ML in the Service of COP: From Prediction to Prescription

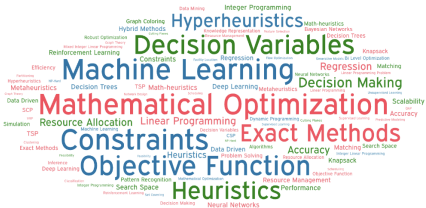
A Hands-On Workshop with Prophet & Gurobi

Dr. Mahdi Khemakhem

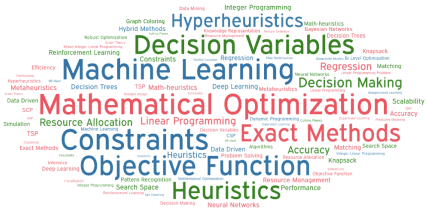
Department of Computer Science
Prince Sattam Bin Abdulaziz University

February 25, 2025

Outline

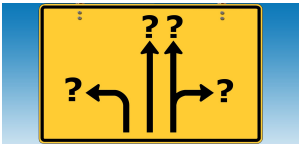


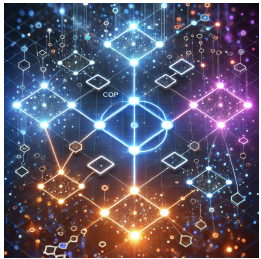
Outline

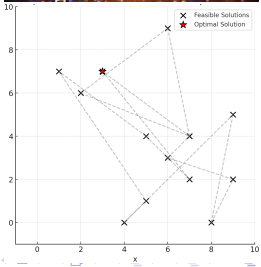


COP Definition

- A combinatorial optimization problem (COP) is a computational problem that involves finding the best solution from a set of feasible solutions (generally extensive).
- The goal is to either maximize or minimize a certain objective function, subject to a set of constraints.







COP Requirements and Components

An COP definition requires a prior definitions of:

1. A set of decision variable(s).
2. An objective function using defined decision variable(s) to precise how good solution is.
3. A set of constraints(s) using defined decision variable(s)to determine the feasible region containing all COP's possible solutions.

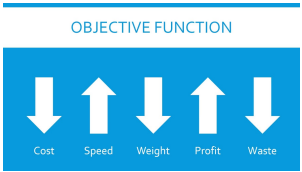
$\Rightarrow$ Solving a COP consists to finding the better solution(s) in the feasible region according to the defined objective function and the constraint(s) set.

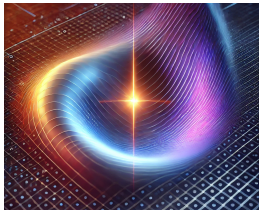
Decision Variables?

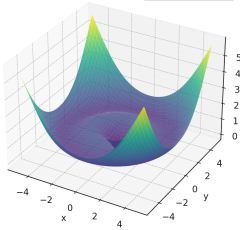
- A decision variable is an unknown-value variable associated to an COP.
- A decision variable has a type and a domain.
- Decision variables are a set of quantities to be determined for solving the COP.
  $\Rightarrow$ The OP is solved when the best values of the decision variables have been identified.
- They are called decision variables because the problem is to decide what value each variable should take.

Objective Function?

- An objective function (OF) is a function associated with an COP which determines how good a solution is.
- An OF can be Minimized or Maximized.
- An OF should be defined using decision variables.

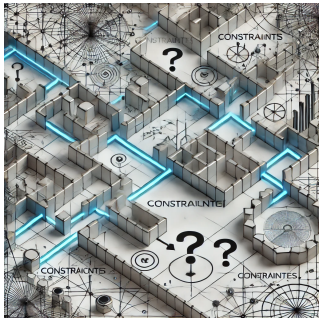






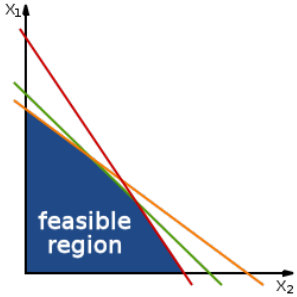
Constraints?

