Optimization Cheat Sheet

BAX 443 Analytic Decision Making

Hao Hao
Lynn Hong
Joey Li
Zhexuan Meng
Yuanyuan Ge

Table of Contents

1. Overview

Optimization refers to the minimization (or maximization) of a given objective function of several decision variables that satisfy functional constraints. A typical optimization model addresses the allocation of scarce resources among possible alternative uses in order to maximize an objective function such as total profit.

1.1 Three Essential Elements

1. **Decision variables**
   - All decisions are continuous: Linear Programming
   - All decisions are discrete: Integer Programming
   - Some decisions are continuous while others are discrete: Mixed-Integer Programming
2. **Objective function**
   - Problems with no objective functions: Feasibility Problems.
   - Problems with multiple objective functions: often addressed by reducing them to a single-objective optimization problem or a sequence of such problems.
3. **Constraints**
   - Problems with constraints: Constrained Optimization Problems
   - Problems that lack constraints: Unconstrained Optimization Problems

1.2 General Formula

Maximize	$f(x, \theta)$
Subject to	$g(x, \theta)$

$f(.)$: Objective Function

$g(.)$: Constraints

x: Decision variables

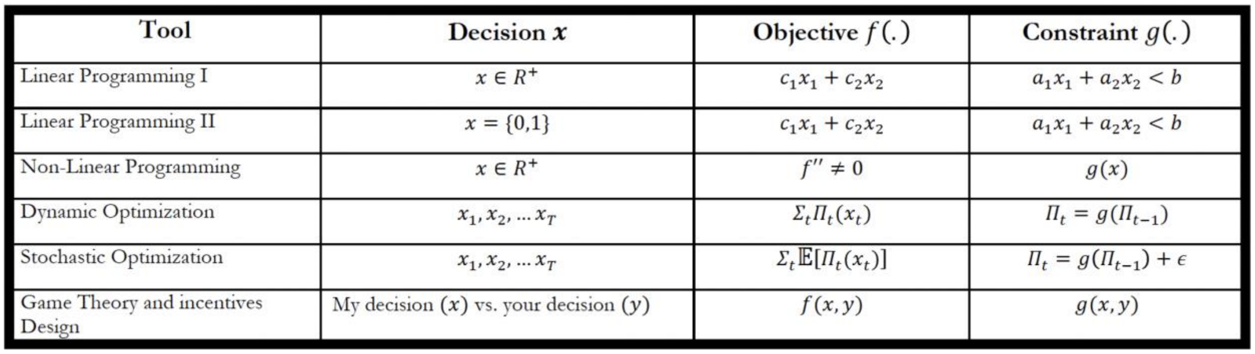
θ: Parameters

Problem Statement: What is x^* that gives the highest $f(.)$ given $g(.)$?

1.3 Types of Feasible Region

- **Infeasible problem:** No set of values for the choice variables satisfies all the constraints.
- **Feasible problem:** At least one set of values for the choice variables satisfy all the constraints.
  - **Unbounded problem:** The objective function can be made to be better than any given finite value.

1.4 Optimization Problems



Tool	Decision $\boldsymbol{x}$	Objective $f(.)$	Constraint $g(.)$
Linear Programming I	$x \in R^+$	$c_1x_1 + c_2x_2$	$a_1x_1 + a_2x_2 < b$
Linear Programming II	$x = \{0,1\}$	$c_1x_1 + c_2x_2$	$a_1x_1 + a_2x_2 < b$
Non-Linear Programming	$x \in R^+$	$f'' \neq 0$	$g(x)$
Dynamic Optimization	$x_1, x_2, \dots x_T$	$\Sigma_t \Pi_t(x_t)$	$\Pi_t = g(\Pi_{t-1})$
Stochastic Optimization	$x_1, x_2, \dots x_T$	$\Sigma_t \mathbb{E}[\Pi_t(x_t)]$	$\Pi_t = g(\Pi_{t-1}) + \epsilon$
Game Theory and incentives Design	My decision (x) vs. your decision (y)	$f(x,y)$	$g(x,y)$

