CSL 356: Analysis and Design of Algorithms

Ragesh Jaiswal

CSE, IIT Delhi

Linear Programming

Solving LP

Linear Programming: Solving LP

- (*Complication 1*) What if the initial basic solution is not a feasible solution?
- Pre-processing algorithm:
  - Given L, check if all b_i's are positive. In that case return L.
  - Consider L'. Perform the pivoting using the equation with smallest b_i to obtain L''.
  - Solve L'' using Simplex and find the optimal objective value Opt.
  - If ($Opt \neq 0$), the output "LP is infeasible".
  - Otherwise, let L_S be the LP obtained at the end of the simplex. Do the following:
    - If x_0 is a basic variable in L_S, then perform a pivoting step to obtain L_S'.
    - Remove all instances of x_0 and rewrite the objective function of L in terms of non-basic variables of L_S'.

Linear Programming: Solving LP

- (*Complication 1*) What if the initial basic solution is not a feasible solution?
- Pre-processing algorithm: Example
- L:
  - $z = 2x_1 - x_2$
  - $x_3 = 2 - 2x_1 + x_2$
  - $x_4 = -4 - x_1 + 5x_2$
- L':
  - $z = -x_0$
  - $x_3 = 2 - 2x_1 + x_2 + x_0$
  - $x_4 = -4 - x_1 + 5x_2 + x_0$
- L'': After Pivot using ($x_4 = \ldots$)
  - $z = -4 - x_1 + 5x_2 - x4$
  - $x_3 = 6 - x_1 - 4x_2 + x_4$
  - $x_0 = 4 + x_1 - 5x_2 + x_4$

Linear Programming: Solving LP

- (*Complication 1*) What if the initial basic solution is not a feasible solution?
- Pre-processing algorithm: Example
- L:
  - $z = 2x_1 - x_2$
  - $x_3 = 2 - 2x_1 + x_2$
  - $x_4 = -4 - x_1 + 5x_2$
- L_S:
  - $z = -x_0$
  - $x_2 = 4/5 - x_0/5 + x_1/5 + x_4/5$
  - $x_3 = 14/5 + 4x_0/5 - 9x_1/5 + x_4/5$
- L_S:
  - $z = 2x_1 - x_2 = 2x_1 - (4/5 + x_1/5 + x_4/5) = -4/5 + 9x_1/5 - x_4/5$
  - $x_2 = 4/5 + x_1/5 + x_4/5$
  - $x_3 = 14/5 - 9x_1/5 + x_4/5$

Linear Programming

Approximation algorithms

Linear Programming: Approx. algo.

- Problem(Weighted Set Cover): Given subsets $S_1, \dots, S_m$ of a universe U of elements and positive weights $w_1, \dots, w_m$ attached with these subsets. Find a subset S of $\{S_1, \dots, S_m\}$ such that S covers all elements of U and $\sum_{i:S_i \in S} w_i$ is minimized. Also, assume that each element appears in at most f subsets.

Linear Programming: Approx. algo.

- Problem(Weighted Set Cover): Given subsets $S_1, \dots, S_m$ of a universe U of elements and positive weights $w_1, \dots, w_m$ attached with these subsets. Find a subset S of $\{S_1, \dots, S_m\}$ such that S covers all elements of U and $\sum_{i:S_i \in S} w_i$ is minimized. Also, assume that each element appears in at most f subsets.
- Integer LP formulation:
  - Minimize $\sum_i w_i \cdot x_i$
  - Subject to:
    - $\sum_{i:S_i \; contains \; e} x_i \geq 1$, for every element e in U
    - $x_i \in \{0, 1\}$, for every i.

Linear Programming: Approx. algo.

