^I isn't even a polyhedron:

Because of $\sqrt{2}$ being irrational, there will be infinitely feasible integer points getting infinitely close, but never touching it. This cannot be handled with finitely many constraints.

plot irrational constraint

Theorem 4.121 (Fundamental theorem of IP)**.** For rational A, b and $P = \{x \mid Ax \leq b\}$ holds that P^I is a polyhedron.

Proof. See [KV18, Thm. 5.1] □

Lecture 16
Th 16 June 2022

content lec16

Question 4.122. How do we find Cutting planes?

Consider the Simplex Tableau, and some $x^{\mathsf{LP}} \notin \mathbb{Z}^n$, e.g. $x_1^{\mathsf{LP}} \notin \mathbb{Z}$. Then, $x_1^{\mathsf{LP}} > 0$, and it must be in the basis. Row 1 of the tableau therefore looks like:

$$\begin{aligned} x_1 + \bar{a}_{1,N}^T x_N &= \bar{b}_1 \\ \implies x_1 + \left\lfloor \bar{a}_{1,N}^T \right\rfloor x_N &\leq \bar{b}_1 \end{aligned}$$

If $x \in P^I$, then the left-hand side is integral, so

$$x_1 + \left\lfloor \bar{a}_{1,N}^T \right\rfloor x_N \leq \left\lfloor \bar{b}_1 \right\rfloor$$

and for their difference

$$f(\overline{a}_{1,N}^T)x_N \geq f(\overline{b}_1)$$

such that f maps to the fractional part of its input. This is clearly feasible for all integral vertices in P, and thus also for P^I. But x^{LP} violates it, since the left-hand side is equal to 0, while the right-hand side must be positive - we found our cutting plane! This particular type is called **Gomory cut**.

Fact 4.123. Repeatedly finding Gomory cuts leads in a finite number of iterations to an optimal integral solution.

By 1963 though, Gomory cuts were abandoned, because there were issues that could not be resolved:

Remark 4.124. Even though they can be easily generated by an optimal simplex tableau, in practice Gomory cuts appear to be slow. Additionally, this procedure can be numerically unstable because of rounding errors.

Maybe we need to think larger - let's try to generalize Gomory Cuts: Starting with $Ax \leq b$, we introduce row multipliers $y \in \mathbb{R}^m$ such that

make more clean

$$\begin{aligned} y^T Ax &\leq y^T b \\ y &\geq 0 \\ y^T A &\in \mathbb{Z}^n \\ y^b &\text{ fractional} \end{aligned}$$

Then

$$y^T Ax \leq \lfloor y^T b \rfloor$$

is a cutting plane, called a **Chvátal Cut**, or more general **Chvátal-Gomory Cut**.

Fact 4.125. The multiplier y is a succinct certificate that this is a valid cutting plane for P^I. This is also called a **Chvátal proof**.

Example 4.126. We try to derive a possible Chvátal proof for general matchings. Recall the base-LP

$$\begin{aligned} \max_x \quad & w^T x \\ \text{s.t.} \quad & x(\delta(i)) \leq 1, \quad \forall i \in V, \\ & x \geq 0 \end{aligned}$$

which produces non-integral solutions in general, e.g. Example 4.72. Therefore, we needed to introduce the blossom constraints for all odd $S \subseteq E$ as cutting planes:

$$x(E(S)) \leq \left\lfloor \frac{|S|}{2} \right\rfloor$$

For Chvátal, we now sum the first constraint over all $i \in S$,

$$2x(E(S)) + x(\delta(S)) \leq |S|,$$

and sum the second constraint multplied by $-\frac{1}{2}$ for all $e \in \delta(S)$

$$-x(\delta(S)) \leq 0,$$

In total and divided by 2, we get

$$x(E(S)) \leq \frac{|S|}{2}.$$

Because all properties hold, we can round down and get our Chvátal plane. While in this case these CPs are already enough to cut down to P^I, in general it usually is more complicated.

Let's find some properties about Chvátal-Gomory cuts.

Theorem 4.127. Define P' as P with *all* possible Chvátal-Gomory cuts added. Then P' is a polyhedron.

tikz graphic

Proof. As a general idea, we will show that every Chvátal cut is equivalent to a Chvátal cut generated from a finite set S. Then, it follows directly that P' is still a polyhedron.

For S, we define

$$S \coloneqq \{x \mid \exists 0 \leq y < 1 : x = y^T A \wedge y^T A \in \mathbb{Z}^n\}$$

which is obviously finite.

Suppose there is a "big" $\overline{y}$ that defines any Chvátal cut via

$$(\overline{y}^T A)x \leq \lfloor \overline{y} b \rfloor .$$

We define

$$\tilde{y} \coloneqq f(\overline{y}).$$

Note $0 \leq \tilde{y} < 1$. Calculations show:

$$\tilde{y}^T A = \underbrace{\overline{y}^T A}_{\text{integral}} - \underbrace{\lfloor \overline{y} \rfloor A}_{\text{integral}}$$

which implies $\tilde{y} \in S$. Also,

$$\tilde{y}^T b = \overline{y}^T b - \underbrace{\lfloor \overline{y} \rfloor b}_{\text{integral}}$$

So, $\tilde{y}^T b$ and $\overline{y}^T b$ have the same fractional part, or

$$\lfloor \tilde{y} b \rfloor = \lfloor \overline{y} \rfloor b = \lfloor \overline{y} b \rfloor$$

finish proof

See also [Coo+97, Thm. 6.34]. □

Theorem 4.128. There is an $k \in \mathbb{N}$ such that repeatedly calculating Chvátal closure P_{i+1} from P_i (that is, $P_{i+1} := P_i'$) will lead to $P^I = P_k$. We call this k the **Chvátal rank** of P_1.

Lecture 17
Th 23 June 2022

content lec17

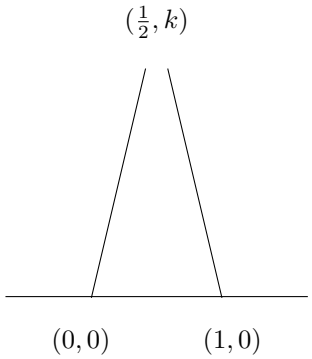
write lec17

Lecture 18
Tu 28 June 2022

content lec18

Conclusion 4.129. The Chvátal rank is *always* finite.

Remark 4.130. Nonetheless, the Chvátal rank can be arbitrarily large. Consider:



Here, the Chvátal rank is $2k$.

We can also give similar qualitative measures for cuts:

Definition 4.131. If the cutting plane $\alpha^T x \leq \beta$ is valid for P^k, but not P^{k-1}, then this cutting plane has **Chvátal** rank k.

Furthermore, we say the cutting plane is **facet-inducing** for P^I if the CP contains a facet of P^I.

