Mathematical Optimization

Chao Ni

January 19, 2020

1 Linear Programming: Geometry part

- General and Canonical form: canonical form is a maximization problem and consists of less equal inequalities and non-negative constraints; converting a genral Lp into canonical form.
  - Standard form: maximation LP with only equality constraints (introduce slack variables) and non-negativity constraints.
- Three types of LPs: 1. with finite optimum (unique solution or infinitely many opt. sol.); 2. unbounded LP (in term of the sol. is unbounded) 3. Infeasible LP;
- For a LP, we expecet it to return: 1. if it is finite, we want it return a corner that is a optimal sol. (or just an opt. sol.); 2. if it is unbounded, provide an improving direction, $y + \lambda v$ is feasible for any nonnegative λ, and $c^T v > 0$. 3. If is is infeasible, provide infeasibility certificate.
- Half-space, hyperplane: $\{x \in \mathcal{R}^n : a^T x \leq \beta\}$,$\{x \in \mathcal{R}^n : a^T x = \beta\}$
- Polyhedron is a finite intersection of half-spaces, a bounded polyhedron is called a polytope.
- Redundance: a linear constaint of a polyhedron is redundant if removing it does not change the polyhedron.
- Dimension of a Polyhedron is the dimension of a smallest-dimensional affine subspace containing P.
  - Think of the affine mapping: it maps from a polyhedron to a polyhedron and is bijective, so some operations can be done in the affine subspace with lower dimension and then mapped it back to the original space;
- Supporting hyperplane: If $H = \{x \in \mathcal{R}^n : a^T x = \beta\}$ satisfies: $P \cap H \neq \emptyset$ and P is contained in one of the two half-spaces defined by H.
- Face: either P itself or the intersection of P with a supporting hyperplane;
  - Following statements are equivalent
    1. F is a face of P
    2. $\exists c \in \mathcal{R}^n$ such that $\delta := \max\{c^T x : x \in P\}$ is finite and $F = \{x \in P : c^T x = \delta\}$
    3. $F = \{x \in P : \bar{A}x = \bar{b}\} \neq \emptyset$ for a subsystem $\bar{A}x \leq \bar{b}$ of $Ax \leq b$.
  - We also need the property that if the optimum δ is finite, then F is non-empty. To show this (problems set), the basic idea is by 1. by induction; 2. In a full-rank case, for any $y \in P$, we can find a $y' \in P$, and it hits one more boundary of P than y. The interesting line direction is to follows the kernel of constraint matrix A. 3. An insight from this is that there are finitely many constraints, so we can achieve the optimum.
  - A face of face of P is itself a face of P.
  - Facet-defining inequality
- vertex: 0-dimensional face of P
  - a vertex is a unique sol. to a subsystem of the constaints. (No other degree of freedom)
  - a vertex cannot be expressed as non trivial convex combinations of other vertexs.
  - a vertex is an extreme point of P.

- If y is a vertex, there needs to be a vector $c \in \mathcal{R}^n$ such that y is the unique maximizer of $\max c^T x : x \in P$.
- Foe every point y in P, there are two kinds of constraints, one is tight, and the other one is not tight. For small deviations from y, i.e. $y + \epsilon w$, the untight constraint will not be violated if devaition is small enough, and if w is a proper direction, say, the kernel of the tight constraint matrix, then the tight constraints will not be violated, either.
- a vertex $y \in vertices(P)$ is called degenerate if the number of y-tight constraints in the linear ineqaulity description is strictly larger than n. The inequality description is called degenerate if there is a degenerate vertex with respect to it.
- a polyhedron is called degenerate if it has a vertex contained on strictly more than dim(P) many facets.

- Edge: an edge of P is a 1-dimensional face of P
  - an edge has one degree of freedom
  - points on a bounded edge can be expressed as convex combination of its two endpoint vertices
  - points on an unbounded edge: direction is of concern, and often combined with a polyhedron cone
- Facet: a facet of P is a (dim(P)-1)-dimensional face of P: (n-1) degree of freedom
  - For any facet of P, there exists at least one inequality that defines F. $F = \{x \in P : a^T x = \beta\}$. The facet has n-1 dimension, hence such methods(interserction with a hyperplane) can be used in induction.
  - Any description inequality that is not facet-defining for P is redundant.
- Extreme point: points of P it it is not the midpoint of two disjoint points of P.
- For full-dimensional polyhedron, it contains interior points and a corresponding ball. If you conv this ball and an outsider point, this convex thing is still full dimensional and can be intesected by a hyperplane, which leads to a n-1 dimensional polyhedron.
- If a polyhedron has number of f facets, then every inquality description of P requires at least f inequalities. Moreover, if P is full-demensional, then an ineqaulity description of P with f inequalities exists.
  - Full dimensional means, there are n+1 affinely independent points in P. Hence n affine independent points in the corresponding facet (and they are all in the same corresponding facet-defining hyperplanes.)
- Dominant: $dom(X) := X + \mathcal{R}^n_{\geq 0} = \{x + y : x \in X, y \in \mathcal{R}^n_{\geq 0}\}$
- Convex combination: two take-away points about the coefficients: 1. nonnegetivity; 2. sum up tp one.
  - convex hull
  - A polytope is the convex hull of its vertices: $P = conv(vertices(P))$
  - Let $X \subset \mathcal{R}^n$ be a finite set, then $conv(X)$ is a polytope.
  - Polyhedral cone: a cone that is a polyhedron. $C = \{x \in \mathcal{R}^n : Ax < 0\}$
- If C is a polyhedral cone, then $C = \{\sum_{i=1}^{k} \lambda_i x_i : \lambda_i \geq 0 \forall i \in [k]\}$, these points are called a set of generators.
  - For a polyhedral cone, we can always bound it with a cube, hence it becomes a polytope and has non-zero right side constraints. The resulting vertices can be used as generators and some of them may useless.
  - One way to consider constructing a polyhedral cone is that, first you have to have zero in the polyhedral, and then if you remove all constraints with non-zero right side, you will get a polyhedral cone.
- Let P be a polyhedron then $P = Q + C$, where Q is a polytope and C is a polyhedral cone.
  - From the left to right: Assume $P = \{x \in \mathcal{R}^n : Ax \leq b\}$, we can construct $Q := conv(vertices(P))$ and $C := \{x \in \mathcal{R}^n : Ax \leq 0\}$.

