## Simplex

**Proposition.** ***Simplex***

1. *If Simplex returns a result, it is a feasible solution*
2. *Simplex will terminate in at most* $\binom{n+m}{m}$ *steps (prevent cycling by choosing entering/leaving variable of least possible index)*
3. *Simplex algorithm yields an optimal solution (proves that on termination, the maximization and minimization problem is tight on a single solution set)*

**Definition.** *Given a* ***primal*** *version of LPP,*

$$\begin{aligned} \textbf{\textit{Maximize}} \quad & c^T x \\ \textbf{\textit{Subject to}} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

*The corresponding* ***dual*** *version is given by*

$$\begin{aligned} \textbf{\textit{Minimize}} \quad & b^T y \\ \textbf{\textit{Subject to}} \quad & A^T y \geq c \\ & y \geq 0 \end{aligned}$$

**Lemma.** *Let $\overline{x}$ be any feasible solution to the primal and $\overline{y}$ be any feasible solution to the dual LPP. Then*

$$\sum_{j=1}^{n} c_j \overline{x}_j \leq \sum_{i=1}^{m} b_i \overline{y}_i \qquad \iff \qquad c^T \overline{x} \leq b^T \overline{y}$$

*Proof.*

$$\begin{aligned} \sum_{j=1}^{n} c_j \overline{x}_j &\leq \sum_{j=1}^{n} \left( \sum_{i=1}^{m} a_{ij} \overline{y}_i \right) \overline{x}_j \\ &= \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} \overline{x}_j \overline{y}_i \\ &\leq \sum_{i=1}^{m} b_i \overline{y}_i \end{aligned}$$

□

**Corollary.** *Let $\overline{x}$ be a feasible solution to the primal and $\overline{y}$ be feasible solution to the dual such that*

$$\sum_{j=1}^{n} c_j \overline{x}_j = \sum_{i=1}^{m} b_i \overline{y}_i$$

*Then $\overline{x}$ is an optimal solution to the primal LPP and $\overline{y}$ is an optimal solution to dual LPP*

**Theorem.** ***LPP Duality*** *Assume simplex returns values $\overline{x} = (\overline{x}_1, \cdots, \overline{x}_n)$ for the LPP $(A, b, c)$. Let $N$ be the nonbasic and $B$ be basic variables for the final slack form $(N, B, A', b', c', v')$. Define $\overline{y} = (\overline{y}_1, \cdots, \overline{y}_m)$ as follows,*

$$\overline{y}_i = \begin{cases} c_{n+i} & \text{if } n+i \in N \\ 0 & \text{otherwise} \end{cases}$$

*Then $\overline{x}$ is an optimal solution to the primal and $\overline{y}$ is an optimal solution to the dual, and*

$$\sum_{j=1}^{n} c_j \overline{x}_j = \sum_{i=1}^{m} b_i \overline{y}_i \quad \iff \quad c^T x = b^T y$$

**Function** `Initialize-Simplex`$(A, b, c)$
    $k \leftarrow$ index of minimum $b_i$
    **if** $b_k \geq 0$ **then**
        **return** $(\{1, \cdots, n\}, \{n+1, \cdots, n+m\}, A, b, c, 0)$
    $L_{aux} \leftarrow$ be $L$ by adding $x_0$ to LHS of each constant,
        and setting the objective function to $-x_0$
    $(N, B, A, b, c, v) \leftarrow$ the resulting slack from $L_{aux}$
    $l \leftarrow n + k$
    $(N, B, A, b, c, v) =$ `Pivot`$(N, B, A, b, c, l, 0)$ //basic solution feasible
    $x \leftarrow$ `Simplex`$(N, B, A, b, c, v)$
    **if** $\overline{x}_0 \neq 0$ **then**
        **return** `Infeasible`
    **else**
        **if** $\overline{x}_0$ *is basic variable* **then**
            do another pivot to make it nonbasic
        **else**
            remove $\overline{x}_0$ from constraints
            restore original objective function of $L$, but replace each basic variable
                in each objective function by RHS of its associated constraints
    **return** *Modified slack form*

**Definition.** *During initialization,*

1. *if LPP is infeasible, stops right away*

2. *if LPP is feasible, but the basic solution is not feasible, transform LPP to another problem such that the basic solution is feasible, do simplex*

   (a) *Define $L_{aux}$*

$$\begin{aligned} \textbf{\textit{Maximize}} \quad & -x_0 \\ \textbf{\textit{Subject to}} \quad & x_1 - x_2 - x_0 \leq -5 \\ & x_1 + x_2 - x_0 \leq 11 \\ & x_1, x_1, x_2 \geq 0 \end{aligned}$$

