1 Discrete optimization examples

1.1 Mixed integer linear program

Formulation: $\min_{x,y} c^T x + d^T y$ such that $Ax + By = b, x \geq 0, x \in \mathbb{Z}^n, y \geq 0$.
Often, we pick $x \in \{0,1\}^n$.

1.2 Knapsack

Formulation: $\max_x \sum_{i=1}^n c_i x_i$, such that $\sum_{i=1}^n w_i x_i \leq B, x_i \in \{0,1\}$

1.3 Packing, covering and partitioning

Let $E = \{1, \ldots, n\}$ be an index set, and let $F = \{F_1, \ldots, F_m\}$ be a collection of subsets of E. We define an incidence matrix $A \in \mathbb{R}^{m \times n}$ by $A_{ij} = 1$ if $j \in F_i$ and 0 otherwise. We let $x \in \mathbb{R}^m$ denote the binary decision vector, where x_i is the indicator variable of choosing F_i.
Formulation: The packing problem is defined as $\max_x c^T x$ such that $A^T x \leq \mathbf{1}$. The program is to find a collection of disjoint subsets to maximize total value.
Formulation: The covering problem is defined as $\min_x c^T x$ such that $A^T x \geq \mathbf{1}$. The program is to find a collection of subsets to cover all indices with minimal cost.
Formulation: The partitioning problem is defined as $\max_x c^T x$ such that $A^T x = \mathbf{1}$.

1.4 Disjunction

We may want to satisfy either one of two constraints.
Formulation: $a^T x \geq yb, (a')^T x \geq (1-y)b', y \in \{0,1\}, x \geq 0$. If $y = 0$, then $(a')^T x \geq b$ is satisfied and $a^T x \geq 0$ is trivial, and vice versa if $y = 1$.
More generally, we want at least k inequality constraints to be satisfied.
Formulation: $a_i^T x \geq b_i y_i, \sum_i y_i \geq k, y_i \in \{0,1\}, x \geq 0$

1.5 Forcing

We may want that a variable taking a certain value forces another variable to take a certain value.
Consider a case of facility location. Suppose there are n services locations, and m customers who require a service from any one of these. For a location j to provide the service, it must be switched on at a cost of c_j. If a customer i gets service from location j, cost d_{ij} will be incurred. We model a program to provide service for all customers and minimize total cost incurred. Let x_{ij} denote the decision variable for customer i to get service from location j, and y_j denote the decision variable to switch on location j.
Formulation: $\min_{x,y} \sum_{j=1}^n c_j y_j + \sum_{j=1} n \sum_{i=1}^m d_{ij} x_{ij}$ such that $\sum_{j=1} x_{ij} = 1 \forall i, x_{ij} \leq y_j \forall i, j$. First constraint ensures that any customer get service from exactly one location. Second constraint ensures that if any customer gets service from a location, that location must be switched on.

1.6 Routing

Given an undirected graph $G(V, E)$ over n nodes. A tour is an ordering of nodes $\{1, \ldots, n\}$ such that we visit all nodes in this order and complete a cycle. Suppose the cost of traversing edge $e \in E$ is c_e. For each $e \in E$, let x_e be the decision variable to traverse the edge. Given a subset $S \subseteq V$, we let $\delta(S) \subseteq E$ be the subset of edges traversing a node in S to another node in S^c. The travelling salesman problem is to find such a route with minimal cost.
Formulation: $\min_{x_e} \sum_{e \in E} c_e x_e$ such that $\sum_{e \in \delta(\{i\})} x_e = 2 \ \forall i \in V, \sum_{e \in \delta(S) x_e} \geq 2 \forall \varnothing \subset S \subset V$. The first set of constraint esures that we visit and exit each node exactly once. The second set of constraint ensures for every proper subset of nodes, we enter and exit this set at least once, so that tours formed by separate sub-tours do not occur.