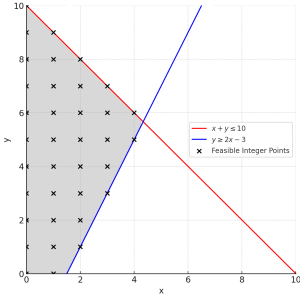
- Constraints are logical conditions that a solution to an COP must satisfy.
- Constraints reflect real-world limits on *production capacity*, *market demand*, *available funds*, and so on.
- Constraints should be defined using decision variables.

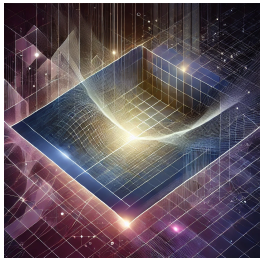


Feasible Region?

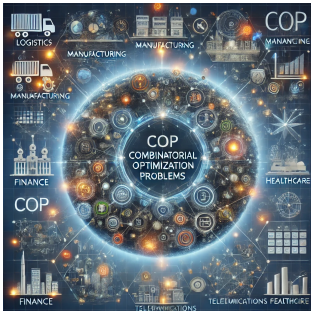
- A feasible region is the set of all possible solutions of an COP.
- In mathematical optimization, a feasible region, feasible set, search space, or solution space is the set of all possible points (sets of values of the choice variables) of an COP that satisfy the problem's constraints.







COP Across Industries



Supply Chain
Facility locations, truck routing, inventory placement

Manufacturing
Production planning and scheduling, workforce planning

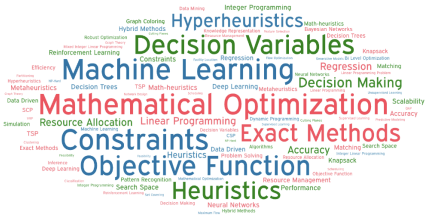
Airlines
Crew scheduling, flight to aircraft assignment

Energy
Balancing supply and demand, optimal energy resource planning to meet CO_2 emission targets

Finance
Investment portfolio optimization, cash allocation and management

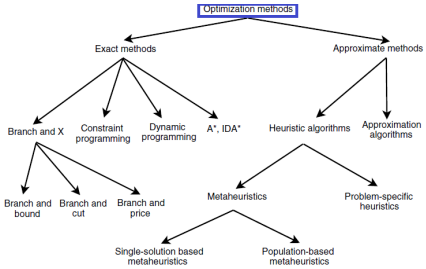
Sales & Marketing
Marketing campaign optimization, sales territory allocation

Outline



Optimization Methods

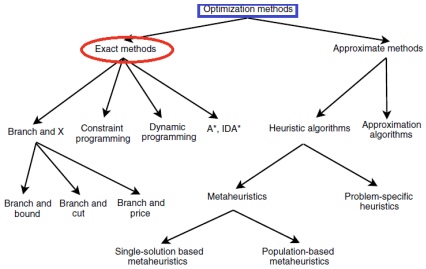
- Following the complexity of the optimization problem, it may be solved by an exact method or an approximate method.



[a] In the artificial intelligence community, those algorithms are also named complete algorithms.

Optimization Methods

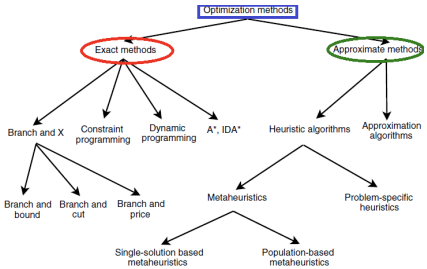
- Following the complexity of the optimization problem, it may be solved by an exact method or an approximate method.
- Exact methods[a] obtain optimal solutions and guarantee their optimality.



[a] In the artificial intelligence community, those algorithms are also named complete algorithms.

Optimization Methods

- Following the complexity of the optimization problem, it may be solved by an exact method or an approximate method.
- Exact methods[a] obtain optimal solutions and guarantee their optimality.
- Approximate (or heuristic) methods generate high-quality solutions in a reasonable time for practical use, but there is no guarantee of finding a global optimal solution.



[a] In the artificial intelligence community, those algorithms are also named complete algorithms.

Exact Methods

In the class of exact methods we can find the following classical algorithms:

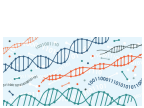
- Dynamic programming.
- Branch and Bound.
- Constraint programming.
- ...

