Universiteit Utrecht

Introduction to optimization + Introduction to Assignment B

Nico Brinkel MSc ENERGY IN THE BUILT ENVIRONMENT GEO4-2522 7-10-2021

Universiteit Utrecht

Outline

- Introduction to optimization
- Examples
- Classes and formulation of optimization problems
- Gurobi
- Introduction to assignment B

What is an optimization problem?

- *The art of making good decisions*
- Find optimal solution to a problem, considering the limitations of the system.
- Helps in answering daily life questions
    - How to make my car as fast as possible?
    - What is the most profitable investment plan?
    - How to discover diseases as early and reliable as possible?

Universiteit Utrecht

Elements of optimization problems

1. Variables: elements that can be varied and influence the outcome of the optimization problem. In the optimization problem, the optimal values of the variables are determined.
2. Objective function: The function for which the optimal value will be determined (e.g., minimization of a problem).
3. Constraints:
    1. Equality constraints
    2. Inequality constraints

Universiteit Utrecht

Optimization problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & g_i(x) = 0, \quad i = 1, \ldots, p \end{array}$$

- **$x = (x_1, \ldots, x_n)$**: optimization variables (also called decision variables)
- **$f_1, \ldots, f_m$** are the inequality constraint functions
- **$g_1, \ldots, g_p$** are the equality constraint functions
- **f_0** is the objective function, to be minimized
- Optimal point (solution) has the smallest value of f_0 among all vectors that satisfy the constraints.

- variations: maximize objective, multiple objectives, . .

Universiteit Utrecht

Examples

- x represents some action, e.g.,
  - trades in a portfolio (how much of some assets to acquire)
  - schedule
  - resource allocation
- constraints limit actions or impose conditions on outcome, e.g.,
  - Budget
  - Number of transactions
  - Ramping rates of generators
- the smaller the objective $f_0(x)$, the better
  - total cost (or negative profit)
  - fuel use
  - CO2 emissions

Examples

portfolio optimization

- objective: overall risk (or return variance)
- variables: amounts invested in different assets
- constraints: budget, max./min. investment per asset, minimum return

generators dispatch (unit commitment)

- objective: power cost minimization
- variables: power to be generated by each generator
- constraints: output power limits of each generator, nonnegativity, demand satisfaction, etc

Universiteit Utrecht

Solving optimization problems

general optimization problem

- very difficult to solve
- methods involve some compromise, e.g., very long computation time, or not always finding the solution

exceptions: certain problem classes can be solved efficiently and reliably

- Depending on optimization class!

Universiteit Utrecht

Variable types

- Continuous variables
  - Can take any value between two values
- Binary variables
  - Only possible values are 0 or 1
- Discrete variables
  - Can only take a specific number of values between two values (e.g., only whole numbers)
- Affects complexity of the problem!

Optimization classes

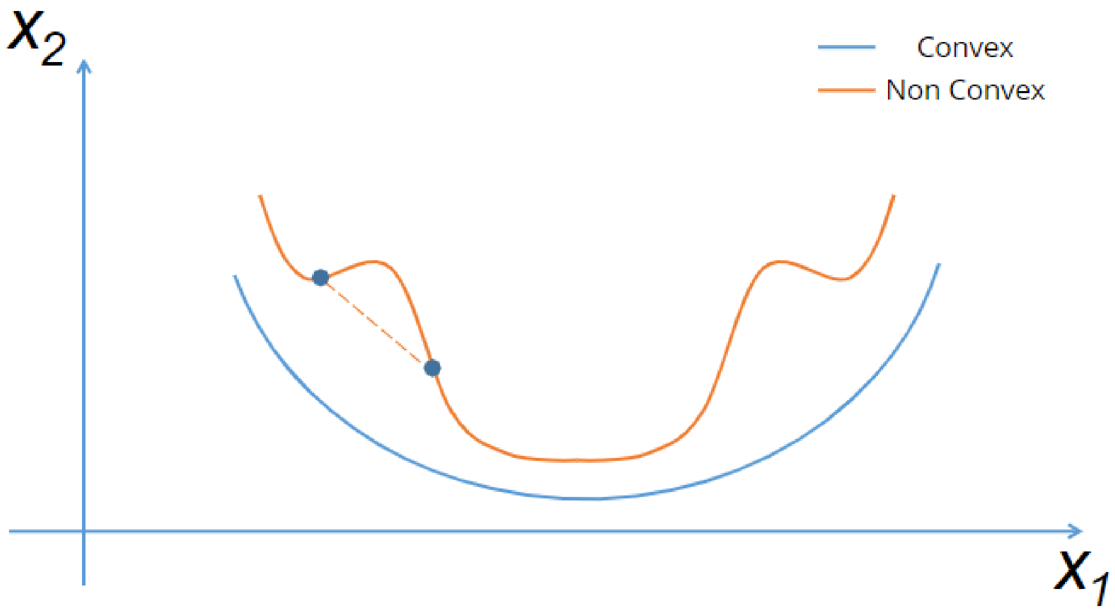
1. Linear optimization
    1. All variables are continuous
    2. Objective function and constraints are linear functions
    3. Easy to find global optimum!
2. Mixed-integer linear optimization
    1. Objective function and constraints are linear functions
    2. Some variables are binary or discrete
    3. Harder to find global optimum!