  - From the right to left, we also need to find the inequality descriptions of P given Q and C. First thing is that we can find a line that slice the cone into a polytope, and make sure the intersection of this line $L = \{v^T x = \beta\}$with Q is non-emnpty. Hence we can find points on this hyperplane that intersecti with $W = \{q + \lambda x_i\} \in L, q \in vertices(Q)$ and x_i is generators from C. For the polytope $P = conv(vertices(Q) \cup W) = \{x \in \mathcal{R}^{n"} Ax \leq b, v^T x \leq \beta\}$, Hence we find the description of the desired polyhedron $\{x \in \mathcal{R}^n : Ax \leq b\}$. And remember to remove the redudant constraints as is clear from the graph.

- An affine image of a polyhedron is a polyhedron. affine function is the sum of a linear function and a translation.

- Dominant of a polyhedron is a polyhedron.

- (Strictly) separating hyperplanes.
  Let $Y, Z \subset \mathcal{R}^n$ be two sets, a hyperplane $H = \{x \in \mathcal{R}^n : a^T x = \beta\}$ is called a (Y, X)-separating hyperplane if Y is contained in one of the half-spaces defined by H and Z in the other one, i.e., either
  $a^T y \leq \beta \leq a^T z, \quad \forall y \in Y, z \in Z$ or
  $a^T y \geq \beta \geq a^T z, \quad \forall y \in Y, z \in Z$

  - Separating a point from a polyhedron.
  - Let $Y, Z \subset \mathcal{R}^n$ be two disjoint closed convex sets with at least one of them being compact(closed and bounded), then there exists a strictly separating hyperplane.
  - If both are just polyhedra, then either there exists a point in the intersection of the two polyhedras, or the two sets admits a strictly separating hyperplane. (using linear programming to check, and there two cases are actually corresponding to infeasible and unbounded (hence admit a sol.) cases of primal and dual).

2 Simplex Method

- Three questions regarding Simplex Method: 1. Given the vertex $y \in vertices(P)$, how to find an improvement edge direction (Phase II); 2. How to find a starting vertex; 3. How to use such a method to show that a linear programming is infeasible (Phase I).

- A tuple (x,y) is a *solution* if it satisfies the equality constraints; a solution is called feasible if it satisfies the non-negativity constaints.

  - Optimal tableau and Optimal basic sol.

- Free variables and dependent variables (non-basic variable and basic variables)

  - Every free variable and dependent variable tuple corresponds to a basic sol.
  - the dimension of the sol. space is determined by the number of free variables, and the dependent variables are in the tabular position the identity matrix corresponds to.
  - Free variables and dependent variables can change with different parameterization.
  - The basic sol. lies on the intersection of m linearly indep. constraints.

- Pivoting is getting from one tableau to another tableau Pivot element, pivot row, pivot column

- Short tableau: makes basis explicit; has some natural dual interpretation; is more compact

  - For short tableau, we have m constraints, and a total number of $m + n$ variables. Notive that every feasible sol. of the tableau is a unique sol. to some square matrix. We can think of it both ways. First, because we have m constraint, if we can eliminate the number of variables to m, then we can get a unique sol. (Hence, setting the n free variables to zero); Second, because the sol. is n dimensional, we need to construct a $n \times n$ matrix to determine the sol., for every free variable that we set to zero, we know it corresponds to a constaint that being set equal, (If $y_i = 0$, then correspondingly, $a_i x = b_i$, if $x_i = 0$, then correspondingly, $y_i = b_i$), this concises with the method that we set free variables to zero.

- After introducing the auxilary variable $z = c^T x + q$, we can then go to the Phase II.

  - auxilary variable will always stay in the basis. Although we now have m+1 equations.

  - Feasible tableau if the right side $b_i \geq 0, i \in [m]$
  - The enqry q is called the value of the tableau.

