Optimization Modeling — Part 1/4

First glances

Parin Chaipunya
KMUTT
└ Mathematics @ Faculty of Science
└ The Joint Graduate School of Energy and Environments

Areas of research:

- Multi-agent optimization: Bilevel programs, Game theory
- Optimization modeling: mainly focused on energy and environmental applications

A coordinated course with Metropolitan Electricity Authority (MEA) on STOCHASTIC OPTIMIZATION

by Parin CHAIPUNYA (KMUTT, Thailand) and Michel DE LARA (ENPC, France)

17 Nov – 18 Dec 2025

parin.cha@kmutt.ac.th
parinchaipunya.com
github.com/parinchaipunya

Overview

The main objective of this lecture is to get you to know the **framework and workflow of optimization and modeling**.

The lecture consists of 4 parts:
Part 1: First glances
Part 2: Deterministic models
Part 3: Recalls on prabability theory
Part 4: Basic stochastic models

Table of contents

Section 1

General ideas about optimization

Subsection 1

An optimization dictionary

What is optimization ?

Optimization
(n.) A process of making something as good as it can be.

Improvement
(n.) The act of making something better.

Mathematical optimization
(n.) Using and/or creating mathematical tools to do optimization.

Where is optimization in the map of data analytics?

The four types of data analytics:

- **Descriptive:** What happened in the past ?
  Classical statistical tools, Data visualization, Clustering, etc.
- **Diagnostic:** Why something happened in the past ?
  Data discovery, Data mining, Root cause analysis, etc.
- **Predictive:** What is likely to happen in the future ?
  Regression, Classification, etc.
- **Prescriptive:** How to make something happen ?
  Optimization, Operations reseach, Reinforcement learning (repeated decision), etc.

!! One should notice that Optimization remains the main tool when it comes to prescriptive analytics.

!! Machine learning tools are largely predictive and descriptive in nature.

!! ML tools are also useful to incorporate with Optimization.

Ingredient dictionary of an optimization problem

Ingredient	**Intuition**	**Expression**
Decision variable	What you control.	$u \in \mathcal{U}$
Optimization space	The scope of what you control.	$\mathcal{U}$
Objective function	The quantitative evaluation (outcome) of the decision.	$J : \mathcal{U} \to \mathbb{R} \cup \{+\infty\}$
Constraints	Limitations, represented as a set of admissible decisions.	U^{ad}

Decision variable

It is typical that $\mathcal{U} = \mathbb{R}^n$ (n real variables), which means u is actually a vector

$$u = (u_1, \ldots, u_n).$$

Optimization sense

An objective function J could represent different quantities, *e.g.*

- **badness:** cost, disutility, etc. (The lower, the better.)
- **goodness:** profit, utility, etc. (The higher, the better.)

If J displays the badness, then we are looking at a **minimization problem**

$$\min_{u \in U^{\mathrm{ad}}} J(u).$$

If J displays the goodness, then we are looking at a **maximization problem**

$$\max_{u \in U^{\mathrm{ad}}} J(u).$$

Useful notations

- $\arg\min_{u \in U^{\mathrm{ad}}} J(u) = \{\text{Admissible decisions that minimizes } J(u)\}$
- $\arg\max_{u \in U^{\mathrm{ad}}} J(u) = \{\text{Admissible decisions that maximizes } J(u)\}$

Optimization sense

Here and forward, it is conventional that the min, max, arg min and arg max are taken over U^{ad}, unless stated otherwise.

Observations

- $\min J(u) = -\max\,[-J(u)]$
- $\max J(u) = -\min\,[-J(u)]$
- $\arg\min J(u) = \arg\max\,[-J(u)]$ (arg min is obtained from arg max of negative objective.)
- $\arg\max J(u) = \arg\min\,[-J(u)]$ (arg max is obtained from arg min of negative objective.)

Useful notes

For $\lambda > 0$ and $\beta \in \mathbb{R}$, we have the following.

