UC-Denver Math 7825, Section 001 — Syllabus
Topics in Optimization "Circuit Walks in Optimization" — Fall 2021

Instructor: **Steffen Borgwardt**	Class Hours: **TuTh 2:00 – 3:15pm**
Email: steffen.borgwardt@ucdenver.edu URL: http://math.ucdenver.edu/~sborgwardt Office: SCB 4313	Class Room: SCB 4119 (SCB 4113 starting Aug. 31) Office Hours: Tu 3:30 – 4:30pm Th 12:30 – 1:30pm

Prerequisites: Graduate Standing in Mathematics. This course will be more accessible and pleasant if you had at least two courses among Math 5593 Linear Programming, Math 7594 Integer Programming, Math 5490 Network Flows.

Course Credits: 3.

Catalog Description / Course Overview:
The so-called circuits are an important concept in polyhedral theory. They generalize the set of edge directions and represent elementary changes between feasible solutions. The first half of this course is on types of optimization not covered in the standard curriculum, which includes matroids and greedy algorithms, approximation theory, and dynamic programming. As part of this, students are exposed to a variety of problems from combinatorial optimization. The second half is on the fundamentals of circuits and walks along them. In project-based work, these concepts are connected to general classes of combinatorial optimization problems and algorithms for them, as well as real-world applications.

Course Goals:
In this course, students learn

1. to identify and work on a range of combinatorial optimization problems
2. to develop a geometric intuition of circuits and circuit walks
3. to understand the connections of circuits and combinatorial algorithms
4. to identify applications and viable algorithms based on circuits
5. to present their project-based work to peers and a more general audience

Required textbook:
B. Korte, J. Vygen, Combinatorial Optimization, 6th edition, Theory and Algorithms, Springer, 2018

Assignments: Expect to spend about 4-8 hours per week on reading and homework. Reading and homework are assigned during class. Homework will be due in large collections twice during the semester as indicated in the schedule. No late homework will be accepted unless prior arrangements are made. **Homework can be submitted alone or in groups of two.** Collaboration among students is encouraged, but every group needs to write up their own solution. You need to share who you collaborated with; higher standards about the presentation may be applied if you worked with more peers.

Tests: There will be a mid-term test in the form of a 20-25 minute oral exam. In the exam, solutions for a take-home assignment from the day before are presented. The exam is scheduled the week before. If you cannot take the test at the appointed time, you must contact me at least one week prior to the test date so that we can make other arrangements. (Exceptions only for serious unpredictable circumstances.) There will be no class on exam day.

Final Project: There is no final exam. There will be a final project starting in the second half of classes. **Work in groups of two is encouraged**, but you can work by yourself or in a group of three. Details on grading and deliverables will be announced in class as soon as possible.

Grading: **40% Homework, 20% mid-term test, 40% final project.** There are a total of 100 points. Final grades will be assigned using the following scale: 89-100 A; 78-88.5 B; 67-77.5 C; 56-66.5 D; less than 56 F. Plusses and minuses will be assigned for borderline cases. The midterm is 20 points. The final project is 40 points.

Homework, midterm and the project will be graded depending on correctness and quality of presentation. It is always expected that you show your work in detail.

Important Course Dates:

- Thursday, Sep. 23rd, Homework 1 Collection.
- Tuesday, Oct. 12th, Midterm
- Thursday, Oct. 21st, Homework 2 Collection.

Course Schedule:

Week	Dates	Topics
1	8/24 8/26	Class Organization and Introduction Matroids and Greedy Algorithms (Chapter 13)
2	8/31 9/2	Matroids and Greedy Algorithms Matroids and Greedy Algorithms
3	9/7 9/9	Matroids and Greedy Algorithms Matroids and Greedy Algorithms
4	9/14 9/16	Approximation Theory (Chapter 16) Approximation Theory
5	9/21 9/23	Approximation Theory Approximation Theory (Homework 1 Collection due)
6	9/28 9/30	Dynamic Programming and Generalized Knapsack Problems (Chapter 17) Dynamic Programming and Generalized Knapsack Problems
7	10/5 10/7	Dynamic Programming and Generalized Knapsack Problems Dynamic Programming and Generalized Knapsack Problems
8	10/12 10/14	Midterm Circuits and Circuit Walks
9	10/19 10/21	Circuits and Circuit Walks Circuits and Circuit Walks (Homework 2 Collection due)
10	10/26 10/28	A Compendium of Circuits 1 (Start of Project) Circuit Walks for Data Analysis 1
11	11/2 11/4	A Compendium of Circuits 2 Circuit Walks for Data Analysis 2
12	11/9 11/11	Student Session 1 (5-minute sales pitches) A Compendium of Circuits 3
13	11/16 11/18	Circuit Walks for Data Analysis 3 Student Session 2 (formal progress reports)
14	11/30 12/2	Extra Time for Topics Extra Time for Topics
15	12/7 12/9	Final Presentations 1 Final Presentations 2

Matroid Theory

Common formulation of comb. opt. problems

Given a set system $(E, \mathcal{F})$, where E is a finite set and $\mathcal{F} \subseteq 2^E$ (the power set on E), and a cost function $c: \mathcal{F} \to \mathbb{R}$, find an element of $\mathcal{F}$ with min/max cost.

Assumptions:

(I) c is a modular function, i.e.,

$$c(X) = c(\emptyset) + \sum_{x \in X} \left(c(\{x\}) - c(\emptyset) \right) \quad \text{for all } X \subseteq E$$

We use the variant $(c(\emptyset) = 0)$

$$c: E \to \mathbb{R} \quad \text{and} \quad c(X) = \sum_{e \in X} c(e)$$

(II) $\mathcal{F}$ is an independence system, i.e., closed under subsets

Independence Systems and Matroids

Def A set system $(E, \mathcal{F})$ is an independence system if

(M1) $\emptyset \in \mathcal{F}$

(M2) If $X \subseteq Y \in \mathcal{F}$, then $X \in \mathcal{F}$

The elements of $\mathcal{F}$ are called independent, the elements of $2^E \setminus \mathcal{F}$ dependent. Minimal dependent sets are called circuits, maximal independent sets are called bases. For $X \subseteq E$, the maximal independent subsets of X are called bases of X.

__Def__ Let $(E, \mathcal{F})$ be an independence system. For $X \subseteq E$, we define the __rank__ of X by

$$r(X) = \max \{ |Y| : Y \subseteq X, Y \in \mathcal{F} \}$$

and the __closure__ of X by

$$\sigma(X) = \{ y \in E : r(X \cup \{y\}) = r(X) \}$$

Two types of problems:

Maximization Problem for Independence Systems

Given: Independence system $(E, \mathcal{F})$ and $c: E \to \mathbb{R}$

Find: $X \in \mathcal{F}$ s.t. $c(X) = \sum_{e \in X} c(e)$ is maximum

Minimization Problem for Independence Systems

Given: Independence system $(E, \mathcal{F})$ and $c: E \to \mathbb{R}$

Find: Basis B s.t. $c(B)$ is minimum

Examples

- **Maximum Weight Stable Set Problem**

  Given: Graph G and weights $c: V(G) \to \mathbb{R}$

  Find: Stable Set in G of max weight

  $E = V(G)$ and $\mathcal{F} = \{F \subseteq E : F \text{ is stable in } G\}$

- **Traveling Salesman Problem**

  Given: Complete, undirected graph G and weights $c: E(G) \to \mathbb{R}_+$

  Find: Minimum weight Hamiltonian circuit

  $E = E(G)$ and $\mathcal{F} = \{F \subseteq E : F \text{ is a subset of edges of a Hamiltonian circuit in } G\}$

Shortest Path Problem

Given: Graph G (directed or undirected)
$c: E(G) \to \mathbb{R}$ and $s, t \in V(G)$ s.t. t is reachable from s

Find: Shortest s-t path in G w.r.t. c

$E = E(G)$ and $\mathcal{F} = \{F \subseteq E : F$ is a subset of edges of an s-t path$\}$

Knapsack Problem

Given: $n \in \mathbb{N}$ and non-neg. c_i, w_i $(1 \leq i \leq n)$ and W

Find: Subset $S \subseteq \{1, \dots, n\}$ s.t. $\sum_{j \in S} w_j \leq W$ and $\sum_{j \in S} c_j$ is maximal

$E = \{1, \dots, n\}$ and $\mathcal{F} = \{F \subseteq E : \sum_{j \in F} w_j \leq W\}$

- ## Minimum Spanning Tree Problem

Given: Connected, undirected graph G, weights $c : E(G) \to \mathbb{R}$

Find: Minimum weight spanning tree in G

$E = E(G)$ and elements of $\mathcal{F}$ are the edge sets of forests in G

- **Maximum Weight Forest Problem**

Given: Undirected graph G and weights $c: E(G) \to \mathbb{R}$
Find: Maximum weight forest in G

$E = E(G)$ and elements of $\mathcal{F}$ are the edge sets of forests in G

- **Steiner Tree Problem**

Given: Connected undirected graph G, weights $c: E(G) \to \mathbb{R}_+$, set $T \subseteq V(G)$ of terminals

Find: Steiner tree for T, i.e., a tree S with $T \subseteq V(S)$ and $E(S) \subseteq E(G)$ s.t. $c(E(S))$ is minimum

$E = E(G)$ and $\mathcal{F}$ contains all subsets of edges of Steiner trees for T

- <u>Maximum Weight Branching Problem</u>

Given: Digraph G and weights $c: E(G) \to \mathbb{R}$

Find: A maximum weight branching ← no cycles + vertices of indegree ≤ 1

$E = E(G)$ and $\mathcal{F}$ contains the edge sets of the branchings in G

- <u>Maximum Weight Matching Problem</u>

Given: Undirected graph G and $c: E(G) \to \mathbb{R}$

Find: Maximum weight matching in G

$E = E(G)$ and $\mathcal{F}$ is the set of matchings in G

Note: The above list has NP-hard and poly-time algorithms.

Def An independence system is a **matroid** if the following extra condition holds:

(M3) if $X, Y \in \mathcal{F}$ and $|X| > |Y|$, then there is an $x \in X \setminus Y$ with $Y \cup \{x\} \in \mathcal{F}$

Examples The following independence systems $(E, \mathcal{F})$ are matroids.

a) E is the set of columns of a matrix A over some field and $\mathcal{F} := \{F \subseteq E :$ The columns in F are lin. independent over that field$\}$

b) E is the set of edges of some undirected graph G and $\mathcal{F} := \{F \subseteq E : (V(G), F)$ is a forest$\}$

c) E is a finite set, k a non-neg. integer, and
$\mathcal{F} := \{F \subseteq E : |F| \leq k\}$

d) E is the set of edges of some undirected graph G,
S is a stable set in G, $k_s \in \mathbb{Z}_+$ $(s \in S)$ and
$\mathcal{F} := \{F \subseteq E : |\delta_F(s)| \leq k_s \text{ for all } s \in S\}$

e) E is the set of edges of some digraph G, $S \subseteq V(G)$,
$k_s \in \mathbb{Z}_+$ $(s \in S)$ and $\mathcal{F} := \{F \subseteq E : |\delta_F^-(s)| \leq k_s \text{ for all } s \in S\}$

a) is called the <u>vector matroid</u> of matrix A. Let M be a matroid. If there is a matrix A over a field F such that M is the vector matroid of A, then M is called <u>representable</u> over F.

b) is called the <u>cycle matroid</u> of G and is often denoted as $M(G)$. A matroid that is the cycle matroid of some graph (that may contain loops) is called a <u>graphic matroid</u>.

c) is called a <u>uniform matroid</u>.

Theorem Let $(E, \mathcal{F})$ be an independence system. Then the following statements are equivalent:

(M3) If $X, Y \in \mathcal{F}$ and $|X| > |Y|$, then there is an $x \in X \setminus Y$ with $Y \cup \{x\} \in \mathcal{F}$

(M3') If $X, Y \in \mathcal{F}$ and $|X| = |Y| + 1$, then there is an $x \in X \setminus Y$ with $Y \cup \{x\} \in \mathcal{F}$

(M3'') For each $X \subseteq E$, all bases of X have the same cardinality.

Proof
- (M3) $\Leftrightarrow$ (M3') follows from $|X| = |Y|+1$ being a special case of $|X| > |Y|$ and there always existing a subset $X' \subseteq X$ with $|X'| = |Y|+1$ if $|X| > |Y|$.
- (M3) $\Rightarrow$ (M3'') follows from the definition of a basis
- (M3'') $\Rightarrow$ (M3): Let $X, Y \in \mathcal{F}$ and $|X| > |Y|$.
  By (M3''), Y cannot be a basis of $X \cup Y$
  $\Rightarrow$ there exists an $x \in (X \cup Y) \setminus Y$ s.t. $Y \cup \{x\} \in \mathcal{F}$ $\square$

<u>Def</u> Let $(E, \mathcal{F})$ be an independence system. For $X \subseteq E$, we define the <u>lower rank</u> by

$$\rho(X) = \min\{|Y| : Y \subseteq X,\ Y \in \mathcal{F} \text{ and } Y \cup \{x\} \notin \mathcal{F} \text{ for all } x \in X \setminus Y\}$$

The <u>rank quotient</u> of $(E, \mathcal{F})$ is defined by

$$q(E, \mathcal{F}) = \min_{F \subseteq E} \frac{\rho(F)}{r(F)}$$

$$r(X) = \max\{|Y| : Y \subseteq X, Y \in \mathcal{F}\}$$

<u>Theorem</u> Let $(E, \mathcal{F})$ be an independence system.

Then $q(E, \mathcal{F}) \leq 1$. Further, $(E, \mathcal{F})$ is a matroid if and only if $q(E, \mathcal{F}) = 1$.

Proof: $q(E, \mathcal{F}) \leq 1$ by definition of ρ and r.

