CS880: Approximation and Online Algorithms	**Scribe:** Jennifer Cao, Dongqiangzi Ye
Lecture 25: Approximations for 2-Stage Stochastic Optimization	**Date:** 11/18/2019

25.1 Two-Stage Resource Model

25.1.1 Model Formulation

Stochastic Optimization aims at capturing the uncertainty of the input data in decision making process. A widely used sketch is the *two-stage resource model[1]*:

Given a *distribution* D over some of the realizations of future data, where each realization is called a *scenario*.

- First Stage: With D, we can initial a decision and construct an anticipatory part of the solution x, incurring a cost of $c(x)$.
- Second Stage: Sampling a scenario A from D, we may augment the decision x that made at stage one by taking resource actions y_A, incurring a cost of $f_A(x, y_A)$.

The goal is to choose the initial decisions so as to minimize the expected total cost, i.e.,

$$\min\{c(x) + \mathbf{E}_A[f_A(x, y_A)]\} \tag{25.1.1}$$

Typically, there's a trade off between two stages: the first stage does not have precise information while encounters a lower cost; but the second stage take advantage of actual data, which leads to a higher cost but can make decision based on precise information.

25.1.2 Curse of Dimensionality

One core problem is how to represent scenario-distribution. A straightforward approach is to consider it as non-stochastic input of scenarios. If we explicitly enumerate each scenario with its probability of occurrence, distribution D may contain up to exponential size of scenarios, which leads to a very high problem complexity.

To overcome the "curse of dimensionality", we can restrict the distribution with a polynomial support, which is known as *polynomial-scenario model*. This model assumes scenario distribution is a product of independent (input) distributions, and ignore the correlation between input data. On the other hand, a *black-box model* is described to capture more precise input information: it assumes distribution D is supported by other polynomial-size of scenarios("black boxes"), and we can sample scenarios from it. Thus, black-box model can capture correlated data as well as be reduced to a polynomial-scenario model.

Actually, a black-box model can be reduced to a polynomial-scenario model by ρ-approximation algorithms, thus can be solved in polynomial time.

25.2 Stochastic Set Cover Problem

We will consider stochastic set cover problem as a illustrative problem to explain 2-stage stochastic optimization.

Definition 25.2.1 (Stochastic Set Cover Problem(SSC)) *Given a sets $S_1, S_2, \cdots, S_m$ over a ground set U of n elements. For each set S, it can either be picked at stage I with cost w_S^I, or be picked at stage II with cost w_S^A, where subset $A \subseteq U$ is drawn according to a specified distribution. We need to make sure that A is contained in the union of both sets selected in two stages. The goal is to minimize the expected cost of the sets picked.*

25.2.1 LP Relaxation

The stochastic set cover problem can be formulated as an integer program with the following LP relaxation:

$$\begin{aligned} \min \quad & \sum_S w_S^I x_S + \sum_{A \subseteq U, S} p_A w_S^A r_{A,S} \\ \text{s.t.} \quad & \sum_{S: e \in S} (x_S + r_{A,S}) \geq 1, \quad \forall A, e \in A \\ & x_S, r_{A,S} \geq 0, \quad \forall A, S \end{aligned} \tag{SSC-P1}$$

Where variable p_A indicates the probability of scenario A (which we do not know explicitly, and could be 0). x_S denotes whether set S is picked at stage I, and $r_{A,S}$ indicates whether set S is picked at scenario A. The first constraint shows that for an arbitrary scenario A, every element in A must be covered either at stage I or stage II. In an integer program, $x_S, r_{A,S} \in \{0, 1\}$, and $x_S = 0, r_{A,S} = 1$ or $x_S = 1, r_{A,S} = 0$ are exactly solutions of the above problem.

25.2.2 Rounding

[2] gives an $(2\rho + \epsilon)$-approximation guarantee for stochastic set cover problem for any ϵ, where ρ is the integrality gap for deterministic set cover problem. The intuition of the approach is to prove that each of the element is at least half covered at stage I or in each scenario at stage II, and by decoupling two stages, we can apply the deterministic result to each stage separately. Let OPT_{Det} denote the optimal solution of deterministic set cover problem, and OPT denote the optimal solution of stochastic set cover problem.