It is simpler when P^I is full-dimensional, i.e. $\dim(P^I) = n$.

It remains to show how we can solve MIPs. Recall the fundamental theorem of MIP Theorem 4.121. Consider following simple-looking MIP for some fractional b:

$$\begin{aligned} x + y &\geq b \\ x &\in \mathbb{R}_{\geq 0} \\ y &\in \mathbb{Z}_{\geq 0} \end{aligned}$$

How can we cut off non-integer points without violating the continuity of x? Define a line

$$l = \{(x, y) \mid x + f(b)y \geq f(b)\lceil b \rceil\}$$

We see this yields our integer hull. What about the general case, that is, $x \in \mathbb{R}^n_{\geq 0}, y \in \mathbb{Z}^p_{\geq 0}$?

$$\begin{aligned} & \sum_j a_j x_j + \sum_k d_k y_k \geq b \\ \Leftrightarrow \quad & \left(\underbrace{\sum_{j:a_j \leq 0} a_j x_j}_{\leq 0} + \sum_{j:a_j > 0} a_j x_j \right) + \left(\sum_k \lfloor d_k \rfloor y_k + \sum_{k:f(d_k) < f(b)} f(d_k) y_k + \sum_{k:f(d_k) \geq f(b)} f(d_k) y_k \right) \geq b \\ \Leftrightarrow \quad & \underbrace{\sum_{j:a_j > 0} a_j x_j + \sum_{k:f(d_k) < f(b)} f(d_k) y_k}_{\in \mathbb{R}_{\geq 0}} + \underbrace{\sum_k \lfloor d_k \rfloor y_k + \sum_{k:f(d_k) \geq f(b)} y_k}_{\in \mathbb{Z}_{\geq 0}} \geq b \\ \Leftrightarrow \quad & \sum_{j:a_j > 0} a_j x_j + \sum_{k:f(d_k) < f(b)} f(d_k) y_k + f(b) \sum_k \lfloor d_k \rfloor y_k + \sum_{k:f(d_k) \geq f(b)} y_k \geq f(b)\lceil b \rceil \end{aligned}$$

We call this **Gomory Mixed Integer Cuts**, shorthand GMI, and **Mixed Integer Rounding**, shorthand MIR. Consider

$$\sum_j a_j^+ x_j + \sum_k \min(f(d_k), f(b)) y_k + f(b) \sum_k \lfloor d_k \rfloor y_k \geq f(b)\lceil b \rceil .$$

We introduce a technique to find GMI cuts. Consider row multipliers $\lambda \geq 0$ on

$$\lambda : Ax + Dy \geq b$$

2 and slack variables

$$Ax + Dy - Is = b$$

so that we get

$$(\lambda^T A)x + (\lambda^T D)y - \lambda^T s = \lambda^T b$$

with λ now being free. Applying GMI yields

$$\sum_j (\lambda^T A)_j^+ x_j + f(\lambda^T b) \sum_k \lfloor (\lambda^T D)_k \rfloor y_k + \\ + \sum_k \min(f(\lambda^T D_k), f(\lambda^T b)) y_k + \sum_{\lambda_i < 0} |\lambda_i| s_i \geq f(\lambda^T b) \lceil \lambda^T b \rceil$$

Now, substitute $s := Ax + Dy - b$. example...

Conclusion 4.132. GMI cuts can be stronger than Chvátal-Gomory cuts.

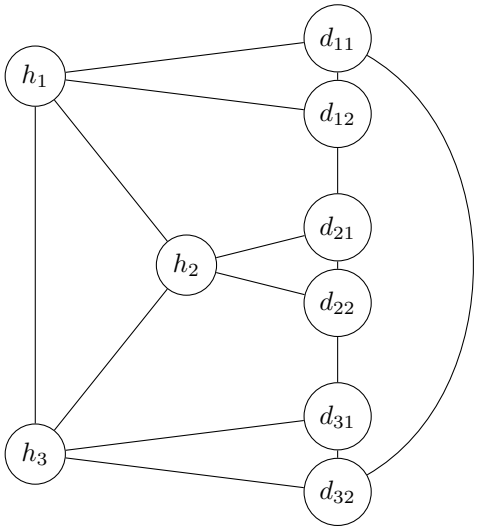
Consider a disjunction $y \leq \beta \vee y \geq \beta + 1$. We can find a **Split Cut** as a generalization of CG and GMI cuts. examples...

Theorem 4.133. Define the polyhedron generated by adding all GMI (split) cuts as P^{GMI} (P^{split}). Then $P^{\text{split}} = P^{\text{GMI}} \subseteq P^1 = P^{\text{CG}}$.

Lecture 19
Th 30 June 2022

Definition 4.134. In the following we use **Template cuts** as a general term for "useful" families of cuts, i.e. they have a specific name.

For TSP we consider 2-matching CPs. We know there is a polynomial algorithm for SEP of 2-matching CPs. For subtour elimination cuts there also exists a polynomial SEP-algorithm via MaxFlow/MinCut. ref script



Lemma 4.135. This x satisfies all degree, subtour, and 2-matching constraints.

Proof. Brute Force. □ define comb

Theorem 4.136 (Chvátal)**.** Let $T_1, \dots, T_k$ be mutually disjoint for odd k such that $T_j \cap H \neq \emptyset$ and $T_j \cap H \setminus \emptyset$ for all j. Then

$$x(E(H)) + \sum_j x(E(T_j)) \leq |H| + \sum_j (|T_j| - 1) - \left\lceil \frac{k}{2} \right\rceil$$

is valid for the TSP integer hull.

Proof. Using degree and subtour constraints for a CG-proof. See homework insert ref □

Remark 4.137. It is not known if SEP for combs is **P** or **NPC**. Nonetheless, for special cases polynomial SEP-algorithms are known. One can also use Gomory-Hu-Trees for a heuristic comb-SEP.

While TSP-cuts are specifically good for TSP, in general they might not be good at all. Therefore, we need to find general templates for MIP/IP that most of the times work satisfactory.

Lots of IPs have $0-1$-decision variables.

Suppose $x \in \mathbb{B}^n$ and a constraint $a^T x \leq b$. If $a_j < 0$, replace

1. x_j by $1 - x_j$,
2. a_j by $-a_j > 0$, and
3. b by $b - a_j > 0$.

So, assume w.l.o.g. $a_j \geq 0$ and $a_j \leq b$. We define $N = \{1, \dots, n\}$. Consider $C \subseteq N$ such that $a(C) > b$. Then, if $x_j = 1$ for all $j \in C$, the solution is infeasible.

Theorem 4.138. As a consequence, $x(C) \leq |C| - 1$ is valid for the integer hull.

Proof. Via Chvátal derivation, see . homework □

Definition 4.139. Such things we call a **Knapsack cover**. Furthermore, a cover C is **minimal** if for all $j \in C$ it holds that $C - j$ is *not* a cover.