$q(E, \mathcal{F}) = 1 \iff (M3'')$

$\square$

Greedy Algorithms

Let $(E, \mathcal{F})$ be an independence system and $c : E \to \mathbb{R}_+$.

We consider a maximization problem for $(E, \mathcal{F}, c)$.

negatively weighted items would not appear in any optimal solution anyway

Best-in Greedy Algorithm

Assume $(E, \mathcal{F})$ is given by an independence oracle, i.e. an oracle which, given a set $F \subseteq E$, decides whether $F \in \mathcal{F}$ or not.

Input: Ind. system $(E, \mathcal{F})$ given by an independence oracle
Weights $c: E \to \mathbb{R}_+$

Output: A set $F \in \mathcal{F}$

1. Sort $E = \{e_1, \dots, e_n\}$ such that $c(e_1) \geq c(e_2) \geq \dots \geq c(e_n)$
2. Set $F = \emptyset$
3. For $i = 1$ to n do: If $F \cup \{e_i\} \in \mathcal{F}$ then set $F = F \cup \{e_i\}$

Worst-out Greedy Algorithm

Assume $(E, \mathcal{F})$ is given by a basis superset oracle, i.e., an oracle which, given a set $F \subseteq E$, decides whether F contains a basis of $(E, \mathcal{F})$.

Input: Ind. system $(E, \mathcal{F})$ given by a basis superset oracle
Weights $c: E \to \mathbb{R}_+$

Output: A basis F of $(E, \mathcal{F})$

1. Sort $E = \{e_1, \dots, e_n\}$ such that $c(e_1) \leq c(e_2) \leq \dots \leq c(e_n)$.
2. Set $F = E$.
3. For $i = 1$ to n do: If $F \setminus \{e_i\}$ contains a basis then
   set $F = F \setminus \{e_i\}$

The two types of oracles are not polynomially equivalent (reducible from/to) for ind. systems.

Examples

- For TSP, it is easy to decide whether a set of edges is independent / a subset of a Hamiltonian circuit (because we assumed a complete graph).
  But it is NP-complete to decide whether a set of edges contains a Hamiltonian circuit.

- For Shortest Path, it is NP-complete to decide whether a given set is independent / a subset of an s-t path.
  But it is easy to decide whether a set of edges contains an s-t path.

For matroids, both oracles are polynomially equivalent, and equivalent to the rank oracle (return the rank) and closure oracle (return the closure of a given subset).

Other natural oracles are not polynomially equivalent. Deciding whether a given set is a basis is weaker than the independence oracle. Finding the minimum cardinality of a dependent subset of $\mathcal{F}$ is stronger than the independence oracle.

Excursion Matroid / Independence System Duality

Def Let $(E, \mathcal{F})$ be an independence system.
We define the dual of $(E, \mathcal{F})$ by $(E, \mathcal{F}^*)$ where
$\mathcal{F}^* = \{F \subseteq E : \text{there is a basis } B \text{ of } (E, \mathcal{F}) \text{ such that } F \cap B = \emptyset\}$

Lemma $(E, \mathcal{F}^{**}) = (E, \mathcal{F})$

Proof: $F \in \mathcal{F}^{**}$

$\Leftrightarrow$ there is a basis B^* of $(E, \mathcal{F}^*)$ such that $F \cap B^* = \emptyset$

$\Leftrightarrow$ there is a basis B of $(E, \mathcal{F})$ such that $F \cap (E \setminus B) = \emptyset$

$\Leftrightarrow$ $F \in \mathcal{F}$ $\square$

$F \subseteq B$

Theorem Let $(E, \mathcal{F})$ be an independence system, $(E, \mathcal{F}^*)$ its dual and let r and r^* be the corresponding rank functions.

a) $(E, \mathcal{F})$ is a matroid if and only if $(E, \mathcal{F}^*)$ is a matroid.

b) If $(E, \mathcal{F})$ is a matroid, then $r^*(F) = |F| + r(E \setminus F) - r(E)$ for $F \subseteq E$

Both greedy algorithms can be formulated for the minimization problem: Best-in Greedy for maximization over $(E, \mathcal{F}, c)$ corresponds to Worst-out Greedy for minimization over $(E, \mathcal{F}^*, c)$.

Note: Adding an element to F in Best-in corresponds to removing an element from $E \setminus F$ in Worst-out.

Also works for Best-in over $(E, \mathcal{F}^*, c)$ and Worst-out over $(E, \mathcal{F}, c)$.

$\Rightarrow$ Both greedy algorithms work for both problems (max/min) but Best-in is naturally associated to max and Worst-out is naturally associated to min.

The term "greedy" comes from algorithms always taking the currently best step. $\rightarrow$ Important Question:

How good is the final solution?

Theorem Let $(E, \mathcal{F})$ be an independence system. For $c: E \to \mathbb{R}_+$, denote by $G(E, \mathcal{F}, c)$ the cost of some solution found by Best-in Greedy for the maximization problem and by $Opt(E, \mathcal{F}, c)$, the cost of an optimal solution. Then

$$q(E, \mathcal{F}) \leq \frac{G(E, \mathcal{F}, c)}{Opt(E, \mathcal{F}, c)} \leq 1$$

for all $c: E \to \mathbb{R}_+$. There is a cost function where the lower bound is attained.

Proof: Let $E = \{e_1, \dots, e_n\}$, $c: E \to \mathbb{R}_+$ and $c(e_1) \geq \dots \geq c(e_n)$.
Let G_n be the solution found by Best-in Greedy while O_n is an opt. solution.
Define $E_j = \{e_1, \dots, e_j\}$, $G_j = G_n \cap E_j$ and $O_j = O_n \cap E_j$ for $j = 0, \dots, n$. Set $d_n = c(e_n)$ and $d_j = c(e_j) - c(e_{j+1})$ for $j = 1, \dots, n-1$.

the "drop" from e_j to e_{j+1}

Since $O_j \in \mathcal{F}$, $|O_j| \leq r(E_j)$. Since G_j is a basis of E_j, $|G_j| \geq \rho(E_j)$

$$\Rightarrow c(G_n) = \sum_{j=1}^{n} \left(|G_j| - |G_{j-1}|\right) \cdot c(e_j) = \sum_{j=1}^{n} |G_j| \, d_j$$

$$\geq \sum_{j=1}^{n} \rho(E_j) \, d_j \geq q(E, \mathcal{F}) \cdot \sum_{j=1}^{n} r(E_j) \, d_j$$

$$\geq q(E, \mathcal{F}) \cdot \sum_{j=1}^{n} |O_j| \, d_j$$

$$q(E, \mathcal{F}) = \min_{F \subseteq E} \frac{\rho(F)}{r(F)}$$

$$= q(E,\mathcal{F}) \sum_{j=1}^{n} \left(|O_j| - |O_{j-1}|\right) c(e_j)$$

$$= q(E,\mathcal{F})\, c(O_n)$$

$$\Rightarrow \quad q(E,\mathcal{F}) \leq \frac{c(G_n)}{c(O_n)}$$

This bound can be tight: choose $F \subseteq E$ and bases B_1, B_2 of F s.t. $\frac{|B_1|}{|B_2|} = q(E,\mathcal{F})$ and define $c(e) = \begin{cases} 1 & \text{for } e \in F \\ 0 & \text{for } e \in E \setminus F \end{cases}$.

Now sort $e_1, \dots, e_n$ s.t. $c(e_1) \geq \dots \geq c(e_n)$ and $B_1 = \{e_1, \dots, e_{|B_1|}\}$.

Then $G(E,\mathcal{F},c) = |B_1|$ and $Opt(E,\mathcal{F},c) = |B_2|$.

$\Rightarrow$ The lower bound is attained. $\square$

Corollary An ind. system $(E, \mathcal{F})$ is a matroid if and only if the Best-in Greedy algorithm finds an optimal solution for the maximization problem for $(E, \mathcal{F}, c)$ for all cost functions $c: E \to \mathbb{R}_+$.

Proof: $q(E, \mathcal{F}) < 1$ if and only if there exists a cost function c for which Best-in Greedy does not find an optimal solution. But $q(E, \mathcal{F}) < 1$ if and only if $(E, \mathcal{F})$ is not a matroid. $\square$

Punchline: Greedy algorithms are "safe" to use if you have a matroid, i.e., they return a true optimum. If you have an ind. system with a good rank quotient, i.e., close to 1, you get an equally good approximation.

Polyhedral description of matroids

Theorem Let $(E,\mathcal{F})$ be a matroid and $r: 2^E \to \mathbb{Z}_+$ its rank function. Then the matroid polytope of $(E,\mathcal{F})$, i.e., the convex hull of the incidence vectors of all elements of $\mathcal{F}$, is equal to

$$\left\{x \in \mathbb{R}^E : x \geq 0, \sum_{e\in A} x_e \leq r(A) \ \forall A \subseteq E\right\}.$$

Proof: The polytope contains all incidence vectors of independent sets. We have to show integrality of the vertices, which can be done by showing that

$$(*) \qquad \max\left\{c^T x : x \geq 0, \sum_{e\in A} x_e \leq r(A) \ \forall A \subseteq E\right\}$$

has an integral solution for any $c: E \to \mathbb{R}$.

W.l.o.g. $c(e) \geq 0 \ \forall e$, since for $e \in E$ with $c(e) < 0$ any optimal solution x of $(*)$ has $x_e = 0$.

Let x be an optimal solution of $(*)$.
Follow the transformation from $c(G_n)$ to $q(E, \mathcal{F}) \cdot c(O_n)$ in the above proof, but replace $|O_j|$ by $\sum_{e \in E_j} x_e \ \forall j = 0, \dots, n$:

$$c(G_n) \geq \underbrace{q(E, \mathcal{F})}_{=1,\ \text{matroid}} \cdot \sum_{j=1}^{n} r(E_j) d_j = \sum_{j=1}^{n} r(E_j) d_j$$

$$\geq \sum_{j=1}^{n} |O_j| d_j = \sum_{j=1}^{n} \Big(\sum_{e \in E_j} x_e \Big) d_j =$$

$$= \sum_{j=1}^{n} \Big(\underbrace{\sum_{e \in E_j} x_e - \sum_{e \in E_{j-1}} x_e}_{x_{e_j}} \Big) c(e_j) = \sum_{e \in E} c(e) \cdot x_e$$

$$\Rightarrow c(G_n) \geq \sum_{e \in E} c(e) \cdot x_e$$

$\Rightarrow$ So the Best-in Greedy algorithm produces a solution G_n whose incidence vector is an optimal solution of $(*)$.

$\Rightarrow$ Independent of the cost function, Best-in Greedy always returns an integral solution in the optimal face of the polyhedron. $\Rightarrow$ All vertices (the 0-dim. faces) have to be integral. $\square$

Recall: Best-in for Max over $(E, \mathcal{F}, c)$ corresponds to Worst-out for Min over $(E, \mathcal{F}^*, c)$.

Theorem Let $(E,\mathcal{F})$ be an ind. system. For $c: E \to \mathbb{R}_+$, let $G(E,\mathcal{F},c)$ denote a solution found by Worst-out Greedy for the minimization problem. Then

$$1 \le \frac{G(E,\mathcal{F},c)}{Opt(E,\mathcal{F},c)} \le \max_{F \subseteq E} \frac{|F| - \rho^*(F)}{|F| - r^*(F)}$$

for all $c: E \to \mathbb{R}_+$, where ρ^* and r^* are the rank functions of the dual independence systems $(E,\mathcal{F}^*)$.
There is a cost function where the upper bound is attained.

Proof: Same notation as before for E_j, G_j, O_j.
By construction, $G_j \cup (E \setminus E_j)$ contains a basis of E, but $(G_j \cup (E \setminus E_j)) \setminus \{e\}$ does not contain a basis of E for any $e \in G_j$ $\forall j = 1, \dots, n$.

$\Rightarrow$ $E_j \setminus G_j$ is a basis of E_j w.r.t $(E, \mathcal{F}^*)$, so

$$|E_j| - |G_j| \geq \rho^*(E_j)$$

Since $O_n \subseteq E \setminus (E_j \setminus O_j)$ and O_n is a basis,

$E_j \setminus O_j$ is independent in $(E, \mathcal{F}^*)$, so

$$|E_j| - |O_j| \leq r^*(E_j)$$

$\Rightarrow$ $|G_j| \leq |E_j| - \rho^*(E_j)$ and $|O_j| \geq |E_j| - r^*(E_j)$

Now follow the transformation from $c(G_n)$ to $q(E, \mathcal{F})\, c(O_n)$ in the first proof again exactly. This gives the upper bound.

To see that the bound can be tight, consider

$$c(e) = \begin{cases} 1 & \text{for } e \in F \\ 0 & \text{for } e \in E \setminus F \end{cases}$$ where $F \subseteq E$ is a set where

the maximum ratio $\dfrac{G(E,\tilde{\mathcal{F}},c)}{Opt(E,\tilde{\mathcal{F}},c)}$ is attained.

Let B_1 be a basis of F w.r.t. $(E,\mathcal{F}^*)$ with $|B_1| = \rho^*(F)$.
If $e_1,\dots,e_n$ are sorted s.t. $c(e_1) \geq \dots \geq c(e_n)$ and
$B_1 = \{e_1,\dots,e_{|B_1|}\}$, we have $G(E,\tilde{\mathcal{F}},c) = |F| - |B_1|$ and
$Opt(E,\tilde{\mathcal{F}},c) = |F| - r^*(F)$. $\square$

Punchline For Best-In & Max and Worst-Out & Min, we have performance guarantees for all ind. systems. That makes these combinations appealing.

For the other combinations (Worst-Out & Max, Best-In & Min), there is no positive lower / finite upper bound for $\dfrac{G(E,\mathcal{F},c)}{\mathrm{Opt}(E,\mathcal{F},c)}$.