Definition 25.2.2 (Deterministic Set Cover Problem(DSC)) *Given a universe U of n elements and m subsets of U, with any subset S having weight w_S. The goal is to choose sets with minimum total weight such that each element e is contained in some chosen set.*

The LP relaxation of deterministic set cover problem can be formulated as

$$\begin{aligned} \min \quad & \sum_S w_S x_S \\ \text{s.t.} \quad & \sum_{S:e\in S} x_S \geq 1, \quad \forall e \\ & x_S \geq 0, \quad \forall S \end{aligned} \tag{P}$$

where x_S indicates whether S is chosen. [3] proposes a greedy algorithm which achieves $\ln n$-approximation: it initializes with empty set, and chooses the set that contains the largest number of uncovered elements at each round.

Theorem 25.2.3 *Suppose that we have a procedure that for every instance of DSC produces a solution of cost at most $\rho \cdot OPT_{Det}$. Then one can convert any solution (x,r) to SSC-P1 to an integer solution losing a factor of at most 2ρ.*

Proof: Let $h(\cdot)$ be the objective function in SSC-P1. Suppose (x, r) is the optimal solution of SSC-P1, and $(\tilde{x}, \tilde{r})$ is the integer (rounding) solution, we are going to argue that the integer solution is obtained by a cost at most $2\rho \cdot h(x, r)$.

Let $E = \{e : \sum_{S:e\in S} x_S \geq \frac{1}{2}\}$, then $(2x)$ is the fractional set cover solution for $e \in E$, so the integer solution of stage I costs at most $\rho \cdot \sum_S w_S^I 2x_S$.

For any scenario A, consider $e \in A\backslash E$. By observation, for an arbitrary element e, it should be covered either to an extent of at least $\frac{1}{2}$ at stage I by x_S, or to an extent of at least $\frac{1}{2}$ at stage II by $r_{A,S}$ in every scenario A contains e. Hence $e \in A\backslash E$ should be covered to an extent of more than a half at stage II, i.e., $A\backslash E = \{e : \sum_{S:e\in S} r_{A,S} \geq \frac{1}{2}\}$. Similarly, $(2r_A)$ is the fractional set cover solution for $e \in A\backslash E$, thus stage II costs at most $\rho \cdot \sum_S w_S^A 2r_{A,S}$.

Sum up two parts, the cost of solution $\tilde{x}$ is at most $2\rho \cdot h(x, r)$. ■

If we apply a polynomial-scenario model, by theorem 25.2.3 we obtain a $2 \ln n$-approximation algorithm for SSC. However, in general, it is hard to solve SSC-P1 as both the variables and constraints are in exponential size. Even if SSC-P1 is "solvable", writing out the solution takes exponential space and time. From the proof of theorem 25.2.3, we only need to examine E, and it only encounters variable x_S in stage I. This conclusion motivates the following formulation with only stage I variable x_S:

$$\begin{aligned} \min \quad & h(x) = \sum_S w_S^I x_S + \sum_{A\subseteq U} p_A f_A(x) \\ \text{s.t.} \quad & 0 \leq x_S \leq 1 \quad \forall S, \\ \text{where} \quad & f_A(x) = \min\{\sum_S w_S^A r_{A,S} : \sum_{S:e\in S} r_{A,S} \geq 1 - \sum_{S:e\in S} x_S, \quad \forall e \in A; \quad r_{A,S} \geq 0, \quad \forall S\} \end{aligned} \tag{SSC-P2}$$

Here the second-stage decisions only appear in the minimization problem $f_A(x)$, which denotes the recourse problem that one needs to solve for scenario A. It is obvious to show that SSC-P1 and SSC-P2 are equivalent programs, and the object function of SSC-P2 is convex.

The idea is to prove that SSC-P2 can be solved efficiently and return a near-optimal first-stage solution x, then we can obtain a 2ρ-approximation algorithm.