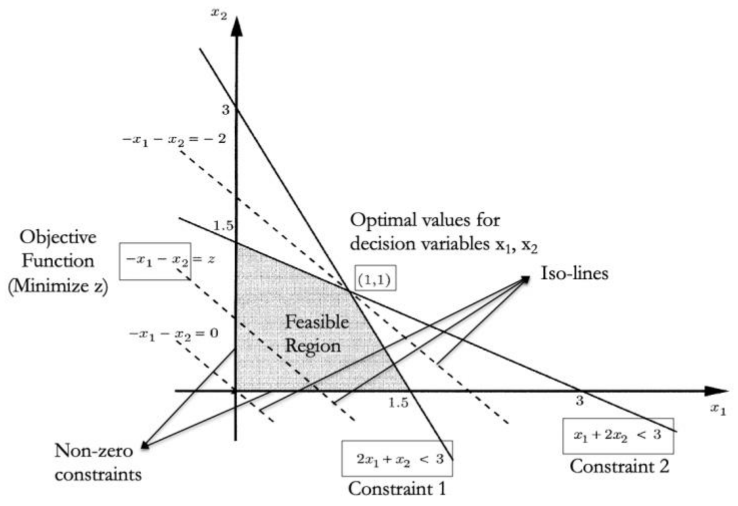
1.5 Convex Optimization

An optimization problem is convex when both the objective function and constraints are convex. It is a framework where differentiability is not necessary. It includes: continuous and discrete decision variables, static and dynamic optimization, deterministic and stochastic optimization

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y)$$

2 Linear Programming (LP)

A method to achieve the best outcome (such as maximum profit or lowest cost) in a mathematical model whose requirements are represented by linear relationships. For example, visually:



2.1 Assumptions

- **Linearity:** The constraints and objective function are linear.
  - **Proportionality:** The value of the objective function and the response of each resource expressed by the constraints is proportional to the level of each activity expressed in the variables.
  - **Additivity:** The effects of the value of each variable on the values of the objective function and the constraints are additive. In other words, there can be no interactions between the effects of different activities; i.e., the level of activity X_1 should not affect the costs or benefits associated with the level of activity X_2.
- **Divisibility:** The values of decision variables can be fractions. Sometimes these values only make sense if they are integers; then we need an extension of linear programming called integer programming.
- **Certainty:** The model assumes that the responses to the values of the variables are exactly equal to the responses represented by the coefficients.

2.2 Duality

Primal vs. Dual

Maximize	$c^T x$		Minimize	$b^T y$
Subject to	$Ax \leq b$		Subject to	$A^T y \geq c$
	$x \geq 0$			$y \geq 0$

Primal *Dual*

e.g. y in the dual is the shadow price for b in the primal. i.e. one unit increase in b will lead to y unit increase in the objective function of the primal.

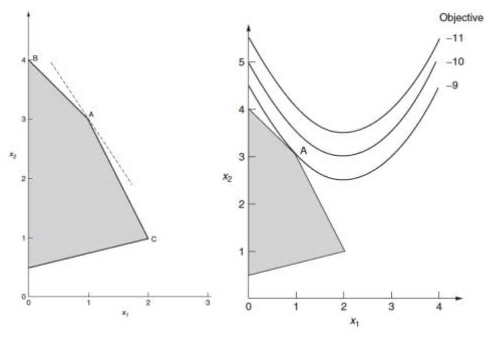
Duality Theorems

Weak Duality Theorem: Let x be any feasible solution to the primal LP and y be any feasible solution to the dual LP. Then $c^T x \leq b^T y$.

Strong Duality Theorem: If the primal (dual) problem has an optimal solution x (y), then the dual (primal) has an optimal solution y (x) such that $c^T x = b^T y$.

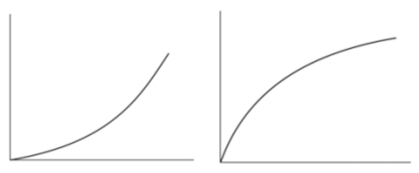
3. Nonlinear Programming (NLP)

Some of the constraints or the objective function are nonlinear. For example, visually:



3.1 Applicability

- **Economies of scale/Non-constant marginal returns**: Costs or profits do not grow linearly with the corresponding activities. (e.g. convex or concave)



- **Probabilistic elements:** Some of the coefficients in the model are random variables.
- **Various connectivities:** Having discontinuities.