- To find the pivot element: legal pivot element for Phase II of Simplex Method. (i,k) is legal pivot element if: 1) $c_k < 0$; 2) $i \in argmin\{\frac{b_j}{A_{jk}} : j \in [m], A_{jk} > 0\}$
  - If pivoting on a legal pivot element, then the new tableau is still feasible (check if b is still nen-negative)
  - The value of the new tableau is no less than that of the original one
  - If $b_i > 0$, then the value of the new tableau stictly improves.
  - Every tableau represents a vertex of the polytope, and Simplex Method is moving from current vertex along the edge to another vertex (or unbounded direction). Hence it happens when a feasible tableau becomes a unbounded tableau after one pivoting.
- If $c_k \geq 0 \quad \forall k \in [n]$, then the basic solution corresponding to this tableau is an optimal sol. to the LP;
- If $\exists k \in [n], c_k < 0, A_{jk} \leq 0 \quad \forall j \in [m]$, then the underlying LP is unbounded, and we can find the certificate (a feasible point and an unboundness direction). And this is also true even if there is a legal pivot element in a different column
- In the Phase II of Simplex Method, it still open which row and which column to choose when we have multiple options. And we still need to consider if the choice method will lead to termination of the algorithm.
- The basic sol to a tableau is called *degenerate*, if there is an index $i \in [m]$ such that $b_i = 0$. A feasible tableau is degenerate if and only in its basic sol is a degenerate vertex of the feasible region of the problem (wrt either canonical or standart form)
  - Because in the basic sol. we already have free variables as zero, which means tight constraints, if at least one b_i is zero again, means another tight constraint, thus we have more tight constraints than number of variables(n), which leads to *degenerate*.
  - If no degeneracy case happens in the pivoting process, then every pivot, the value of the tableau strictly improves, and the total number of tableau is finite, so the pivoting will terminate in finite times.
  - If there is degeneracy and some methods is applied in choosing pivoting columns or rows when there are more than one options, there may exist cycles. Here the cycles means that you encounter with same set of basic variables.
- In the cycle, take a look at $z = c^T x + q$, bacause the basic variables don't change after a cycle, hence the value of the tableau never changes, considering that each pivoting improves or equal to the previous one.
  - In the cycle, basic solution never change and is all-zeros vectorsb. Because If in one tableau, the right side has $b_i > 0$, then at least once these row will become the pivot row and the value of the tableau will strictly improve.
  - Hence in the cycle, all right side are zeros (and we assume wlog that the value of the tableau is zero as well)
- Bland's pivot rule: All variables are first ordered in an arbitrary way, whenever a pivot column or row is to be selected and there are several feasible options, we choose the column or row corresponding to the variable appearing first in the fixed order.
  - to prove this, consider the case when the last variable is leaving basis to the non basis and then draw the contradition.
  - Lexicographic pivoting.
- In Phase I of Simplex Method, an auxiliary LP introduced. The initial LP is feasible if and only if the auxiliary LP has optimal value 0.

  - Actually, we are converting a Phase I problem into a Phase II problem, because the right side of the tableau is negative for at least one member, otherwise the original LP admits an easy feasible solution already. Choose x_0 as the variable entering the basis, and choose a row with most negative b-value as pivot row. Then the tableau immediately becomes feasible. After that, using Phase II method to do pivoting. If the optimal value of this tableau is zero, then the basic sol. of this tableau is exactly a feasible tableau of the original LP (remove x_0 column,)
  - If the opt. sol. of the tableau has x_0 in the basis (Normally, x_0 should in the non-basis and has coefficient 1 only in the corresponding auxilary objective row), do another pivoting to let x_0 leave basis again.
  - One can simply add an additional row during phase I of the simplex Method that keeps track of original objective.
  - If the auxiliary Lp has strictly negative value, then we can obtain a certificate of infeasibility from the corresponding optimal simplex tableau.

- Linear Duality: Introduce non-negative variables representing how we add up the constraints, and duality is basically to find an optimal upper bound (and can be guarenteed to be achieved by strong duality). In the same time, we should also guarentee that the sum of corresponding initial primal variables in the dual is larger equal to the coefficient in the primal objective function.
  - Primal: $\max c^T x, s.t. Ax \leq b, x \geq 0$, Dual: $\min b^T y, s.t. A^T y \geq c, y \geq 0$
  - Primal: $\min c^T x, s.t. Ax = b, x \geq 0$, Dual: $\min b^T y, s.t. A^T y \leq c$
  - Dual of Dual is Primal.

- Weak duality: $c^T x \leq b^T y$, both are just feasible sol.
  - If the primal is ubnounded, then dual is infeasible
  - if the dual is unbounded, then primal is infeasible
  - If one finds a primal feasible sol. x and a dual feasible sol. y with $c^T x = b^T y$, then x, y is optimal for primal and dual, respectively.

- Strong duality: if the primal has a finite optimum, then the dual also has a finite optimum and their values are equal, i.e., there exists a feasible sol. x for primal and a feasible sol. y for dual such that $c^T x = b^T y$

- Complementary slackness theorem: Consider a pair of primal and dual linear programs with finite optima (in canonical form), Let $\hat{x}, \hat{y}$ be feasible primal and dual sols. Then they are both optimal for the primal and dual. respectively, if and only if $(b - A\hat{x})^T y = 0$, and $(A^T \hat{y} - c)^T \hat{x} = 0$.

3 Computational Complexity

- The running time of an algorithm is the number of elementary operations that it performs, including plus, minus, mutplication, devide, comparisons and assignment(with integer number). For example, the inner product of two vectros require $2n - 1$ elementary operations.