25.2.3 Solving the Convex Problem

Definition 25.2.4 *Fully Polynomial Approximation Scheme (FPAS) for 2-stage Stochastic LPs: The algorithm returns a solution of value within $(1+\kappa)$ times the optimum (with high probability), for any $\kappa > 0$, in time polynomial in the input size, $\frac{1}{\kappa}$, and a parameter λ, which is the maximum ratio between the second- and first-stage costs.*

Algorithm 1 High-Level Ellipsoid Method

Start: $P_0 \subseteq E_0$ [containing the feasible region within a ball]
for each step i **do**
  if x_i (current ellipsoid center) is infeasible **then**
    One uses an inequality violated by it as the hyperplane H_i
  else
    One uses an objective function cut as the hyperplane H_i, to eliminate points whose objective function value is no better than the current center. e.g. $h(x) \leq h(x_i)$
    Update feasible region $P_{k+1} = P_k \cap E_i$, $k \leftarrow k+1$
  end if
  Set E_{i+1} to be the ellipsoid of minimum volume containing the half-ellipsoid $E_i \cap H_i$
end for
return $\bar{x} : \min_{x_0, x_1, \cdots, x_k} h(x_i)$

Let $P = P_0$ denotes the polytope $\{x \in \mathbb{R}^m : 0 \leq x \leq 1 \quad \forall S\}$, and x_i be the current iterate. The ellipsoid method (Algorithm 1) can be adapted to find a near-optimal solution to SSC-P2 in polynomial time, despite the fact that evaluating object function $h(.)$ may in general be #P-hard. However, without the ability to evaluate the object function $h(.)$ (or even estimate), it's non-trivial to find an objective function cut. One simple way is to add a constraint $h(x) \leq h(x_i)$, which is not a "linear" cut. But then, in subsequent iterations, one would need to check if the current iterate is feasible, and generate a separating hyperplane if not. Since evaluating $h(.)$ is #P-hard, this appears to pose a formidable difficulty.

So the core issue is how we find an objective function cut. An alternative possibility is to use cuts generated by a subgradient, which essentially plays the role of the gradient when the function is not differentiable.

Definition 25.2.5 *Subgradient: We say that $d \in \mathbb{R}^m$ is a subgradient of a function $g : \mathbb{R}^m \to \mathbb{R}$ at the point u, if $g(v) - g(u) \geq d \cdot (v - u)$ for every $v \in \mathbb{R}^m$*

Lemma 25.2.6 *If d_i is the subgradient at point x_i, we can add the subgradient cut $d_i \cdot (x - x_i) \leq 0$ and proceed with the (smaller) polytope $P_{i+1} = P_i \cap \{x : d_i \cdot (x - x_i) \leq 0\}$. The remaining polytope must contain OPT.*

Proof: Consider what points we remove $\{x^- : d_i \cdot (x^- - x_i) > 0\}$. Since d_i is the subgradient at point x_i, we have $h(x^-) - h(x_i) \geq d_i \cdot (x^- - x_i) > 0$, then $h(x^-) > h(x_i)$. But $OPT \leq h(x_i)$, we will

never remove OPT and the overall objective functions of remaining feasible region will decrease. ■

Unfortunately, even computing a subgradient is hard to do in polynomial time for the objective functions that arise in stochastic programs. To circumvent this obstacle, we define the notion of an approximate subgradient which is crucial to the adapted ellipsoid algorithm.

Definition 25.2.7 *Approximate Subgradient: We say that $\hat{d} \in \mathbb{R}^m$ is an (ω, D)-subgradient of a function $g : \mathbb{R}^m \to \mathbb{R}$ at the point $u \in D$, if for every $v \in D$, we have $g(v) - g(u) \geq \hat{d} \cdot (v - u) - \omega g(u)$*