- $\min[\lambda J(u)] = \lambda \min J(u)$
- $\min[J(u) + \beta] = [\min J(u)] + \beta$
- $\arg\min[\lambda J(u)] = \arg\min J(u)$ (The positive scaling does not affect arg min.)
- $\arg\min[J(u) + \beta] = \arg\min J(u)$ (Shifting up and down do not affect arg min.)

That being said, we shall focus mainly on the theory of minimization.

Constraints

Usually the constraint set U^{ad} is **explicit**, which means it is defined using equations and inequalities

$$U^{\text{ad}} = \left\{ u \;\middle|\; \begin{array}{ll} g_i(u) \le 0 & i = 1, \dots, r \\ h_j(u) = 0 & j = 1, \dots, \ell. \end{array} \right\}$$
$$= \{\text{Decisions satisfying certain equations and inequalities.}\}.$$

Additionally, we may require some slices of u to be integers. In this case, we have $\mathcal{U} = \mathbb{R}^n$ and

$$U^{\text{ad}} = \left\{ u \;\middle|\; \begin{array}{ll} g_i(u) \le 0 & i = 1, \dots, r \\ h_j(u) = 0 & j = 1, \dots, \ell \\ u_k \in \mathbb{Z} & k \in K_0 \end{array} \right\}.$$

Here, $K_0 \subset \{1, \dots, n\}$ represents the set of indices of u_i's that are required to be integers.

General framework

An **optimization problem** (or a **mathematical program**, or a **mathematical programming problem**) takes the following form.

$$
\begin{cases}
\overbrace{\min / \max}^{\text{optimization sense}} & \overbrace{J(u)}^{\text{objective function}} \\
\text{s.t.} & \left.\begin{array}{ll} g_i(u) \le 0 & i = 1, \dots, r \\ h_j(u) = 0 & j = 1, \dots, \ell \\ \underbrace{u \in C}_{\text{possibly further limitations}} & \end{array}\right\} \text{constraints}
\end{cases}
$$

Subsection 2

A tale of a hungry wizard — a toy modeling example

A tale of a hungry wizard

Once upon a time in a fruit market...

- A hungry wizard has a magic sack that carries 1 ton of anything.
- He loves apple and guava, so he wants to fill his magic sack full with these fruits.
- So he goes to a fruit merchant and asks for a combination of apple and guava.
- A ton of apple is sold for 3 gold bars, and a ton of guava for 2 gold bar.
- The merchant has 0.8 ton of each fruit.

Since the merchant knows optimization, he formulates his decision model.

A mathematical tale of a hungry wizard

Decision variable
In the merchant's model, he sets $\mathcal{U} = \mathbb{R}^2$ and

$$u = (\underbrace{u_1}_{\text{sales amount of apple}}, \underbrace{u_2}_{\text{sales amount of guava}}).$$

Objective function
The objective function is an evaluation of his sales performance (income), which is

$$J(u) = J(u_1, u_2) = \underbrace{3u_1}_{\text{apple sales}} + \underbrace{2u_2}_{\text{guava sales}}.$$

Constraints
There are a few constraints.

- The two fruits fill the sack full: $u_1 + u_2 = 1$
- Stock limitations: $u_1 \leq 0.8,\ u_2 \leq 0.8$
- Non-negativity: $u_1 \geq 0,\ u_2 \geq 0$

Subsection 3

What could it be ?

What could it be ? — Manufacturing

$$\begin{cases} \text{min} & \text{Manufacturing cost} \\ \text{s.t.} & \text{Demand constraint} \\ & \text{Manufacturing limit} \\ & \text{Material limit} \end{cases}$$

What could it be ? — Resources

$$\begin{cases} \text{min} & \text{Resources used} \\ \text{s.t.} & \text{Order quantity} \\ & \text{Product quality} \\ & \text{Production formula} \end{cases}$$

What could it be ? — Delivery

$$\begin{cases} \text{min} & \text{Distance travelled} \\ \text{s.t.} & \text{Visits all customers} \\ & \text{No subtour} \end{cases}$$

$$\begin{cases} \text{min} & \text{Travel time} \\ \text{s.t.} & \text{Visits all customers} \\ & \text{No subtour} \end{cases}$$