3.2 Optimality Conditions: Unconstrained Problems

One Variable

Let $f(x)$ be a differentiable function, if x^* is a maximum, then $f'(x^*) = 0$ and $f''(x^*) < 0$.

Let $f(x)$ be a differentiable function, if x^* is a minimum, then $f'(x^*) = 0$ and $f''(x^*) > 0$.

Multivariate Case

Let $f: \mathbb{R}^n \to \mathbb{R}$ be a differentiable function, if x^* is a maximum, then $\nabla f(x^*) = 0$ and the Hessian of f is negative semi-definite

- When $n = 2$, i.e., $f(x_1, x_2)$, the Hessian is $H = \begin{pmatrix} f_{x_1x_1} & f_{x_1x_2} \\ f_{x_2x_1} & f_{x_2x_2} \end{pmatrix}$

  x^* is a maximum when $f_{x_1x_1} < 0$ and $f_{x_1x_1}f_{x_2x_2} - f_{x_1x_2}f_{x_2x_1} > 0$

- When $n > 2$, i.e., $f(x_1, \ldots, x_n)$, the Hessian is $H = \begin{pmatrix} f_{x_1x_1} & \cdots & f_{x_1x_n} \\ \vdots & \ddots & \vdots \\ f_{x_nx_1} & \cdots & f_{x_nx_n} \end{pmatrix}$

  x^* is a maximum when eigenvalues of the Hessian are non-positive.

Let $f: \mathbb{R}^n \rightarrow \mathbb{R}$ be a differentiable function, if x^* is a minimum, then $\nabla f(x^*) = 0$ and the Hessian of f is positive semi-definite

4 Quadratic Programming (QP)

QPs are problems of minimizing a quadratic function subject to linear constraints. They are a special class of NLP.

Standard QP problem:

$$\begin{aligned} \text{Minimize} \quad & f(x) = \frac{1}{2}x^T Qx + c^T x \\ \text{Subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

4.1 Duality

$$\begin{aligned} \text{Maximize} \quad & b^T y - \frac{1}{2}x^T Qx \\ \text{Subject to} \quad & A^T y - Qx + s = c \\ & x, s \geq 0 \end{aligned}$$

4.2 Optimality Conditions

Solutions can be tested for optimality using Karush-Kuhn-Tucker conditions just as is done for other nonlinear problems:

- **Condition 1:** primal feasibility:

$$Ax = b$$
$$x \geq 0$$

- **Condition 2:** dual feasibility:

$$A^T y - Qx + s = c$$
$$s \geq 0$$

- **Condition 3:** complementary slackness:

$$For\ each\ i = 1, \ldots, n, we\ have\ x_i s_i = 0$$

4.3 Unconstrained cases

Most straightforward to solve, simply set the derivative (gradient) of the objective function equal to zero and solve, as solving NLP problems above.

4.4 Constrained cases: Interior-point method

Generic Interior Point Algorithm

Write the QP in matrix form:

$$F(x,y,s) = \begin{bmatrix} AT\, y - Qx + s - c \\ Ax - b \\ XSe \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \quad (x, s) \geq 0$$

Set of feasible points:

$$F = \{(x, y, s) : Ax = b, A\, y - Qx + s = c, x \geq 0, s \geq 0\}$$

Strictly feasible solution:

$$F = \{(x,y,s) : Ax = b, A\ y - Qx + s = c, x > 0, s > 0\}$$

Choose $(x^0, y^0, s^0) \in F^o$. For $k = 0,1,2,\ldots$ repeat the following steps:

1. *Choose* $\sigma^k \in [0,1]$, *let* $\mu^k = \frac{(x^k)^T s^k}{n}$. *Solve*

$$\begin{bmatrix} -Q & A^T & I \\ A & 0 & 0 \\ S^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta x^k \\ \Delta y^k \\ \Delta s^k \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \sigma^k \mu^k e - X^k S^k e \end{bmatrix}.$$

2. *Choose* α^k *such that*

$$x^k + \alpha^k \Delta x^k > 0, \quad \textit{and } s^k + \alpha^k \Delta s^k > 0.$$

Set

$$(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha_k (\Delta x^k, \Delta y^k, \Delta s^k),$$

and $k = k + 1$.