The algorithm only uses (ω, P)-subgradients which we denote as ω-subgradients from now on. Later, We will show that one can compute with high probability an ω-subgradient of $h(.)$ at any point x, by sampling from the black box on scenarios. Like subgradient, we can add a cut $\hat{d}_i \cdot (x - x_i) \leq 0$ and proceed with the (smaller) polytope $P_{i+1} = P_i \cap \{x : \hat{d}_i \cdot (x - x_i) \leq 0\}$. For any removed points $y \in P_{i+1} \backslash P_i$, we have that $h(y) - h(x_i) \geq \hat{d}_i \cdot (y - x_i) - \omega h(x_i)$ and $\hat{d}_i \cdot (y - x_i) > 0$, then $h(y) > (1 - \omega)h(x_i)$, which means no such point has $h(.)$ value much smaller than $h(x_i)$.

Definition 25.2.8 *Given a function $g : \mathbb{R}^m \to \mathbb{R}$, we say that g has Lipschitz constant (at most) K if $|g(v) - g(u)| \leq K||v - u||$ for all $u, v \in \mathbb{R}^m$*

Continue this way, we obtain a polynomial number of points $x_0, x_1, \cdots, x_k$ such that $x_i \in P_i \subseteq P_{i-1}$ for each i, and the volume of the ellipsoid centered at x_k containing P_k, and hence that of P_k is small. If if $h(.)$ has a bounded Lipschitz constant, then one can show that $\min_i h(x_i)$ is close to the optimal value OPT with high probability. The entire adapted ellipsoid algorithm is formulated below.

Algorithm 2 FindOpt(γ, ϵ)

[Returns a point $\bar{x}$ such that $h(\bar{x}) \leq \frac{OPT}{1-\gamma} + \epsilon$. Assume $\gamma \leq \frac{1}{2}$]
Set: $k \leftarrow 0$, $y_0 \leftarrow 0$, $N \leftarrow 2m^2 \ln \frac{16KR^2}{V\epsilon}$, $n \leftarrow N \log \frac{8NKR}{\epsilon}$, and $\omega \leftarrow \frac{\gamma}{2n}$. Let $E_0 \leftarrow B(0, R)$ and $P_0 \leftarrow P$
for $i = 0, 1, \cdots, N$ **do**
  [We maintain the invariant E_i is an ellipsoid centered at y_i containing the current polytope P_k]
  if $y_i \in P_k$ **then**
    Set $x_k \leftarrow y_i$, Let $\hat{d}_k$ be an ω-subgradient of $h(.)$ at x_k. Let H denote the half space $\{x \in \mathbb{R}^m : \hat{d}_i \cdot (x - x_k) \leq 0\}$. Set $P_{k+1} \leftarrow P_k \cap H$ and $k \leftarrow k + 1$
  else
    Let $a \cdot x \leq b$ be a violated inequality, that is, $a \cdot y_i > b$, whereas $a \cdot x \leq b$ for all $x \in P_k$. Let H be the half space $\{x \in \mathbb{R}^m : a \cdot (x - y_i) \leq 0\}$
  end if
  Set E_{i+1} to be the ellipsoid of minimum volume containing the half-ellipsoid $E_i \cap H_i$
end for
Set $k \leftarrow k - 1$. We now have a collection of points $x_0, x_1, \cdots, x_k$ such that each $x_l \in P_l \subseteq P_{l-1}$
return FindMin(ω; $x_0, x_1, \cdots, x_k$)

Lemma 25.2.9 *The points $x_0, x_1, \cdots, x_k$ generated by FindOpt satisfy $\min_{i=0}^{k} h(x_i) \leq \frac{OPT + \frac{\epsilon}{4}}{1-\omega}$*

Proof: Let x^* be an optimum solution. If x^* is removed from the feasible region P, which means

$\hat{d}_l \cdot (x^* - x_l) \geq 0$ for some l, then $h(x_l) \leq \frac{h(x^*)}{1-\omega}$ since $\hat{d}_l$ is an ω-subgradient at x_l. Otherwise using a volume reduction argument, one can show that there must be a point y lying on some hyperplane $\hat{d}_l \cdot (x - x_l) = 0$ such that $||y - x^*|| \leq \frac{\epsilon}{4K}$, so $h(x_l) \leq \frac{h(y)}{1+\omega} \leq \frac{h(x^*)+\frac{\epsilon}{4}}{1-\omega}$ ■