For matroids, it does not matter whether you use Best-In or Worst-Out: all bases have the same cardinality, so the Minimization problem for $(E,\mathcal{F},c)$ is equivalent to the Maximization problem for $(E,\mathcal{F},c')$ where $c'(e) = M - c(e)$ for all $e \in E$ and $M = 1 + \max\{c(e) : e \in E\}$.

The characterization of a matroid through the greedy algorithm always (for all c) working also gives a characterization of optimal k-element solutions of the max problem.

<u>Theorem</u> Let $(E, \mathcal{F})$ be a matroid, $c: E \to \mathbb{R}$, $k \in \mathbb{N}$, and $X \in \mathcal{F}$ with $|X| = k$. Then $c(X) = \max\{c(Y) : Y \in \mathcal{F}, |Y| = k\}$ if and only if the following two conditions hold:

(a) For all $y \in E \setminus X$ with $X \cup \{y\} \notin \mathcal{F}$ and all $x \in \underline{C(X,y)}$, we have $c(x) \geq c(y)$.

(b) For all $y \in E \setminus X$ with $X \cup \{y\} \in \mathcal{F}$ and all $x \in X$, we have $c(x) \geq c(y)$.

Fact and Notation from 13.2: For $X \in \mathcal{F}$ and $e \in E$ s.t. $X \cup \{e\} \notin \mathcal{F}$:
There is a unique circuit in $X \cup \{e\}$ and we denote it by $C(X,e)$.

Proof: Necessity can be seen by observing that if one of the conditions is violated for some y and x, the k-element set $X' = (X \cup \{y\}) \setminus \{x\} \in \mathcal{F}$ has greater weight than X.

It remains to show sufficiency.
Let $\mathcal{F}' = \{F \in \mathcal{F} : |F| \leq k\}$ and $c'(e) = c(e) + M$ for all $e \in E$, where $M = \max\{|c(e)| : e \in E\}$. Sort $E = \{e_1, \ldots, e_n\}$ such that $c'(e_1) \geq \ldots \geq c'(e_n)$ and, for any i, $c'(e_i) = c'(e_{i+1})$ and $e_{i+1} \in X$ imply $e_i \in X$.
(i.e., elements of X come first among equal-weight ones)

Now run Best-In Greedy to find solution X' for instance $(E, \mathcal{F}', c')$ where Step 1 sorts as above.
Since $(E, \mathcal{F}')$ is a matroid, we know

$$c(X') + kM = c'(X') = \max\{c'(Y) : Y \in \mathcal{F}'\} =$$
$$= \max\{c(Y) : Y \in \mathcal{F}, |Y| = k\} + kM$$

It remains to prove that $X = X'$. We know that $|X| = k = |X'|$.
So suppose $X \neq X'$ and let $e_i \in X' \setminus X$ with i minimum.
Then $X \cap \{e_1, \dots, e_{i-1}\} = X' \cap \{e_1, \dots, e_{i-1}\}$.
Now if $X \cup \{e_i\} \notin \mathcal{F}$, then (a) implies $C(X, e_i) \subseteq X'$. ↯
If $X \cup \{e_i\} \in \mathcal{F}$, then (b) implies $X \subseteq X'$. ↯

$\square$

Approximation Theory

Approximation algorithms find provably good approximations for problems. Algorithms without a guarantee are called **heuristics**.

There are several concepts for "provably good" approximations. Often they imply that the approximation algorithm is polynomial (and the underlying exact problem is NP-hard).

Def An absolute approximation algorithm for an opt. problem P is a poly-time algorithm A for P for which there exists a constant k s.t.

$$|A(I) - Opt(I)| \le k$$

for all instances I of P.

$A(I)$, $Opt(I)$: obj. fun. values of app. algorithm / true optimum

Rare in practice.

More common: relative performance guarantees

Def Let P be an opt. problem with non-negative weights (c) and $k \geq 1$. A <u>k-factor approximation algorithm</u> for P is a poly-time algorithm A for P such that

$$\frac{1}{k} Opt(I) \leq A(I) \leq k \cdot Opt(I)$$

for all instances I of P. We also say that A has <u>performance ratio / guarantee k</u>.

Note: $\frac{1}{k} Opt(I) \to$ max guarantee

$k\, Opt(I) \to$ min guarantee

Note: for instances I with $Opt(I) = 0$, exact sol $A(I) = 0$ is required

Sometimes k is a function of the instance I and the same terms are used.

First example: Best-in Greedy Algorithm for maximization over ind. system $(E, \mathcal{F})$ has performance ratio $\frac{1}{q(E,\mathcal{F})}$.

Example **Min Weight Set Cover Problem**

Instance: A set system $(U, \mathcal{S})$ with $\bigcup_{S \in \mathcal{S}} S = U$, weights $c : \mathcal{S} \to \mathbb{R}_+$

Task: Find a min weight set cover of $(U, \mathcal{S})$, i.e., a subfamily $\mathcal{R} \subseteq \mathcal{S}$ s.t. $\bigcup_{R \in \mathcal{R}} R = U$, of minimal cost

Special Case: $c = 1$: Minimum Set Cover Problem

Special Case if $|\{S \in \mathcal{S} : x \in S\}| = 2 \ \forall x \in U$:

Minimum Weight Vertex Cover Problem

Given Graph G, $c : V(G) \to \mathbb{R}_+$, instance is defined by

$U := E(G)$, $\mathcal{S} = \{\delta(v) : v \in V(G)\}$ and

$c(\delta(v)) = c(v) \ \forall v \in V(G)$

set of all incident edges

Minimum Weight Vertex Cover is NP-hard

$\Rightarrow$ Minimum Weight Set Cover is NP-hard

Greedy Algorithm for Set Cover

Input: Set system $(U, \mathcal{S})$ with $\bigcup_{S \in \mathcal{S}} S = U$, weights $c : \mathcal{S} \to \mathbb{R}_+$

Output: Set cover $\mathcal{R}$ of $(U, \mathcal{S})$

1. Set $\mathcal{R} := \emptyset$, $W := \emptyset$

2. While $W \neq U$ do

   - choose a set $R \in \mathcal{S} \setminus \mathcal{R}$ with $R \setminus W \neq \emptyset$ and $\dfrac{c(R)}{|R \setminus W|}$ is minimum
   - set $\mathcal{R} := \mathcal{R} \cup \{R\}$ and $W := W \cup R$

Running Time $O(|U||S|)$: $|U|$ is size of ground set and an upper bound on the number of iterations. $|S|$ is the size of the set system and each iteration runs in $O(|S|)$.

<u>Theorem</u> For any instance (U, S, c) of Min Weight Set Cover Problem, the Greedy Alg. for Set Cover finds a set cover whose weight is at most $H(r) \cdot Opt(U, S, c)$ where $r := \max_{S \in S} |S|$ and $H(r) = 1 + \frac{1}{2} + \dots + \frac{1}{r}$.

Proof: Let (U, S, c) be an instance of Min Weight Set Cover and let $R = \{R_1, \dots, R_k\}$ be the solution of the greedy alg. with R_i chosen in the i-th iteration.

$\rightarrow$ For $j = 0, \dots, k$, let $W_j := \bigcup_{i=1}^{j} R_i$.

For each $e \in \mathcal{U}$, let $j(e) := \min\{j \in \{1, \dots, k\} : e \in R_j\}$ be the iteration where e is covered. (first covered)

Let $$y(e) := \frac{c(R_{j(e)})}{|R_{j(e)} \setminus W_{j(e)-1}|}$$

Let $S \in \mathcal{S}$ and let $k' := \max\{j(e) : e \in S\}$.

latest pickup of any item in S

$$\Rightarrow \sum_{e \in S} y(e) = \sum_{i=1}^{k'} \sum_{e \in S : j(e) = i} y(e) =$$

$$= \sum_{i=1}^{k'} \frac{c(R_i)}{|R_i \setminus W_{i-1}|} \cdot |S \cap (W_i \setminus W_{i-1})|$$

$$= \sum_{i=1}^{k'} \frac{c(R_i)}{|R_i \setminus W_{i-1}|} \cdot \left(|S \setminus W_{i-1}| - |S \setminus W_i| \right)$$

$$\leq \sum_{i=1}^{k'} \frac{c(S)}{|S \setminus W_{i-1}|} \cdot \left(|S \setminus W_{i-1}| - |S \setminus W_i| \right)$$

by choice of R_i in 2 and noting $S \setminus W_{i-1} \neq \emptyset \ \forall i = 1, \dots, k'$

By writing $s_i = |S \setminus W_{i-1}|$, we get

$$\sum_{e \in S} y(e) \leq c(S) \cdot \sum_{i=1}^{k'} \frac{s_i - s_{i+1}}{s_i}$$

$$\leq c(S) \sum_{i=1}^{k'} \left(\frac{1}{s_i} + \frac{1}{s_i - 1} + \dots + \frac{1}{s_{i+1} + 1} \right)$$

$$= c(S) \sum_{i=1}^{k'} \left(H(s_i) - H(s_{i+1}) \right)$$

$$= c(S) \left(H(s_1) - H(s_{k'+1}) \right)$$

$$\leq c(S) H(s_1)$$

Since $s_1 = |S|$, we get

$$\sum_{e \in S} y(e) \leq c(S) H(r)$$

as $r = \max_{S \in \mathcal{S}} |S|$.

We sum over all $S \in \mathcal{O}$ for an opt. set cover $\mathcal{O}$ and get

$$c(\mathcal{O}) \cdot H(r) \geq \sum_{S \in \mathcal{O}} \sum_{e \in S} y(e) \geq \sum_{e \in U} y(e)$$

$$= \sum_{i=1}^{k} \sum_{e \in U : j(e) = i} y(e) = \sum_{i=1}^{k} c(R_i) = c(\mathcal{R})$$

$\square$

Comment: Not the tightest analysis possible, but

- there exists a constant $c > 0$ s.t. no performance ratio of $c \cdot \ln |U|$ can be achieved (if $P \neq NP$)
- no ratio $c \cdot \ln |U|$ for any $c < 1$ can be achieved unless each problem in NP can be solved in $O(n^{O(\log \log n)})$ time

Minimum Weight Edge Cover is a special case of Minimum Weight Set Cover with $r = 2$

$r = \max_{S \in \mathcal{S}} |S|$

$\Rightarrow$ the greedy algonithm then is a $\frac{3}{2}$ factor approximation

$H(2) = 1 + \frac{1}{2}$

but it is poly-time soloable anyway

For Minimum Vertex Cover, the greedy algorithm becomes

Greedy Algorithm for Vertex Cover

Input: Graph G

Output: Vertex cover R of G

1. Set $R := \emptyset$.
2. While $E(G) \neq \emptyset$ do
   - choose a vertex $v \in V(G) \setminus R$ with maximal degree
   - set $R := R \cup \{v\}$ and delete all edges incident to it.

There is no k s.t. this is a k-factor app. algorithm.

The bound from the previous theorem is best possible.

Theorem For all $n \geq 3$, there is an instance G of the Minimum Vertex Cover Problem such that

$$nH(n-1)+2 \leq |V(G)| \leq nH(n-1)+n,$$

the max degree of G is $n-1$, $Opt(G)=n$ and the above algorithm finds a vertex cover containing all but n vertices.

Theorem For all $n \geq 3$, there is an instance G of the Minimum Vertex Cover Problem such that

$$nH(n-1)+2 \leq |V(G)| \leq nH(n-1)+n,$$

the max degree of G is $n-1$, $Opt(G)=n$ and the above algorithm finds a vertex cover containing all but n vertices.

Proof: For each $n \geq 3$ and $i \leq n$, we define $A_n^i := \sum_{j=2}^{i} \lfloor \frac{n}{j} \rfloor$ and

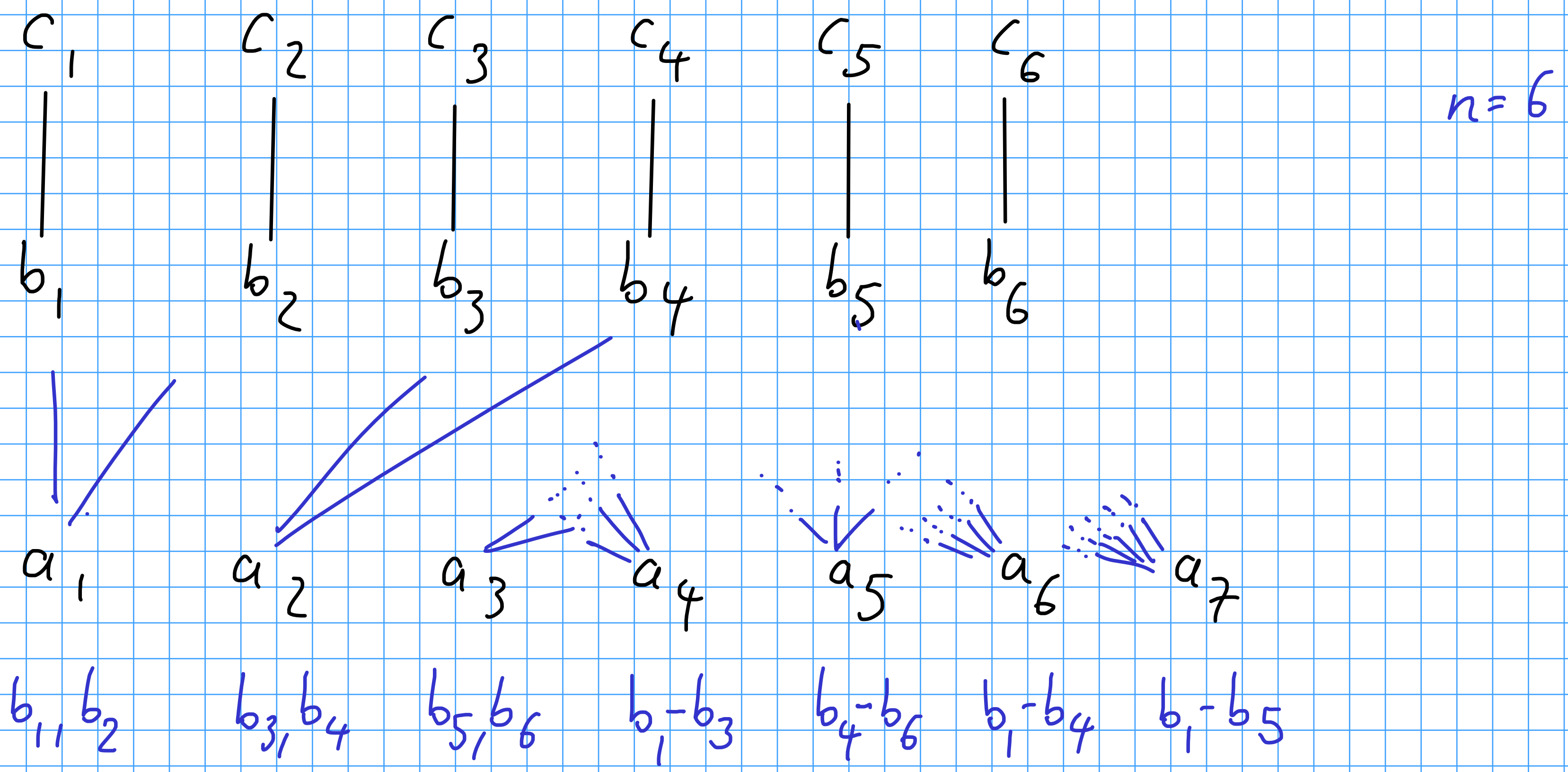
$$V(G_n) := \{a_1, \dots, a_{A_n^{n-1}}, b_1, \dots, b_n, c_1, \dots, c_n\}$$