What could it be ? — Load shifting

$$
\begin{cases}
\min & \text{Bill payment} + \text{discomfort} \\
\text{s.t.} & \text{Demand dynamics} \\
& \text{Shift limits} \\
& \text{Total energy preserved}
\end{cases}
$$

What could it be ? — Electricity production

$$\begin{cases} \text{min} & \text{Production cost} + \text{Startup cost} + \text{Shutdown cost} \\ \text{s.t.} & \text{Generation limits} \\ & \text{Transmission limits} \\ & \text{Demand balance} \\ & \text{Reserve margin constraint} \\ & \text{Emission limit} \\ & \text{Generator ramping constraints} \end{cases}$$

What could it be ? — Transmission expansion planning

$$\begin{cases} \text{min} & \text{Investment cost} + \text{Operation cost} \\ \text{s.t.} & \text{Power flow balance} \\ & \text{Material constraint} \\ & \text{Budget constraint} \\ & \text{Reliability constraint} \end{cases}$$

What could it be ? — Electricity market clearing

$$
\begin{cases}
\max & \text{Social welfare} = \text{Consumer benefit} - \text{Generation cost} \\
\text{s.t.} & \text{Market balance} \\
& \text{Network limits} \\
& \text{Nodal pricing constraints}
\end{cases}
$$

What could it be ? — Storage management

$$
\begin{cases}
\min & \text{Generation cost} + \text{Battery operation cost} \\
\text{s.t.} & \text{Power balance: generation} + \text{discharge} = \text{demand} + \text{charge} \\
& \text{Storage limits} \\
& \text{Storage behaviors}
\end{cases}
$$

What could it be ? — Microgrid optimization

$$\begin{cases} \min & \text{Cost of local generation} + \text{Import/export cost} \\ \text{s.t.} & \text{Power balances} \\ & \text{Renewable constraints} \end{cases}$$

Section 2

Some theory

Subsection 1

Linearity, quadraticity and convexity

Linear programs (LP)

A **linear program: LP** is the problem where all the functions are defined by linear (affine) equations, equalities and inequalities.
A linear program has the form

$$\begin{cases} \min & e_1 u_1 + e_2 u_2 + \cdots + e_n u_n \ (+e_0) \\ \text{s.t.} & a_{i1} u_1 + a_{i2} u_2 + \cdots + a_{in} u_n \leq b_i \quad i = 1, \ldots, r \\ & c_{j1} u_1 + c_{j2} u_2 + \cdots + c_{jn} u_n = d_j \quad j = 1, \ldots, \ell. \end{cases}$$

In vector-matrix form, we write

$$\begin{cases} \min & e^\top u \ (+e_0) \\ \text{s.t.} & Au \leq b \\ & Cu = d. \end{cases}$$

Quadratic programs (QP)

A **quadratic program: QP** is an optimization problem taking the form

$$\begin{cases} \min & u^\top E u + e^\top u + e_0 = \sum_{i,j} e_{ij} u_i u_j + \sum_i e_i u_i + e_0 \\ \text{s.t.} & Au \leq b \\ & Cu = d. \end{cases}$$

Convexity

A set $C \subset \mathbb{R}^n$ is **convex** if we have

$$(1-t)u + tv \in C \quad \text{for any } t \in [0,1] \text{ and any } u, v \in C.$$

Examples

- A line segment is convex.
- A hyperplane (a set defined by linear equalities) is convex.
- A half space (a set defined by linear inequalities) is convex.
- An intersection of convex sets is convex.

Convexity

A function $J : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ is

- **convex** if
  for any $u, v \in \mathbb{R}^n$ and any $t \in [0,1]$,
  $$J((1-t)u + tv) \leq (1-t)J(u) + tJ(v),$$
- **strictly convex** if
  for any $u, v \in \mathbb{R}^n$, $u \neq v$ and any $t \in (0,1)$,
  $$J((1-t)u + tv) < (1-t)J(u) + tJ(v),$$
- **strongly convex** (with modulus $a > 0$) if
  for any $u, v \in \mathbb{R}^n$ and any $t \in [0,1]$,
  $$J((1-t)u + tv) \leq (1-t)J(u) + tJ(v) - \frac{a}{2}t(1-t)\|u-v\|^2.$$

Fact

A strongly convex function is strictly convex, and a strictly convex function is convex.