- Landau notation

- Worst-case assumption: when analysing the running time of an algorithm for a certain problem class, we want to get a running time upper bound that holds for any problem instance of a certain size.

- Input size: number of bits needed to save problem instances.
  - Saving interger numbers in binary: $\lceil \log_2(a+1) \rceil + 1$

- An algorithm is polynomial or efficient if its running time $f(< input >)$ is bounded by a polynomial in the size of the input, i.e., there is a polynomial g such that $f = O(g)$. A problem is solvable in polynomial time if it can be solved by a polynomial algorithm.

- $\mathcal{P}$: Class of all decision problems that can be solved in polynominal time

- $\mathcal{NP}$: Class of all decision problems for which a yes-instance can be certified efficiently, which means that there is a polynoial-size certificate and an efficient algrithm such that, given the instance and the certificate, the algorithm can verify that it is a yes-instance.

4 Basics on Graphs

- undirected graph, directed graph, simple graph (with no loops or parallel edges)
- $\delta^+(v)$: arcs leaving v (directed graph)
- walk, path (a walk that contains no vertex more than once, cycle
- subpraphs, induced subgraph;
- cut is a vertex subset; s-t cut: contains s and not contain t.
- Data structure: adjacency matrix vs. incidence list
- Breadth-first search: for every graph $G = (V, E)$ and every vertex $v_1 \in V$, BFS has a running time bounded by $O(m+n)$. BFS can return the distance from v_1 to any other verteces. And it can natually generate the shortest distance from s to t with same computational complexity.
- Connectivity in undirected graphs, directed graphs, strong connected normally means that each vertex can be reached from every other vertex, and this is a term for directed graph only;
- connected components, strong connected components (for directed graphs, considering the arc direction, every vertex can be reached from any other vertex)

5 Flows and Cuts

- an s-t flow defined in a Graph is a function $f : A \to \mathcal{R}_{\geq 0}$ satisfying two conditions: 1. Capacity constraints; 2. Balance constraints. The value of a flow is $v(f) := f(\delta^+(s)) - f(\delta^-(s))$. And we assume the capacities are integer. (thus avoiding the Zeno's paradox).
- an s-t cut is a set $c \subset V$, such that $s \in C$ and $t \notin C$. The value of an s-t cut is defined as $u(\delta^+(C))$.
- Value of a flow can be expressed via an s-t cut: $v(f) = f(\delta^+(C)) - f(\delta^-(C))$.
- The Weak max-flow min-cut theorem is then from above: $v(f) \leq u(\delta^+(C))$, which means the value of a maximum s-t flow is upper bounded by the value of a minimum s-t cut.
- Ford-Fulkerson Algorithm
  - f-residual graph and f-residual capacities: f-residual graph contains all initial arcs and the arcs that is antiparallel to the original arcs.
  - f-augmenting path: is an s-t path in the residual graph with positive capacities along the path. augmented with the augmentation volume of the aumentation path $\gamma := \min\{u_f(b) : b \in P\}$
  - If there exists a augmenting path, then we can use BFS to find it with O(m+n); Because each augmentation step has at least one unit improvement, we can thus bound the running time of FF algorithm in $O(\alpha(m+n))$. However, it is not polynomial.
  - For a maximum s-t flow f, there does not exist an f-augmenting path in G_f, which means if we define the set C as all vertex s can be reached from in the residual graph, there is no arcs with positive capacity leaving C, otherwise you can expand the set C to contain that vertex, and that means in the original graph, all arcs leaving C is f-saturated and all arcs entering C is zero flow. Hence this set C satisfies that: $u(\delta^+(C)) = v(f)$.
- Polynomial-time variations and extentions of FF
  - Capacity scaling algorithm: main modification to FF is that in the augmentation step, only pay attention to the augmenting path with at least Δ capacity in the residual graph. Starting from a big $\Delta = 2^{\lfloor \log_2(U) \rfloor}$, explore all paths in the $\Delta - phase$ and the half Δ. The complexity needed is $O(m^2 \log(U))$.
    An interesting observation is $u_f(\delta^+_{G_f}(C)) = u(\delta^+(C)) - f(\delta^+(C)) + f(\delta^-(C)) = u(\delta^+(C)) - v(f)$

  - Edmonds-Karp algorithm: augmentation is always on shortest paths in term of number of arcs. An observation is that distances from s and distances to t become larger in residual graphs after a shortest augmenting path in the residual graph when only consider strictly positive residual capacity. Computing the shortest path from s to t by BFS is natually already done.
    Define k-phase as augmenting on the path with length k, Hence O(n) phases; for each phase, at most we have m augmentations (because we have m arcs can get saturated and each augmentation will saturate at least one arc per augmentation, also this is because in each phase, for every arc, augmentations either never use the arc or never use its reverse arc.), Hence, the complexity is $O(nm^2)$ bacause each augmentation needs O(m) by BFS.