$\Rightarrow$ Exact methods can be applied to small instances of difficult problems.

Heuristics

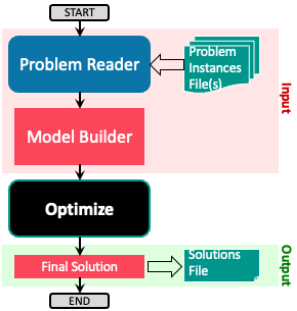
Heuristics find "good" solutions on large-size problem instances.

- They allow to obtain acceptable performance at acceptable costs in a wide range of problems.
- In general, heuristics do not have an approximation guarantee on the obtained solutions.
- They may be classified into many families: Heuristics, Metaheuristics, Matheuristics, Hyperheuristics.
- Examples: Local Search (LS), Variable Neighborhood Search (VNS), Tabu search (TS), Simulated Annealing (SA), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), etc.



Macroscopic View of Optimization Methods

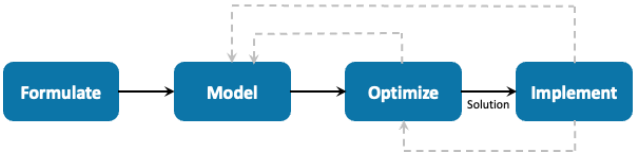
- For now, we consider that an Optimization Method as a black box!
- Since an Optimization Method is an algorithm so it should have Input and Output.
- Two modules[a] are responsible to build the Input:
  - Problem reader: a module that reads the problem instance (example) then provides the problem parameters stored in some data structures.
  - Model builder: a module that, from the problem parameters, build a model considered as a starting point to the Optimization Method.
- Starting from a model, the Optimization Method provides a final solution (optimal or near-optimal) considered as an Output to be stored in a solution file.



[a] functions, methods, procedures, etc.

Decision Making Steps in Optimization

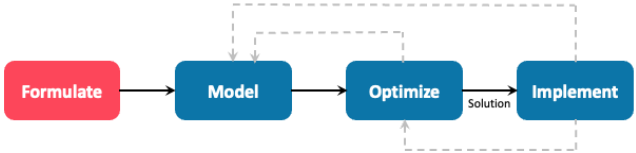
- As scientists, engineers, and managers, we always have to take decisions.
- Decision making is everywhere.
- As the world becomes more and more complex and competitive, decision making must be tackled in a rational and optimal way.
- Decision making consists in the following steps:



The classical process in decision making: formulate, model, solve, and implement.
⇒In practice, this process may be iterated to improve the optimization model or algorithm until an acceptable solution is found.

Decision Making Steps in Optimization

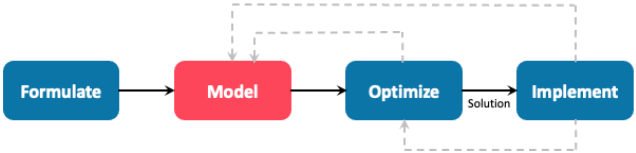
- As scientists, engineers, and managers, we always have to take decisions.
- Decision making is everywhere.
- As the world becomes more and more complex and competitive, decision making must be tackled in a rational and optimal way.
- Decision making consists in the following steps:



The classical process in decision making: formulate, model, solve, and implement.
⇒In practice, this process may be iterated to improve the optimization model or algorithm until an acceptable solution is found.

Decision Making Steps in Optimization

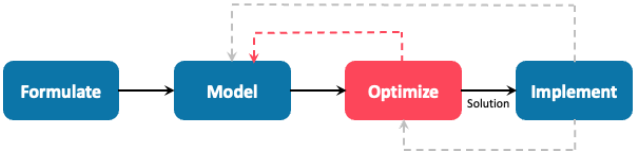
- As scientists, engineers, and managers, we always have to take decisions.
- Decision making is everywhere.
- As the world becomes more and more complex and competitive, decision making must be tackled in a rational and optimal way.
- Decision making consists in the following steps:



The classical process in decision making: formulate, model, solve, and implement.
⇒In practice, this process may be iterated to improve the optimization model or algorithm until an acceptable solution is found.