Convexity

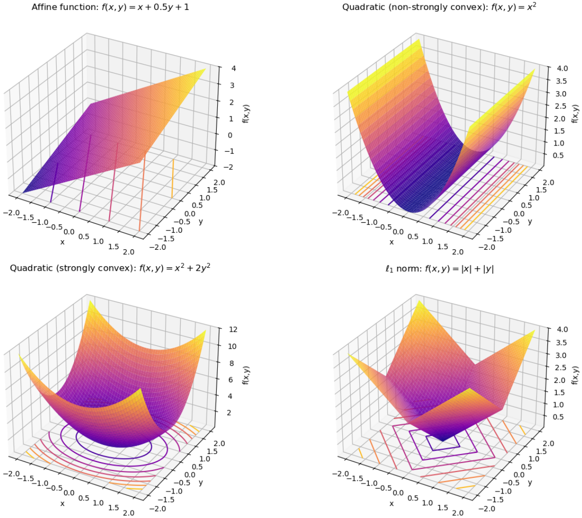


Figure: TL: affine TR: convex quadratic / BL: Strongly convex quadratic BR: Convex

Convexity

Examples (and facts)

- An affine function (a function defined by a linear equation) is convex.
- A quadratic function is convex if E has no negative eigenvalues.
- A quadratic function is strongly convex if E has all positive eigenvalues. In this case $a = \min\{\text{eigenvalues of } E\}$.
- A positive scaling of a convex function is convex.
- A sum of convex functions is convex.
- A sum of convex functions with a strongly convex function is strongly convex.
- A pointwise supremum (maximum) of convex functions is convex.

- The set $\{u \mid g_i(u) \leq 0, \quad i = 1, \ldots, r\}$, where g_i's are convex, is convex.

Convexity with calculus

If J is continuously twice differentiable, then its **Hessian matrix** is defined by

$$H_J(u) = \left[\frac{\partial^2 J}{\partial u_i \partial u_j}(u) \right]_{i,j}.$$

In this case, we have

$$\begin{aligned} J \text{ is convex } &\iff H_J(u) \text{ has nonnegative eigenvalues at every } u \\ J \text{ is strictly convex } &\iff H_J(u) \text{ has positive eigenvalues at every } u \\ J \text{ is strongly convex (with modulus } a > 0) &\iff H_J(u) \text{ has eigenvalues } \geq a \text{ at every } u \end{aligned}$$

Examples

- A quadratic function $J(u) = u^\top E u + e^\top u + e_0$ has a Hessian $H_J(u) = 2E$ at all u.
- An affine function $J(u) = e^\top u + e_0$ has a Hession $H_J(u) = 0$ at all u.

Convex programs

A **convex program** is an optimization problem where J is a convex function and U^{ad} is a convex set.
Usually, U^{ad} is given by linear equalities and convex inequalities

$$U^{\text{ad}} = \left\{ u \;\middle|\; \begin{array}{ll} g_i(u) \le 0 & i = 1, \dots, r \\ Au = b & \end{array} \right\},$$

where g_i's are convex, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Mixed-integer programs

A **mixed-integer program** is an optimization problem where some slices of u are required to be integers.
This is often combined with LPs or QPs, which respectively result in MILPs or MIQPs.