Can we separate x^0 from Knapsack covers, that is decide if

$$x^0(C) \leq |C| - 1$$

for all minimal covers C? Note

$$(1 - x^0)(C) = \sum_{j \in C} (1 - x_j^0) = |C| - x^0(C),$$

therefore

$$\begin{aligned} & x^0(C) \leq |C| - 1 \\ \Leftrightarrow \quad & (1 - x^0)(C) = |C| - x^0(C) \geq |C| - (|C| - 1) = 1, \end{aligned}$$

so deciding

$$1 \geq \min_{\text{cover } C} (1 - x^0)(C)$$

suffices if x^0 satisfies all Knapsack covers. Unfortunately, this is **NP**-hard, but in practice we can use Dynamic Programming, see . homework

Observation 4.140. It seems to be ideal to find CPs that are **facet-inducing** on P^I, i.e. the intersection with P^I is a facet of it. Knowing $\dim(P^I)$ is therefore necessary.

TSP itself is not full-dimensional because of its degree equality-constraints.

Theorem 4.141. The integer hull for knapsack cover *is* full-dimensional (if $a_j \leq b$).

Proof. Note $0 \in P^I$. Furthermore, for all j it holds that $a_j \leq b$ implies the unit vector $e^j \in P^I$. So we have $n + 1$ affinely independent points, immediately leading to P^I being full-dimensional. □

Now, consider P^I is full-dimensional. There exists several methods of proving if a CP $\alpha^T x \leq \beta$ is facet-inducing:

1. Find $\bar{x}$ such that $\alpha^T \bar{x} > \beta$, but $\bar{x}$ satisfies all constraint of P^I. However, we don't know all constraints of P^I.
2. Find feasible, affinely independent $v^1, \ldots, v^n \in P^I$ such that for all i it holds $\alpha^T v^i = \beta$
3. If $c^T x \leq d$ is some other valid constraint for P^I with

   $$P^I \cap \{\alpha^T x \leq \beta\} = P^I \cap \{c^T x \leq d\},$$

   then (c, d) is a scalar multiple of (α, β).

Lecture 20
Tu 05 July 2022
contents lec20

Definition 4.142. Consider a binary constraint $a^T x \leq b$ and some corresponding minimal Knapsack cover C. We define

$$E(C) \coloneqq C \cap \{j \in N \mid a_j \geq \max_{k \in C} a_k\}$$

as the **extended cover**.

Observation 4.143. For P^I and a extended cover $E(C)$ it always holds that

$$\sum_{j \in E(C)} a_j \leq |C| - 1$$

is also a valid cutting plane. Moreover, the extended cover is *sometimes* a facet, see [NW99, Ch. II.2, Prop. 23] for a detailed breakdown.

Example 4.144. Consider for $x \in \mathbb{B}$

$$7x_1 + 8x_2 + 9x_3 + 8x_4 + 22x_5 \leq 30.$$

We see $C = \{1, 2, 3, 4\}$ is a minimal cover, therefore we can derive the cutting plane

$$x_1 + x_2 + x_3 + x_4 \leq 3$$

and its extended cover $E(C) = \{1, 2, 3, 4, 5\}$ with the constraint

$$x_1 + x_2 + x_3 + x_4 + \alpha x_5 \leq 3.$$

How big can α be such that the constraint can still be valid? For this, assume $x_5 = 1$ and look at the remaining RHS of 8. Here, a maximum of 1 variable of C can be chosen, so by rearranging

$$\alpha \leq 3 - (x_1 + x_2 + x_3 + x_4) = 3 - 1 = 2$$

and we get a **lifted cover constraint** of

$$x_1 + x_2 + x_3 + x_4 + 2x_5 \leq 3$$

which is a facet.

Definition 4.145. Consider constraints of the form

$$\begin{aligned} \sum_j y_j &\leq b \\ y_j &\leq a_j x_j \\ y &\geq 0 \\ x &\in \mathbb{B}^n. \end{aligned}$$

We call these constraints with **variable upper bounds**. For these, we define a **flow cover** as follows: Let C be any cover. We define $\lambda = a(C) - b > 0$ as the **excess cap** of C.

what exactly flow cover?

For each

ahahhhhhhhhh

4.8 Branch and Bound

Consider again TSP with binary variables for every edge. Note that there are quadratically many variables - for the IP solver this is way too much! But we might consider heuristic ideas that use only "good" edges. For example, we can consider only *short* edges under some threshold.

Lecture 21
Th 07 July 2022

content lec21

Note. I missed the remaining three lectures and will try to incorporate the missing material in a future point in time when I'm less stressed out.

Lecture 22
Tu 12 July 2022

content lec22

...

Lecture 23
Th 14 July 2022

content lec23

restructure table of contents

clean up vocabulary index

Part II
Appendix

A Exercise sheets

1. exercise sheet

Exercise 1.1. Did on paper.

2. exercise sheet

Exercise 2.1. An correct ordering is given by:

$$\mathrm{O}\left(\varepsilon^n\right) \subseteq \mathrm{O}\left(n^{\varepsilon-1}\right) \subseteq \mathrm{O}\left(n^{-\varepsilon}\right) \subseteq \mathrm{O}\left(\frac{\log n}{n^{\varepsilon}}\right) \tag{7}$$

$$\subseteq \mathrm{O}\left(\frac{1}{\log n}\right) \subseteq \mathrm{O}\left(\frac{\log^2 n}{\log n}\right) \subseteq \mathrm{O}\left(\frac{1}{\log^2 n}\right) \tag{8}$$

$$\subseteq \mathrm{O}\left(e^{\frac{1}{n}}\right) = \mathrm{O}\left(1\right) = \mathrm{O}\left(\left(1-\frac{1}{n}\right)^n\right) \tag{9}$$

$$\subseteq \mathrm{O}\left(\log n\right) \subseteq \mathrm{O}\left(\frac{n^{\varepsilon}}{\log n}\right) \subseteq \mathrm{O}\left(n^{\varepsilon}\right) \subseteq \mathrm{O}\left(n^{\varepsilon}\log n\right) \subseteq \mathrm{O}\left(n^{1-\varepsilon}\right) \tag{10}$$

$$\subseteq \mathrm{O}\left(\frac{n}{\log n}\right) \subseteq \mathrm{O}\left(n\log n\right) \subseteq \mathrm{O}\left(n^2\right) \subseteq \mathrm{O}\left(n^2\log n\right) \subseteq \mathrm{O}\left(n^e\right) \tag{11}$$

$$\subseteq \mathrm{O}\left(n^{\log n}\right) \subseteq \mathrm{O}\left(e^n\right) \subseteq \mathrm{O}\left((\log n)^n\right) \subseteq \mathrm{O}\left(n!\right) \tag{12}$$

These can mostly achieved by the fact that $n^x \in \mathrm{O}(n^y)$ if $x \leq y$, and $(\log n) \cdot n^x \in \mathrm{O}(n^y)$ if $y > x$, otherwise the other way around. Additionally, it is often useful to consider the logarithm of the functions we compare, because it maintains monotonocity.