1.7 Scheduling tasks

We operate a machine that can operate m types of tools. The machine can only load B tools simultaneously. There is a list of jobs to be performed. Each job requires a subset of tools to be loaded. The subset $J_i \subseteq \{1, \ldots, m\}$ denote the tools required. For tool i, we incur a cost of s_j to load or remove the tool. We let $x_{i,r}$ be the decision variable to execute job i at time r. We let $y_{j,r}$ be the indicator variable to indicate tool j is active on the machine at time r. We try to complete all jobs with minimal cost of loading and removing incurred. The total cost of loading and removing is $\sum_{r,j} s_j |y_{j,r} - y_{j,r-1}|$, we introduce slack variables to remove modulus.
Formulation: $\min_{x,y,z} \sum_{j,r} s_j z_{j,r}$, such that $z_{j,r} \geq y_{j,r} - y_{j,r-1}, z_{j,r} \geq y_{j,r-1} - y_{j,r}, \sum_r x_{i,r} = 1 \forall i, \sum_i x_{i,r} = 1 \forall r, x_{i,r} \leq y_{i,r} \forall j \in J_i, \sum_j y_{i,r} \leq B \forall r$. The first and second set of constraints is for slack variables. The third set of constraints ensures all jobs are performed. The fourth set of constraints ensures that exactly one job is performed at each time. The fifth set of constraints ensures that for all job to be performed, the tools required are loaded. The sixth set of constraints ensures that at any time, the machine has at most B tools loaded.

2 The geometry of LP

2.1 Standard form

$\min_x c^T x$ such that $Ax = b, x \geq 0$. To convert any LP to standrd form, we could swap max to min by $-c$, add slack variables to inequality constraints to make equality, and write $x = x^+ - x^-$ for any unconstrained variable.

2.2 Polyhedra

Definition: In standard form, $\{x : Ax = b, x \geq 0\}$. If a polyhedra is bounded, we call it a polytope.
Definition: Let $S \subset \mathbb{R}^n$ be a set. We say that $y \in S$ is an extreme point of S if $y = \theta y_1 + (1-\theta) y_2$ for $\theta \in (0,1)$ and $y_1, y_2 \in S \implies y = y_1 = y_2$.
Definition: Let $P \subset \mathbb{R}^n$ be a polyhedron. We say that $y \in P$ is a vertex of P if there is a direction $c \in \mathbb{R}^n$ such that $c^T y < c^T z$ for all $z \in P - \{y\}$
Definition: Suppose $P \subset \mathbb{R}^n$ is a polyhedron. Assume $P \neq \varnothing$, assume A has full row rank. We call any collection of m independent columns of A a basis. WLOG up to permutation, we assume last m columns of A are linearly independent and write the $m \times m$ submatrix corresponding to the columns as B, we have $A = [*|B]$. We let $x = (0 \ B^{-1}b)^T$, then $Ax = b$. We call x a basic solution. If x is further feasible, then it is a basic feasible solution. Intuitively, BFS is a feasible solution that contains as many 0s as possible.

Theorem 1. Suppose $P = \{x : Ax = b, x \geq 0\}$ is a non-empty polyhedron, and let $x \in P$. The following are equivalent: x is a vertex; x is an extreme point; x is a BFS. In other words, there exists a set S of m column indices such that $x_S = B^{-1}b, x_{S^c} = 0$.

3 Unimodularity and total unimodularity

3.1 Relaxation of MILP

We relax an integer program into a linear program by removing the integer constraints. The feasible region of the relaxed linear program contains the feasible region of the integer program, thus the optimal value of the linear program is at most equal to the optimal value of the integer program. Hence, if an optimal solution of the linear program is also integral, as it is a feasible solution of the integer program, it must be an optimal solution of the integer program. In this case, the relaxation is tight.

3.2 Integral polyhedra

Definition: A polyhedron $P \subset \mathbb{R}^n$ is integral if all extreme points are integral.