Subsection 2

Local vs global optima

Local vs global optima

Definition

A decision $\bar{u}$ is

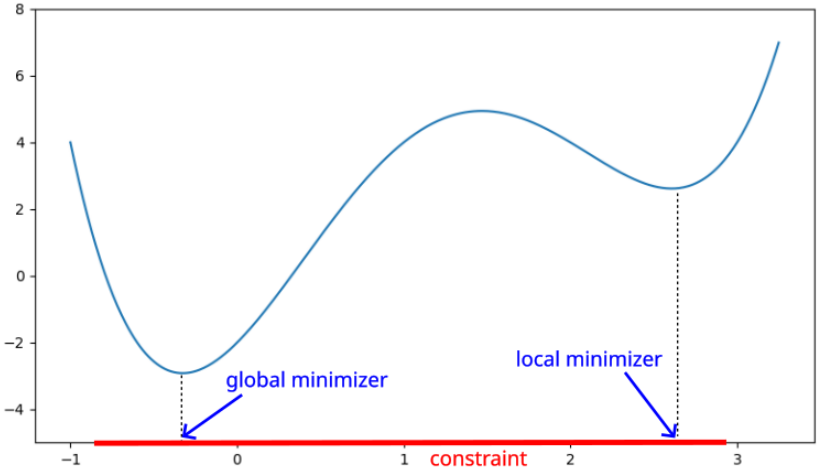
- a **global minimizer** of J if

$$J(\bar{u}) \leq J(u) \quad \text{for any admissible decision } u \in U^{\text{ad}}.$$

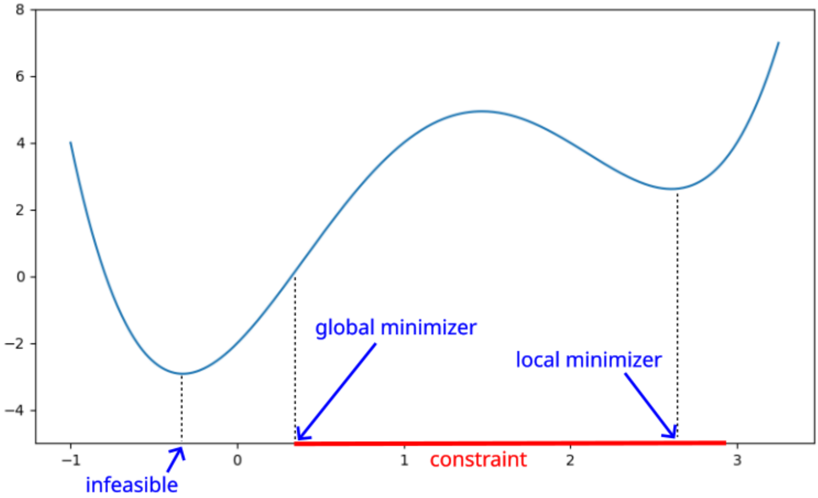
- a **local minimizer** of J if

$$J(\bar{u}) \leq J(u) \quad \text{for any admissible decision } u \in U^{\text{ad}} \text{ near } \bar{u}.$$

Local vs global optima



Local vs global optima



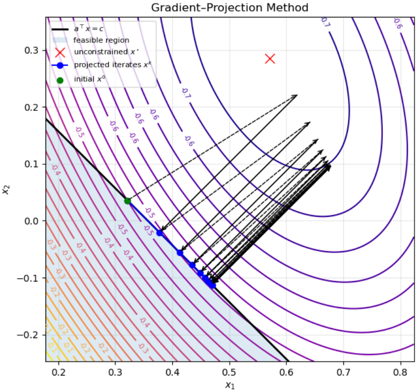
Detecting local optima

Fact

- In a convex program, a local minimizer is a global minimizer. (Local becomes global)
- An explicit convex program has a global minimizer if U^{ad} is bounded. (Existence)
- A strongly convex function has exactly one global minimizer. (Existence and uniqueness)

Follow the slopes — Gradient-projection algorithm

A useful way to find a minimizer is to follow the slopes, using the gradient-projection algorithm.



Gradient-projection algorithm.

Input. Initial guess $u^{(0)}$, step length $\rho > 0$, tolerance $\varepsilon > 0$.
Output. Optimal decision $\bar{u}$.
Repeat
| $\quad u^{(k+1)} \leftarrow \mathrm{proj}_{U^{\mathrm{ad}}}[u^{(k)} - \rho \nabla J(u^{(k)})]$
until $\|\nabla J(u^{(k)})\| < \varepsilon$.