Exercise 2.2. Analoguous to the lecture we can introduce constraints, such that $y_{ij} = x_i \wedge x_j$:

$$\begin{aligned} y_{ij} &\leq x_i \\ y_{ij} &\leq x_j \\ y_{ij} &\geq x_i + x_j - 1 \\ y_{ij} &\in [0,1] \end{aligned}$$

Exercise 2.3. We can show that $f(x_1) = \max(c_1x_1, c_1p + c_2x_1 - c_2p)$ using a case distinction.

- $x_1 = p$: Trivial.
- $x_1 > p$: Consider $c_1 < c_2$. Multiplying by $x_1 - p$ (which is positive) and rearranging yields $c_1x_1 < c_1p + c_2x_1 - c_2p$.
- $x_1 < p$: Analoguous, but now $x_1 - p$ is negative, which reverses the inequality.

As shown in ADM1, the maximum of linear functions can be written as an LP by

introducing a helper variable as follows:

$$
\begin{aligned}
\min \quad & z + \sum_{i=2}^{n} c_i x_i \\
\text{s.t.} \quad & Ax = b \\
& l \leq x_1 \leq u \\
& x_2, ..., x_n \leq 0 \\
& z \geq c_1 x_1 \\
& z \geq c_1 p + c_2 x_1 - c_2 p
\end{aligned}
$$

3. exercise sheet

Exercise 3.1. 1. We can show easily that $\mathsf{SPATH} \in \mathbf{P}$ by using the fact from ADM1, that breadth-first search started from s finds a shortest path to t in polynomial time. Therefore, if the shortest path has length $k^* \leq k$, we can return true, and false otherwise.

2. We first show $\mathsf{LPATH} \in \mathbf{NP}$: Suppose an instance of LPATH is true, then there is a path of at least length k. Therefore, we can simply use this path as a succinct certificate and verify in polynomial time that the path is indeed valid.

   It remains to show that we can reduce a **NP**-complete problem to LPATH. It suffices to show $\mathsf{UHAMPATH} \propto \mathsf{LPATH}$: Suppose we have an instance $((V, E), s, t)$ of $\mathsf{UHAMPATH}$. We can simply reduce it to the problem of finding a path of at least length $|V| - 1$ starting in s and ending in t, because every such path is indeed a hamiltonian path, because every vertex needs to be visited exactly once. Therefore, if there is a hamiltonian path, it is already a path of at least length $|V| - 1$. For the other direction, if there is a path of at least length $|V| - 1$, then it must visit every node exactly once in order to be a valid path.
   This shows that the reduction is Yes-preserving.

Exercise 3.2. If $\mathsf{DOUBLESAT}$ is true, then we can choose any two valid assignments as a succinct certificate and easily verify their correctness in polynomial time.

It remains to show $\mathsf{SAT} \propto \mathsf{DOUBLESAT}$: Starting from our SAT-instance, we can simply introduce two new variables a, b and a new clause $a \vee b$. This construction is Yes-preserving, because if the original instance is infeasible, the new instance still has no assignments. On the other hand, if there is a valid assignment in the original, then we now have at least 3 valid instances for different assignments of a and b.

Exercise 3.3. We notice that $a \vee b$ is equivalent to $\neg a \implies b$, and $\neg b \implies a$. By doing this for all clauses, we can construct a graph with the literals as vertices, and the implications as directed edges. Now, checking for each literal pair $l, \neg l$ if both can reach one another by a directed path suffices to show feasibility:

If previous condition holds true, then by logic it must hold that a feasible assignment satisfies $l \Leftrightarrow \neg l$, which is impossible. On the other hand, if this is never the case, then there must be a feasible solution:

We can construct this solution by iteratively setting either l or $\neg l$ to true, depending if $l \implies \neg l$ holds, and then also set every implied variable to true. If we would encounter a variable r which is already false, then $\neg r$ must be true, and therefore all further implications would need to be true by construction. Because $l \implies r$,

also $\neg r \implies \neg l$, meaning that l would be already false - contradiction!
Therefore, our construction always works.

4. exercise sheet

Exercise 4.1. We're given an instance $(v^1, ..., v^k, x^0)$ as described. Iff $x_0 \in P$, then $x_0 \in \text{conv}(v^1, \ldots, v^k)$. Thus there exist $\lambda_1, \ldots, \lambda_k$ such that

$$\begin{aligned} \sum_{i=1}^{k} \lambda_i v^i &= x_0 \\ \sum_{i=1}^{k} \lambda_i &= 1, \\ \lambda &\geq 0 \end{aligned}$$

which we can find by LP. Otherwise, the system is found infeasible, and we know there must exist c with $c^T x \geq c^T x^0$. We first show it suffices that this property holds for all vertices.

Suppose $c^T v^i \geq c^T x^0$. Let $x = \sum_i \lambda'_i v^i \in P$. Then

$$\begin{aligned} c^T x &= \sum_i \lambda'_i c^T v^i \\ &\geq \sum_i \lambda'_i c^T x^0 \\ &= c^T x^0. \end{aligned}$$

Note that we used $\lambda' \geq 0$ and $\sum_i \lambda'_i = 1$ in the second and third step.

Using

$$c^T v^i \geq c^t x^0 \Leftrightarrow (v^i - x^0)^T \cdot c \geq 0,$$

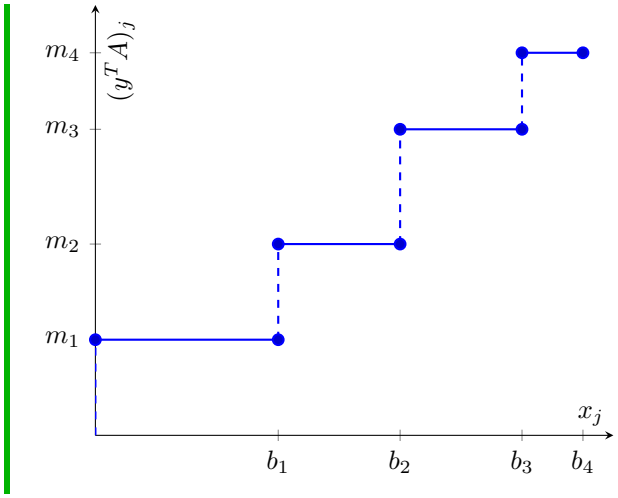
this means we can find c with following constraints via LP:

$$(v^i - x^0)^T \cdot c \geq 0, \quad i = 1, ..., k.$$

Exercise 4.2. We solve:

1.
2.
3.