Decision Making Steps in Optimization

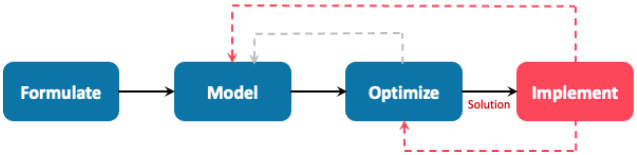
- As scientists, engineers, and managers, we always have to take decisions.
- Decision making is everywhere.
- As the world becomes more and more complex and competitive, decision making must be tackled in a rational and optimal way.
- Decision making consists in the following steps:



The classical process in decision making: formulate, model, solve, and implement.
⇒In practice, this process may be iterated to improve the optimization model or algorithm until an acceptable solution is found.

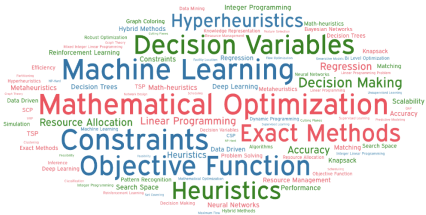
Decision Making Steps in Optimization

- As scientists, engineers, and managers, we always have to take decisions.
- Decision making is everywhere.
- As the world becomes more and more complex and competitive, decision making must be tackled in a rational and optimal way.
- Decision making consists in the following steps:



The classical process in decision making: formulate, model, solve, and implement.
⇒In practice, this process may be iterated to improve the optimization model or algorithm until an acceptable solution is found.

Outline



What is Mathematical Optimization (MO)?

Mathematical Optimization (MO)
solves complex Decision Problems by:

What is Mathematical Optimization (MO)?

Mathematical Optimization (MO) solves complex Decision Problems by:

1. Identifying the best line of action
2. Among *many* possibilities
3. That the decision maker can set
4. Given specified limitations
5. With some particular objective(s) in mind

What is Mathematical Optimization (MO)?

Mathematical Optimization (MO) solves complex Decision Problems by:

1. Identifying the best line of action
2. Among *many* possibilities
3. That the decision maker can set
4. Given specified limitations
5. With some particular objective(s) in mind

- This leads to an Optimization Model, defined by:
  - one objective function
  - n decision variables
  - m constraints

Decision problems to Optimization Models

$$\text{Min (or Max)} Z = c^T x$$
$$\text{subject to (s.t.):}$$
$$Ax \begin{cases} \leq \\ \geq \\ = \end{cases} b$$
$$\forall x_i \in x, x_i \in \mathbb{R}^+ (\text{or } \mathbb{N} \text{ or } \{0,1\})$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}$$

where x is a vector of n continuous (or integer, or mixed) decision variables, and c and b (resp. A) are constant vectors (resp. matrix) of problem parameters.

Example of MO

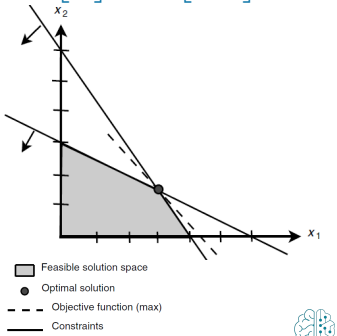
Consider the following small optimization model:

$$\begin{aligned} & MaxZ = 5X_1 + 4X_2 \\ & \text{(s.t.):} \\ & 6X_1 + 4X_2 \leq 24 \\ & 1X_1 + 2X_2 \leq 6 \\ & X_1, X_2 \in \mathbb{R}^+ \end{aligned}$$

- The figure on the right shows the graphical interpretation of the model.
- Each constraint can be represented by a line.
- The objective function is an infinity of parallel lines.
- The optimum solution will always lie at an extreme point.
- The optimal solution is $(X_1 = 3, X_2 = 1.5)$ with a maximum profit of $Z = 21$.

$$x = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}, \; c = \begin{bmatrix} 5 \\ 4 \end{bmatrix},$$

$$b = \begin{bmatrix} 24 \\ 6 \end{bmatrix}, \; A = \begin{bmatrix} 6 & 4 \\ 1 & 2 \end{bmatrix}$$

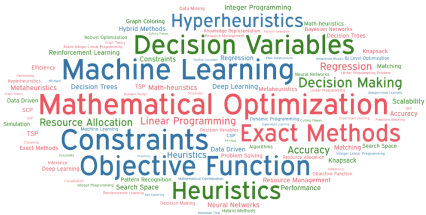


Different Types of MO Models

Mathematical Optimization Models Come in Many Forms:

- Linear Program (LP)
  1. linear objective function
  2. linear constraints
  3. continuous decision variables, $x \in \mathbb{R}^+$
- Integer [Linear] Program (ILP)
  - Same as 1. and 2. above
  - Integer or binary decision variables, $x \in \mathbb{N}$ or $x \in \{0, 1\}$
- Mixed-Integer [Linear] Program (MIP or MILP)
  - Same as 1. and 2. in LP
  - Decision variables are mixture of continuous and integer/binary
- Objective function includes **quadratic terms OR** either the objective function or constraints (or both) involve **nonlinear functions**
  - Quadratic Program (QP), MIQP,...
  - Quadratically Constrained Program (QCP), MIQCP,...
  - Nonlinear Program (NLP), MINLP,...

Outline



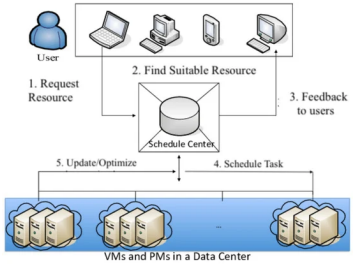
VM Placement Problem in Cloud Data Center

- Cloud computing data centers (a set of physical machines (PMs)) host numerous virtual machines (VMs) that require computing resources such as CPU, memory, and disk storage.
- Efficiently managing these resources while minimizing energy consumption is a crucial challenge.

Objective: Minimize the total energy consumption of the data center.

Constraints:

- Each VM must be assigned to exactly one active PM.
- The total resource usage of VMs on any PM cannot exceed the PM's capacity.
- The number of active PMs should be minimized to reduce energy consumption.
- Binary decision variables indicate whether a VM is allocated to a PM and whether a PM is active.



BLP Model: Parameters and Decision Variables

Parameters: Let:

- I be the set of virtual machines (VMs).
- J be the set of physical machines (PMs).
- $r_i^{(c)}$, $r_i^{(m)}$, and $r_i^{(d)}$ be the CPU, memory, and disk requirements of VM $i \in I$.
- C_j, M_j, and D_j be the CPU, memory, and disk capacity of PM $j \in J$.
- P_j be the power consumption of PM j when active.

Decision Variables: Let:

- $x_{ij} \in \{0, 1\}$ be a binary variable indicating whether VM i is assigned to PM j.
- $y_j \in \{0, 1\}$ be a binary variable indicating whether PM j is active or not.

Objective Function: Minimize the total power consumption:

$$\min \sum_{j \in J} P_j y_j \tag{1}$$

BLP Model: Constraints

Each VM is assigned to one server:

$$\sum_{j \in J} x_{ij} = 1, \quad \forall i \in I \tag{2}$$

CPU capacity constraint:

$$\sum_{i \in I} r_i^{(c)} x_{ij} \leq C_j y_j, \quad \forall j \in J \tag{3}$$

Memory capacity constraint:

$$\sum_{i \in I} r_i^{(m)} x_{ij} \leq M_j y_j, \quad \forall j \in J \tag{4}$$

Disk capacity constraint:

$$\sum_{i \in I} r_i^{(d)} x_{ij} \leq D_j y_j, \quad \forall j \in J \tag{5}$$

Binary decision variables:

$$x_{ij} \in \{0,1\}, \quad y_j \in \{0,1\}, \quad \forall i \in I, j \in J \tag{6}$$

Equations from (1) to (6) define the mathematical model for the cloud resource optimization problem.

Small Example

Parameters: $I = \{1, 2, 3\}$, $J = \{1, 2\}$.

