

Optimization Course Notes

Rasmus Kirk Jakobsen

2022-06-28

Contents

1. Linear Programming Problems	3
Disposition	3
Examples	3
Definitions	3
Linear Programming Problems	3
Standard form	3
Fundamental Theorem of Linear Programming	4
Simplex	4
Example	4
2. Duality	7
Disposition	7
Examples	7
Duality theorems/properties	8
3. Network Flows	9
Disposition	9
Network Flow	9
Balances	9
Arc Constraints	9
The Maximum (s, t) -flow Problem	9
Ford-Fulkerson Algorithm	9
4. P, NP and Cook's theorem	10
Disposition	10
NP completeness teori	10
Language Complexity	11
SAT & CSAT	11
5. NP-Complete Problems	12
Disposition	12

6. Approximation Algorithms and Search Heuristics	13
Disposition	13
Deterministic Max-Cut Example	13
Randomized Max-Cut Example	14
Appendix	16
CSAT gates to CNF proofs	16
NOT	16
COPY	16
AND	16
OR	16
XOR	16
EQ	17

1. Linear Programming Problems

Disposition

- Linear Programming Problems
- Standard form
- Fundamental Theorem of Linear Programming
- Simplex
 - Two Phase
 - Example
 - (Degeneracy)

Examples

$$z = 0 + x_1 + x_2$$

$$1 \geq x_1 + x_2$$

$$2 \geq x_1 + x_2$$

$$\begin{array}{ll} \max/\min & z = c^T x \\ s.t. & Ax \begin{pmatrix} \leq \\ \geq \\ = \end{pmatrix} b \\ & x \geq 0 \end{array}$$

Definitions

- **Objective Function:** A Linear Function, over all variables, to be maximized.
- **Polytope:** The geometric shape formed by the constraints.
- **Solution:** Any possible values that can be assigned to the variables, ignoring constraints.
- **Feasible Solution:** Any *Solution*, satisfying all constraints.
- **Basic Solution:** A *Feasible Solution* which lies in the geometric corner of the polytope.
- **Optimal Solution:** A *Feasible Solution* that maximizes the target function.

Linear Programming Problems

- Problems of the form: $z = \sum_{i=0}^n c_i x_i$ where we want to maximize or minimize z
- Must have constraints

Standard form

1. Must be maximization problem
2. All constraints must be \geq

Fundamental Theorem of Linear Programming

1. If no optimal solution exists, the problem is infeasible or unbounded
 2. If 1. and there exists a feasible solution there exists a Basic Feasible Solution
 3. If there exists an optimal solution, there exists a Basic Optimal Solution
- **TLDR:** If an optimal solution exists, there must be one in a corner of the convex polytope.

Simplex

- Takes a Linear Programming Problem in Standard Form,
- Returns the optimal solution
- Simplex Tableau:

$$\begin{bmatrix} 1 & -\vec{c}^T & 0 \\ 0 & A & \vec{b} \end{bmatrix}$$

- **Slack variables:** Constraints of the form $c \leq c_1x_1 + c_2x_2 \rightarrow x_{n+1} = c - (c_1x_1 + c_2x_2)$
- **Cycles:** Suppose we have some Dictionary D_0 , and we pivot some number of times to get Dictionary D_k , where we have already seen D_k :
 - $D_0 \rightarrow D_1 \cdots D_k : D_k \in [D_0, D_{k-1}]$
- **Bland's Rule:**
 - Why?
 - * Prevents cycles
 - How?
 - * Start by choosing the left-most non-basic variable with a positive coefficient