$$E(G_n) := \{\{b_i, c_i\} : i = 1, \dots, n\} \cup \bigcup_{i=2}^{n-1} \bigcup_{j=A_n^{i-1}+1}^{A_n^i} \{\{a_j, b_k\} : (j - A_n^{i-1} - 1) \cdot i + 1 \leq k \leq (j - A_n^{i-1}) \cdot i\}$$

Note that : $\cdot\ |V(G_n)| = 2n + A_n^{n-1}$

$\cdot\ A_n^{n-1} \leq n H(n-1) - n$

$\cdot\ A_n^{n-1} \geq n H(n-1) - n - (n-2)$



Apply greedy algorithm to G_n $\rightarrow$ It may first choose $a_{A_n^{n-1}}$ because it is of max degree, and subsequently $a_{A_n^{n-1}-1}, a_{A_n^{n-1}-2}, \dots, a_1$. Then there still are n pairwise disjoint edges left, so n more vertices are needed.

$\Rightarrow$ constructed vertex cover has $A_n^{n-1} + n$ vertices,
but the optimum vertex cover $\{b_1, \dots, b_n\}$ just n $\square$

So this algorithm can be arbitrarily bad. But there exists a 2-factor approximation algorithm: find any maximal matching M and take the ends of all edges in M

inclusion-maximal

$\Rightarrow$ $2|M|$ vertices and any vertex cover must contain at least $|M|$ vertices

This is the best <u>known</u> approximation algorithm for Minimum Vertex Cover. There exists a $k > 1$ such that no k-factor approximation algorithm exists unless $P = NP$.

A 1.36-factor approximation algorithm does not exist unless $P = NP$.

Approximation Schemes

The existence of an absolute approximation algorithm does not imply the existence of a k-factor approximation algorithm for all $k > 1$.

$\Rightarrow$ new term to "combine" the concepts

<u>Definition</u> Let P be an opt. problem with non-negative weights. An <u>asymptotic k-factor approximation algorithm</u> for P is a poly-time algorithm A for which there exists a constant c such that

$$\frac{1}{k}\,\mathrm{Opt}(I) - c \leq A(I) \leq k\,\mathrm{Opt}(I) + c$$

for all instances I of P. We say that A has <u>asymptotic performance ratio k</u>.

The <u>asymptotic approximation ratio</u> of an Opt. Problem P with non-negative weights is defined to be the infimum of all numbers k for which there exists an (asymptotic) k-factor approximation algorithm for P, or ∞ if there is none such

<u>Example</u> Edge Coloring has approximation $\frac{4}{3}$ (i.e., there exists a polynomial-time $\frac{4}{3}$-factor approximation algorithm), but asymptotic approximation ratio 1 (because of the existence of an absolute approximation algorithm).

Note: The $\frac{4}{3}$ ratio for edge coloring comes from

- a graph with max degree k requiring at least k colors
- the existence of a poly-time algorithm that returns a k+1 coloring of a graph with max degree k

- the ability to easily check for 1- or 2-colorability

$\Rightarrow$ the approximation ratio for a graph with max degree k is $\frac{k+1}{k}$, which is worst for $k=3$ and $\frac{4}{3}$

<u>Definition</u> Let P be an optimization problem with non-negative weights. An <u>approximation scheme</u> for P is an algorithm A accepting as input an instance I of P and an $\varepsilon > 0$ such that, for each fixed ε, A is a $(1+\varepsilon)$-factor approximation algorithm for P.

<u>Definition</u> An <u>asymptotic approximation scheme</u> for P is a pair of algorithms (A, A') with the following properties:

- A' is a poly-time algorithm accepting a number $\varepsilon > 0$ as input and computing a number c_ε.

- A accepts an instance I of P and an $\varepsilon > 0$ as input, and its output consists of a feasible solution for I satisfying

$$\frac{1}{1+\varepsilon} \operatorname{Opt}(I) - c_\varepsilon \le A(I, \varepsilon) \le (1+\varepsilon) \operatorname{Opt}(I) + c_\varepsilon .$$

For fixed ε, the running time of A is polynomially bounded in $\operatorname{size}(I)$.

<u>Definition</u> An (asymptotic) approximation scheme is called a <u>fully polynomial (asymptotic) approximation scheme</u> if the running time, as well as the maximum size of any number occurring in the computation, is bounded by a polynomial in $\operatorname{size}(I) + \operatorname{size}(\varepsilon) + \frac{1}{\varepsilon}$, and c_ε is a polynomial in $\frac{1}{\varepsilon}$.

Abbreviations: PTAS, FPTAS

Apart from absolute approximation algorithms, an FPTAS can be considered the best one may hope for when faced with NP-hard optimization problems (under mild assumptions: all-feasible solutions non-negative integers).

Fixed Numbers vs. Free Input

$n := \text{size}(I)$, $s_\varepsilon = \text{size}(\varepsilon)$

$n^{\varepsilon}, n^{\frac{1}{\varepsilon}}, n^{s_\varepsilon}$ are polynomial terms for fixed ε,

$\to$ okay as running time for PTAS

but not polynomial for free ε

$\to$ not okay as running time for FPTAS

note: $\text{size } x = \lceil \log x \rceil + 2$ for $x \in \mathbb{Z}$

$\text{size } \frac{x}{y} = \text{size } x + \text{size } y$

$\Rightarrow$ encoding length s_ε is logarithmic in $\varepsilon \in \mathbb{Z}$

$\Leftrightarrow$ ε is exponential ($\to$ not polynomial) in s_ε

What is an example of a running time polynomial in $n + s_\varepsilon + \frac{1}{\varepsilon}$, but not $n + s_\varepsilon$?

FPTAS "better than FPTAS $\Rightarrow$ poly-time exact" theorem

Answer: a running time of $\frac{1}{\varepsilon}$ itself is not poly in $n + s_\varepsilon$

(but of course poly in $n + s_\varepsilon + \frac{1}{\varepsilon}$)

can assume you look at given instance I ($\Rightarrow n = \text{size}(I)$ fixed) and $\varepsilon \to 0$. think of $\varepsilon = \frac{1}{y}$ as a rational number with nominator $y \to \infty$, $y \in \mathbb{Z}$

$y = \frac{1}{\varepsilon} \to \infty$ and y grows exponentially compared to s_ε

(and thus is not polynomial w.r.t. $n + s_\varepsilon$)

Theorem Let $P = \big(\overset{\text{instances}}{X},\ \overset{\text{set of feasible solutions}}{(S_x)_{x \in X}},\ c,\ \max/\min\big)$ be an NP optimization problem where the values of obj. function c are non-negative integers. Let A be an algorithm which, given an instance I of P and a number $\varepsilon > 0$, computes a feasible solution of I with

$$\frac{1}{1+\varepsilon}\, Opt(I) \leq A(I,\varepsilon) \leq (1+\varepsilon)\, Opt(I)$$

and whose running time is bounded by a polynomial in $\text{size}(I) + \text{size}(\varepsilon)$. Then P can be solved exactly in polynomial time.

Proof Given instance I, we first run A on $(I, 1)$.
We set $\varepsilon = \frac{1}{1+2A(I,1)}$ and see that $\varepsilon \cdot Opt(I) < 1$

Now we run A on (I, ε). Since size(ε) is poly. bounded in size(I), this procedure constitutes a poly-time algorithm.

If P is a min problem, we have

$$A(I, \varepsilon) \leq (1+\varepsilon)\, Opt(I) < Opt(I) + 1$$

which, since c is integral, implies optimality.

Similarly, if P is a max problem, we have

$$A(I, \varepsilon) \geq \frac{1}{1+\varepsilon}\, Opt(I) > (1-\varepsilon)\, Opt(I) > Opt(I) - 1.$$

$\square$

The Knapsack Problem

"The easiest NP-hard problem"

- without integrality, it becomes linear time
- there exists an exact pseudo-poly. time algorithm
- there exists an FPTAS

Definition Knapsack Problem

Instance: $n, c_1, \ldots, c_n, w_1, \ldots, w_n, W \in \mathbb{Z}_+$

Task: Find a subset $S \subseteq \{1, \ldots, n\}$ s.t. $\sum_{j \in S} w_j \leq W$ and $\sum_{j \in S} c_j$ is maximum

Knapsack Problems without Integrality

Definition Fractional Knapsack Problem

Instance: $n, c_1, \dots, c_n, w_1, \dots, w_n, W \in \mathbb{Z}_+$

Task: Find numbers $x_1, \dots, x_n \in [0,1]$ s.t.

$$\sum_{j=1}^{n} x_j w_j \leq W \text{ and } \sum_{j=1}^{n} x_j c_j \text{ is maximum}$$

Can be solved exactly through an algorithm sorting elements appropriately.

Theorem Let $c_1, \dots, c_n, w_1, \dots, w_n, W \in \mathbb{Z}_+$ with $\sum_{i=1}^{n} w_i > W$,

$\{1 \le i \le n : w_i = 0\} = \{1, \dots, h\}$ and

$$\frac{c_{h+1}}{w_{h+1}} \ge \frac{c_{h+2}}{w_{h+2}} \ge \dots \ge \frac{c_n}{w_n},$$

and let

$$k := \min\left\{ j \in \{1, \dots, n\} : \sum_{i=1}^{j} w_i > W \right\}.$$

Then an opt. solution of the given instance of Fractional Knapsack is defined by

$$x_j = 1 \quad \text{for } j = 1, \dots, k-1$$

$$x_k = \left(W - \sum_{j=1}^{k-1} w_j\right) / w_k$$

$$x_j = 0 \quad \text{for } j = k+1, \dots, n$$

Sorting: $O(n \log n)$

Computing k: $O(n)$ $\to$ efficient exact solution

One can do better as a special case of weighted median search, which can be solved in linear time.

<u>Definition</u> Let $n \in \mathbb{N}$, $z_1, \dots, z_n \in \mathbb{R}$, $w_1, \dots, w_n \in \mathbb{R}_+$ and $W \in \mathbb{R}$ with $0 < W < \sum_{i=1}^{n} w_i$. Then the <u>$(w_1, \dots, w_n; W)$-weighted median with respect to $(z_1, \dots, z_n)$</u> is defined to be the unique numbers z^* for which

$$\sum_{i: z_i < z^*} w_i < W \leq \sum_{i: z_i \leq z^*} w_i$$

Definition **Weighted Median Problem**

Instance: $n \in \mathbb{N}$, $z_1, \dots, z_n \in \mathbb{R}$, $w_1, \dots, w_n \in \mathbb{R}_+$, and $W > 0$ with $W \leq \sum_{i=1}^{n} w_i$

Task: Find the $(w_1, \dots, w_n, W)$-weighted median w.r.t. $(z_1, \dots, z_n)$.

Special Case:

Definition **Selection Problem**

Instance: $n \in \mathbb{N}$, $z_1, \dots, z_n \in \mathbb{R}$, and an integer $k \in \{1, \dots, n\}$

Task: Find the k-smallest numbers among $z_1, \dots, z_n$, i.e., an index $i \in \{1, \dots, n\}$ with

$$|\{j : z_j < z_i\}| < k \leq |\{j : z_j \leq z_i\}|$$

Weighted median can be solved in $O(n)$.

Weighted Median Algorithm

Input: $n \in \mathbb{N}$, $z_1, \dots, z_n \in \mathbb{R}$, $w_1, \dots, w_n \in \mathbb{R}^+$, and $W > 0$ with $W \leq \sum_{i=1}^{n} w_i$

Output: The $(w_1, \dots, w_n; W)$-weighted median w.r.t. $(z_1, \dots, z_n)$

1. Partition the list $z_1, \dots, z_n$ into blocks of five elements each (possible less in the last block)

   Find the (non-weighted) median of each block

   Let M be the list of these $\lceil \frac{n}{5} \rceil$ median elements.

(2) Find the non-weighted median of M. Let it be z_m.