Subsection 3

Optimization with only equality constraints

Optimization with only equality constraints

We consider here the following optimization problem

$$\begin{cases} \min & J(u) \\ \text{s.t.} & h_j(u) = 0 \quad j = 1, \dots, \ell \end{cases}$$

with J and h_j's being differentiable.
Thus we may take advantage of the gradients

$$\nabla J(u) = \left(\frac{\partial J}{\partial u_i}(u) \right)_i \quad \text{and} \quad \nabla h_j(u) = \left(\frac{\partial h_j}{\partial u_i}(u) \right)_i$$

Lagrange's approach

If we define a Lagrange function $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$ by

$$L(u, \lambda) = \underbrace{J(u)}_{\text{original cost}} + \underbrace{\sum_{j=1}^{\ell} \lambda_j h_j(u)}_{\text{weighted penalty}} ,$$

then we have the gradient

$$\nabla L(u, \lambda) = (\nabla_u L(u, \lambda), \nabla_\lambda L(u, \lambda))$$

where

$$\nabla_u L(u, \lambda) = \nabla J(u) + \sum_{j=1}^{\ell} \lambda_j \nabla h_j(u)$$

$$\nabla_\lambda L(u, \lambda) = (h_j(u))_j .$$

Necessary condition (Sieve)

Theorem

Let $\bar{u} \in \mathbb{R}^n$. We suppose that the constraints are qualified*at $\bar{u}$.
Then a necessary condition for $\bar{u}$ to be a local minimizer of J over U^{ad} is:
there exist $\bar{\lambda}_j$'s (called **Lagrange multipliers**) in which

$$\nabla L(\bar{u}, \bar{\lambda}) = 0. \tag{1}$$

Note that (1) is exactly the same as

$$\nabla J(\bar{u}) + \sum_{j=1}^{\ell} \bar{\lambda}_j \nabla h_j(\bar{u}) = 0, \quad h_j(\bar{u}) = 0 \quad \forall j = 1, \dots, \ell.$$

*For example, either (1) all h_j's are linear, or (2) the gradients of $h_j(\bar{u})$'s are linearly independent, or (3) there is only one constraint.

Sufficient condition

Theorem

Let $\bar{u} \in \mathbb{R}^n$. We suppose that the objective function is convex and the constraints are linear. Then a sufficient condition for $\bar{u}$ to be a global minimizer of J over U^{ad} is:

there exist $\bar{\lambda}_j$'s (called **Lagrange multipliers**) in which

$$\nabla L(\bar{u}, \bar{\lambda}) = 0.$$

Section 3

Solvers and packages

Some off-the-shelf solvers

Solver	Comments
`CPLEX`	Commercial. Free for small problems. Fully free for academics.
`Gurobi`	Commercial. Free for small problems. Fully free for academics.
`GAMS`	Commercial. Free trial. Fully free for academics.
`SCIP`†	Free and open-source. Also works with nonlinear problems.
`HiGHS`	Free and open-source. Only for LPs.
`CBC`	Free and open-source.

†We shall use `SCIP` in our demo problems.

Some off-the-shelf packages

Package	Language	Description
`PySCIPOpt`	Python	Interface to SCIP (MIP, MINLP)
`Pyomo`	Python	General modeling language; supports many solvers
`PuLP`	Python	Lightweight LP/MIP modeling
`CVXPY`	Python	Convex optimization modeling
`JuMP`	Julia	High-performance modeling for LP/QP/MIP/NLP

The classes of optimization problems

$$\begin{array}{ccccc} & & \text{Convex Quadratic} & & \text{Nonconvex Quadratic} \\ & & \cap & & \cap \\ \text{Linear} & \subset & \text{Convex} & \subset & \text{Nonlinear} \end{array}$$

Large mixed-integer problems are known to be slow to solve exactly. Hence we might have to rely on heuristic methods to *improve* the outcome and hope it is close enough to the minimizer.

Getting started with SCIP

Let's see a demo of how to use SCIP through PySCIPOpt.

-» Continue to **Part 2**.