3.3 Unimodularity

Definition: We say that a square matrix $A \in \mathbb{Z}^{m \times m}$ is unimodular if $\det A = \pm 1$. We say that $A \in \mathbb{Z}^{m \times n}$ with full row rank is unimodular if all $m \times m$ submatrices are either singular or unimodular.

Theorem 2. Let $A \in \mathbb{Z}^{m \times n}$ be a matrix with full row-rank. A is unimodular $\iff$ the set $P(b) = \{x : Ax = b, x \geq 0\}$ is integral for all $b \in \mathbb{Z}^m$ for which $P(b) \neq \varnothing$.

Theorem 3. A non-empty bounded polyhedron has at least one extreme point.

Theorem 4. Let P be a non-empty polyhedron with at least one extreme point. Consider the LP $\min_{x \in P} c^T x$, then the optimal solution is either $-\infty$ or attained at an extreme point.

Theorem 5. Any non-empty polyhedron of the form $\{x : Ax = b, x \geq 0\}$ has at least one BFS.

Theorem 6 (Cramer's rule)**.** Suppose $A \in \mathbb{R}^{n \times n}$ is invertible, then the solution to the linear system $Ax = b$ is given by $x_i = \frac{\det A_i}{\det A}$, where A_i is the matrix attained by replacing column i of A by b.

3.4 Total unimodularity

Definition: We say that an integral matrix $A \in \mathbb{Z}^{m \times n}$ is totally unimodular if the determinant of each square sub-matrix of A is in $\{-1, 0, 1\}$.
This implies that a TU matrix must have all entries in $\{-1, 0, 1\}$. TU implies U but converse is not true.

Theorem 7. Let $A \in \mathbb{Z}^{m \times n}$ be a matrix. A is TU $\iff$ the set $P(b) = \{x : Ax \leq b, x \geq 0\}$ is integral for all $b \in \mathbb{Z}^m$ for which $P(b) \neq \varnothing$.

Theorem 8. Suppose $A \in \mathbb{Z}^{m \times n}$, then A is TU $\iff$ $[A, I]$ is U, where I is the $m \times m$ identity matrix.

Theorem 9. The point x is an extreme point of the polyhedron $P(b) = \{x : Ax \leq b, x \geq 0\}$ $\iff$ $(x, y) = (x, b - Ax)$ is an extreme point of the polyhedron $Q(b) = \{(x, y) : Ax + y = b, x \geq 0, y \geq 0\}$.

Theorem 10. Suppose $A \in \mathbb{Z}^{m \times n}$, then A is TU $\iff$ A^T is TU.

Theorem 11. Suppose $A \in \mathbb{Z}^{m \times n}$, then we have: A is TU $\iff$ $[A, I]$ is TU. A is TU $\iff$ $[A, A]$ is TU. A is TU $\iff$ $[A, -A]$ is TU. A is TU $\iff$ $[A, -A, I]$ is TU. A is TU $\iff$ $[A, -A, I, -I]$ is TU.

3.5 Sufficient conditions to prove TU

Theorem 12. A matrix $A \in \{-1, 0, 1\}^{m \times n}$ is TU if both of the following conditions hold: Each column of A contains at most two non-zero entries. It is possible to split the row indices $\{1, \ldots, m\}$ into two disjoint sets I_1, I_2 such that whenever a column j has two non-zero entries, it holds that $\sum_{i \in I_i} A_{ij} = \sum_{i \in I_2} A_{ij}$. In other words, if the two non-zeros have the same sign then one lies in I_1 and one lies in I_2, where as if the two non-zeros have different signs they both lie in I_1 or I_2.

Theorem 13. A matrix $A \in \{-1, 0, 1\}^{m \times n}$ is TU if each of its columns has at most one $+1$ entry and at most one -1 entry.

3.6 Equitable coloring

Definition: We say that an integer-valued matrix A admits an equitable column bicoloring if it is possible to partition the column indices J into two sets J_a, J_b so that the different between the sums of the columns in these subsets is a vector with entries in $\{-1, 0, 1\}$: $\sum_{i \in J_a} A_i - \sum_{j \in J_b} A_j \in \{-1, 0, 1\}^m$, where A_i is column i of A. Equivalently, there exists some $z \in -1, 1^n$ such that each entry of Az has absolute value at most one.