- Strong max-flow min-cut theorem: $\max\{v(f)$:f is s-t flow in G$\} = \min\{u(\delta^+(C)) : c \in V, s \in C, t \notin C\}$

6 Applications of s-t flow

- arc-connectivity
  - A directed graph is k-arc-connected if for any two vertices s and t, there are at least k arc-disjoint s-t paths in G.
  - In determining the maximum number of arc-disjoint s-t paths, we can set all arcs with unit vapacity, and then find the maximum s-t flow.
  - Also notice that there is still difference between the k-arc-connected graph where all pair should be explored and the arc-disjoint path for given two vertices.
  - To prove the correctness of the method, we show first that given k arc-disjoint paths, we can then build a flow that is with value k; then we show that for a flow f with value $v(f)$, we can find a path and then set the flow on that path to zero, hence sequentially find $v(f)$ arc-disjoint s-t paths in G.
  - Menger's Theorem: the maximum number of arc-disjoint s-t paths is equal to the number of arcs in a minimum cardinality s-t cut, i.e. $\min\{|\delta^+(C)| : C \subseteq V, s \in C, t \notin C\}$.
    Edge version: the size of the minimum edge cut for s and t(the minimum number of edges whose removal disconnects s and t)(the degree of the cut) is equal to the maximum number of pairwise edge-independent paths from s to t.
  - Reducing the problem of checking k-edge-connectivity in undirected graphs to the problem of checking k-arc-connectivity in directed graphs: using gadget.
- Maximum cardinality bipartite matching
  - Bipartite (undirected) graph: if there is a bipartition of its vertices $V = X \cup Y$ such that each edge has one endpoint in X and the other one in Y. ALSO, bipartite graph is a graph that contains no odd cycles.
  - Matching: Let $G = (V, E)$ be an undirected graph, a set $M \subset E$ is a matching if M doesnot contain loops and no two edges of M share a common endpoint. Hence one interesing question is to determin the maximum cardinality matching in an undirected bipartite graph.
  - Hall's Theorem: Let $G = (V, E)$ be a bipartite graph with bipartition $V = X \cup Y$. Then there exists a matching $M \subset E$ in G that touches all vertices in X iff $|N(X_0)| \geq |X_0|, \forall X_0 \subset X$.
  - The value of a maximum s-t flow in $\mathcal{D}$ equals the vardinality of a maximum mathching; the value of a minimum s-t cut in $\mathcal{D}$ equals the cardinality of a minimum vertex cover, where $\mathcal{D}$ is an auxilary graph wich virtual verteces s and t, and the capacity 1 and ∞.
  - To show two values equal, we can show that every flow correspinds to a match with same value, while every match corresponds to a flow with same value.
- Multiple sources and sinks
- Roster planning: several workers, every worker can operate on some machines, some machine is suitable for some projects, aim to maximize the number of projects
- Optimal project selection: there are some projects, among whom some are profitable and some are generating a loss, the arc represents the precedence relations. Ask how to choose projects to maximize the total profit.
- Open pit mining

- Image segmentation: two points should be considered, one is that the set chosen should not deviate from the manual one too much; the second one is to encourage separating points whose color difference is huge by setting huge constraints.
- Theoretical winning possibilities in sports competitions. If one team A cannot become sole leader anymore, consider a subset of all remaining teams, games between them generate fixed number of points and the points have to be absorbed by these teams, and the average point is larger than the maximum point the team A can achieve.

7 Combinatorial Optimization

- Combinatorial optimization problems can often by described by :1. A finite set $\mathcal{N}$, called ground set; 2. A family $\mathcal{F} \subset 2^N$ of feasible sets, also called solutions; and 3. an objection function $w : \mathcal{N} \to \mathcal{R}$ to maximize or minimize. Hence $\max / \min w(F) := \sum\limits_{e \in F} w(e)$, $F \in \mathcal{F}$. Here such notation is very common in later notes. An exmaple: $\mathcal{N} = \{e_1, e_2\}$, then $\mathcal{F} = \{\emptyset, \{e_1\}, \{e_2\}, \{e_1, e_2\}\}$, if we take $F = \{e_1, e_2\}$, then $w(F) = w(e_1) + w(e_2)$.
- Define characteristic vectors/incident vectors: $\mathcal{X}^U$, where $U \subseteq \mathcal{N}$, $\mathcal{X}^U(e) = 1_{e \in U}$ and this incident vectos has dim of $|\mathcal{F}|$. The combinatorial polytope that corresponds to the feasible sol. set $\mathcal{F}$ is the polytope $P_{\mathcal{F}} \subseteq [0,1]^N$ whose vertices are precisely $\{\mathcal{X}^F : F \in \mathcal{F}\}$, i.e. $P_{\mathcal{F}} = conv(\{\mathcal{X}^F : F \in \mathcal{F}\})$. Therefore, combinatorial optimization problem is cast into a linear program (Because optimal sol. to LP is always at the vertex, and the vertex corresponds to an feasible sol. to the original combinatorial optimization porblem. In the sametime, the number of facets of $P_{\mathcal{F}}$ is always much smaller than the number of vertices, which encourages us to find inequality description of this polytope).
- To find the inequality description (maybe exponential large, but we can still find a way to handle it later by laminar family), we follow the recipe: 1. determine the candidate description $P = \{x \in \mathcal{R}^N : Ax \leq b\} \subseteq [0,1]^N$; 2. Prove that this polytope contains correct integral points, i.e., $P \cap \{0,1\}^N = \{\mathcal{X}^F : F \in \mathcal{F}\}$; 3. Show that P don't contain vertices that are not integral, i.e P is integral. We can derive from above three points that $P = P_{\mathcal{F}}$ because $P = conv(vertices(P))$.
- To prove the integrality of polytope, one way is proving TU-ness of the constraint matrix A.: a matrix is totally unimodular if the determinant of any square submatrix of it is either 0, 1, or -1.
  - If $A \in \mathcal{R}^{m \times n}$ is TU, then so is A^T, $[A \quad -A]$, $[A \quad I]$;
  - Characterization of Ghouila-Houri: a matrix $A \in \mathcal{R}^{m \times n}$ is TU iff for every subset of the rows $R \subseteq [m]$, there is a partition $R = R_1 \cup R_2$ such that: $\sum\limits_{i \in R_1} A_{ij} - \sum\limits_{i \in R_2} A_{ij} \in \{-1, 0, 1\}$, $\forall j \in [n]$.
  - Consecutive-ones matrices are TU.
- Vertex cover polytope for bipartite graph: $P = \{x \in [0,1]^V, x(u) + x(v) \geq 1, \forall \{u,v\} \in E\}$;
- Bipartite matching polytope: $P_{\mathcal{M}} = \{x \in \mathcal{R}^E_{\geq 0} : x(\delta(v)) \leq 1 \quad \forall v \in V\}$;
  - To show integral vertex sol. of the polytope, one way is by TU-ness, but we can also show by disproving existence of fractional exterme points, i.e. by showing that a fractional vertex sol. $y = \frac{1}{2}(y^\epsilon + y^{-\epsilon})$, a general try would be $y^\epsilon = y + \epsilon(\mathcal{X}^A - \mathcal{X}^B)$; hence by modifying each element of y that is not integral to another fractional one and is also in the polytope.
  - Perfect bipartite matching polytope: $P = \{x \in \mathcal{R}^E_{\geq 0} : x(\delta(v)) = 1 \quad \forall v \in V\}$;
  - A graph is called d-regular for some $d \in \mathcal{Z}_{\geq 1}$ if every vertex has degree d. Theorem goes: every d-regular bipartite graph admits a perfect matching. (by showing non-empty of the perfect match polytope)
- Polyhedral description of shortest s-t paths: Notice that the polyhedral for all s-t path is difficult and in the shortest path problems, we typically have non-negative or even positive arc length, consider:

$$P = \left\{ x \in [0,1]^A \middle| x(\delta^+(v)) - x(\delta^-(v)) = \begin{cases} 1 & \text{if } v = s, \\ -1 & \text{if } v = t, \\ 0 & \text{otherwise} \end{cases} \right\}$$

However, the integral points in P do not correspond one-to-one to s-t paths. Each s-t path is an integral point in P, however, integral points in P corresponds to disjoint union of an s-t path and possibly some additional cycles. If we optimize on positive weights $w : A \to \mathcal{Z}_{>0}$, the basic sol. to $\min\{w^T x : x \in P\}$ will correspond to a shortest s-t path. The positive constraints can be relaxed to cases when all cycles must have strictly negative weights.

  – The vertex-arc incidence matrix $D \in \{-1, 0, 1\}^{V \times A}$ of any directed graph $G = (V, A)$ is TU.

- Spanning tree: a spanning tree in $G = (V, E)$ is an edge set $T \subseteq E$ that connects all vertices and does not contain a cycle. Notice that this polytope can have exponential many constraints and all constraints may be facet-defining. Consider also the spanning tree constraints and non-negativity constraints.

$$P = \left\{ x \in \mathcal{R}^E_{\geq 0} \middle| \begin{array}{l} x(E) = |V| - 1 \\ x(E(S)) \leq |S| - 1, \forall S \subset V, |S| \geq 2 \end{array} \right\}$$

  – Combinatorial uncrossing is a technique to extract out a well-structed set system out of a large family of sets. The goal is given a heavily overdetermined linear system that uniquely determines a point, find a well-structured full-rank susbsystem.
  – One trick often used in proof is to delete the edges with $y(e) = 0$ (or assume positivity of weights) once a vertex y is introduced, because such an edge will not destory the constraints.
  – Interesting lemma: $\mathcal{X}^{E[A]} + \mathcal{X}^{E[B]} + \mathcal{X}^{E(A \backslash B, B \backslash A)} = \mathcal{X}^{E[A \cup B]} + \mathcal{X}^{E[A \cap B]}$, for any sets $A, B \subseteq V$;
  – TU-NESS about laminar family: $x(S)$, $x(E[S])$, $x(\delta^-(S))$, $x(\delta^+(S)$, $S \in \mathcal{H}$ and the latter two are directed graphs, the corresponding system matrix are all TU. Proof can be shown by numbering the positive-negative sign of the laminar element.
  – Basic idea to show the integerity of spanning tree polytope:
    a) to handle exponential many constraints, extract out a maximum laminar family to uniquely define the vertex sol. y for two inherent reasons: 1: Laminar family induced system is TU; 2. We have proved in the problem sets that the maximum cardinality of a laminar family $\mathcal{L}$ that contains no non-trivial partition of some set $L \in \mathcal{L}$ is n, hence the system is actually square. Because if we have a set L such that $L = L_1 \cup L_2$, then the equality description on L_1 and L_2 is enough to derive the equality description for L, hence redundant system.
    b) to show that the larminar family induced system is enough to represent the whole equality system. Every eqaulity in the original can be implied by the laminar system, to show that, use contradiction and define $\mathcal{H}_S := \{H \in \mathcal{H} : \text{H and S are intersecting}\}$.