5 Integer Programming (IP)

Mathematical optimization or feasibility program in which some or all of the variables are restricted to be integers.

The standard IP problem:

$$\begin{aligned} &\text{Minimize} \quad && f(x) = c^T x \\ &\text{Subject to} && Ax + s = b \\ & && s \geq 0 \\ & && x \geq 0 \\ & && x \in Z^n \end{aligned}$$

5.1 Linear Relaxation

Given above IP, the associated LP is called liner relaxation:

$$\begin{aligned} &\text{Minimize} \quad && f(x) = c^T x \\ &\text{Subject to} && Ax + s = b \\ & && s \geq 0 \\ & && x \geq 0 \end{aligned}$$

Since LP is less constrained than IP, following hold:

- **If IP is minimization:** optimal objective function value for LR is less than or equal to that of IP.
- **If IP is maximization:** optimal objective function value for LR is greater than or equal to that of IP.
- **If LP is infeasible:** the so is IP.
- **If LP is optimized by integer variables:** the solution is also optimal for IP.
- **If the objective function coefficients are integer:** for minimization (maximization), the optimal objective for IP is greater than or equal to the 'round up (down)' of the optimal objective for LP.

5.2 Branch and Bound

A recursive approach to solve the LPs.

1. Solve the linear relaxation of the problem. If the solution is integer, the done. Otherwise, create two subproblems by branching on a fractional variable.

2. A subproblem is not active when any of the following occurs:

- You used the subproblem to branch on.
- All variables in the solution is integer.
- The subproblem is infeasible.
- You can fathom the subproblem by a bounding argument.

3. Choose an active subproblem and branch on a fractional variable. Repeat until there are no active subproblems.

5.3 Applicability

Two main reasons for using IPs:

1. Variables quantities can only be integers.

2. Variables represent decisions so that can only be 0 or 1.

- **Production planning**: A possible objective is to maximize the total production, without exceeding the available shared resources.
- **Scheduling:** These problems involve service and vehicle scheduling in transportation networks.
- **Telecommunications networks:** Design a network of lines to install so that a predefined set of communication requirements are met and the total cost of the network is minimal.

6 Dynamic Programming (DP)

Dynamic Programming (DP) refers to simplifying a decision by breaking it down into a sequence of decision steps over time.

6.1 Categories

Depending on the **sequential decision process** (Sequence of decisions to take, with an objective to optimize).

Discrete time/ Continuous time

Discrete time: At each step, one takes a decision, depending on the available information.

Continuous time: The sequence of decisions if replaces by a control, function of time.

Deterministic/ Stochastic

Deterministic decision process: A decision at a particular state uniquely determines the transition state.

Stochastic decision process: Probabilistic events affect the transition state.

6.2 Common elements of DP Problems

1. decision stages(steps)
2. states
3. Decision(control)
4. policy
5. objective function (value function)
6. optimal policy & optimal trajectory

6.3 Bellman's Principle of Optimality

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

6.4 Components of dynamic programming models

Component	Description
State	$\mathbf{s} = (s_1, s_2,..., s_m)$, where s_i is the value of state variable i and m is the number of state variables
Initial state set	$\boldsymbol{I} = \{\mathbf{s}$: nodes in decision network with only leaving arcs$\}$
Final state set	$\boldsymbol{F} = \{\mathbf{s}$: nodes in decision network with only entering arcs$\}$
State space	$\boldsymbol{S} = \{\mathbf{s} : \mathbf{s}$ is feasible$\}$
Decision	$\mathbf{d}(\mathbf{s}) = (d_1, d_2, . . . , d_p)$, where d_j is the value of the jth decision variable and p is the number of decision variables
Feasible decision set	$\boldsymbol{D}(\mathbf{s}) = \{\mathbf{d} : \mathbf{d}$ leads to a feasible state from state $\mathbf{s}\}$
Transition function	$\mathbf{s}' = T(\mathbf{s}, \mathbf{d})$, a function that determines the next state, $\mathbf{s}'$, reached when decision $\mathbf{d}$ is taken from state $\mathbf{s}$
Decision objective	$z(\mathbf{s}, \mathbf{d})$, the measure of effectiveness associated with decision $\mathbf{d}$ taken in state $\mathbf{s}$
Path objective	$z(\mathbf{P})$, the measure of effectiveness defined for path $\mathbf{P}$. This function describes how the objective terms for each state on the path and the final value function are combined to obtain a measure for the entire path.
Final value function	$f(\mathbf{s})$ given for all $\mathbf{s} \in \boldsymbol{F}$