VMs resource requirements			
VM	CPU	Memory	Disk
i=1	$r_1^{(c)}$ =2	$r_1^{(m)}$ =4	$r_1^{(d)}$ =10
i=2	$r_2^{(c)}$ =3	$r_2^{(m)}$ =6	$r_2^{(d)}$ =12
i=3	$r_3^{(c)}$ =1	$r_3^{(m)}$ =2	$r_3^{(d)}$ =5

PMs Capacities				
PM	CPU	Memory	Disk	Power
j =1	C_1 =5	M_1 =10	D_1 =20	P_1 =100
j =2	C_2 =4	M_2 =8	D_2 =15	P_2 =80

Decision Variables: $x_{ij} \in \{0,1\}$, $y_j \in \{0,1\}$, $\forall i \in I, j \in J \implies x_{11}, x_{12}, x_{21}, x_{22}, x_{31}, x_{32}, y_1, y_2 \in \{0,1\}$.
Objective Function: Minimize $P_1 y_1 + P_2 y_2 =$ Minimize $100y_1 + 80y_2$.
Constraints:

- Each VM is assigned to one PM: $x_{11} + x_{12} = 1$ $x_{21} + x_{22} = 1$ $x_{31} + x_{32} = 1$.
- CPU capacity constraint: $2x_{11} + 3x_{21} + x_{31} \leq 5y_1$ $2x_{12} + 3x_{22} + x_{32} \leq 4y_2$.
- Memory capacity constraint: $4x_{11} + 6x_{21} + 2x_{31} \leq 10y_1$ $4x_{12} + 6x_{22} + 2x_{32} \leq 8y_2$.
- Disk capacity constraint: $10x_{11} + 12x_{21} + 5x_{31} \leq 20y_1$ $10x_{12} + 12x_{22} + 5x_{32} \leq 15y_2$.

Solution Approach

Optimal VMs Assignment to PMs in Cloud Data Center

The solution approach of the problem consists of one main component: ⇒Optimization Component to determine the optimal assignment of each VM to a PM.

- Use the Gurobi solver[a][b] to solve the optimization model defined below.
- The optimization component is in the `MO_optimal_VM_to_PM_Assignment.ipynb` notebook.

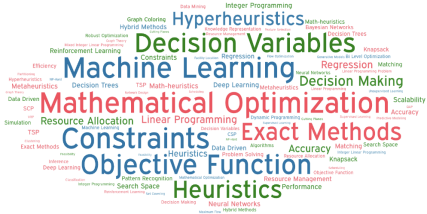
GUROBI OPTIMIZATION

[a] https://www.gurobi.com
[b] To install Gurobi via this link

Hands-on Use Case

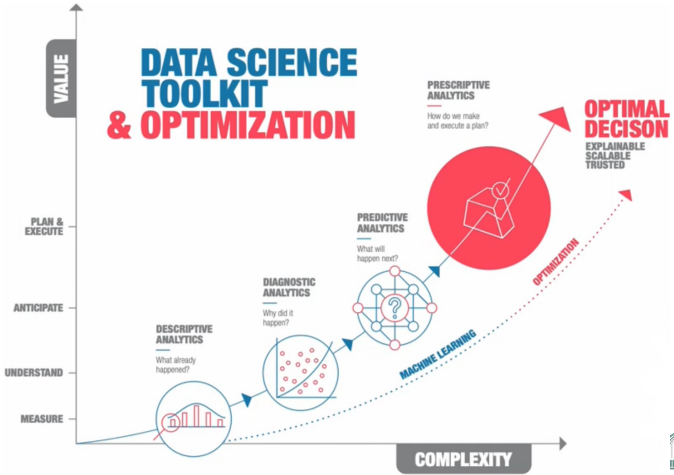
https://drive.google.com/drive/folders/1qe5muZbgviDeauH2EA45U54wW7zcyBHa?usp=drive_link

Outline



Predictive vs Prescriptive

- What questions does MO address?
- How does this differ from Machine Learning (ML)?



How ML and MO Can Coexist

How Do ML and MO Work Together?

1. **MO Enhances ML by Structuring Model Training**

   ML itself is an Optimization Problem, where the goal is to find the **best model parameters** that **minimize a loss function** while **satisfying various constraints**.

   - Minimizing loss functions (e.g., mean squared error, cross-entropy) to improve predictive accuracy.
   - Using constrained optimization for feature selection, reducing model complexity while preserving performance.

Examples:

- Hyperparameter tuning uses optimization techniques (e.g., Bayesian optimization, grid search).
- Training deep learning models requires gradient-based optimization (e.g., Adam, SGD).
- Reinforcement learning uses mathematical programming to optimize policies.

How ML and MO Can Coexist

How Do ML and MO Work Together?