(3) Compare each element with z_m. W.l.o.g. let $z_i < z_m$ for $i = 1, \dots, k$. $z_i = z_m$ for $i = k+1, \dots, l$ and $z_i > z_m$ for $i = l+1, \dots, n$.

(4) If $\sum_{i=1}^{k} w_i < W \le \sum_{i=1}^{l} w_i$ then stop with $z^* = z_m$

If $\sum_{i=1}^{l} w_i < W$ then find recursively the $(w_{l+1}, \dots, w_n; W - \sum_{i=1}^{l} w_i)$-weighted median w.r.t. $(z_{l+1}, \dots, z_n)$. stop

If $\sum_{i=1}^{k} w_i \ge W$ then find recursively the $(w_1, \dots, w_k; W)$-weighted median w.r.t. $(z_1, \dots, z_k)$. stop

Theorem The Weighted Median Algorithm works correctly and takes time $O(n)$.

Proof: Correctness is the "easy" part of the proof.
We focus on the running time.
The worst-case running time of n elements is denoted by $f(n)$. It satisfies

$$f(n) = \underset{①}{O(n)} + \underset{②}{f\left(\left\lceil \frac{n}{5} \right\rceil\right)} + \underset{③}{O(n)} + \underset{④}{f\left(\frac{1}{2}\left\lceil \frac{n}{5}\right\rceil 5 + \frac{1}{2}\left\lceil \frac{n}{5}\right\rceil 2\right)}$$

because the recursive call in ④ omits at least three elements of at least half of the 5-element blocks.

The above recursion formula yields $f(n) = O(n)$:

as $\lceil \frac{n}{5} \rceil \leq \frac{9}{41} n$ for all $n \geq 37$, one obtains

$$f(n) \leq c \cdot n + f\left(\frac{9}{41} n\right) + f\left(\frac{7}{2} \cdot \frac{9}{41} n\right)$$

for a suitable c and $n \geq 37$. Given this, one obtains

$$f(n) \leq (82c + f(36))\, n \text{ by induction}$$

$$\Rightarrow \quad f(n) \in O(n)$$

$\square$

<u>Corollary</u> The Selection Problem can be solved in $O(n)$ time.

<u>Corollary</u> The Fractional Knapsack Problem can be solved in $O(n)$ time.

Integral Knapsack

Some of the previous ideas help.

Theorem Let $c_1, \dots, c_n, w_1, \dots, w_n,$ and $W \in \mathbb{Z}_+$ with $w_j \le W \;\forall j = 1, \dots, n$, $\sum_{i=1}^{n} w_i > W$, and

$$\frac{c_1}{w_1} \ge \dots \ge \frac{c_n}{w_n}.$$

Let $k := \min\{ j \in \{1, \dots, n\} : \sum_{i=1}^{j} w_i > W \}$.

Then choosing the better of the two feasible solutions $\{1, \dots, k-1\}$ and $\{k\}$ constitutes a 2-factor approximation algorithm for Knapsack with running time $O(n)$.

Proof Items i with $w_i > W$ do no fit in the Knapsack and can be ignored. If $\sum_{i=1}^{n} w_i < W$, then the whole set $\{1, \dots, n\}$ is optimal. Else we compute number k in $O(n)$ by solving a Weighted Median Problem.

Recall $\sum_{i=1}^{k} c_i$ is an upper bound on the optimal value for fractional Knapsack, so also for integral Knapsack (which is formally a restriction).

$\Rightarrow$ The better of the two feasible solutions $\{1, \dots, k-1\}$ and $\{k\}$ achieves at least half optimal value $\square$

So there is a linear-time 2-approximation.

Theorem The Knapsack Problem is NP-hard

"strongly NP-hard" $\hat{=}$ NP-hard even if actual numbers are polynomial in size of the input

Knapsack is <u>not</u> strongly NP-hard!

We are going to describe a pseudo-polynomial algorithm.
The existence of a pseudo-polynomial algorithm disproves strong NP-hardness.

There is a pseudo-polynomial algorithm for Subset Sum.

$\rightarrow$ generalize and you get an algorithm with running time $O(n \cdot W)$

We take a different approach to get $O(nC)$ running time, where $C = \sum_{j=1}^{n} c_j$.

Idea: <u>Dynamic Programming</u>

- based on a recursion formula
- solving a problem by recursively breaking it down into simpler subproblems
- memory-less incremental build-up of partial solutions

Dynamic Programming Knapsack Algorithm

Input: $n, c_1, \dots, c_n, w_1, \dots, w_n, W \in \mathbb{Z}_+$

Output: Subset $S \subseteq \{1, \dots, n\}$ such that $\sum_{j \in S} w_j \leq W$ and $\sum_{j \in S} c_j$ is maximum

1. Let C be any upper bound on the value of the opt. solution, e.g., $C = \sum_{j=1}^{n} c_j$ (could use lower C, like 2 times 2-factor-opt)

2. Set $x(0,0) = 0$ and $x(0,k) = \infty$ for $k = 1, \dots, C$

3. For $j = 1$ to n do

   For $k = 0$ to C do

   Set $s(j,k) = 0$ and $x(j,k) = x(j-1,k)$

For $k = c_j$ to C do

If $x(j-1, k-c_j) + w_j \leq \min\{W, x(j,k)\}$ then

Set $x(j,k) = x(j-1, k-c_j) + w_j$ and $s(j,k) = 1$

(4) Let $k = \max\{i \in 0, \dots, C : x(n,i) < \infty\}$. Set $S = \emptyset$.

For $j = n$ down to 1 do

If $s(j,k) = 1$ then set $S = S \cup \{j\}$ and $k = k - c_j$.

<u>Theorem</u> The Dynamic Programming Knapsack Algorithm finds an exact opt. solution in $O(n \cdot C)$ time.

Proof Running time: constant effort for each of the $n \cdot C$ entries of arrays x and s

Variable $x(j,k)$ denotes the minimal total weight of a subset $S \subseteq \{1,\dots,j\}$ with $\sum_{i\in S} w_i \le W$ and $\sum_{i\in S} c_i = k$.

The algorithm computes these values using the recursion formula:

$$x(j,k) = \begin{cases} x(j-1,k-c_j)+w_j & \text{if } c_j \le k \text{ and } x(j-1,k-c_j)+w_j \le \min\{W, x(j-1,k)\} \\ x(j-1,k) & \text{otherwise} \end{cases}$$

for $j=1,\dots,n$, $k=0,\dots,C$.

The variables $s(j,k)$ indicate which of the two cases apply.

So the algorithm enumerates all subsets $S \subseteq \{1,\dots,n\}$ except those that are not feasible or "dominated" by others. In 4, the best feasible subset is chosen. $\square$

An FPTAS for Knapsack

Knapsack has no absolute app. algorithm

But there exists an FPTAS!

Recall: running time of Dynamic Programming algorithm depends on C

$\Rightarrow$ Idea: Divide all numbers $c_1, \dots, c_n$ by 2 or more and round

$\Rightarrow$ reduced running time, but inaccurate solution because of rounding

In general: Setting $\bar{c}_j = \lfloor \frac{c_j}{t} \rfloor \quad \forall j = 1, \dots, n$

will reduce the running time by about a factor t

A tradeoff of running time and accuracy is the hallmark feature of approximation schemes.

For $S \subseteq \{1, \dots, n\}$, $c(S) = \sum_{i \in S} c_i$.

Knapsack Approximation Scheme

Input: $n, c_1, \dots, c_n, w_1, \dots, w_n, W \in \mathbb{Z}_+$, $\varepsilon > 0 \in \mathbb{R}$

Output: A subset $S \subseteq \{1, \dots, n\}$ s.t. $\sum_{j \in S} w_j \leq W$ and

$$c(S) \geq \frac{1}{1+\varepsilon} c(S') \quad \forall S' \subseteq \{1, \dots, n\} \text{ with } \sum_{j \in S'} w_j \leq W$$

1. Run the 2-factor app. algorithm and let S_1 be its solution. If $c(S_1) = 0$ then set $S := S_1$ and stop.

2. Set $t = \max\left\{1, \dfrac{\varepsilon \cdot c(S_1)}{n}\right\}$

   Set $\bar{c}_j = \left\lfloor \dfrac{c_j}{t} \right\rfloor$ for $j = 1, \dots, n$

3. Apply the Dynamic Programming Knapsack Alg. to the instance $(n, \bar{c}_1, \dots, \bar{c}_n, w_1, \dots, w_n, W)$ and use $C = \dfrac{2 \cdot c(S_1)}{t}$.

   Let S_2 be the solution.

4. If $c(S_1) > c(S_2)$ then set $S := S_1$.

   Else set $S := S_2$.

Theorem The Knapsack App. Scheme is an FPTAS for Knapsack. Its running time is $O(n^2 \frac{1}{\varepsilon})$.

Proof: If alg. stops in 1, then S_1 is optimal.

So assume $c(S_1) > 0$.

Let S^* be an opt. solution of original instance. Since $2 \cdot c(S_1) \geq c(S^*)$, C in 3 is a correct upper bound on the value of the opt. solution of the instance solved in 3 where costs were divided by t and rounded down.

$\Rightarrow$ S_2 is an exact opt. solution of <u>this</u> instance in 3.

$$\Rightarrow \sum_{j \in S_2} c_j \geq \sum_{j \in S_2} t \bar{c}_j = t \sum_{j \in S_2} \bar{c}_j \geq t \sum_{j \in S^*} \bar{c}_j =$$

$$= \sum_{j \in S^*} t \bar{c}_j > \sum_{j \in S^*} (c_j - t) \geq c(S^*) - \underline{nt}$$

If $t = 1$, then S_2 is optimal. Else, the above inequality implies $c(S_2) \geq c(S^*) - \underline{\varepsilon c(S_1)}$

$$\Rightarrow (1+\varepsilon) c(S) \geq c(S_2) + \varepsilon c(S_1) \geq c(S^*)$$

$\Rightarrow$ We have a $(1+\varepsilon)$-factor app. alg. for any fixed $\varepsilon > 0$.

Running time bottleneck is 3 and gives us

$$O(nC) = O\left(\frac{n\, c(S_1)}{t}\right) = O\left(n^2 \cdot \frac{1}{\varepsilon}\right) \qquad \square$$

Not many problems have an FPTAS.

→ We want to state this fact more precisely.

Consider the Maximization Problem for Independence Systems.

(Knapsack is such a problem.)

In Dyn. Prog. & App. Scheme for Knapsack, we used a certain Dominance relation, which we now generalize.

<u>Definition</u> Given an ind. system $(E, \mathcal{F})$, a cost function

$c: E \to \mathbb{Z}_+$, subsets $S_1, S_2 \subseteq E$ and $\varepsilon \geq 0$.

S_1 <u>ε-dominates</u> S_2 if $\frac{1}{1+\varepsilon} c(S_1) \leq c(S_2) \leq (1+\varepsilon)\, c(S_1)$

and there is a basis B_1 with $S_1 \subseteq B_1$, s.t. for each basis B_2 with $S_2 \subseteq B_2$, we have $(1+\varepsilon)\, c(B_1) \geq c(B_2)$

ε-Dominance Problem

Instance: An ind. system $(E, \mathcal{F})$, a cost function $c: E \to \mathbb{Z}_+$, $\varepsilon \geq 0$ and two subsets $S_1, S_2 \subseteq E$.

Question: Does S_1 ε-dominate S_2?

Ind. system is given by an independence oracle.

Dynamic Prog. Knapsack made frequent use of 0-dominance (and that was easy/efficient → but this is quite unusual).

The existence of an efficient algorithm for the ε-Dominance problem is essential for an FPTAS!

Theorem Let $\mathcal{I}$ be a family of ind. systems. Let $\mathcal{I}'$ be the family of instances $(E, \mathcal{F}, c)$ of the Max Problem for ind. systems with $(E, \mathcal{F}) \in \mathcal{I}$ and $c: E \to \mathbb{Z}_+$ and let $\mathcal{I}''$ be the family of instances $(E, \mathcal{F}, c, \varepsilon, S_1, S_2)$ of the ε-Dominance Problem with $(E, \mathcal{F}) \in \mathcal{I}$.

Then there exists an FPTAS for the Max Problem for ind. systems restricted to $\mathcal{I}'$ <u>if and only if</u> there exists an algorithm for the ε-Dominance problem restricted to $\mathcal{I}''$ whose running time is bounded by a polynomial in the length of the input and $\frac{1}{\varepsilon}$.

If an FPTAS exists at all, then it is basically a modified version of the Knapsack App. Scheme.

A Tutorial on Circuits and Circuit Walks

Some Background on Polyhedral Theory

general polyhedron $\quad P = \{x \in \mathbb{R}^n : Ax = b,\ Bx \leq d\}$

standard form $\quad P = \{x \in \mathbb{R}^n : Ax = b,\ x \geq 0\}$

canonical form $\quad P = \{x \in \mathbb{R}^n : Bx \leq d\}$

special cases with $B = -I$, $d = 0$

$\ker A = \mathbb{R}^n$

- $Ax = b$: affine subspace that contains P
- $Bx \leq d$, $x \geq 0$ (facets that) bound the feasible region

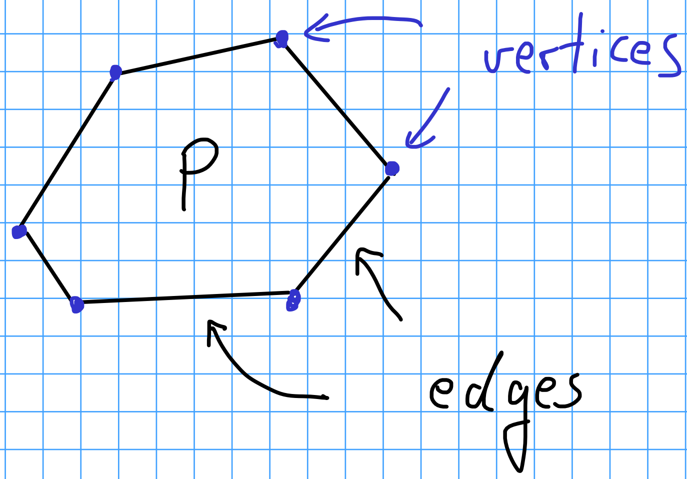
Vertices and Edges: Graph of a Polyhedron

A <u>k-dim. face</u> F of a polyhedron P is a set $F \subseteq P$ fully contained in a k-dim. affine subspace S formed from all equalities and a subset of the inequalities (satisfied with equality), but no $(k-1)$-dim. affine subspace.