Since we cannot evaluate $h(.)$ at any given x, we will not be able to choose the point $\bar{x} = argmin_i h(x_i)$. Instead, by using ω-subgradient we will find a point $\bar{x}$ in the convex hull $x_0, x_1, \cdots, x_k$ at which the objective function value is close to $\min_i h(x_i)$. We start at $\bar{x} = x_0$. At each step i, we perform a bisection search to find $\bar{x} = \min(\bar{x}, x_i)$, using the ω-subgradient to infer which direction to move along the line segment. The whole algorithm is showed in Algorithm 3.

Algorithm 3 FindMin(ω; $x_0, x_1, \cdots, x_k$)

Set $\rho \leftarrow \frac{\epsilon}{4k}$, $\bar{x} \leftarrow x_0$, $N' \leftarrow \log \frac{8kKR}{\epsilon}$
for $i = 1, 2, \cdots, k$ **do**
  [We maintain the invariant that $h(\bar{x}) \leq \frac{\min_{l=0}^{i-1} h(l)+(i-1)\rho}{(1-\omega)^{(i-1)N'}}$]
  We use binary search to find y on the $\bar{x} - x_i$ line segment with value close to $\min(h(\bar{x}), h(x_i))$.
  Initialize $y_1 \leftarrow \bar{x}$, $y_2 \leftarrow x_i$
  for $j = 1, 2, \cdots, N'$ **do**
    [We maintain that $h(y_1) \leq \frac{h(\bar{x})}{(1-\omega)^{j-1}}$, $h(y_2) \leq \frac{h(x_i)}{(1-\omega)^{j-1}}$]
    Let $y = \frac{y_1+y_2}{2}$. Computer ω-subgradient $\hat{d}$ of $h(.)$ at point y. If $\hat{d} \cdot (y_1 - y_2) = 0$, then exit the loop. Otherwise exactly one of $\hat{d} \cdot (y_1 - y)$ and $\hat{d} \cdot (y_2 - y)$ is positive
    if $\hat{d} \cdot (y_1 - y) > 0$ **then**
      Set $y_1 \leftarrow y$
    else
      Set $y_2 \leftarrow y$
    end if
  end for
  Set $\bar{x} \leftarrow y$
end for
return $\bar{x}$

Lemma 25.2.10 *Procedure FindMin returns a point $\bar{x}$ such that $h(\bar{x}) \leq \frac{\min_{i=0}^{k} h(x_i)+\frac{\epsilon}{4}}{(1-\omega)^{kN'}}$*

Proof: From the Algorithm 3, it's obvious that the inner "for $j = 1, 2, \cdots, N'$" loop returns a point y such that $h(y) \leq \frac{\min(h(\bar{x}),h(x_i))}{(1-\omega)^{N'}} + \rho$ (due to the ω-subgradient $\hat{d}$). So if the invariant hold at the start of iteration i, setting $\bar{x} \leftarrow y$ at the end of iteration i ensures that it holds at the beginning of $i + 1$. ■

To convert the performance guarantee of procedure FindOpt into a purely multiplicative $(1 + \kappa)$-guarantee, we need to obtain the lower bound on OPT (and set ϵ, γ accordingly), or return 0 as optimal solution (with high probability). The procedure is summarized in Algorithm 4.

Now we proves ConvOpt works well with high probability. Before that, we make the very mild assumption that for an optimal solution x^*, in any scenario $A \neq \phi$, the total cost of scenario A is at least 1, that is, $w^I \cdot x^* + f_A(x^*) \geq 1$.