Example

We start with:

$$z = 0 + 10x_1 + 22x_2$$

$$11 \geq 3x_1 + 4x_2$$

$$15 \geq 5x_1 + 20x_2$$

Introducing slack variables:

$$z = 0 + 10x_1 + 22x_2$$

$$x_3 = 11 - 3x_1 - 4x_2$$

$$x_4 = 15 - 5x_1 - 20x_2$$

1. We need to choose the *Entering Variable*, using Bland's rule we choose x_1 .

2. Then we need to choose an *Exiting Variable*, using Bland's rule, $x_3 : 11/3 = 3 + \frac{2}{3}$ and $x_4 : 15/5 = 3$, x_4 has the smallest non-negative value, so x_4 is the exiting variable.
3. Isolate the entering variable, (x_1) , from the definition of our exiting variable (x_4) .
4. Repeat 1-3 until all coefficients in the objective function are non-negative (We don't need to repeat for this example):

$$x_4 = 15 - 5x_1 - 20x_2$$

$$x_4 + 5x_1 = 15 - 20x_2$$

$$5x_1 = 15 - 20x_2 - x_4$$

$$x_1 = 3 - 4x_2 - \frac{1}{5}x_4$$

Inserting x_1 in z :

$$z = 0 + 10x_1 + 22x_2$$

$$z = 0 + (30 - 2x_4 - 40x_2) + 22x_2$$

$$z = 30 - 2x_4 - 18x_2$$

Because we are done, we don't actually need to do it for x_3 , but for completeness, we finish step 3:

$$x_3 = 11 - 3x_1 - 4x_2$$

$$x_3 = 11 - (9 - 12x_2 - \frac{3}{5}x_4) - 4x_2$$

$$x_3 = 2 + 8x_2 + \frac{3}{5}x_4$$

So our final dictionary:

$$z = 30 - 2x_4 - 18x_2$$

$$x_1 = 3 - 4x_2 - \frac{1}{5}x_4$$

$$x_3 = 2 + 8x_2 + \frac{3}{5}x_4$$

This means that we have found our maximum, 30. If we want to find the necessary variables to produce 30. We know $x_2 = 0$, to isolate x_4 :

$$x_1 = 3 - 4x_2 - \frac{1}{5}x_4$$

$$x_1 = 3 - 4 \cdot 0 - \frac{1}{5} \cdot 0$$

$$x_1 = 3$$

We can make a last sanity check:

$$z = 0 + 10 \cdot 3 + 22 \cdot 0$$

$$z = 0 + 30 + 0$$

$$z = 30$$

2. Duality

Disposition

- Duality
 - Motivation
 - Geometric intuition
 - Strong & Weak Duality Theorems
 - Complimentary Slackness
- Matrix Games
 - Example
 - Nash Equilibrium
 - Fair Game
 - Principle of Indifference

Examples

General Example Primal:

$$\begin{array}{ll}
 \mathbf{P:} & \\
 \max & z = c^T x \\
 s.t. & Ax \leq b \\
 & x \geq 0
 \end{array}$$

Dual:

$$\begin{array}{ll}
 \mathbf{D:} & \\
 \min & w = b^T y \\
 s.t. & A^T y \geq c \\
 & y \geq 0
 \end{array}$$

Rock Paper Scissors Our matrix A :

	Rock	Paper	Scissors
Rock	0	1	-1
Paper	-1	0	1
Scissors	1	-1	0

Primal (Column Player):

$$\begin{array}{ll}
 \mathbf{P:} & \\
 \max & z = v \\
 s.t. & A \vec{p} \leq \vec{v} \\
 & \vec{p} \geq \vec{0} \\
 & \vec{p}^T \vec{1} = 1
 \end{array}$$

Dual (Row Player):

$$\begin{array}{ll}
 \mathbf{D:} & \\
 \max & w = u \\
 s.t. & A^T \vec{q} \geq \vec{u} \\
 & \vec{q} \geq \vec{0} \\
 & \vec{q}^T \vec{1} = 1
 \end{array}$$

Duality theorems/properties

- For any feasible solution $p \in P$, and any feasible solution $d \in D$, $p \leq d$
- **Weak Duality Theorem:**
 - $p \leq d$
- **Strong Duality Theorem:**
 - $p = d \Leftrightarrow p = \text{optimal}(P) \wedge d = \text{optimal}(D)$
- If P is unbounded then D is infeasible and vice versa