Exercise 4.3. The Kilter diagram of the primal is given by


$(y^T A)_j$
m_4
m_3
m_2
m_1
x_j
b_1
b_2
b_3
b_4

5. exercise sheet

Exercise 5.1. TBD

Exercise 5.2. Let r be submodular as defined, and let $A, B \subseteq E$ be sets. We prove by induction over $n \coloneqq |B - A| \in \mathbb{N}$ that

$$r(A) + r(B) \geq r(A \cup B) + r(A \cap B). \tag{13}$$

Suppose $n = 0$. Then $A \cup B = A$ and $A \cap B = B$, implying equality of (13).

Now, consider (13) to be true for sets $A, B \subseteq E$ such that $|B - A| = n$. Suppose $A, B \subseteq E$ with $|B - A| = n + 1 > 0$, and let $e \in B - A$. Notice that $A \cap B \subseteq A \subseteq E$ with $e \notin A$. Therefore, by applying rearranged submodularity, we get

$$r((A + e) \cap B) - r(A + e) \geq r(A \cap B) - r(A). \tag{14}$$

Additionally, $|B - (A + e)| = n$ by choice of e, enabling us to use the (rearranged) induction step (13):

$$r(B) \geq r((A + e) \cup B) + r((A + e) \cap B) - r(A + e)$$

Knowing $(A + e) \cup B = A \cup B$, and applying (13) yields

$$r(B) \geq r(A \cup B) + r(A \cap B) - r(A),$$

which was to be shown.

Exercise 5.3. TBD

6. exercise sheet

Exercise 6.1. TBD

Exercise 6.2. We show that r is a polymatroid rank function. We already know $r(\emptyset) = 0$. It is also easy to prove that r monotonically increasing for $R \subseteq S$, because every flow in R is also feasible in S. It remains to show submodularity.

7. exercise sheet

Exercise 7.1. For even-sized subsets $B \subseteq V$ we introduce

$$\sum_{e \in E(B)} x_e \leq \frac{|B|}{2}.$$

We can merge this without problems with the odd blossom constraints, and get

$$\begin{aligned} \max_x \quad & \mathbf{1}^T x \\ \text{s.t.} \quad & \sum_{e \in \delta(v)} x_e \leq 1, && \forall v \in V, \\ & \sum_{e \in E(B)} x_e \leq \left\lfloor \frac{|B|}{2} \right\rfloor, && \forall B \subseteq V,\ |B| \geq 2, \\ & x \geq 0 \end{aligned}$$

and construct the dual

$$\begin{aligned} \min_{y,z} \quad & \mathbf{1}^T y + \sum_{\substack{B \subseteq V, \\ |B| \geq 2}} \left\lfloor \frac{|B|}{2} \right\rfloor z_B \\ \text{s.t.} \quad & y_i + y_j + \sum_{B : e \in E(B)} z_B \geq 1, \quad \forall (i,j) \in E, \\ & y, z \geq 0 \end{aligned}$$

Now, given an optimal matching M (with corresponding solution x) and its Tutte-Berge subset $X \subseteq V$, we construct the optimal dual solution by setting for every $i \in V$

$$y_i \coloneqq \begin{cases} 1, & \text{if } i \in X \\ 0, & \text{else} \end{cases}.$$

Furthermore, consider $G^- \coloneqq G(V \setminus X)$, the graph without nodex X. For every $B \subseteq V$ with at least 2 nodes, we set

$$z_B \coloneqq \begin{cases} 1, & \text{if } B \text{ is maximal connected component in } G^- \\ 0, & \text{else} \end{cases}.$$

This solution is feasible, because now every edge is covered either by a node $i \in X$, or otherwise is part of one of the connected components of G^-, thus satisfying our only dual constraint. It remains to show its optimality via strong duality - it holds

that

$$\begin{aligned}\mathbf{1}^T x = |M| &= \frac{1}{2}(|N| + |X| - oc(X)) \\ &= |X| + \frac{1}{2}(|N - X| - oc(X)) \\ &= \mathbf{1}^T y + \frac{1}{2}(|N - X| - oc(X)).\end{aligned}$$

What is left is to show $\frac{1}{2}(|N - X| - oc(X)) = \sum_{B \text{ is cc in } G^-} \left\lfloor \frac{|B|}{2} \right\rfloor z_B$. Note that every node of G^- is in exactly one connected component, so our sum on the right side is at most $\frac{1}{2}|N - X|$. It holds that

$$\left\lfloor \frac{|B|}{2} \right\rfloor = \begin{cases} \frac{|B|}{2}, & B \text{ is even-sized} \\ \frac{|B|-1}{2}, & B \text{ is odd-sized,} \end{cases}.$$

Because $oc(X)$ is based on G^- by construction, this immediately yields the part we need to subtract from $|N - X|$, proving optimality.

Exercise 7.2. TBD

8. exercise sheet

Exercise 8.1. TBD

Exercise 8.2. Observe that G has a T-join if and only if all connected components of G have a T-join. So it suffices to show the case when G is connected.

Now suppose that G has a T-join $J \subseteq E$. Consider

$$a := \sum_{v \in T} \deg_J(v) \text{ and } b := \sum_{v \in N-T} \deg_J(v).$$

The term $a + b$ precisely double-counts the cardinality of J. Hence, $a + b$ is even. Since b is a sum of even addends it also must be even. We can follow that $a = a+b-b$ is even. Finally, because all summands of a are odd there must be an even number of elements in T.

Assume conversely T to have even cardinality $2t$ and let $v_1, \dots v_{2t}$ be the distinct nodes of T. For G is connected there exists a path P_i connecting v_i and v_{i+t} for all $i = 1 \dots t$. Now define J to be the edges of E occurring in an odd number of these t paths. For every $i \in [t]$ the sum

$$\sum_{j=1}^{t} \deg_{P_j}(v_i) = \deg_{P_i}(v_i) + \sum_{j=1, j\neq i}^{t} \deg_{P_j}(v_i) = 1 + \sum_{j=1, j\neq i}^{t} \deg_{P_j}(v_i)$$

is odd because for every $i \in [i] \backslash \{j\}$ the addend $\deg_{P_j}(v_i)$ is either 0 or 2. Analogously, we can argue that this sum is also odd for all the other nodes of T and even for all nodes not in T. By construction of J we have

$$\deg_J(v) \equiv \sum_{j=1}^{t} \deg_{P_j}(v) \mod 2$$

for all nodes $v \in N$. This shows that J is a T-join.