Algorithm 4 ConvOpt(κ, δ)

[Returns a point $\bar{x}$ such that $h(\bar{x}) \leq (1+\kappa) \cdot OPT$ with high probability. Assume $\delta \leq \frac{1}{2}$]

Define $\lambda = \max(1, \max_{A,S} \frac{w_S^A}{w_S^I})$. Sample $M = \lambda \ln(\frac{1}{\delta})$ times from the distribution on scenarios. Let X = number of times a non-null scenario occurs

if $X = 0$ **then**

  return $x = 0$ as optimal solution

else

  With high probability, $OPT \geq \frac{\varrho}{\lambda}$, where $\varrho = \frac{\delta}{\ln \frac{1}{\delta}}$. Set $\epsilon = \frac{\kappa \varrho}{2\lambda}$, $\gamma = \frac{\kappa}{3}$

  return FindOpt(γ, ϵ)

end if

Lemma 25.2.11 *ConvOpt determines with probability at least $1-\delta$, that $OPT \geq \frac{\varrho}{\lambda}$, or that $x = 0$ is an optimal solution.*

Proof: Note that $\varrho \leq 1$ since $\delta \leq \frac{1}{2}$. Since in every non-null scenario, we incur a cost of at least 1, $OPT = E(\text{minimal cost}) \geq 1 \cdot q + 0 \cdot (1-q) = q$, where $q = \sum_{A \subseteq U, A \neq \phi} p_A$ is the probability of occurrence of a non-null scenario. Let $r = Pr[X = 0] = (1-q)^M$. So $r \leq e^{-qM}$ and $r \geq 1 - qM$. If $q \geq \frac{-\ln(\delta)}{M}$, then $Pr[X = 0] \leq \delta$. So with probability at least $1-\delta$ we will say that $OPT \geq \frac{\varrho}{\lambda}$ which is true since $OPT \geq q \geq \frac{1}{\lambda}$. If $q \leq \frac{\delta}{M}$, then $Pr[X = 0] \geq 1 - \delta$. We return $x = 0$ as an optimal solution with probability at least $1-\delta$ which is indeed an optimal solution, because $q \leq \frac{1}{\lambda} = \frac{-\ln(\delta)}{M}$ implies that it is always at least good to defer to stage II since the expected stage II cost of set S is at most $q \cdot w_S^A \leq W_S^I$. If $\frac{\delta}{M} < q < \frac{-\ln(\delta)}{M}$, then we always return a correct answer since it is both true that $x = 0$ is an optimal solution, and that $OPT \geq q \geq \frac{\varrho}{\lambda}$. ■

The last important thing is how to compute an ω-subgradient at $x \in P$ efficiently (with high probability). Consider the dual version of SSC-P2 $f_A(x)$.

$$\begin{aligned} \max \quad & \sum_e (1 - \sum_{S: e \in S} x_S) z_{A,e} \\ \text{s.t.} \quad & z_A \in \mathcal{Q}_A \\ \text{where} \quad & \mathcal{Q}_A = \{z \in \mathbb{R}^{|U|} : \sum_{e \in A \cap S} z_e \leq w_S^A \quad \forall S; z_e = 0 \quad \forall e \notin A; z \geq 0\} \end{aligned}$$

The Lemma 25.2.12 and Lemma 25.2.13 show that each component of the subgradient vector is the expectation of a random variable with bounded variance.

The Lemma 25.2.13 also gives a bound on the Lipschitz constant K.

Lemma 25.2.12 *Let d be a subgradient of $h(.)$ at the point $x \in P$, and suppose $\hat{d}$ is a vector such that $d_S - \omega w_S^I \leq \hat{d}_S \leq d_S$ for all S. Then $\hat{d}$ is an ω-subgradient of $h(.)$ at x.*

Lemma 25.2.13 *Let $x \in \mathbb{R}^m$, and let z_A^* be an optimal dual solution for scenario A with x as the stage I vector. The d with components $d_S = w_S^I - \sum_A p_A \sum_{e \in S} z_{A,e}^*$ is a subgradient at x, and $||d|| \leq \lambda ||w^I||$.*

Using standard Chebyshev and Chernoff bounds one can show the following.