3. Network Flows

Disposition

- Network Flow
 - Balances
 - Arc constraints
- Maximum (s, t) -flow
- Max flow-min cut theorem
- Ford-Fulkerson example

Network Flow

Balances

- A flow network: $D = (N, A)$
- Outgoing flow from node i : $\sum_{ij \in A} x_{ij}$
- Ingoing flow to node i : $\sum_{ji \in A} x_{ji}$
- Balance at node i : $b_i = \text{out} - \text{in} = \sum_{ij \in A} x_{ij} - \sum_{ji \in A} x_{ji}$
- Balance restriction: $\sum_{i \in N} b_i = 0$
- If $b_i > 0$ then node i is a source, if $b_i < 0$ then node i is a sink

Arc Constraints

- Lower (l_{ij}) and upper (u_{ij}) bound for flows in nodes: $l_{ij} \leq x_{ij} \leq u_{ij}$
- Assumption: $0 \leq l_{ij} \leq u_{ij}$

The Maximum (s, t) -flow Problem

- One source (s)
- One sink (t)
- Flow conservation restriction: $b_i = 0 \mid i \in N \setminus \{s, t\}$
- Flow is feasible if:
 - No negative flows
 - Flow conservation restriction
 - Must satisfy arc constraints
- Convert maximum (s, t) -flow problem $(D = (N, A))$ to minimum flow problem:
 - New edge s to t is added to D' with cost -1 and upper bound ∞
 - All other edges has cost 0
 - All other nodes has balance $+$
 - Feasible flows x in D is feasible flows $x' \in D'$ where cost of $x' = -x$

Ford-Fulkerson Algorithm

See the following [video](#)

4. P, NP and Cook's theorem

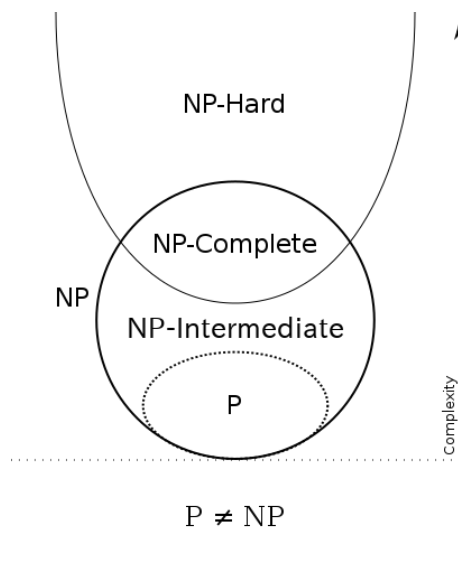
Disposition

- Decision Problems
 - Modelling inputs
 - Modelling boolean functions
 - Modelling optimization problems
- P, NP, NPC, NPH
 - Draw graph
- Cook's theorem
 - $\text{CSAT} \leq \text{SAT}$

NP completeness teori

- Model definition:
 - Operates on bits and bites
 - Input size is equal to number of bits in input
 - Time complexity of an algorithm is number of bit operations done.
- Decision problems, yes-no answers from input
- Inputs are bits
- Pair function:
 - $\langle x, y \rangle = x_1 0 \cdots x_n 0 \quad 11 \quad y_1 0 \cdots y_n 0$
- Church-Turing Thesis:
 - **Alanzo Church:** *No computational procedure will be considered as an algorithm unless it can be represented as a Turing Machine.*
 - **Polynomial Church-Turing thesis:** A decision problem can be solved in polynomial time **iff** it can be solved in polynomial time by a turing machine.