2 ML Supports MO by Providing Data-Driven Parameters

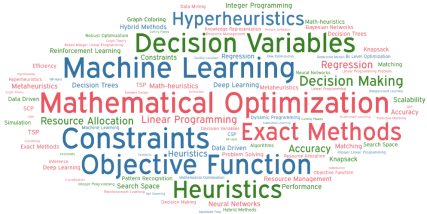
ML models can be used to **estimate key parameters** for **optimization models**, such as cost coefficients, demand forecasts, or resource availability. This enables dynamic and adaptive decision-making based on real-world data.

$$\begin{aligned} &\textbf{Optimization Model:} \quad \min / \max \quad Z = c^T x \\ &\text{Subject to:} \\ &Ax \leq, \geq, \text{ or } = b \\ &\forall x_i \in x, \quad x_i \in \mathbb{R}^+ \quad (\text{or } \mathbb{N}, \text{ or } \{0,1\}) \end{aligned}$$

Examples:

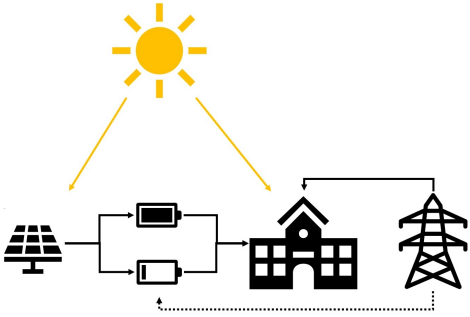
- Predictive models estimate customer demand for supply chain optimization.
- ML-based forecasting helps determine pricing parameters in revenue optimization.
- Deep learning models refine constraint values in scheduling problems.

Outline



Prerequisites

- A university building operates courses over six days (Monday–Saturday), with electricity demand varying by course load.
- The building is equipped with a solar panel and two batteries for energy storage and supply.
- Batteries can also be charged from the grid, and additional electricity can be purchased when needed. At the same time, battery capacity and charge/discharge limits impose constraints, adding complexity to the problem.
- Energy demand includes both the building's baseline requirements and course-specific needs, which vary by class size and resource type (e.g., labs).



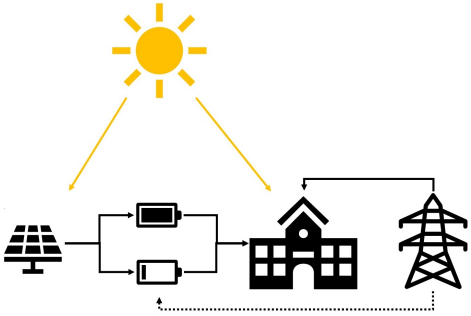
Time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
08.00–09.00						
09.00–10.00	Course 1	Course 8	Course 1	Course 6	Course 1	
10.00–11.00	Course 2	Course 3	Course 2	Course 7	Course 2	
11.00–12.00						
12.00–13.00						
13.00–14.00	Course 3	Course 8	Course 3	Course 8	Course 3	Course 11
14.00–15.00	Course 4		Course 4	Course 9	Course 4	
15.00–16.00	Course 5		Course 5		Course 5	
16.00–17.00						
17.00–18.00						
18.00–19.00			Course 10			
19.00–20.00						

The information used for this example have been adopted from IEEE's Predict+Optimize Technical Challenge.

Problem Statement and Objective

- Given a course schedule for the upcoming week and historical data about the solar potential, the objective is to determine the optimal batteries charging and discharging schedule in order to satisfy the energy demand for a building where the courses are held.
- The goal of the problem is to find the battery charge and discharge schedule that will:
  1. Minimize the total electricity purchased from the grid during the upcoming week.
  2. Since electricity prices fluctuate, minimize the total cost of electricity purchased from the grid.

⇒This example combines ML (time series forecasting) with MO, using solar generation forecasts as key inputs.



Time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
08:00–09:00						
09:00–10:00	Course 1	Course 8	Course 1	Course 6	Course 1	
10:00–11:00	Course 2	Course 3	Course 2	Course 7	Course 2	
11:00–12:00						
12:00–13:00						
13:00–14:00	Course 3	Course 8	Course 3	Course 8	Course 3	Course 11
14:00–15:00	Course 4		Course 4	Course 9	Course 4	
15:00–16:00	Course 5		Course 5		Course 8	
16:00–17:00					Course 5	
17:00–18:00						
18:00–19:00			Course 10			
19:00–20:00						

The information used for this example have been adopted from IEEE's Predict+Optimize Technical Challenge.