Exercise 8.3. We prove:

(a) Let J be a T-join and $S \subseteq V$. Then the sum of all degrees of S limited to J is:

$$\begin{aligned} \sum_{v \in S} \delta_J(v) &= \sum_{v \in S \cap T} \delta_J(v) + \sum_{v \in S \backslash T} \delta_J(v) \\ &\equiv |S \cap T| \cdot 1 + 0 \\ &\equiv |S \cap T| \pmod 2 \end{aligned}$$

Thus, $|\delta(S) \cap J|$ must have the same parity as $|S \cap T|$, because otherwise the

theoretical sum of degrees of $G(S)$ would be odd. This would contradict that the sum of all degrees in a (sub-)graph must be always even!

$J \cap \delta(S)$ is non-empty because $|S \cap T|$ has odd parity by definition.

(b) We prove both directions.

"⇒" Suppose F includes a T-join J. Consider the T-cut of a T-odd $S \subseteq V$. By (a), $F \cap \delta(S) \supseteq J \cap \delta(S) \neq \emptyset$.

"⇐" Suppose F has only non-empty intersections with T-cuts. We construct a T-join $J \subseteq F$ as follows:

Algorithm 6: Construct T-join

```
J ← ∅
U ← ∅
while v ∈ T − U do
    S ← {v}
    K ← ∅
    while true do
        Choose any (u, w) ∈ δ(S) ∩ F
        S ← S + u
        K ← K + (u, w) if w ∈ U then
            Add any node reachable from w via J to S and corresponding edges
              to K
        end
        if w ∈ T then
            U ← U + {v, w}
            Let P ⊆ K be the (unique) path from v to w
            J ← JΔP
            break
        end
    end
end
```

We start from any unused node $v \in T$ and construct a path to another unused node in T by repeatedly adding any edge in F from the T-cut formed by all visited nodes S. For this, we need to assure S is always T-odd: If our new node is in T, but used, we need to repair our T-odd-property of S by adding all nodes in the connected component of our partial solution J. In particular, there must be an even number of nodes in this component we add.

Therefore, our algorithm is well-defined and terminates.

It is clear that our final $J \subseteq F$. Furthermore, by Lemma 4.104, our partial J is always a T-join for U (because P is a T-join for $\{u, w\}$). Thus, in the end we have $T = U$ and J is a T-join. (This feels more complicated than necessary...)

Any feasible solution now has a non-empty intersection with any T-cut, therefore the solution contains a T-join. Because we only have non-negative costs, an optimal solution does not need any edges that aren't part of the included T-join. Since every T-join is also feasible by our theorem, an optimal IP-solution is also a minimal T-join.

(c) We prove both directions.

"$\Rightarrow$" Suppose F includes a T-cut $\delta(S)$. Consider a T-join J, then $F \supseteq \delta(S) \supseteq \delta(S) \cap J \neq \emptyset$.

"$\Leftarrow$" We prove the contraposition: Suppose F does not include a T-cut. Then, for every T-cut $\delta(S)$ it holds $\delta(S) \cap (E - F)$ is non-empty. By (b), $E - F$ includes a T-join J. Therefore, $J \cap F = \emptyset$.

9. exercise sheet

Exercise 9.1. TBD

Exercise 9.2. We derive:

(a) We start with $(2,1)$-matchings. Because $b \equiv 2$, for every subset $H \subseteq N$ it holds that $b(H)$ is even, therefore for $D \subseteq \delta(H)$ it follows $b(H) + u(D)$ is odd iff $|D|$ is odd. Furthermore, it holds that

$$\begin{aligned}\left\lfloor \frac{b(H) + u(D)}{2} \right\rfloor &= \left\lfloor \frac{2|H| + |D|}{2} \right\rfloor \\ &= |H| + \left\lfloor \frac{|D|}{2} \right\rfloor.\end{aligned}$$

In summary, both sets of constraints are equivalent.

Consider now $(1,1)$-matchings. We can easily see that blossom constraints form a subset of our b-matching constraints if for odd $H \subseteq N$ we choose $D = \emptyset \subseteq \delta(H)$. It remains to show every other b-matching constraint is redundant. Let $H \subseteq N$ and $D \subseteq \delta(H)$ such that $|H| + |D|$ is odd. Let

$$H_D \coloneqq \{j \mid (i,j) \in D \wedge i \in H\}$$

be the set of outside-nodes directly connected by D. H_D is empty iff D is empty, which is the case already considered. For some $v \in H_D \neq \emptyset$, define

$$H' \coloneqq \begin{cases} H \cup H_D & \text{if } |H \cup H_D| \text{ odd} \\ H \cup H_D - v & \text{if } |H \cup H_D| \text{ even} \end{cases}.$$

Then, H' is odd and generates a blossom constraint. Consider the second case (the first case works analogous):

$$\begin{aligned} x(E(H)) + x(D) \leq x(E(H \cup H_D)) &= x(E(H' + v)) \\ \leq x(E(H')) + 1 \quad &\leq \left\lfloor \frac{|H'|}{2} \right\rfloor + 1 \\ = \frac{|H'| + 1}{2} \quad &= \frac{|H| + |H_D|}{2} \\ \leq \left\lfloor \frac{|H| + |D|}{2} \right\rfloor \quad &= \left\lfloor \frac{b(H) + u(D)}{2} \right\rfloor \end{aligned}$$

This shows that the blossom constraints suffice to make every b-constraint feasible.

(b) Given a (b, u)-matching and H, D as specified. Recall our LP constraints:

$$x(\delta(v)) = b_v, \quad \forall v \in N \tag{15}$$
$$x_e \leq u_e, \quad \forall e \in E \tag{16}$$
$$x_e \geq 0, \quad \forall e \in E \tag{17}$$

Consider following row-multiplied expressions:

$$\sum_{i \in H} (15) \equiv \quad 2 \cdot x(E(H)) + x(\delta(H)) = b(H)$$
$$\sum_{e \in D} (16) \equiv \quad x(D) \leq u(D)$$
$$-\sum_{e \in \delta(H) \setminus D} (17) \equiv \quad -x(\delta(H) \setminus D) \leq 0$$

Summing everything up and dividing by two yields

$$x(E(H)) + x(D) \leq \frac{b(H) + u(D)}{2}$$

as wished. In particular, every LHS coefficient is integral. Therefore, all Chvátal conditions are satisfied, and we get our b-matching constraint as a cutting plane.

10. exercise sheet

Exercise 10.1. TBD

Exercise 10.2. We proof by induction over $k \geq 3$: If $C_k = \{1, \ldots, k\}$ is a clique in a graph $G = (V, E)$, then

$$\sum_{j \in C_k} x_j \leq 1$$

can be derived as a valid CP for the integer hull of the Max Stable set problem using Chvátal cuts.