Theorem 14. Let $A \in \mathbb{Z}^{m \times n}$, and for any $J \subseteq \{1, 2, \ldots, n\}$, let A_J denote the $m \times |J|$ submatrix obtained by keeping only the columns in J. A is TU $\iff$ A_J admits an equitable column bicoloring for all non-empty J.

Theorem 15. A is TU $\iff$ every sub-matrix obtained by taking a non-empty subset of rows of A admits an equitable row bicoloring.

3.7 Bipartite graphs

Given an undirected graph, we denote the node-edge incidence matrix A with entries $A_{ij} = 1$ if node i is in edge j and 0 otherwise.

Theorem 16. The node-edge incidence matrix of an undirected bipartite graph is TU.

Proof follows directly from theorem 12.
Definition: Given a bipartite graph $G = (V, E)$, a matching is a subset of non-intersecting edges(edges for which no two of them have a common node). A matching is said to be perfect if all nodes are selected. The maximum cardinality bipartite matching problem seeks the largest matching in a bipartite graph.
Formulation: $\max_{e \in E} x_e$ such that $\sum_{e \in \delta(\{v\})} x_e \leq 1 \forall v \in V$. The constraint ensures that no two edges can meet at any node. We can use the node-edge incidence matrix A to rewrite the problem as $\max_{e \in E} x_e$ such that $Ax \leq 1, x \in \{0,1\}^n$.
By theorem 16, A is TU, and thus by theorem 7, the feasible region of the linear relaxation is integral, and thus we can solve IP by solving LP directly.
Same conclusion applies to maximum weight bipartite matching, where weight c_e is assigned to each edge and we find a matching with maximum weight, or minimum.
By replacing $\sum_{e \in \delta(\{v\})} x_e \leq 1 \forall v \in V$ with $\sum_{e \in \delta(\{v\})} x_e = 1 \forall v \in V$, we could ensure that we are searching only perfect matching.

3.8 Directed graphs

Given a directed graph, the node-incidence matrix is a matrix with entries $A_{ij} = 1$ if edge j starts from node i, and -1 if edge j ends at node i, and 0 otherwise.

Theorem 17. The node-edge incidence matrix of a directed graph is TU.

Proof follows directly from theorem 13.

Given a subset $S \subseteq V$, we let $\delta^+(S)$ denote the set of edges starting from a node in S and ending at a node in S^c. We let $\delta^-(S)$ denote the set of edges starting from a node in S^c and ending at a node in S.

Let l_e be the length of an edge. We could define shortest path problem from node s to node t.

Formulation: $\min_{x_e} l_e x_e$ such that $\sum_{e\in\delta^-(v)} x_e - \sum_{e\in\delta^+(v)} x_e = 0 \forall v \in V - \{s,t\}, \sum_{e\in\delta^-(s)} x_e - \sum_{e\in\delta^+(s)} x_e = -1, \sum_{e\in\delta^-(t)} x_e - \sum_{e\in\delta^+(t)} x_e = 1$, The first set of constraint ensures that for any node, we either enter and leave or never visit. The second constraint ensures that an edge leaves s. The third constraints ensures that an edge enters t.

The constraints can be summarized by $Ax \leq b$, where A is the incidence matrix. b is $+1$ in the entry corresponding to s, -1 corresponding to t, and 0 in the remaining entries. Since A is TU, the linear relaxation is tight.

4 Approximation

(Minimization ILP) Suppose x^* is the optimal solution to ILP, and $\tilde{x}$ is a feasible solution to ILP, then the solution is an α-approximation if $c^T\tilde{x} \leq \alpha c^T x^*$.

Definition: The integrality gap is the ratio between OPT(ILP) and OPT(LP), IG=OPT(ILP)/OPT(LP).