- The r-arborescence polytope in $G = (V, A)$ is an arc set $T \subseteq A$ such that: 1. T is a spanning tree when disregarding the arc direction, 2. there is one vertex r such that, every vertex can be reached from r using a directed path in T. The vertex r is called the root, and T is called an r-arborescence. 2': for every vertex, there is at most one incoming arc (zero for root)

$$P = \left\{ x \in \mathcal{R}^A_{\geq 0} \middle| \begin{array}{l} x(A) = |V| - 1 \\ x(A(S)) \leq |S| - 1, \forall S \subset V, |S| \geq 2 \\ x(\delta^-(v)) \leq 1, \forall v \in V \end{array} \right\}$$

Minimizing over P is equivalent to minimizing over dom(P) if the weights are positive. The dominant of P is:

$$P = \left\{ x \in \mathcal{R}^A_{\geq 0} : x(\delta^-(S)) \geq 1, \forall S \subseteq V \backslash \{r\} \right\}$$

  – To prove the integrity of dominant of r-arborescence polytope, the process is almost the same as before in the spanning tree case. The eqaulity trick here is:
    $\mathcal{X}^{\delta^-(S_1)} + \mathcal{X}^{\delta^-(S_2)} = \mathcal{X}^{\delta^-(S_1 \cap S_2)} + \mathcal{X}^{\delta^-(S_1 \cup S_2)} + \mathcal{X}^{A(S_1 \backslash S_2, S_2 \backslash S_1)} + \mathcal{X}^{A(S_2 \backslash S_1, S_1 \backslash S_2)}$

- Non-bipartite matching: The perfect matching polytope of an undirected graph $G = (V, E)$ is given by:

$$P = \left\{ x \in \mathcal{R}^E_{\geq 0} \middle| \begin{array}{ll} x(\delta(v)) = 1 & \forall v \in V \\ x(\delta(S)) \geq 1 & \forall S \subseteq V, |S| \text{ odd} \end{array} \right\}$$

Consider that, if y is a vertex, then it is the unique sol. of a square submatrix of the original system, and we can divide the constraints into two kinds. (degree constraints and cut constraints, the non-negativity constraints can be deleded by smartly choose the contradiction example (smallest $|V| + |E|$), etc.)

  – The matching polytope of an undirected graph $G = (V, E)$ is given by:

$$P = \left\{ x \in \mathcal{R}^E_{\geq 0} \middle| \begin{array}{ll} x(\delta(v)) \leq 1 & \forall v \in V \\ x(E[S]) \leq \frac{|S|-1}{2} & \forall S \subseteq V, |S| \text{ odd} \end{array} \right\}$$

    Prove this by constructing a perfect matching graph. Some interesting eqautions:

  - $|S| \geq \sum\limits_{v \in S} x(\delta(v)) \geq 2x(E[S])$ with the assumption $x(\delta(v)) \leq 1$;
  - $x(\delta(A)) + x(\delta(B)) = x(\delta(A \backslash B)) + x(\delta(B \backslash A)) + 2x(E(A \cap B, V \backslash (A \cup B)))$
  - Another way to showing that the vertex of the polytope is integral (and hence the characteristic vectors of the corresponding matches) is to show every $x \in P$ satisfies: $x = \sum\limits_i \mathcal{X}^{M_i}$, where M_i is the matching.

- Minimaly k-edge-connected graph G: 1) $|\delta(S)| \geq k$, $\forall S \subseteq V, S \neq \emptyset$, and 2) $\forall e \in E, \exists S \subseteq V$ s.t. $e \in \delta(S)$ and $|\delta(S)| = k$; The theorem goes: if $G = (V, E)$ be a minimally k-edge-connected graph, then $|E| \leq (|V| - 1)$.
  - Show by constructing a certifying family, and this certifying family is kind of like a laminar family after uncrossing operation and delete all non-trivial partition cases. We call $\mathcal{H} \subseteq 2^V$ a certifying family if: a) $|\delta(H)| = k$, $\forall H \in \mathcal{H}$; b) $\bigcup\limits_{H \in \mathcal{H}} \delta(H) = E$
  - An interesting lemma: Let $S_1, S_2 \subseteq V$ be two crossing sets that are minimum cuts, then $S_1 \cup S_2$ and $S_1 \cap S_2$ are also minimum cuts and $E(S_1 \backslash S_2, S_2 \backslash S_1) = \emptyset$;
  - we need to show $|\mathcal{H}| \leq |V| - 1$, and then $|E| = |\bigcup\limits_{H \in \mathcal{H}} \delta(H)| \leq \sum\limits_{H \in \mathcal{H}} |\delta(H)| = k \cdot |\mathcal{H}| = k(|V| - 1)$

8 Ellipsoid Method

- General idea about Elliposoid Method: it is used for solving optimization problem with exponential many constraints, which Simplex Method cannot handle. The draft goes like this: for the Lp problem $\max\{w^T x : x \in P\}$, we first need to find a point $y \in P$, and for the $\{0, 1\}$ polytope, we then can cast the optimization problem into figuring out if some polytope is empty, which can be shown by the fact that we can/cannot find a point x in the corresponding auxiliary polytope. In this part, we will strict the analysis in the full dimensional polytope. An intuition to the non-full dimentioanl polytope is that the we can see it as a very very flat ellipsoid in iteration process.