For $k = 3$:

$$\begin{array}{lrrrrr}
1/2 \times (& x_1 & +x_2 & & & \leq 1) \\
1/2 \times (& & x_2 & +x_3 & & \leq 1) \\
1/2 \times (& x_1 & & +x_3 & & \leq 1) \\
\hline
& x_1 & +x_2 & +x_3 & & \leq 1.5
\end{array}$$

This satisfies all Chvátal conditions, and thus

$$x_1 + x_2 + x_3 \leq 1 = \lfloor 1.5 \rfloor$$

is a valid cutting plane.

Suppose we already have Chvátal derivation for some $k \geq 3$. We prove that we can also find a derivation for any clique C_{k+1}. Since subsets of cliques are itself cliques, we can use $C_{k+1} \setminus \{1\}$ and $C_{k+1} \setminus \{k+1\}$ and derive by our inductive assumption that

$$\begin{array}{lrrrrrr}
1/2 \times (& x_1 & +x_2 & \ldots & +x_k & & \leq 1) \\
1/2 \times (& & x_2 & \ldots & +x_k & +x_{k+1} & \leq 1) \\
1/2 \times (& x_1 & & & & +x_{k+1} & \leq 1) \\
\hline
& x_1 & +x_2 & \ldots & +x_k & +x_{k+1} & \leq 1.5
\end{array}$$

and immediately get our constraint we wanted to derive by rounding down again.

11. exercise sheet

Exercise 11.1. TBD

Exercise 11.2. We derive the Gomory cuts as discussed for each basic variable as

$$\begin{array}{llll} 0.7x_2 & +0.2x_3 & +0.6x_4 & \geq 0 \\ 0.0x_2 & +0.1x_3 & +0.5x_4 & \geq 0.1 \\ 0.2x_2 & +0.5x_3 & +0.0x_4 & \geq 0.7 \end{array}$$

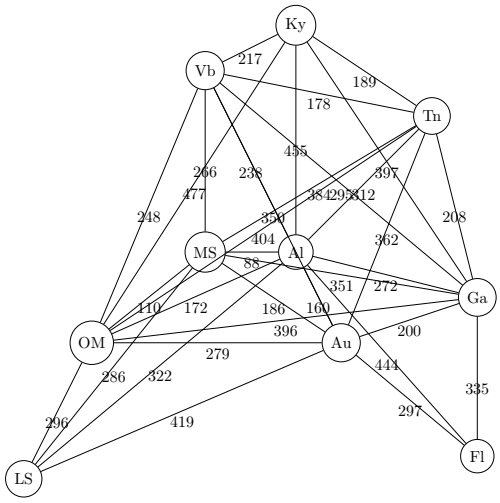
Note the first constraint is useless and can be dropped.

B Programming Project

In this section we will solve TSP instances. For the project, we used Gurobi with its Python-API in a Jupyter Notebook, see subsection B.3

B.1 First instance

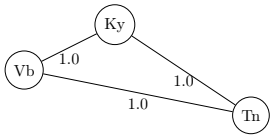
First, have a look at our "good" graph (V, E_G):

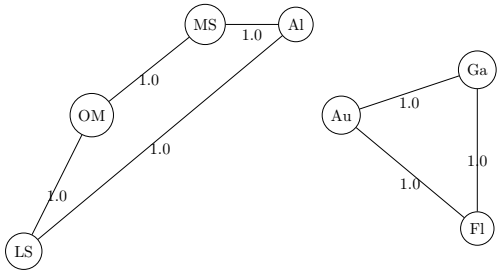


Solving following LP

$$\begin{aligned} \min_{x} \quad & \sum_{e \in E_G} x_e \\ \text{s.t.} \quad & \sum_{\{i,j\} \in \delta(i)} x_{\{i,j\}} = 2, \quad \forall i \in V, \\ & x \leq 1, \\ & x \geq 0, \end{aligned}$$

i.e. using only degree constraints, we get:

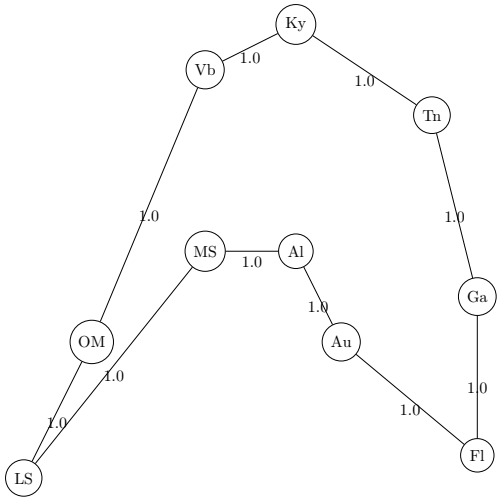




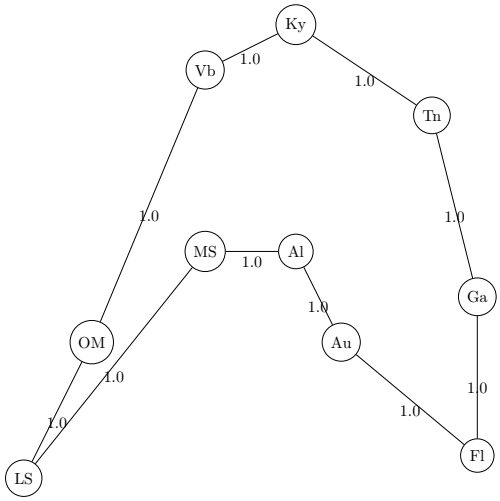
We see three subtours which we will eliminate by adding corresponding subtour constraints

$$\sum_{e \in \delta(S)} x_e \geq 2, \qquad \forall S \in \{\{\text{LS}, \text{OM}, \text{MS}, \text{Al}\}, \{\text{Vb}, \text{Ky}, \text{Tn}\}, \{\text{Au}, \text{Ga}, \text{Fl}\}, \}$$

and get an integral Hamiltonian tour:



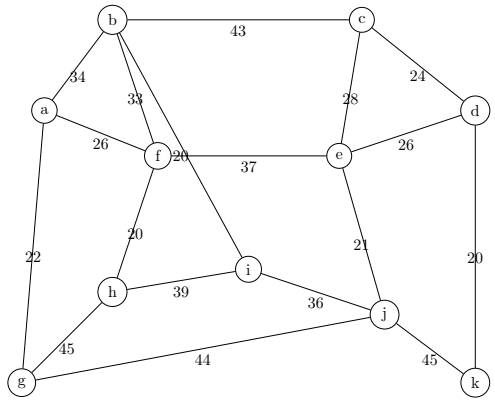
Adding back all edges V and modifying the constraints to include the new edges does indeed not generate a better solution:



The total length of the optimal integral Hamilton tour is thus 2324 .