Universiteit Utrecht

Optimization classes

3. Quadratic optimization
   1. Quadratic objective function, linear constraints
   2. Complexity of model depends on

4.



Universiteit Utrecht

Linear programming

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & a_i^T x \le b_i, \quad i = 1, \ldots, m \end{array}$$

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b, \end{array}$$

with variable $x \in \mathbf{R}^n$

solving linear programs

- no analytical formula for solution
- reliable and efficient algorithms and software

Universiteit Utrecht

Formulation of optimization problems

Objective function:

$$\text{minimize} \sum_{t=1}^{T} (\mathrm{C}_{\mathrm{el},t} + \mathrm{C}_{\mathrm{battdeg},t})$$

Constraints:

$$\mathrm{P}_{\mathrm{grid},t} = \mathrm{P}_{\mathrm{res},t} - \mathrm{P}_{\mathrm{PV},t} + \frac{1}{\eta_{\mathrm{ch}}} \sum_{n=1}^{N} \mathrm{P}_{\mathrm{ch},n,t} - \eta_{\mathrm{disch}} \sum_{n=1}^{N} \mathrm{P}_{\mathrm{disch},n,t} \quad \forall t,$$

$$\mathrm{E}_{\mathrm{ch},t,s} = \mathrm{E}_{\mathrm{ch},t-1,s} + \left(\mathrm{P}_{\mathrm{ch},n,t} - \mathrm{P}_{\mathrm{disch},n,t}\right)\Delta\mathrm{t} \quad \forall t \in \{2, \dots, T\}, s.$$

Universiteit Utrecht

Formulation of optimization problems

Objective function:

$$\text{minimize} \sum_{t=1}^{T} (\mathrm{C}_{\mathrm{el},t} + \mathrm{C}_{\mathrm{battdeg},t})$$

Constraints:

$$\mathrm{P}_{\mathrm{grid},t} = \mathrm{P}_{\mathrm{res},t} - \mathrm{P}_{\mathrm{PV},t} + \frac{1}{\eta_{\mathrm{ch}}} \sum_{n=1}^{N} \mathrm{P}_{\mathrm{ch},n,t} - \eta_{\mathrm{disch}} \sum_{n=1}^{N} \mathrm{P}_{\mathrm{disch},n,t} \quad \forall t,$$

$$\mathrm{E}_{\mathrm{ch},t,s} = \mathrm{E}_{\mathrm{ch},t-1,s} + \left(\mathrm{P}_{\mathrm{ch},n,t} - \mathrm{P}_{\mathrm{disch},n,t}\right)\Delta\mathrm{t} \quad \forall t \in \{2, \dots, T\}, s.$$

Specify for which timesteps, scenarios etc. a constraint applies!