Parameters

B	Set of batteries.
T	Set of time periods.
$c_b, \forall b \in B$	Capacity of battery $b \in B$.
$p_b, \forall b \in B$	Loss of energy (as a percentage) during transfer into battery $b \in B$.
$q_b, \forall b \in B$	Quantity of initial energy in battery $b \in B$.
$d_t, \forall t \in T$	Total building and class energy demand for time period $t \in T$.
maxcd	Maximum charging/discharging rate.
$solar_t, \forall t \in T$	Expected Solar power generation of the panel for time period $t \in T$.
$cgrid_t, \forall t \in T$	Expected cost/used unit of grid energy during time period $t \in T$.

Decision Variables

$f^{in}_{b,t} \in \mathbb{R}^+, \forall b \in B, t \in T$	Energy charged into battery $b \in B$ at time period $t \in T$.
$f^{out}_{b,t} \in \mathbb{R}^+, \forall b \in B, t \in T$	Energy discharged from battery $b \in B$ at time period $t \in T$.
$\text{grid}_t \in \mathbb{R}^+, \forall t \in T$	Energy taken from the grid at time period $t \in T$.
$\text{s}_{b,t} \in \mathbb{R}^+, \forall b \in B, t \in T$	Stored energy in battery $b \in B$ at the end of time period $t \in T$.
$\text{gen}_t \in \mathbb{R}^+, \forall t \in T$	Generated solar energy at time period $t \in T$.
$\text{zwitch}_{b,t} \in \{0,1\}, \forall b \in B, t \in T$	Binary switch indicating battery charge/discharge mode.

MILP Model

Minimize $\sum_{t \in T} grid_t$ **OR** $\sum_{t \in T} cgrid_t \times grid_t$ (Minimize grid cost)

Subject to:

$$\sum_{b \in B} (f^{out}_{b,t} - p_b \times f^{in}_{b,t}) + gen_t + grid_t = d_t, \quad \forall t \in T \qquad \text{(Power balance)}$$

$$s_{b,0} = q_b + p_b \times f^{in}_{b,0} - f^{out}_{b,0}, \quad \forall b \in B \qquad \text{(Initial state)}$$

$$s_{b,t} = s_{b,t-1} + p_b \times f^{in}_{b,t} - f^{out}_{b,t}, \quad \forall b \in B, t \geq 1 \qquad \text{(Battery state update)}$$

$$\sum_{b \in B} f^{in}_{b,t} + gen_t \leq solar_t, \quad \forall t \in T \qquad \text{(Solar energy limit)}$$

$$f^{in}_{b,t} \leq \text{maxcd} \times zwitch_{b,t}, \quad \forall b \in B, t \in T \qquad \text{(Charging limit)}$$

$$f^{out}_{b,t} \leq \text{maxcd} \times (1 - zwitch_{b,t}), \quad \forall b \in B, t \in T \qquad \text{(Discharging limit)}$$

$$s_{b,t} \leq c_b, \quad \forall b \in B, t \in T \qquad \text{(Battery capacity)}$$

Solution Approach

The solution approach of the problem consists of two components:

1. Solar Power Forecasting
   ⇒Forecasting Component to forecast the solar availability
   - Use the Prophet model[a] to forecast the building's total demand, including baseline and course-specific needs.
   - The forecasting component is in the `ML_solar_energy_forecast.ipynb` notebook.

[a] https://facebook.github.io/prophet

2. Optimal Battery Schedule
   ⇒Optimization Component to determine the battery schedule as well as the amount of electricity purchased from the grid.
   - Use the Gurobi solver[a] to solve the optimization problem for the battery schedule.
   - The optimization component is in the `MO_optimal_battery_schedule.ipynb` notebook.

[a] https://www.gurobi.com

Solution Approach

Hands-on Use Case

https://drive.google.com/drive/folders/1qe5muZbgviDeauH2EA45U54wW7zcyBHa?usp=drive_link

Thank You!

m.khemakhem@psau.edu.sa

For Survey and Certificate