- Problem(Weighted Set Cover): Given subsets $S_1, \dots, S_m$ of a universe U of elements and positive weights $w_1, \dots, w_m$ attached with these subsets. Find a subset S of $\{S_1, \dots, S_m\}$ such that S covers all elements of U and $\sum_{i:S_i \in S} w_i$ is minimized. Also, assume that each element appears in at most f subsets.
- *Relaxed* LP:
  - Minimize $\sum_i w_i \cdot x_i$
  - Subject to:
    - $\sum_{i:S_i \text{ contains } e} x_i \geq 1$, for every element e in U
    - $x_i \geq 0$, for every i.

Linear Programming: Approx. algo.

- Problem(Weighted Set Cover): Given subsets $S_1, \dots, S_m$ of a universe U of elements and positive weights $w_1, \dots, w_m$ attached with these subsets. Find a subset S of $\{S_1, \dots, S_m\}$ such that S covers all elements of U and $\sum_{i:S_i \in S} w_i$ is minimized. Also, assume that each element appears in at most f subsets.
- Let O_I be the optimal solution to the Integer LP and let O_R be the optimal solution to the relaxed LP.
- Claim: $O_I \geq O_R$.

Linear Programming: Approx. algo.

- Problem(Weighted Set Cover): Given subsets $S_1, \ldots, S_m$ of a universe U of elements and positive weights $w_1, \ldots, w_m$ attached with these subsets. Find a subset S of $\{S_1, \ldots, S_m\}$ such that S covers all elements of U and $\sum_{i:S_i \in S} w_i$ is minimized. Also, assume that each element appears in at most f subsets.
- Let O_I be the optimal solution to the Integer LP and let O_R be the optimal solution to the relaxed LP.
- Claim: $O_I \geq O_R$.
- Rounding:
  - Let $(r_1, \ldots, r_m)$ give the optimal solution to the relaxed LP.
  - If $r_i \geq 1/f$, then put S_i in the set S.

Linear Programming: Approx. algo.

- Problem(Weighted Set Cover): Given subsets $S_1, \ldots, S_m$ of a universe U of elements and positive weights $w_1, \ldots, w_m$ attached with these subsets. Find a subset S of $\{S_1, \ldots, S_m\}$ such that S covers all elements of U and $\sum_{i: S_i \in S} w_i$ is minimized. Also, assume that each element appears in at most f subsets.
- Let O_I be the optimal solution to the Integer LP and let O_R be the optimal solution to the relaxed LP.
- Claim: $O_I \geq O_R$.
- Rounding:
  - Let $(r_1, \ldots, r_m)$ give the optimal solution to the relaxed LP.
  - If $r_i \geq 1/f$, then put S_i in the set S.
- Claim: S is a set cover of size at most $f \cdot OPT$.

Linear Programming

Randomized approximation algorithms

Linear Programming: rand. approx.

- Recall the 2-SAT problem. Given a 2-SAT formula, determine if the formula has a satisfying assignment.
- 2-SAT is in **P**.
- Problem(L-2-SAT): Given a 2-SAT formula (with n variables and m clauses) and an integer k, determine if there is an assignment that makes at least k clauses true.
- Claim 1: L-2-SAT is **NP** complete.
  - Proof:
    - Claim 1.1: L-2-SAT is in **NP**.
    - Claim 1.2: CLIQUE $\leq_p$ L-2-SAT.

Linear Programming: rand. approx.

- Claim 1.2: CLIQUE $\leq_p$ L-2-SAT.
- Proof: Given an instance of the CLIQUE problem (G, k), we construct the following instance of the L-2-SAT problem.
  - For each vertex i, we have a variable x_i.
  - We use one extra variable z.
  - We construct the following clauses:
    - $C_1 = \{(x_1 \vee z), (x_2 \vee z), \ldots, (x_n \vee z)\}$
    - $C_2 = \{(x_1 \vee z'), (x_2 \vee z'), \ldots, (x_n \vee z')\}$
    - $C_3 = \{(x_i' \vee x_j') \mid (i, j) \text{ is not in } E\}$
  - The 2-SAT formula α contains all the above clauses.
  - Claim: G has a clique of size at least k if and only if there is an assignment that makes at least $(|V| + |C_3| + k)$ clauses of α to be true.

End