$\Rightarrow$ proper faces lie on the relative boundary of P

(improper: $\dim F = \dim P$)

- facets: $\dim F = \dim P - 1$
- vertices: $\dim F = 0$
- edges: $\dim F = 1$



Vertices V and edges E define a graph/skeleton/1-skeleton $G = (V, E)$ of a polyhedron.

Fundamental Theorem of Linear Programming

A linear program min/max $c^T x$ s.t. $x \in P$ has an opt. solution that is a vertex, or it is unbounded.

Geometric interpretation: "Walk as far as possible in direction of c/$-c$"

The famous Simplex Method (Phase II) begins at a vertex of a standard form polyhedron and then computes an optimal solution to the LP through a walk along the edges of the polyhedron.

$\Rightarrow$ interest in edge walks, combinatorial diameter, combinatorial distance

Combinatorial diameter: maximal combinatorial distance between any pair of vertices

Combinatorial distance: minimal distance (number of edges) of any path between a given pair of vertices in the graph of the polyhedron

The bound $f-d$, #facets − dim, on the combinatorial diameter is called the Hirsch Conjecture and took more than 50 years to disprove (for bounded polytopes). (Disproof for unbounded polyhedra within 10 years)

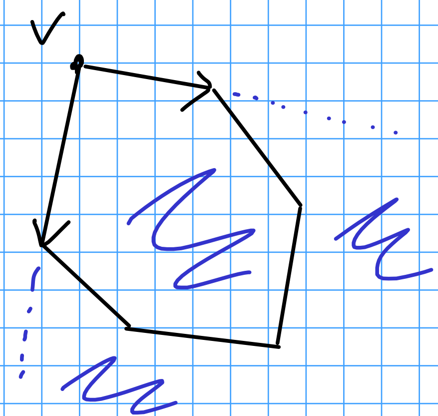
The question whether a bound $p(f-d)$, for some polynomial p, holds is open and called the <u>polynomial Hirsch Conjecture</u>.

The vertices and edges of polyhedra in combinatorial optimization encode a lot of human-interpretable information on the underlying application.

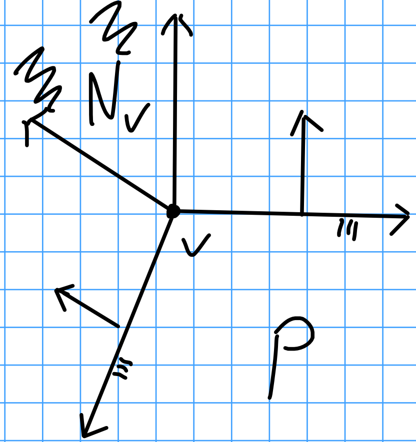
Even for general linear programs, they are valuable:

- conv(V) is an alternative definition of a bounded polytope
- given $v \in V$, $P \subseteq v + I_v$, where I_v is the set of conic combinations of the edges incident to v

I_v is called the <u>inner cone</u> of v



- the normal cone N_v, defined as the polar of I_v, encodes all objective function vectors c for which $c^T v$ would be maximal at v over P



polar of P: $\quad P^o = \{x : x^T y \leq 1 \ \forall y \in P\}$

for cone C: $\quad C^o = \{x : x^T y \leq 0 \ \forall y \in C\}$

Characterization of Vertices

Vertices of $P = \{x : Ax = b, x \geq 0\}$ are formed algebraically as <u>basic feasible solutions</u>.

$A \in \mathbb{R}^{m \times n} \Rightarrow \operatorname{rank} A \leq m \leq n$

for P to be well-defined

Let $A \in \mathbb{R}^{m \times n}$ and $B \subset \{1, \dots, n\}$.
Choose B such that submatrix A_B formed from the columns of indices in B has full rank m. (w.l.o.g. rank $A = m$)

$\Rightarrow$ the columns of A_B are lin. independent for $|B| = m$

$\Rightarrow$ such a choice leads to $A_B \in \mathbb{R}^{m \times m}$ satisfying that

$$A_B x' = b \text{ has a unique solution for all } b$$

$\Rightarrow$ B or A_B are called a basis

N or A_N, denoting all other indices / columns is called nonbasis complement x' with 0's for N to obtain x

$\Rightarrow (A_B \, A_N)\, x = b$. x is called a <u>basic solution</u>

If $x \geq 0 \Rightarrow$ basic feasible solution

If $x \ngeq 0 \Rightarrow$ basic infeasible solution

$A_N \in \mathbb{R}^{m \times (n-m)}$

$A = (A_B \, A_N)$

Vertices correspond to basic feasible solutions (for one or more $A_B \, A_N$ splits of A). If there is more than one to which it belongs, the vertex is <u>degenerate</u>.

Note: B is of minimal size to guarantee solvability of $A_B x' = b$

$\Rightarrow$ A vertex is specified through an inclusion-maximal set N of variables forced to $= 0$.

For $Ax = b$, $x \geq 0$, the constraints $x \geq 0$ give the facets of the polyhedron. (some may be redundant)

Constraints are called <u>active</u> if they are satisfied with equality.
($\Rightarrow$ $Ax = b$, the equality part of the description, is always active)
$\Rightarrow$ A vertex is specified through an inclusion-maximal set of active constraints. This holds for general polyhedra, too.

$\Rightarrow$ Construction of a vertex through an inclusion-maximal nonbasis is a great tool for a characterization in terms of the underlying application.

A second, great tool: Find an objective function c such that the feasible solution x^* under consideration is the <u>unique</u> optimizer of $\max\ c^T x$ s.t. $Ax = b$, $x \geq 0$.

Example Partition Polytopes (fixed cluster sizes)

These arise when partitioning a set of data points $\{1,\dots,n\}$ into k clusters of prescribed sizes $\kappa_1,\dots,\kappa_k \in \mathbb{N}$.

$$\sum_{j=1}^{n} x_{ij} = \kappa_i \qquad \forall i \le k$$

$$\sum_{i=1}^{k} x_{ij} = 1 \qquad \forall j \le n$$

$$x_{ij} \ge 0 \qquad \forall i \le k, j \le n$$

feasible 0, 1 solutions $\hat{=}$ feasible partitions of the data points

Note: These are special transportation polytopes. The constraint matrix is totally-unimodular, a property that gives you integrality of vertices if the right-hand sides are integer.

Characterization of Edges

Recall: For a bounded polytope, the edges are the 1-dim faces and connect two vertices.

Let $B, B' \subseteq \{1, \dots, n\}$ give bases for vertices v and v' connected by edge $e = (v, v')$. Let non-bases N, N' correspond to B, B'.

N, N' were inclusion-maximal for the vertices. $\Rightarrow$ What do we know about B_e / N_e, the basis-nonbasis split for edge e?

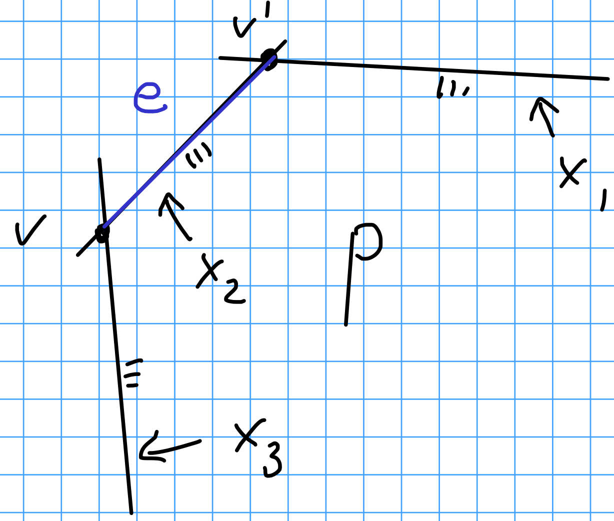
The edge also satisfies $Ax = b$, $x \geq 0$ by $(B_e \; N_e)x = b$, $x \geq 0$ and $x_i = 0$ for all $i \in N_e$.

Example

B_v, N_v-split: $(\{1\}, \{2,3\})$

$B_{v'}, N_{v'}$-split: $(\{3\}, \{1,2\})$

B_e, N_e-split: $(\{1,3\}, \{2\})$



Let v, v' be two vertices and $e = (v, v')$ the edge connecting them.
Let B_v, N_v and $B_{v'}, N_{v'}$ be basis-nonbasis splits for v and v'.

$\Rightarrow$ $N_e \subsetneq N_v, N_{v'}$ actually $N_e = N_v \cap N_{v'}$ is possible

$B_e \supsetneq B_v, B_{v'}$ actually $B_e = B_v \cup B_{v'}$ is possible

For non-degenerate polyhedra, $N_v \setminus N_{v'} = \{i_v\}$ and $N_{v'} \setminus N_v = \{i_{v'}\}$

For general polyhedra, there exist $N_v, N_{v'}$ such that the same holds.

$\Rightarrow$ These observations give a great tool for characterizing the edges: you try to understand what this nonbasis exchange corresponds to.

Note: For a vertex v, $|N_v| = \dim P$ for polyhedron P

For an edge e, $|N_e| = \dim P - 1$

This is a second tool to characterize an edge.

A third tool: Find an objective function such that two given vertices v and v' both are optimal, but no other vertex is.

$\Rightarrow$ v and v' are connected by an edge $e = (v, v')$.

"v and v' are adjacent"

Note: if $c^T v = c^T v'$ then $c^T x = c^T v = c^T v'$ for any $x = v + \lambda (v' - v)$ for any $\lambda \in [0, 1]$ (which is the set of all points on the edge)

$\Rightarrow$ the whole edge is optimal if and only if $c^T v = c^T v'$ is

Example Partition Polytope continued

$$\sum_{j=1}^{n} x_{ij} = \kappa_i \qquad \forall i \le k$$

$$\sum_{i=1}^{k} x_{ij} = 1 \qquad \forall j \le n$$

$$x_{ij} \ge 0 \qquad \forall i \le k, j \le n$$

Recall: Vertices are 0,1-vectors, i.e., feasible clusterings

You can represent an assignment of items to clusters in a bipartite graph.

The difference between two adjacent vertices is a simple cycle in the bipartite graph. We will call these cycles **cyclical exchanges** of items between clusters.

Essentially, edges describe the minimal difference between two vertex solutions. We generalize this concept.

Circuits

$P = \{x : Ax = b, Bx \leq d\}$

Classical concept from the 1970s

- elementary vectors of a subspace (Rockafellar)

  "the minimal difference between two feasible solutions"

  in the sense of an inclusion-minimal set of components in a vector that change

  $u, v \in P \Rightarrow u - v \in \ker A$

- universal test set for LPs (Graver)

  "if you cannot further improve a linear objective along any circuit, you are optimal"

Modern applications

- elementary modes in math biology
  smallest self-contained processes in metabolic networks
- augmentation schemes to solve LPs
- explicit description of the difference of two solutions
- circuit walks and diameters

<u>Definition</u> The <u>set of circuits</u> $\mathcal{C}(A,B)$ of a polyhedron $P = \{x \in \mathbb{R}^n : Ax = b, Bx \leq d\}$ consists of all $g \in \ker A \setminus \{0\}$ normalized to coprime integer components for which Bg is support-minimal over $\{Bx : x \in \ker A \setminus \{0\}\}$.

support of a vector: collection of indices where vector is nonzero

Step by step:

- Right hand sides b, d are <u>not</u> used in the definition of $C(A,B)$

$\Rightarrow$ $C(A,B)$ is a set / property of the whole family of polyhedra

$P(b,d) = \{x : Ax = b, Bx \leq d\}$ where b and d vary

- Circuits are nonzero vectors in $\ker A$

$g \in \ker A \Rightarrow \lambda g \in \ker A$ for any $\lambda \in \mathbb{R}$ (even negative)

$\Rightarrow$ circuits could be arbitrarily scaled

$\Rightarrow$ normalization is used to get a unique representative for all λg, $\lambda \in \mathbb{R}_+$

normalization to coprime integer components $\hat{=}$ two representatives $g, -g$ for $\{\lambda g : \lambda \in \mathbb{R}\}$

- Support-minimality refers to <u>inclusion-minimality</u>

  → there exists no other vector that satisfies all properties and has a support that is strictly included in the support of a circuit

- Support-minimality becomes easier for standard form polyhedra

$$P = \{x \in \mathbb{R}^n : Ax = b,\ x \geq 0\}$$

⇒ $B = -I$ and the property becomes

$g \in \ker A \setminus \{0\}$ with g support-minimal over $\{x : x \in \ker A \setminus \{0\}\}$

⇔ nonzero kernel elements with inclusion-minimal support

→ goes back to Rockafellar's interpretation

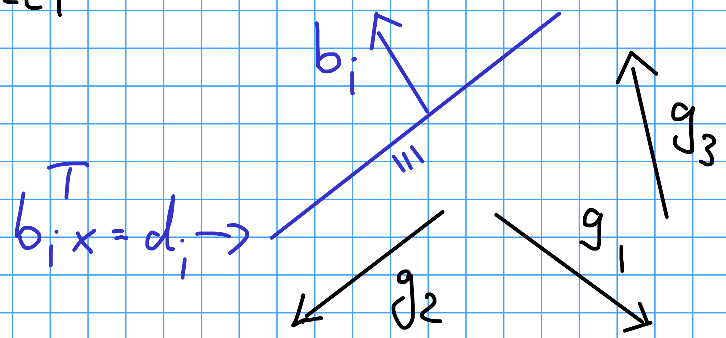
- In the general form, using $\{Bx : x \in \ker A \setminus \{0\}\}$ to check support minimality of a Bg means we are checking/considering the "behavior" of g with respect to the facets described by $Bx \leq d$.