B.2 Second instance

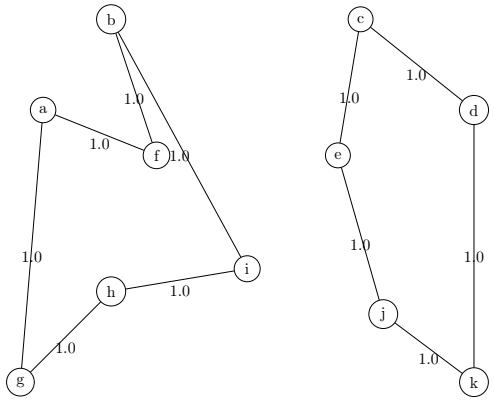
First, have a look at our "good" graph (V, E_G):



Solving following LP

$$\begin{aligned} \min_{x} \quad & \sum_{e \in E_G} x_e \\ \text{s.t.} \quad & \sum_{\{i,j\} \in \delta(i)} x_{\{i,j\}} = 2, \quad \forall i \in V, \\ & x \leq 1, \\ & x \geq 0, \end{aligned}$$

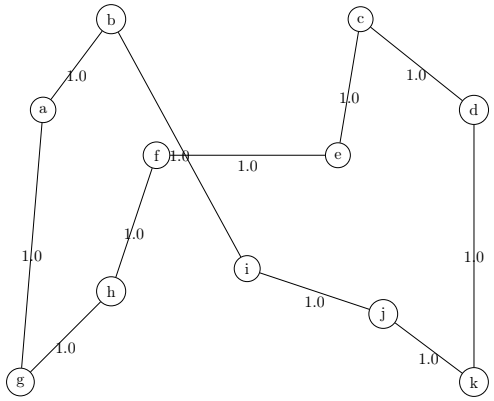
i.e. using only degree constraints, we get:



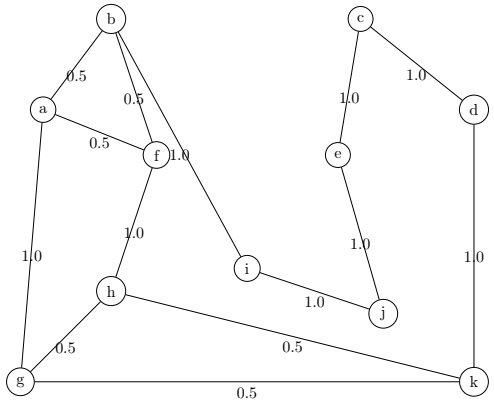
We see two subtours which we will eliminate by adding corresponding subtour constraints

$$\sum_{e \in \delta(S)} x_e \geq 2, \qquad \forall S \in \{\{\mathrm{b,a,f,i,h,g}\}, \{\mathrm{c,d,e,j,k}\}, \}$$

and get following integral Hamiltonian path:



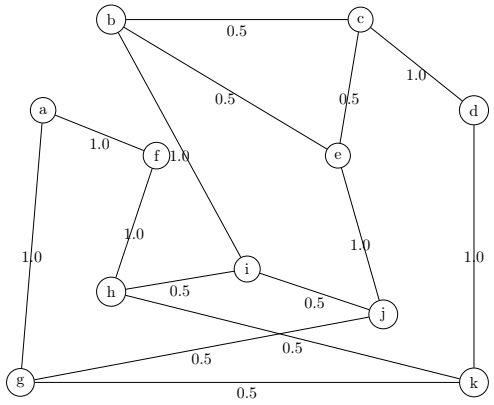
However, if we add all edges now, we get a non-integral solution



Therefore we add 2-matching constraints

$$x(E(H)) + x(D) \leq 4, \qquad H = \{\mathrm{b}, \mathrm{a}, \mathrm{f}\},\ D = \{\{\mathrm{a}, \mathrm{g}\}, \{\mathrm{b}, \mathrm{i}\}, \{\mathrm{f}, \mathrm{h}\}\}$$
$$x(E(H)) + x(D) \leq 4, \qquad H = \{\mathrm{h}, \mathrm{g}, \mathrm{k}\},\ D = \{\{\mathrm{a}, \mathrm{g}\}, \{\mathrm{f}, \mathrm{h}\}, \{\mathrm{d}, \mathrm{k}\}\}$$

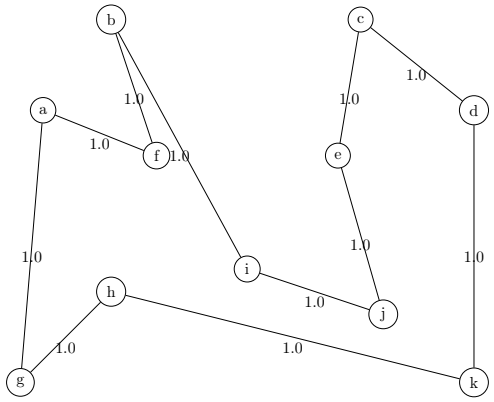
and still get a non-integral solution



We add even more 2-matching constraints

$$x(E(H)) + x(D) \leq 4, \qquad H = \{\mathrm{b}, \mathrm{c}, \mathrm{e}\},\ D = \{\{\mathrm{b}, \mathrm{i}\}, \{\mathrm{e}, \mathrm{j}\}, \{\mathrm{c}, \mathrm{d}\}\}$$
$$x(E(H)) + x(D) \leq 7, \qquad H = \{\mathrm{h}, \mathrm{g}, \mathrm{k}, \mathrm{i}, \mathrm{j}\},\ D = \{\{\mathrm{a}, \mathrm{g}\}, \{\mathrm{f}, \mathrm{h}\}, \{\mathrm{d}, \mathrm{k}\}, \{\mathrm{b}, \mathrm{i}\}, \{\mathrm{e}, \mathrm{j}\}\}$$

and finally get



The total length is 319 .

B.3 Code

See the folder `exercises/code` in our project.

Index

Literature

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows: theory, algorithms, and applications*. Prentice Hall, 1993. ISBN: 0-13-617549-X. URL: https://tu-berlin.hosted.exlibrisgroup.com/primo-explore/fulldisplay?docid=TUB_ALMA_DS21506259970002884&context=L&vid=TUB&lang=de_DE&search_scope=TUB_ALL&adaptor=Local%20Search%20Engine&tab=tub_all&query=any,contains,network%20flows%20ahuja&offset=0.

[Coo+97] William J. Cook et al. *Combinatorial Optimization*. Wiley, 1997. ISBN: 978-0-471-55894-1. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9781118033142.

[KV18] Bernhard Korte and Jens Vygen. *Kombinatorische Optimierung*. 3rd ed. Springer, 2018. ISBN: 978-3-662-57690-8. URL: https://link.springer.com/book/10.1007/978-3-662-57691-5.

[NW99] George Nemhauser and Laurence Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1999. ISBN: 978-0-471-82819-8. URL: https://onlinelibrary.wiley.com/doi/book/10.1002/9781118627372.