6.5 Resource Allocation Problem

Resource allocation problems can be viewed as generalizations of the investment problem.

6.5.1 Components of Resource Allocation Problem Models

Component	Description
State	$\mathbf{s} = (s_1, s_2,..., s_{m+1})$, where s_1 = alternative currently under consideration s_i = amount of resource i–1 used up prior to the current decision, i = 2,..., m +1
Initial state set	$\boldsymbol{I} = \{(1, 0,...,0)\}$ We start with alternative 1 and no resources used.

Final state set	$\boldsymbol{F} = \{\mathbf{s} : s_1 = n + 1\}$ After all the alternatives have been considered we are finished.
State space	$\boldsymbol{S} = \boldsymbol{I} \cup \{\mathbf{s} : 2 \le s_1 \le n + 1, 0 \le s_i \le b_{i-1}, i = 2,..., m +1\}$ Integrality is also required for all elements of $\boldsymbol{S}$ which consists of the initial state set plus all the integer values within the specified ranges.
Decision	$\mathbf{d}(\mathbf{s}) = (d)$, where d is the investment level for alternative s_1
Feasible decision set	$\boldsymbol{D}(\mathbf{s}) = \{d : 0 \le s_i + a(i–1, s_1, d) \le b_{i-1}, i = 2,..., m +1; d \ge 0$ and integer$\}$ All decisions that do not exceed the resources are feasible.
Transition function	$\mathbf{s}' = T(\mathbf{s}, d)$, where $s'_1 = s_1 + 1$ $s'_i = s_i + a(i–1, s_1, d), i = 2,..., m + 1$ We move to the next alternative with the amount of resources used up by the decisions made previously.
Decision objective	$z(\mathbf{s}, \mathbf{d}) = c(s_1, d)$ The contribution to the objective is determined by the payoff function of the current alternative for the current decision. It does not depend on the successor state $\mathbf{s}'$.
Path objective	Maximize $z(\mathbf{P}) = \sum z(\mathbf{s},\mathbf{d}) + f(\mathbf{s}_f)$ $\mathbf{s}\in\boldsymbol{S},\mathbf{d}\in\boldsymbol{D}(\mathbf{s})$ The total return is the sum of the decision objectives over the opportunities.
Final value function	$f(\mathbf{s}) = 0$ for $\mathbf{s} \in \boldsymbol{F}$ If there is a value for leftover resources we can include it in the final value function. Here we assume the value is 0.

6.5.2 Resource Allocation Problem Examples

- Binary Knapsack Problem
- Personnel Assignment Problem

6.6 Line Partitioning Problems

Dynamic Programming can be effectively applied to the partitioning of a line into non-overlapping segments. Applications include cutting sheet metal and cloth, developing a machine overhaul schedule, and setting up inspection stations along an assembly line.