   *$L$ is feasible if and only if the optimal objective value of $L_{aux}$ is 0*

3. *Otherwise, do simplex*

# NP-Completeness

**Definition.** ***Polynomial-time Reducibility*** *Let $X$ and $Y$ be two problems. We say $Y$ is polynomial-time reducible to $X$ if arbitrary instances of problem $Y$ can be solved using a polynomial number of standard computational steps, plus a polynomial number of calls to a blackbox that solves problem $X$. We specify this as*

$$Y \leq_p X$$

*$X$ is at least as hard as $Y$*

1. *Suppose $Y \leq_p X$, If $X$ can be solved in polynomial time, then so can be $Y$*
2. *Suppose $Y \leq_p X$, If $Y$ cannot be solved in polynomial time, then $X$ cannot be solved in polynomial time either*
3. *If $X$, $Y$ and $Z$ are 3 problems, and $Z \leq_p Y$ and $Y \leq_p X$, then $Z \leq_p X$*

*If want to show $X \in NP$, we reduce $Y \leq_p X$ such that $Y \in NP$*

**Example. Independent set of vertex cover**

1. **Independence Set** Given a set $S \subseteq V$ of nodes in graph $G = (V, E)$ is called independent if no two vertices in $S$ is connected in $G$
2. **Independent Set problem** Find the independent set of maximum size
3. **An equivalent decision problem** Given a graph $G$ and a number $k$, ask if there exists an independent set of size $k$.
4. **Vertex cover** A set $S \subseteq V$ in $G = (V, E)$ is called a vertex cover if every $e \in E$ has at least one end in $S$

5. **Vertex cover problem** Given a graph $G$, find a vertex cover of minimum size.

6. **An equivalent decision problem** Given a graph $G$ and $k \in \mathbb{R}$, ask if there exists a vertex cover of size of at most $k$

**Lemma.** *Let $G = (V, E)$ be a graph, then $S \subseteq V$ is an independent set if and only if $V \setminus S$ is a vertex cover*

*Proof.* 1. => If $S$ is independent, want to show $V \setminus S$ is a vertex cover. Let $e = (u, v) \in E$. Assume $u, v \notin V \setminus S$, then $u, v \in S$, then $S$ not independent

□

**Corollary.** *Independent set problem $\leq_p$ vertex cover problem. Similarly for vertex cover problem $\leq_p$ independent set problem (equally hard)*

*Proof.* If we have a black box to solve vertex cover, then we can decide whether $G$ has an independent set of size at least $k$ by asking the blackbox if $G$ has a vertex cover of size at most $n - k$ □

**Example. Set cover** Given a set $U$ of $n$ elements, a collection $S_1, S_2, \cdots, S_m \subseteq U$, and a number $k$, does there exists a collection of at most $k$ of these sets whose union is $U$ Relation between vertex cover and set cover. For $i \in V$, define

$$S_i = \{\text{edges going out of } i\}$$

so $\cup S_i = E$ so $U = E$.

**Lemma.** *Vertex cover $\leq_p$ Set cover*

**Example. Set packing** Given a set $U$ of $n$ elements, a collection $S_1, \cdots, S_,$ of subsets of $U$ and a number $k$¡ does there exists a collection of at least $k$ of these sets with the property that no two of them intersect?

**Lemma.** *Independent set $\leq_p$ set packing*

**Definition.** ***Problem***

1. *Solving a problem, i.e. finding a solution*

2. *Verification, i.e. checking if something is a solution*

***Formulation***

1. ***Input*** *A binary string $s \in \{0, 1\}^*$, where*

$$I = \{0, 1\}^* = \bigcup_{n \in N} \{0, 1\}^n$$

2. ***Decision problem***

$$X = \{s \in I \mid X(s) = yes\}$$

3. ***Algo for decision problem*** *An algorithm A for a decision problem is a function that takes a binary string input and outputs yes or no.*

4. ***Solution*** *We say A solves problem X if for all strings $s \in I$, we have*

$$A(s) = yes \qquad \text{if and only} \qquad \text{if } s \in X$$

5. ***Polynomial-time solution*** *We say A is a polynomial-time algorithm if there is a polynomial function P such that for every input $s \in I$, the algorithm termiantes in at most $O(P(|s|))$ steps.*

$$\mathbb{P} = \{X \mid \text{if there exists a polynomial-time algorithm } A \text{ that solves } x\}$$