IG is at least 1 since LP should have an optimal not worse than ILP.

Definition: The rounding ratio is the ratio between the objective of the integer solution $\tilde{x}$ obtained by rounding from LP and that of the optimal LP solution. RR=$c^T\tilde{x}$/OPT(LP). Again RR is at least 1.

If integer solution $\tilde{x}$ has a rounding ratio α, then $c^T\tilde{x} \leq \alpha$OPT(LP)$\leq \alpha$OPT(ILP), which is immediately an α-approximation to ILP.

4.1 Rounding approaches

Deterministic rounding: Round by threshold such as $\frac{1}{2}$, or always round up.

Randomized rounding: Round up x_i with probability of x_i for $x_i \in [0,1]$, or round up x_i if $x_i > \lambda$, where λ is sampled from uniform distribution on $[0,1]$.

5 Submodularity

We now interpret binary optimization problem as optimizing $f(S)$ over $S \subseteq V = \{1,\ldots,n\}$ by interpreting S as the set of binary variables chosen to be 1. We consider maximization.

For constrained maximization over a constrain set $\mathcal{S}$, we focus on $\mathcal{S} = \{S : |S| \leq k\}$.

Definition: A set function f is said to be monotone if for all $S \subseteq T \subseteq V$, it holds that $f(S) \leq f(T)$.

Definition: A real-valued set function f is said to be submodular if for all $S \subseteq T \subseteq V$ and $e \in V - T$, we have $\Delta(e|S) \geq \Delta(e|T)$, where $\Delta(e|S) = f(\{e\} \cup S) - f(S)$(Diminishing return), modular if always holds that $\Delta(e|S) = \Delta(e|T)$, supermodular if always holds that $\Delta(e|S) \leq \Delta(e|T)$

Theorem 18. For any function $f : 2^V \to \mathbb{R}$, the following are equivalent: (1)$\Delta(e|S) \geq \Delta(e|T)$ for all $S \subseteq T$ and all $e \in V - T$. (2)$f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$ for all S, T. (3)$\Delta(e|S) \geq \Delta(e|S \cup \{e'\})$ for all S, e, e'. (4)If f is monotone, then $f(T) \leq f(S) + \sum_{e\in T-S} \Delta(e|S)$ for all $S \subseteq T$.

Theorem 19. Let f_1, f_2 be submodular functions, then: (1)for positive c_1, c_2, $f(S) = c_1 f_1(S) + c_2 f_2(S)$ is submodular. (2)If $g : 2^V \to \mathbb{R}$ is modular and $h : \mathbb{R} \to \mathbb{R}$ is concave, then $f(S) = h(g(S))$ is submodular. An important special case is $g(S) = |S|$. (3)$f(S) = f_1(S \cup B) - f_1(B)$ is submodular for any B. The subtraction term is optional. (4)$f(S) = f_1(S \cap A)$ is submodular for any A. (5)$f(S) = f_1(V - S)$ is submodular.(6)If f_1 is monotone, then $f(S) = \min\{c, f_1(S)\}$ is submodular for any $c \in \mathbb{R}$. (7) If $f_1 - f_2$ or $f_2 - f_1$ is monotone, then $f(S) = \min\{f_1(S), f_2(S)\}$ is submodular.

5.1 Submodular optimization

If $\mathcal{S}$ contains all 2^n possible choices of S, we say that the problem is unconstrained. For a maximization problem over a monotone function with cardinality constrain set $\mathcal{S}$, we have a greedy algorithm with guaranteed $1 - \frac{1}{e}$ approximation to the optimal.

The greedy algorithm: (1) Initialize $S_0 = \varnothing$ (2)For $i = 1,\ldots,k$, find $e_i = \arg\max_{e\in V - S_{i-1}} \Delta(e|S_{i-1})$, set $S_i = S_{i-1} \cup \{e_i\}$ (3)Return S_k.