Let $b_i^T x \leq d_i$ be a facet (and part of the system $Bx \leq d$).
Recall: b_i is the outer normal of the facet, d_i is the "position" in the space

$\Rightarrow$ $b_i^T g_1 < 0$: g_1 walks away from facet

$b_i^T g_2 = 0$: g_2 walks parallel to facet

$b_i^T g_3 > 0$: g_3 walks towards facet

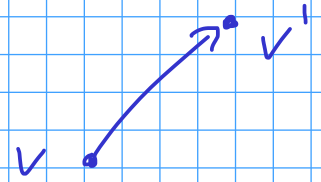


$\Rightarrow$ "circuits are neutral/parallel to as many facets as possible"

Circuits are a direct generalization of edges:

<u>Theorem</u> Let v, v' be adjacent vertices of a polyhedron P. Then $v' - v = \lambda \cdot g$ for some circuit $g \in \mathcal{C}(A, B)$.

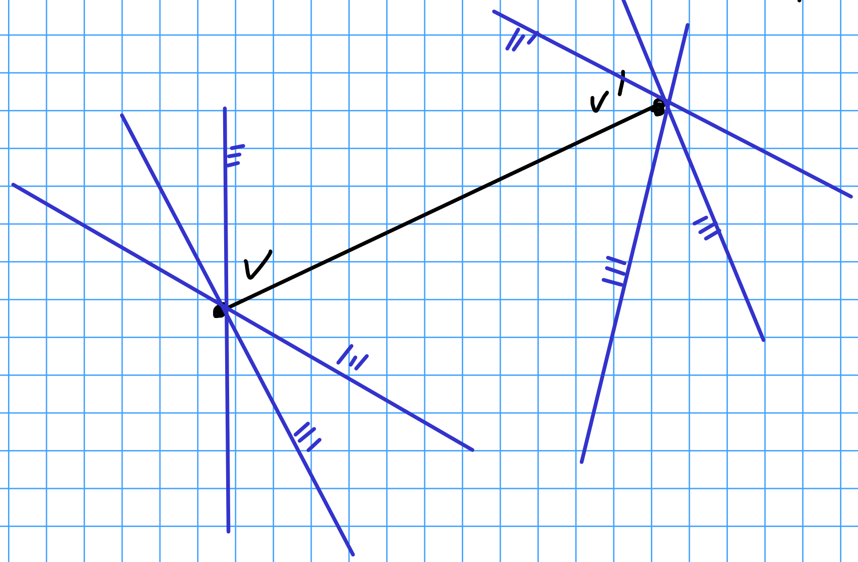
Note: $v'-v$ can be called an "edge direction"



Proof Idea:
- v and v' are feasible $\Rightarrow v'-v \in \ker A$
- scaling/normalization is irrelevant for this statement
- edges are the 1-dim faces formed from inclusion-maximal sets of active facets

<u>Corollary</u> $\mathcal{C}(A, B)$ consists of the edge directions of $P(b, d) = \{x : Ax = b, Bx \leq d\}$ where b and d vary.

Proof Idea: Given $g \in C(A,B)$, create a line segment with endpoints v and v'. Then choose b and d such that all facets run through v and v' and $\{x : x = (1-\lambda)v + \lambda \cdot v',\ \lambda \in [0,1]\}$ is feasible.



Theorem $C(A,B)$ corresponds to the directions of one-dim. subspaces obtained by intersecting any subset of $\dim(P)-1$ facets (or equalities) with lin. independent outer normals placed at the origin.

Proof Idea:

- the statement describes how to construct an edge direction
- subspaces (and not affine subspaces) are used because we are constructing directions
- placing all inequalities and equalities at the origin ($\hat{=}$ choice of right-hand sides 0) allows the intersection of any subsets of them, and emulates working over the family of polyhedra $P(b, d)$ and always choosing b, d appropriately

Comment: This theorem provides an alternative definition of circuits, which leads to a tool to characterize or enumerate them.

Circuits are valuable to understand / characterize when you want to talk about the difference of solutions.

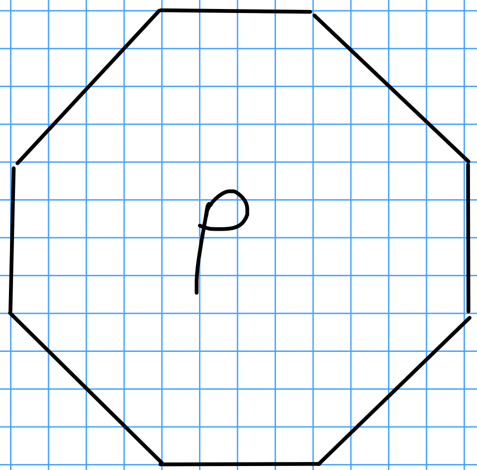
Reason: Walking along a circuit keeps feasibility
and changes an inclusion-minimal support
$\Rightarrow$ best-possible human interpretation

Example

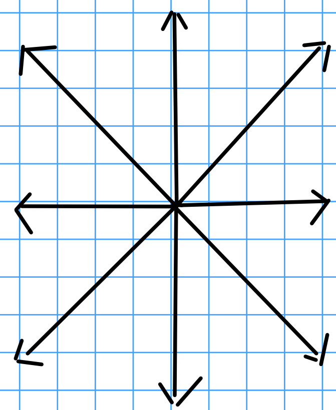
Partition Polytopes continued

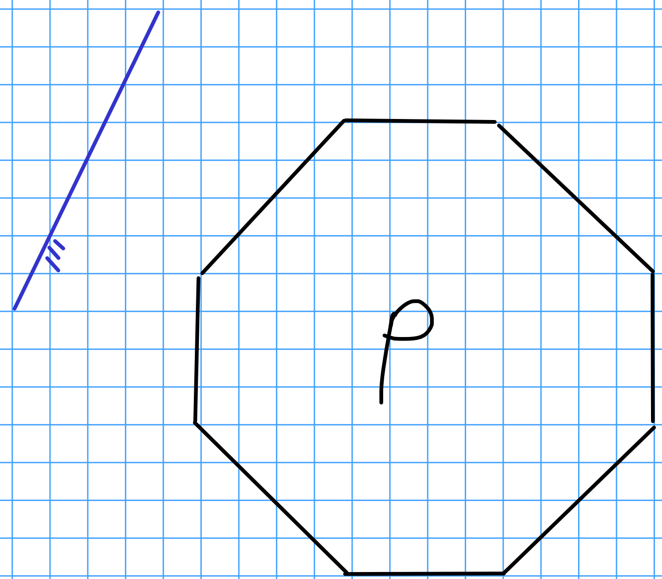
- vertices $\hat{=}$ clusterings
- edges $\hat{=}$ some specific simple cycles in the bipartite graph representation
  "cyclic exchanges"
- circuits $\hat{=}$ all simple cycles / all cyclic exchanges
  (some information irrelevant, like r.h.s κ or 1, or actual clusterings)

Example

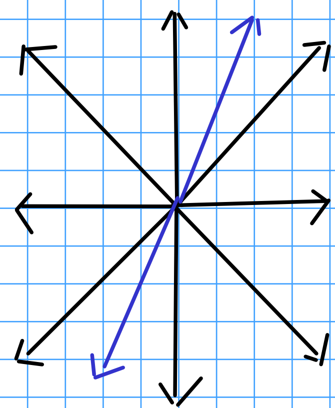


set of circuits





all facets (of this 2-d polytope) were run through the origin



Set of circuits depends on the representation of the underlying P.

Sign-compatible Circuits

Two vectors $x, y \in \mathbb{R}^n$ are said to be <u>sign-compatible</u> if they belong to the same orthant of $\mathbb{R}^n$, that is, if $x_i \cdot y_i \geq 0$ for $i = 1, \dots, n$.

We say x, y are <u>sign-compatible w.r.t. matrix B</u> if the corresponding vectors Bx and By are sign-compatible.

Interpretation for circuits: Sign-compatible circuits "behave the same" with respect to all facets. For any facet, they all walk closer (or closer & parallel) or away from (or away & parallel).

But they never mix walking closer and walking further away!

Partition Polytopes continued

Recall: set of circuits are the simple cycles in a bipartite graph / cyclical exchanges

- What does the difference of two clusterings look like represented in a bipartite graph?
  - drop all edges where the clusterings are identical
  - the remaining edges form a set of simple cycles, alternating between edges of clustering 1 and 2
  - the difference of two clusterings can be decomposed into a set of cycles (each edge only appeared in one cycle)

$\Rightarrow$ the difference of two clusterings decomposes into a sum of sign-compatible circuits

The Conformal Sum Property (Graver)

Proposition Let $P = \{x \in \mathbb{R}^n : Ax = b, Bx \leq d\}$ be a polyhedron with circuits $\mathcal{C}(A,B)$. Any vector $u \in \ker A$ is a sum of sign-compatible circuits. That is, $u = \sum_{i=1}^{t} \lambda_i g_i$ where $g_i \in \mathcal{C}(A,B)$, $\lambda_i > 0$, and g_i is sign-compatible with u w.r.t. B for each $i \leq t$.

$u = \sum_{i=1}^{t} \lambda_i g_i$, where $\lambda > 0$ and circuits $g_1, \dots, g_t \in \mathcal{C}(A,B)$ are sign-compatible with each other and with u is called a **conformal sum.**

Graver's Proposition holds also with the requirement that you have a conformal sum.

We are interested in writing $u \in \ker A$ as a conformal sum, where $u = v - w$, where $v, w \in P$ (i.e., u is a line segment between two feasible solution - often vertices).

$\Rightarrow$ Then $w + \sum_{i \in S} \lambda_i g_i$, where $S \subseteq \{1, \dots, t\}$, is in P.

Any partial sum $x + \sum_{i \in S} \lambda_i g_i$ for any $S \subseteq \{1, \dots, t\}$ lies in P (if $x \in P$ and $x + \sum_{i \in S} \lambda_i g_i \in P$). This is because of sign-compatibility:

If any ordering of the steps would leave the polyhedron, it "overshoots" w.r.t. a facet and a different circuit would have to correct that ($\hat{=}$ end at a point on the correct side of the facet). These two circuits are not sign-compatible (different sign for the violated facet).

Let's assume we have $u, v \in P$ and want to walk from u to v. Further, let $u_i = v_i$ for all $i \in S \subseteq \{1, \dots, n\}$.
Then any conformal sum $v - u = \sum_{j=1}^{t} \lambda_j g_j$ can only have circuits g_j, where $(g_j)_i = 0$ (i-th component) for all $i \in S$.

⇒ This allows us to ignore those components/variables that had the same value in start and end of the sign-compatible walk / conformal sum.

The set of circuits is the unique inclusion-minimal set of directions that has the <u>conformal sum property</u>, i.e., a conformal sum can be constructed for all $v-u \in \ker A$.

It can be shown there exists a conformal sum $v-u = \sum_{i=1}^{t} \lambda_i g_i$, for which $\operatorname{supp}(Bg_i) \not\subseteq \bigcup_{j>i} \operatorname{supp}(Bg_j)$ for each $i \leq t$.

index set of support (i.e., non-zero components)

This implies linear independence of the Bg_i's and thus $t \leq n - \operatorname{rank} A$.

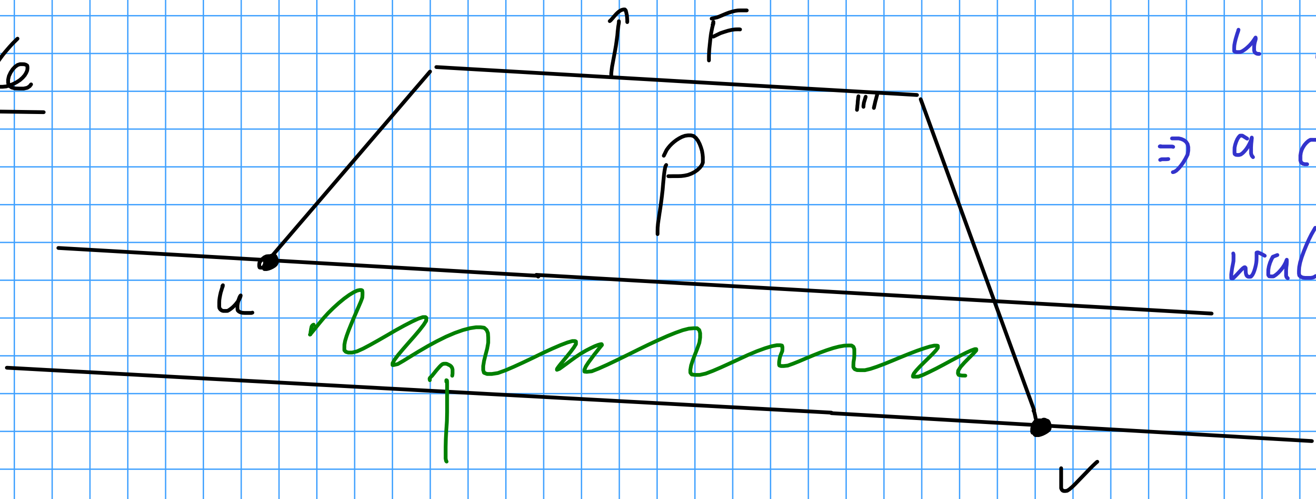
We are going to construct such (special) conformal sums, where we "fill up" a component in each step.