- Seperation oracle for the polyhedron over which to optimize: Given a point $y \in \mathcal{R}^n$, we should decide if $y \in P$, and if not, find a vector c such that $P \subseteq \{c^T x < x^T y, x \in \mathcal{R}^n\}$, notice that the ineqaulity is strict.
  - An easy way to design the oracle is by looking at the constraints, see if one constraint is violated, then a hyperplane can be decided easily.

- We use ellipsoid method to return a final ellipsoid such that is contains the polytope, and its origin is in the polytope. Such process of finding iterative ellipsoids has a bounded number it iteration $l \leq 2(n + 1) \ln \dfrac{vol(E_0)}{vol(P)}$; by the way, the mathematical forula of an ellisoid is given by:

$$E(a, A) = \{x \in \mathcal{R}^n : (x - a)A^{-1}(x - a)^T \leq 1\}$$

  With some properties:
  - A positive symmetric definite: $A = QQ^T$, with $Q \in \mathcal{R}^{n \times n}$ full-rank;
  - Ellipsoid is the image(output of a function) of the unit ball under an affine bijection, and the bijection is define by $y = Qx + a$; the vol relation is $|\det(Q)|$ times.
  - In proving the upper bound the iteration numbers, show that the ratio of consecutive minimum ellipsoid is the same as that of a minimum vol ellipsoid with respect to the unit ball. $\dfrac{vol(E_{i+1})}{vol(E_i)} = \dfrac{vol(E_B)}{vol(E(0, I))}$, and we show that the minimum ellisoid containing the union of half plane and the unit ball can be given by:

$$E_B = \big\{x \in \mathcal{R}^n \big| (\frac{n+1}{n})^2 (x_1 - \frac{1}{n+1})^2 + \frac{n^2 - 1}{n^2} \sum_{j=2} x_j^2 \leq 1\big\}$$

    Hence, we can show that $\frac{vol(E_{i+1})}{vol(E_i)} < e^{-\frac{1}{2(n+1)}}$

- In order to make the ellipsoid method explicit, for the general case, we do the following, first do an inverse affine transformation and map the E_i to a unit ball as well as the hyperplane $\{c^T x \leq 0\}$, for a unit ball, the minimum ellipsoid can be easily obtained though the hyperplane is not trivial here. $a_{i+1} = -\frac{1}{n+1}c$, $A_{i+1} = \frac{n^2}{n^2-1}(I - \frac{2}{n+1}cc^T)$, the do the affine transformation, and get it back to the next ellipsoid.

- Auxilary optimization problem: $P(\gamma) = P \cup \{x \in \mathcal{R}^n : w^T x \geq \gamma - \frac{1}{2}\}$, we can check the emptyness of this polytope to find that if the maximum $\gamma \in [-nw_{max}, nw_{max}]$ can be achieved by binary search. By bounding the minimum vol of $P(\gamma)$ and properly choose a starting ellipsoid, we can bound the iteration of Ellpsoid Method at $O(n^2 \cdot (\log n + \log w_{max}))$, which is polynomial in input size $O(\log n + \log w_{max})$.

- From Ellipsoid method, we already get a point x which is close to the vertex sol. One way to continue to find the vertex is by local search, we can also use another tricky way to define another auxiliary linear programming with slight modification in weight function. Let $S \subseteq [n]$, $w_i^S = \begin{cases} w_i + 1 & \text{if } i \in S \\ w_i & \text{otherwise} \end{cases}$, then we have following equivalent statements:
a) There is an optimal sol x^* to $\max\{w^T x : x \in P\}$ with $x_i^* = 1$ for $i \in S$.
b) $\gamma^* + |S| = \max\{(w^S)^T x : x \in P\}$
Hence, by iteratively choose S, we can obtain the exact sol. cordinate by cordinate. (Notice that γ is still has to be chosen first, which is the largest possible integer such that P_γ is not empty).

9 Equivalence between optimization and separation

- Recall that when we use Ellipsoid Method to solve for the optimization problem, we are actually using given separation oracles. We iteratively using separation oracles to find a point in the polytope and solve for the optimization problem. Inversely, if we want to solve for the separation problem given optimization oracles, which, for any $c \in \mathcal{R}^n$, returns an optimal sol. to $\max\{c^T x : x \in P\}$. For the separation problem, given a y and a polytope, we then want to find a hyperplane that separetes y and that polytope. We then can do is by first compute P^o, and then compute a maximizer $z \in \arg\max\{y^T x : x \in P\}$ using the optimizatrion oracle for P^o, if $y^T z \leq 1$, then $y \in P$, otherwise we have a separation hyperplane $x \in \mathcal{R}^n : z^T x \leq 1\}$

- $X^o = \{y \in \mathcal{R}^n : x^T y \leq 1, \forall x \in X\}$;

- Let $P \subseteq \mathcal{R}^n$ be a polytope containing the origin in its interior. Then P^o is a polytope. Moreover, we have following statements equavilent:
a) x is a vertex of P
b) $\{y \in \mathcal{R}^n : x^T y \leq 1\}$ is facet-defining for P^o

10 Integer Programming

- Branch-Bound; Branch-cut: the basic idea is to reduce branching numbers.