Lemma 25.2.14 *Let $X \in [-a, b]$ be a random variable, $a, b > 0$, computed by sampling from a distribution π. Let $\mu = E[X]$ and $\alpha = \max(1, \frac{a}{b})$. Then for any $c > 0$, by taking $\frac{100\alpha^2}{3c^2}\ln(\frac{1}{\delta})$ independent samples from π, one can compute an estimate $\hat{X}$ such that $\mu - 2cb \leq X \leq \mu$ with probability at least $1 - \delta$.*

Corollary 25.2.15 *At any point $x \in P$, one can compute an ω-subgradient with probability at least $1 - \delta$ using at most $\frac{400\lambda^2}{3\omega^2}\ln(\frac{m}{\delta})$ independent samples from the probability distribution on scenarios.*

By all of previous algorithms and analysis, we can conclude Lemma 25.2.16.

Lemma 25.2.16 *Using the above sampling method to compute ω-subgradients, ConvOpt computes a feasible soulution to SSC-P2 of cost at most $(1+\kappa)\cdot OPT$ with probability at least $1 - 2\delta$, in time poly(input size, $\lambda, \frac{1}{\kappa}, \ln(\frac{1}{\delta})$).*

25.3 Stochastic Facility Location Problem

The above algorithm can be generalized to solve a broad class of 2-stage stochastic problems. By a convex programming relaxation, the rounding approach bounds the integrality gap of stochastic problem by 2 times the integrality gap of the corresponding deterministic problem.

In this section, we take 2-stage stochastic uncapacitated facility location problem(SUFL) as an application of the above algorithm.

Definition 25.3.1 (Deterministic Uncapacitated Facility Location(DUFL)) *Given a set of candidate facility locations F, and a set of clients D. We want to open facilities at a subset of F, and assign each client to an open facility. Opening a facility at location i incurs a cost of f_i, while assigning a client j to facility i costs c_{ij}. The goal is to minimize the total cost of opening facilities and assigning clients.*

The integrality gap for DUFL is $\rho_{DUFL} \leq 1.52$[4].

In a 2-stage SUFL, given a probability distribution of the demand(assign which client to which facility) of clients. facility is either opened at cost f_i^I at stage I, or opened at cost f_i^A in scenario A at stage II. Thus, the LP relaxation can be formulated as following:

$$\begin{aligned} \min \quad & \sum_i f_i^I y_i + \sum_{A \subseteq U} p_A \Big(\sum_i f_i^A y_{A,i} + \sum_{j \in A, i} c_{ij} x_{A,ij}\Big) && \text{(SUFL-P1)} \\ \text{s.t.} \quad & \sum_i x_{A,ij} \geq 1 \quad \forall A, j \in A \\ & x_{A,ij} \leq y_i + y_{A,i} \quad \forall i, A, j \in A \\ & y_i, y_{A,i}, x_{A,ij} \geq 0 \quad \forall i, A, j \in A \end{aligned}$$

Similarly, we can compactify SUFL-P1 to obtain a convex program:

$$\min \quad \sum_i f_i^I y_i + \sum_{A \subseteq U} p_A g_A(y) \quad \text{s.t.} \quad 0 \le y_i \le 1, \quad \forall i \qquad \text{(SUFL-P2)}$$

$$\begin{aligned} \text{where} \quad g_A(y) = \min \quad & \sum_i f_i^A y_{A,i} + \sum_{j \in A,i} c_{ij} x_{A,ij} \\ \text{s.t.} \quad & \sum_i x_{A,ij} \ge 1 \quad \forall A, j \in A \\ & x_{A,ij} \le y_i + y_{A,i} \quad \forall i, A, j \in A \\ & y_i, y_{A,i}, x_{A,ij} \ge 0 \quad \forall i, A, j \in A \end{aligned}$$

Theorem 25.3.2 *The integrality gap of SUFL-P2 is at most* $2\rho_{DUFL}$.

Proof: Let $h(\cdot)$ be the objective function in SUFL-P2. Suppose y is the optimal solution of SUFL-P2, and (x_A^*, y_A^*) be the optimal solution of $g_A(y)$. The proof is similar to theorem 25.2.3, we will show that we can decouple the two stages and solving two DUFL problems.