Recall: For two vertices $x, y \in P$ and a conformal sum for $y-x$, you have to walk away from facets incident to x but not y and closer to facets incident to y but not x (and stay in any shared facet). This principle also holds for facets that do not contain x or y (e.g., if $b_i^T x > b_i^T y$ then x was closer to the facet $b_i^T z \leq d$ and you walk away)

$\Rightarrow$ For <u>all</u> facets, you know the sign of the corresponding component of circuits in a conformal sum and you know how "far" you can walk: each facet bounds the length of any circuit step that takes the current feasible solution closer.

However, we can get a tighter bound.

<u>Example</u>



u is closer to F than v
$\Rightarrow$ a conformal sum for $v-u$ walks "away from" F

a conformal sum only walks in this narrow corridor where for $F = \{x : b^T x \leq d\}$, the $u + \sum_{i \in S} \lambda_i g_i$ for $S \subseteq \{1, \dots, t\}$ give $b^T v \leq b^T (u + \sum_{i \in S} \lambda_i g_i) \leq b^T u$

$\Rightarrow$ $b^T u$ and $b^T v$ are lower and bounds of any intermediate point in the conformal sum

Because of sign-compatibility, the values $u + \sum_{i=1}^{t'} \lambda_i g_i$ form a weakly monotone sequence w.r.t. $b^T(u + \sum_{i=1}^{t'} \lambda_i g_i)$ for increasing t' (up to $t' = t$).

Combine this fact with having an index that is "filled up" in each step, we see a greedy principle to construct a conformal sum:

Pick any sign-compatible circuit g w.r.t. B.
Walk as far as the corridor between Bu and Bv allows.
Repeat.

Greedy Algorithm for Conformal Sums / Sign-compatible Walks

Input : $x, y \in P$, $\mathcal{C}(A,B)$

Output : sign-compatible walk from x to y

- $s = x$, $u = y - s$, $T = \emptyset$
- while $(s \neq y)$ do

  Pick any $g \in \mathcal{C}(A,B)$ that is sign-compatible to u w.r.t. B

  Choose $\lambda > 0$ minimal such that $B(s + \lambda g)$ matches By in an additional component

  $s = s + \lambda g$, $u = y - s$, $T = T \cup \{\lambda g\}$

- return T

You can improve this greedy algorithm through optimizing the g picked w.r.t. some objective function.

Popular choice: steepest-descent circuits w.r.t. c

$$\underset{g \in \mathcal{C}(A,B)}{\arg\min} \frac{c^T g}{\|g\|_1}$$

can be computed efficiently

For your projects, you could try to search circuits with more "valuable", more complicated objective functions. → does not have to be efficient

How would you get/find the set of circuits of a polyhedron?

For example through an algorithm that tries all combinations of facets and checks rank.

Naive Circuit Enumeration Algorithm

$A \in \mathbb{R}^{m_A \times n}$, $B \in \mathbb{R}^{m_B \times n}$

Input: Matrices A, B

Output: $\mathcal{C}(A, B)$

$S \leftarrow \emptyset$

For each $I \subseteq \{1, \dots, m_B\}$ where $|I| = n - \operatorname{rank} A - 1$ do

$\quad B_I \leftarrow$ row submatrix of B indexed by I

$\quad$ if $\operatorname{rank}\begin{pmatrix} A \\ B_I \end{pmatrix} = n - 1$ then $\quad$ full rank

$\quad\quad g \leftarrow$ any $x \in \ker\begin{pmatrix} A \\ B_I \end{pmatrix} \setminus \{0\}$ normalized to coprime integers

$\quad\quad$ if $g \notin S$ then

$\quad\quad\quad S \leftarrow S \cup \{g, -g\}$

return S

Theorem Let $P = \{x \in \mathbb{R}^n : Ax = b, Bx \leq d\}$ be a pointed polyhedron.
there exists a vertex
Let $g \in \ker(A) \setminus \{0\}$ be given, and let B' be the maximal row submatrix of B satisfying $B'g = 0$. Then g is a circuit direction of P if and only if $\operatorname{rank}\begin{pmatrix} A \\ B' \end{pmatrix} = n-1$.

Proof: Note first that since P is pointed, $\operatorname{rank}\begin{pmatrix} A \\ B' \end{pmatrix} \leq n-1$ for any $g \in \ker A \setminus \{0\}$.

Suppose that g is a circuit direction of P.
If $\operatorname{rank}\begin{pmatrix} A \\ B' \end{pmatrix} < n-1$, there exist rows of B that can be added to B' in order to form a new row submatrix B'' of B such that $\operatorname{rank}\begin{pmatrix} A \\ B'' \end{pmatrix} = n-1$.

However, then there exists a nonzero $y \in \ker\begin{pmatrix} A \\ B'' \end{pmatrix}$ which must satisfy $\operatorname{supp}(By) \subsetneq \operatorname{supp}(Bg)$, contradicting the fact that g is a circuit direction of P.

Conversely, if $\operatorname{rank}\begin{pmatrix} A \\ B' \end{pmatrix} = n-1$, it holds that $\ker\begin{pmatrix} A \\ B' \end{pmatrix}$ is one-dimensional and generated by g.

Thus, any $y \in \ker(A) \setminus \{0\}$ satisfying $\operatorname{supp}(By) \subseteq \operatorname{supp}(Bg)$ must be a scalar multiple of g, implying that g is a circuit direction of P. $\square$

The set of circuits $\mathcal{C}(A,B)$ of a polyhedron, in general, is exponentially large. (Exponential number of $I \subseteq \{1, \dots, m_B\}$ can lead to circuits.) $\Rightarrow$ Enumeration cannot be efficient in general.

Often, one wants to optimize over the set <u>without</u> enumeration.

Two approaches make / can make this possible:

- a polyhedral model that allows (certain types of) linear optimization over $C(A,B)$
- an explicit characterization of $C(A,B)$

$\rightarrow$ Combinatorial Optimization Algorithm

Polyhedral Model for Circuits

Goal: Write a polyhedron such that the circuits appear as vertices.

Benefit: · Can do linear optimization over $C(A,B)$

· Can use "vertex enumeration" algorithms to find the whole set.

Theorem Let $P = \{x \in \mathbb{R}^n : Ax = b,\ Bx \leq d\}$ be a pointed polyhedron. The pointed cone

$$C_{A,B} = \{(x, y^+, y^-) \in \mathbb{R}^{n+2m_B} : Ax = 0,\ Bx = y^+ - y^-,\ y^+, y^- \geq 0\}$$

is generated by the set of extreme rays $S \cup T'$ where

- The set $S = \{(g, y^+, y^-) : g \in C(A,B),\ y_i^+ = \max\{(Bg)_i, 0\},$
$$y_i^- = \max\{-(Bg)_i, 0\}\}$$
gives the set of circuits of P.

- The set T' is a subset of $T = \{(0, y^+, y^-) : y_i^+ = y_i^- = 1$
$$\text{for some } i \le m_B,\ y_j^+ = y_j^- = 0 \text{ for } j \neq i\}$$
and has size at most m_B.

Proof idea: • y^+, y^- are the positive and negative parts of Bx and bounded below by 0: those domain constraints form the facets of $C_{A,B}$

• for an extreme ray of $C_{A,B}$, an inclusion-maximal set of those variables / domain constraints has to be $= 0$

$\Leftrightarrow$ (y^+, y^-) needs to have inclusion-minimal support

$\Leftrightarrow$ By needs to have inclusion-minimal support (the property in the definition of circuits) $\square$

Note that there are most m_B rays in T' and the are easy to detect / to distinguish from S: the "x-part" is 0 for T', but non-zero for S.

$\Rightarrow$ Ray enumeration would find all circuits.

Since (y^+, y^-) consists of non-negative entries for any $(x, y^+, y^-) \in C_{A,B}$, a normalizing constraint

$$\sum_{i=1}^{m_B} y_i^+ + \sum_{i=1}^{m_B} y_i^- = 1 = \|y^+\|_1 + \|y^-\|_1 = \|(y^+, y^-)\|_1$$

"cuts off" all extreme rays and leaves us with a bounded polytope $P_{A,B}$, where the circuits g are normalized to 1-norm / norm 1 $\frac{g}{\|Bg\|_1}$.

$$\Rightarrow P_{A,B} = \{(x, y^+, y^-) \in \mathbb{R}^{n+2m_B} : Ax = 0,\ Bx = y^+ - y^-,\ \|y^+\|_1 + \|y^-\|_1 = 1,\ y^+, y^- \geq 0\}$$

__Algorithm:__ Circuit Enumeration via Polyhedral Model

Input: matrices A, B

Output: $\mathcal{C}(A, B)$

$S \leftarrow \emptyset$

Use any vertex enumeration algorithm to compute the set of vertices of $P_{A,B}$

for each $(x, y^+, y^-) \in V$ do

if $(x \neq 0)$ then   if $(y^+ - y^- \neq 0)$ then

$g \leftarrow x$ scaled to coprime integers

$S \leftarrow S \cup \{g, -g\}$

return S

Vertex enumeration algorithms :

- Double Description
- Avis - Fukuda

Modeling subsets of circuits

We look at two desirable properties:

- feasible circuit directions at a given $x_0 \in P$
- sign-compatible circuits to a $u \in \ker A$

Feasible circuits

Given $x_0 \in P$, a direction u is said to be feasible at x_0 if $x_0 + \alpha \cdot u \in P$ for some $\alpha > 0$.

Idea: Can use a face of $C_{A,B}$ or $P_{A,B}$ to model all feasible circuits at a given point x_0.

Note: A face of a polyhedron is a polyhedron, a face of a cone is a cone.

$\Rightarrow$ We will be able to enumerate or optimize over feasible circuits only?

What extra constraints do you need to add to a description of $C_{A,B}$ or $P_{A,B}$?

Idea: a circuit direction is feasible if it does not "immediately" violate one of the active facets at x_0 $(Bx_0)_i = d_i$

Theorem Let $P = \{x \in \mathbb{R}^n : Ax = b,\ Bx \leq d\}$ be a pointed polyhedron and let $x_0 \in P$ be given. Consider the face $C_{A,B,x_0,d}$ formed by intersecting the cone

$$C_{A,B} = \{(x, y^+, y^-) \in \mathbb{R}^{n+2m_B} : Ax = 0,\ Bx = y^+ - y^-,\ y^+, y^- \geq 0\}$$

with the hyperplanes $y_i^+ = 0$ for each $i \leq m_B$ such that $(Bx_0)_i = d_i$.

The extreme rays of $C_{A,B,x_0,d}$ (with the exception of at most m_B rays with $x = 0$ or $y^+ - y^- = 0$) give the feasible circuit directions at x_0 in P.

Further, each feasible direction (does not have to be a circuit) u at x_0 has a representation $(u, y^+, y^-) \in C_{A,B,x_0,d}$.

Sign-compatible circuits

Goal: Given a $u \in \ker A$, restrict the set of circuits to only sign-compatible ones to u with respect to B.

What extra constraints do you need to add to a description of $C_{A,B}$ or $P_{A,B}$?

Idea: Describe the "behavior" of u with respect to B and then force the y^+, y^- to exhibit the same behavior.

Theorem Let $P = \{x \in \mathbb{R}^n : Ax = b, Bx \leq d\}$ be a pointed polyhedron and let $u \in \ker A$ be given. Consider the cone $C_{A,B,u}$ formed by intersecting

$$C_{A,B} = \{(x, y^+, y^-) \in \mathbb{R}^{n+2m_B} : Ax = 0,\ Bx = y^+ - y^-,\ y^+, y^- \geq 0\}$$

with the following hyperplanes for each $i \in m_B$:

- $y_i^- = 0$ if $(Bu)_i > 0$
- $y_i^+ = 0$ if $(Bu)_i < 0$

Then $C_{A,B,u}$ is a face of $C_{A,B}$ whose extreme rays correspond to circuits of P' which are sign-compatible with u with respect to B.

You can also set $y_i^- = 0$ if $(Bu)_i \geq 0$ and $y_i^+ = 0$ if $(Bu)_i \leq 0$.

Types of Circuit Walks

In this class, we used two prototypical examples: edge walks and sign-compatible walks.

Common Types:

CD_m • Circuit Walks (without context): maximal step lengths & stay feasible

CD_f • Feasible circuit walks: any step lengths & stay feasible

CD_e • Edge walks: edge directions only & maximal step lengths & stay feasible

CD_s • Sign-compatible walks: sign-compatible directions & stay feasible

The Hirsch Conjecture claimed a combinatorial diameter bound (→ edge walks) of $f-d$ for any polyhedron with f facets in dimension d.

Let CD_s, CD_m, CD_f, CD_e denote the circuit diameters for the above types of circuit walks for a polyhedron/polytope of f facets in dimension d.

circuit diameter (circuit distance)

Then $CD_e \geq CD_m \geq CD_f \leq CD_s$.

The Hirsch Conjecture is wrong for CD_e.
It is open for CD_m.
It is true for CD_f and CD_s.

The question for CD_m is called the Circuit Diameter Conjecture:

Claim: For any d-dim. polyhedron with f facets,

$$CD_m \leq f - d$$

only thing currently known: The Hirsch counterexample for CD_e for unbounded polyhedra ($f=8$, $d=4$) does not give a counterexample for CD_m.

Studies of this conjecture reveal what is the reason for violation of the Hirsch bound (for CD_e):

- If it is true, the problem with CD_e is the use of edges instead of circuits.
- If it is wrong, the problem is the use of maximal step lengths.

It is actually possible that $CD_e > CD_m > CD_f$, so you have a gap between CD_e and CD_m and CD_m and CD_f. Then both the restriction to edges and the restriction to maximal step lengths are problems.