Language Complexity



- **P:** All decision problems that can be solved by a deterministic Turing machine in polynomial time.
- **NP:** All decision problems, where the solutions that evaluates to “yes,” can be verified in polynomial time.
- **NP-Hard:** All Problems that NP problems can be reduced to.
- **NPC:** $NP \cap NPH$
- **NPI:** $NP - NPC - NPH$
- $L_1 \leq L_2 \Leftrightarrow (x \in L_1 \Leftrightarrow r(x) \in L_2)$

SAT & CSAT

- For every boolean function $f(x)$, an equivalent circuit $C(x)$ exists.
 - **Lemma:** For every boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}^m$, $\exists C$ s.t. $\forall x \in \{0, 1\}^n, C(x) = f(x)$
- **Literals:**
 - **Positive Literal:** An atom x
 - **Negative Literal:** A negation of an atom $\neg x$
- **Clause:** Collection of literals and logical connectives.
- **CNF:** Conjunctive Normal Form is a conjunction (AND's) of clauses.
- **DNF:** Disjunctive Normal Form is a disjunction (OR's) of clauses.
- **SAT:** Can the variables of a given CNF be replaced by either TRUE or FALSE such that the CNF evaluates to TRUE
- **Cook's Theorem:** $SAT \in NPC$

5. NP-Complete Problems

Disposition

- P, NP, NPC, NPH
- CSAT
- 3SAT
- Clique

6. Approximation Algorithms and Search Heuristics

Disposition

- P, NP, NPC, NPH
- What is Approximation Algorithms?
- Max-Cut Deterministic
- Max-Cut Randomized

Deterministic Max-Cut Example

Given a graph $G = (V, E)$

Algorithm 1: Deterministic Max-Cut Algorithm

```
1  $S := \emptyset, T := \emptyset$ 
2 for  $v \in V$  do
3   if  $w(\{v\}, S) > w(\{v\}, T)$  then
4      $T := T \cup \{v\}$ 
5   else
6      $S := S \cup \{v\}$ 
7   end
8 end
9 return  $(S, T)$ 
```

Most optimal case is where all edges cross S and T , in short, the sum of all weights in V :

$$OPT = w(V)$$

Now to derive ρ for the deterministic algorithm:

$$\begin{aligned} w(S, T) &\geq w(S, S) + w(T, T) \\ w(S, T) + w(S, T) &\geq w(S, T) + w(S, S) + w(T, T) \\ 2w(S, T) &\geq w(V) \\ w(S, T) &\geq \frac{w(V)}{2} \end{aligned}$$

We chose our C to be the worst case scenario that our algorithm can come up with:

$$C = \frac{w(V)}{2}$$

Finding approximation ratio:

$$\rho = \frac{OPT}{C} = \frac{w(V)}{\frac{w(V)}{2}} = 2 \cdot \frac{w(V)}{w(V)} = 2$$

Randomized Max-Cut Example

Given a graph $G = (V, E)$

Algorithm 2: Randomized Max-Cut Algorithm

```

1  $S := \emptyset, T := \emptyset$ 
2 for  $v \in V$  do
3   Let  $b \in_R \{0, 1\}$ 
4   if  $b = 1$  then
5      $T := T \cup \{v\}$ 
6   else
7      $S := S \cup \{v\}$ 
8   end
9 end
10 return  $(S, T)$ 

```

Optimal solution same as in the deterministic:

$$OPT = w(V)$$

The probability that an edge will connect S and T :

$$P(w_{(i,j) \in (S,T)}) = \frac{1}{2}$$

$$E[|E_{\in (S,T)}|] = |E_{\in G}| P(w_{(i,j) \in (S,T)})$$

$$E[|E_{\in (S,T)}|] = \frac{|E_{\in G}|}{2}$$

The expected value of a randomly chosen edge:

$$E[e \in E] = \frac{w(V)}{|E_{\in G}|}$$

To find C :

$$C = E[\mathbf{RAN}]$$

$$C = E[e \in E] \cdot E[|E_{\in (S,T)}|]$$

$$C = \frac{w(V)}{|E|} \cdot \frac{|E|}{2}$$

$$C = \frac{w(V)}{2}$$

To find ρ :

$$\rho = \frac{OPT}{C} = \frac{OPT}{E[\mathbf{RAN}]} = \frac{w(V)}{\frac{w(V)}{2}} = 2 \cdot \frac{w(V)}{w(V)} = 2$$