Fix a scenario A with client $j \in A$, i.e., $x_{A,ij} > 0$. we can split it into two stages: $x_{A,ij} = xA, ij^I + xA, ij^{II}$. Thus, $x_{A,ij}^I \le y_i^*$, and $x_{A,ij}^{II} \le y_{A,i}^*$. By observation, client j must either be assigned to a facility opened at stage I, or be assigned to a facility opened at stage II, so $\sum_i x_{A,ij}^I \ge \frac{1}{2}$ or $\sum_i x_{A,ij}^{II} \ge \frac{1}{2}$.

In the first case, take a set of scenarios $S_j = \{j \in A : \sum_i x_{A,ij}^I \ge \frac{1}{2}\}$. Stage I can be decoupled as a DUFL that facility opens at cost f_i^I, assigning a client j to facility i cost c_{ij} per demand, and the demand of j is $\sum_{A \in S_j} p_A$. As we know $\sum y_i \ge \sum_i x_{A,ij}^I \ge \frac{1}{2}$, for scenario A a feasible fractional solution can be set as $x_{\hat{A},ij} = min(1, 2x_{A,ij}^I)$, $\hat{y_i} = min(1, 2y_i^*)$. As $\hat{y_i}$ does not depend on the scenario, we can re-optimize the assignment (j, A) which only depends on $\hat{(}y_i)$ variables: first reset $x_{\hat{A},ij} = 0$, then put i in non-decreasing order of c_{ij}; then for each i, set $x_{\hat{A,}i_t j} = \min(\hat{y_{i_t}}, x_{A,ij})$. In this manner we can obtain a mininum cost of assignment with current opened facilities, motivates us to treat all clients (j, A) as j with demand $\sum_{A \in S_j} p_A$. Hence, the facility cost is at most $2\sum_i f_i^I y_i^*$, and the client cost is at most $2\sum_{i,j} \sum_{A \in S_j} p_A c_{ij} x_{A,ij}^I \le 2\sum_{i,j} \sum_{A \in S_j} p_A c_{ij} x_{A,ij}$. So for the set of facilities open in stage I, the integer solution costs at most $2\rho_{DUFL} \cdot (\sum_i f_i^I y_i^* + \sum_{i,j} \sum_{A \in S_j} p_A c_{ij} x_{A,ij}^*)$.

For the remaining set $D_i = j \in A : A \notin S_j$, we have $\sum_i x_{A,ij}^{II} \ge \frac{1}{2}$. Similarly, the fractional solution should be set as $x_{\hat{A},ij} = min(1, 2x_{A,ij}^{II})$, $\hat{y_i} = min(1, 2y_{A,i}^*)$. So we can bound the integer solution with a cost at most $2\rho_{DUFL} \cdot (\sum_i f_i^A y_{A,i}^* + \sum_{i,j \in D_A} c_{ij} x_{A,ij}^*)$.

Combine the two cases, we can conclude that the overall cost is at most $2\rho_{DUFL} OPT$. ■

Note that without knowing the demand of client j $\sum_{A \in S_j} p_A$ explicitly, we may not be able to apply rounding technique with the 1.52-approximation DUFL. However, [5] gives an approximation algorithm for DUFL without knowing clients' demands explicitly, which only increases the cost by a factor of at most 1.705. Hence rounding gives a 3.225-approximation algorithm for SUFL.

References

[1] C. Swamy and D. B. Shmoys, "Approximation algorithms for 2-stage stochastic optimization problems," *ACM SIGACT News*, vol. 37, no. 1, pp. 33–46, 2006.

[2] D. B. Shmoys and C. Swamy, "Stochastic optimization is (almost) as easy as deterministic optimization," in *45th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 2004, pp. 228–237.

[3] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of operations research*, vol. 4, no. 3, pp. 233–235, 1979.

[4] M. Mahdian, Y. Ye, and J. Zhang, "Improved approximation algorithms for metric facility location problems," in *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer, 2002, pp. 229–242.

[5] C. Swamy and D. Shmoys, *Approximation algorithms for clustering problems*. Citeseer, 2004.