6.6.1 Components of Line Partitioning Problem Models

Component	**Description**
State	$\mathbf{s} = (s)$, where s is a node on the line that begins or ends a segment.
Initial state set	$\boldsymbol{I} = \{(0)\}$, the process begins at state 0.
Final state set	$\boldsymbol{F} = \{(n)\}$, the process ends at state n.
State space	$\boldsymbol{S} = \{0, 1, 2,\ldots,n\}$, the state space has $n + 1$ elements.
Decision	$\mathbf{d}(s) = (d)$, the decision is the number of intervals to include in the segment starting at s.
Feasible decision set	$\boldsymbol{D}(s) = \{1, 2,\ldots,\mathrm{Min}(m, n - s)\}$ for $s < n$, where m is the maximum number of intervals in a segment. Feasible decisions must remain within the state space.
Transition function	$s' = T(s, d)$, where $s' = s + d$ The new state is the old state plus the number of intervals traversed.
Decision objective	$z(s, d) = c(s, d)$, where the function may be given analytically or in tabular form.
Path objective	Minimize $z(\mathbf{P}) = \sum z(s, d) + f(\mathbf{s}_f)$ $\mathbf{s}\in\boldsymbol{S}$, $\mathbf{d}\in\boldsymbol{D}(\mathbf{s})$
Final value function	$f(\mathbf{s}) = 0$ for all $\mathbf{s} \in \boldsymbol{F}$ For specific problems the final value function may be nonzero.

6.6.2 Examples of Line Partitioning Problems

- Capacity Expansion
- Production Scheduling
- Integer Knapsack Problem(Unbounded Knapsack)

6.7 Path Problems

Find an optimal path through a network using dynamic programming

6.7.1 Components of Path Problem Models

Component	**Description**

State	s = (s1, s2), where
	s1 = x1 -coordinate
	
	s2 = x2 -coordinate

Initial state set	I ={(1,1)}
Final state set	F = {(m, n)}, we generalize the model to allow n rows and m columns.
State space	S = {s : 1 ≤ s1 ≤ m, 1 ≤ s2 ≤ n, s1 and s2 integer}
Decision	d = (d), where d indicates the direction traveled d = 0, go up one node; d = 1, go to right one node
Feasible decision set	D(s) = {0, 1: s1 + d ≤ m and s2 + (1 – d) ≤ n}
Transition function	s ' = T(s , d) s '1 = s 1 + d , s '2 = s 2 + 1 – d
Decision objective	z(s , d) = a(s , d)
Path objective	Minimize z(P) = ∑ z(s, d) + f(sf) s∈S, d∈D(s)
Final value function	f(s) = 0 for s ∈ F

6.7.2 Examples of Path Problems

- Grid Problem
- Grid Problem with Turn Penalties

6.8 Sequencing Problems

Operational problems in manufacturing, service and distribution require the sequencing of various types of activities or items. Examples include a production facility in which chassis must be sequenced through an assembly line, an express mail service where parcels and letters must be routed for delivery, and a utility company that must schedule repair work.

6.8.1 Components of Sequencing Problem Models

Component	**Description**

State	To determine the state definition, consider the information neces- sary to specify the set of feasible decisions and to evaluate the cost associated with a decision. At a particular step in the sequence, a job is a feasible choice if it hasn't been chosen before. Thus the minimal information the state must provide is the set of jobs previ- ously included in the sequence. This also is the information nec- essary to compute the time of completion of the job and hence the associated cost. To describe the state we need a vector with n components s = (s1, s2,..., sn), where sj = 0 if job j has not been included in the sequence sj = 1 if job j has already been included in the sequence
Initial state set	I = {(0, 0,...,0)} None of the jobs has been scheduled.
Final state set	F = {(1, 1,...,1)} All jobs are scheduled.
State space	The state vector is a binary vector with n components. Therefore, there are 2n members of the state space representing all possible combinations. S = {s : sj = 0 or 1, j= 1,..., n}
Decision	The decision vector has a single component that identifies the next job to be processed. d = (d), where d = the next job in the sequence
Feasible decision set	The feasible decisions at a given state are the jobs not already chosen. D(s) = {j : sj = 0, j= 1,..., n}
Transition function	The transition function changes the state to reflect the inclusion of an additional job in the sequence. s' = T(s, d), where s'd = 1 and s'j = sj for j ≠ d
Decision objective	z(s , d) = c(d, t), where t = $\sum_{j=1}^{n} s$j p(j) + p(d) The cost function is problem dependent. For the job sequencing problem with tardiness penalties we use the cost function defined above.

Path objective	Minimize $z(P) = \sum z(s, d) + f(sf) \quad s \in S, d \in D(s)$
Final value function	$f(s) = 0$ for $s \in F$

6.8.2 Examples of Sequencing Problems

- Job Sequencing with Tardiness Penalties
- Traveling Salesman Problem