Appendix

CSAT gates to CNF proofs

NOT

$$\begin{aligned} z &\leftrightarrow \neg x \\ (\bar{z} + \bar{x})(z + \bar{x}) \\ (\bar{x} + \bar{z})(x + z) \end{aligned}$$

COPY

$$\begin{aligned} z &\leftrightarrow x \\ (\bar{z} + x)(z + \bar{x}) \\ (x + \bar{z})(\bar{x} + z) \end{aligned}$$

AND

$$\begin{aligned} z &\leftrightarrow xy \\ (z + \overline{(x \cdot y)})(\bar{z} + xy) \\ (z + \bar{x} + \bar{y})(\bar{z} + xy) \\ (z + \bar{x} + \bar{y})(\bar{z} + x)(\bar{z} + y) \end{aligned}$$

OR

$$\begin{aligned} z &\leftrightarrow xy \\ (z + \overline{(x + y)})(\bar{z} + (x + y)) \\ (z + (\bar{x} \cdot \bar{y}))(\bar{z} + x + y) \\ (\bar{x} + z)(\bar{y} + z)(x + y + \bar{z}) \end{aligned}$$

XOR

$$\begin{aligned} z &\leftrightarrow x \oplus y \\ z &\leftrightarrow (\bar{x} + \bar{y})(x + y) \\ (\bar{z} + (\bar{x} + \bar{y})(x + y)) &\cdot (z + \overline{(\bar{x} + \bar{y})(x + y)}) \end{aligned}$$

We start with the left side:

$$\begin{aligned} \bar{z} + ((\bar{x} + \bar{y})(x + y)) \\ (\bar{x} + \bar{y} + \bar{z})(x + y + \bar{z}) \end{aligned}$$

Then the right:

$$\begin{aligned}
& z + (\overline{(x + y)} + \overline{(x + y)}) \\
& z + ((xy) + (\overline{xy})) \\
& z + (((xy) + \overline{x}) \cdot ((xy) + \overline{y})) \\
& z + ((x + \overline{x})(y + \overline{x})(x + \overline{y})(y + \overline{y})) \\
& z + (1 \cdot (y + \overline{x})(x + \overline{y}) \cdot 1) \\
& z + ((\overline{x} + y)(x + \overline{y})) \\
& ((\overline{x} + y + z)(x + \overline{y} + z))
\end{aligned}$$

Finally giving us:

$$(\overline{x} + \overline{y} + \overline{z})(x + y + \overline{z})(\overline{x} + y + z)(x + \overline{y} + z)$$

EQ

$$\begin{aligned}
& z \leftrightarrow x \odot y \\
& z \leftrightarrow (x + \overline{y})(\overline{x} + y)
\end{aligned}$$

We start with the left side:

$$\begin{aligned}
& \overline{z} + ((x + \overline{y})(\overline{x} + y)) \\
& (x + \overline{y} + \overline{z})(\overline{x} + y + \overline{z})
\end{aligned}$$

Then the right:

$$\begin{aligned}
& z + (\overline{(x + y)} + \overline{(x + y)}) \\
& z + ((x\overline{y}) + (\overline{x}y)) \\
& z + (((x\overline{y}) + \overline{x}) \cdot ((x\overline{y}) + y)) \\
& z + ((x + \overline{x})(\overline{y} + \overline{x})(x + y)(\overline{y} + y)) \\
& z + (1 \cdot (\overline{y} + \overline{x})(x + y) \cdot 1) \\
& z + ((\overline{x} + \overline{y})(x + y)) \\
& (\overline{x} + \overline{y} + z)(x + y + z)
\end{aligned}$$

Leaving us with:

$$(x + \overline{y} + \overline{z})(\overline{x} + y + \overline{z})(\overline{x} + \overline{y} + z)(x + y + z)$$