Theorem 20. For any monotone submodular function with $f(\varnothing) = 0$, after k iterations it holds that $f(S_k) \geq (1 - \frac{1}{e}) f(S_k^*)$. The bound is tight.

Theorem 21. For any monotone submodular function with $f(\varnothing) = 0$, after l iterations it holds that $f(S_l) \geq (1 - e^{-\frac{l}{k}}) f(S_k^*)$.

6 Matroids

Definition: Let N be a finite set, and let $\mathcal{I}$ be a collection of subsets of N. We say that the tuple $(N, \mathcal{I})$ is an independence system if $\varnothing \in \mathcal{I}$, and $A \in \mathcal{I} \implies B \in \mathcal{I}$ for all $B \subset A$.

Let $(N, \mathcal{I})$ be an independence system, we say that $S \subset N$ is dependent if it is not independent.i.e.$S \notin \mathcal{I}$. We say that an independent set $S \in \mathcal{I}$ is maximal with respect to T if $S \cup \{i\}$ is dependent for all $i \in T - S$. We call maximally independent subsets of T bases of T, and we call a single maximally independent set a basis. The rank function $r(\cdot)$ of a subset T is the cardinality of the largest sized basis of T.

Definition: An independence system $(N, \mathcal{I})$ is a matroid if for any two independence sets $A \in \mathcal{I}, B \in \mathcal{I}$, if B contains more elements than A, then there exists $x \in B - A$ such that $A \cup \{x\} \in \mathcal{I}$.

Definition: An independence system $(N, \mathcal{I})$ is a matroid if for all subsets $F \subseteq N$, every maximal independent set in F has cardinality $r(F)$.

Theorem 22. Let $(N, \mathcal{I})$ be an independence system, then $(N, \mathcal{I})$ is a matroid $\iff$ $r(\cdot)$ is submodular.

7 Exact solution by greedy

We consider an optimization problem: Given a matroid, find the independent size with the highest weight. We formulate $\max_{\mathcal{S}\in\mathcal{I}} \sum_{i\in S} c_i \iff \max_{x\in\{0,1\}^n : S_x \in \mathcal{I}} c^T x$. It is indeed equivalent to ILP $\max_{x_j} \sum_{j\in N} \sum c_j x_j$, such that $\sum_{j\in T} x_j \leq r(T)$ for all subsets $T \subseteq N$, and $x_j \in \{0,1\}$. r is rank function for the matroid $(\mathcal{N}, \mathcal{I})$.

Linear relaxation: $x_j \geq 0$, $f : 2^N \to \mathbb{R}$ is non-negative, non-decreasing, submodular and $f(\varnothing) = 0$.

We consider a greedy algorithm: First relabel the set such that all weights are sorted in decreasing order: $c_1 \geq c_2 \geq \cdots \geq c_k \geq 0 \geq \cdots \geq c_n$. Define the set $S^j = \{1,\ldots,j\}$ and $S^0 = \varnothing$. We do the following: For i from 1 to n, we increase x_i as much as possible without violating constraint $\sum_{k=1}^i x_k \leq f(\{1,\ldots,i\})$, which is just $x_i = f(S^i) - f(S^{i-1})$ if $1 \leq i \leq k$, and 0 otherwise. The dual problem is $\min_{\{y_T\}_{T\subseteq N}} f(T) y_T$ such that $\sum_{T : j\in T} y_T \geq c_j$ for all $j \in N$ and $y_T \geq 0$ for all $T \subseteq N$. We claim the dual optimal is $y_S = c_j - c_{j+1}$ if $S = S^j, 1 \leq j \leq k$ and c_k if $S = S^k$, and 0 otherwise. The proof for optimality is to show primal and dual feasible, and argue the primal and dual objective are equal by the two solutions constructed.

Discrete optimization: We come back to ILP, where $f : 2^N \to \mathbb{Z}$ and $x_j \in \mathbb{Z}$. Since f is integral, the values of $f(S^j) - f(S^{j-1})$ is also integral, hence the solution constructed by greedy solution of LP relaxation is integral and feasible for ILP, but it is optimal for LP, thus optimal for ILP.

Greedy algorithm for matroid: We now consider $x_j \in \{0,1\}$ and $f = r$. The greedy algorithm is still to sort the weights first and add x_i as long as $\mathcal{J}$ is still an independent set.(feasible)

8 Complexity

Definition: Let f and g be real functions. We say that $f(n) = O(g(n))$ if there exists c, N such that $f(n) \leq cg(n)$ when $n \geq N$. $f(n) = \Omega(g(n))$ if there exists c, N such that $f(n) \geq cg(n)$ when $n \geq N$. $f(n) = \Theta(g(n))$ if previous two holds.

Definition: A decision problem is one that has a binary answer. In other words, the problem is either a YES instance or a NO instance.

Definition: We say that a decision problem is in $\mathcal{P}$ if it is solvable by an algorithm whose runtime is polynomial with respect to number of bits used to specify the problem input.

Definition: We say that a decision problem is in $\mathcal{NP}$ if there exists a certifying procedure such that the following two conditions are true: (1) Any YES instance of the problem has an associated certificate whose size is polynomial with respect to original input. (2) Given the original input and the certificate, the certifying procedure is able to confirm with certainty that the answer is YES in polynomial time.

Definition: We say that ρ_1 reduces to ρ_2 if it is possible to solve ρ_1 by solving at most a polynomial number of instances of ρ_2, plus polynomial-time additional computation.

Definition: We say that a problem ρ_0 is NP-hard if all problems in $\mathcal{NP}$ reduces to ρ_0. Furthermore, ρ_0 is NP-complete if it is in $\mathcal{NP}$.

If ρ_1 reduces to ρ_2, then ρ_2 is at least as hard as ρ_1.

Theorem 23. If ρ_0 is NP-complete and we can reduce it to $\rho_1 \in \mathcal{NP}$, then ρ_1 is NP-complete.

9 Global optimization

9.1 Branch and bound

Consider a minimization problem $\min_x c^T x$ such that $x \in \mathcal{F}$. We partition $\mathcal{F}$ into a finite collection of subsets $\{\mathcal{F}_1,\ldots,\mathcal{F}_k\}$ We solve each sub-problems $\min_x c^T x$ such that $x \in \mathcal{F}_i$. Hopefully with smaller domain the problem is easier, and we pick the smallest from all optimal answers. It is possible that we may continue to break down $\mathcal{F}$ even further. To prevent exponential growth, we could efficiently maintain lower bounds and upper bounds to the global optimal.

Pseudo-algorithm: (1) Select a sub-problem $\mathcal{F}_i$ that is active. (2) If $\mathcal{F}_i = \varnothing$, eliminate it. Otherwise, compute $l(\mathcal{F}_\rangle)$, the LP relaxation. (3) If $l(\mathcal{F}_\rangle) > U$, then eliminate $\mathcal{F}_i$. If we are able to solve ILP for $\mathcal{F}_i$, we do so and update U, and eliminate $\mathcal{F}_i$. Otherwise, we may break $\mathcal{F}_i$ into sub-problems and add them into list of sub-problems.

Some possible ways to split: Once we solve LP for some $\mathcal{F}_i$, if the optimal x' has non-integral entry x'_i, we create two sub-problems: (1) Same as $\mathcal{F}_i$ with additional constraint $x_i \geq \lceil x'_i \rceil$ (2) Same as $\mathcal{F}_i$ with additional constraint $x_i \leq \lfloor x'_i \rfloor$

9.2 Cutting plane method

Consider the LP relaxation of the general MILP. Suppose solving the LP we obtain a solution $(\tilde{x}, \tilde{y})$, but $\tilde{x}$ is not integral. We seek an inequality of the form $\alpha^T x + \gamma^T y \leq \beta$ as an additional constraint to LP, such that all feasible solutions of MILP are still feasible but $(\tilde{x}, \tilde{y})$ is no longer feasible.