∀ means ‘for all’

Universiteit Utrecht

Formulation of optimization problems

Objective function:

$$\text{minimize} \sum_{t=1}^{T} (\mathrm{C}_{\mathrm{el},t} + \mathrm{C}_{\mathrm{battdeg},t})$$

In subscript, use italics for indices (i.e., t), while roman should be used for subscripts used for naming (i.e., $\mathrm{el}, \mathrm{battdeg}$).

Constraints:

$$\mathrm{P}_{\mathrm{grid},t} = \mathrm{P}_{\mathrm{res},t} - \mathrm{P}_{\mathrm{PV},t} + \frac{1}{\eta_{\mathrm{ch}}} \sum_{n=1}^{N} \mathrm{P}_{\mathrm{ch},n,t} - \eta_{\mathrm{disch}} \sum_{n=1} \mathrm{P}_{\mathrm{disch},n,t} \quad \forall t,$$

$$\mathrm{E}_{\mathrm{ch},t,s} = \mathrm{E}_{\mathrm{ch},t-1,s} + \left(\mathrm{P}_{\mathrm{ch},n,t} - \mathrm{P}_{\mathrm{disch},n,t}\right)\Delta\mathrm{t} \quad \forall t \in \{2, \dots, T\}, s.$$

Formulation of optimization problems

Objective function:

$$\text{minimize} \sum_{t=1}^{T} (\mathrm{C}_{\mathrm{el},t} + \mathrm{C}_{\mathrm{battdeg},t})$$

Minimize/maximize should be in roman.

Constraints:

$$\mathrm{P}_{\mathrm{grid},t} = \mathrm{P}_{\mathrm{res},t} - \mathrm{P}_{\mathrm{PV},t} + \frac{1}{\eta_{\mathrm{ch}}} \sum_{n=1}^{N} \mathrm{P}_{\mathrm{ch},n,t} - \eta_{\mathrm{disch}} \sum_{n=1}^{N} \mathrm{P}_{\mathrm{disch},n,t} \quad \forall t,$$

$$\mathrm{E}_{\mathrm{ch},t,s} = \mathrm{E}_{\mathrm{ch},t-1,s} + \left(\mathrm{P}_{\mathrm{ch},n,t} - \mathrm{P}_{\mathrm{disch},n,t}\right)\Delta \mathrm{t} \quad \forall t \in \{2, \dots, T\}, s.$$

Universiteit Utrecht

Formulation of optimization problems

Objective function:

$$\text{minimize} \sum_{t=1}^{T} (\mathrm{C}_{\mathrm{el},t} + \mathrm{C}_{\mathrm{battdeg},t})$$

Constraints:

$$\mathrm{P}_{\mathrm{grid},t} = \mathrm{P}_{\mathrm{res},t} - \mathrm{P}_{\mathrm{PV},t} + \frac{1}{\eta_{\mathrm{ch}}} \sum_{n=1}^{N} \mathrm{P}_{\mathrm{ch},n,t} - \eta_{\mathrm{disch}} \sum_{n=1}^{N} \mathrm{P}_{\mathrm{disch},n,t} \quad \forall t,$$

$$\mathrm{E}_{\mathrm{ch},t,s} = \mathrm{E}_{\mathrm{ch},t-1,s} + \left(\mathrm{P}_{\mathrm{ch},n,t} - \mathrm{P}_{\mathrm{disch},n,t}\right)\Delta\mathrm{t} \quad \forall t \in \{2, \dots, T\}, s.$$

When defining your variables, be specific! E.g., is $\mathrm{E}_{\mathrm{ch},t,s}$ the battery energy volume at the beginning or end of timestep t?

Universiteit Utrecht

Modeling languages and solvers

- high level language support for linear optimization
  - describe and solve problem in high level language
  - ideal for teaching (can do a lot with short scripts)
- implementations and interfaces:
  - Matlab
  - Python
- solvers:
  - Gurobi
  - CPLEX (IBM)
  - GAMS

Summary

- **summary**: optimization arises everywhere
- **the bad news**: most optimization problems are intractable i.e., we cannot solve them
- **an exception**: linear optimization problems (among some other classes) are tractable i.e., we (generally) can solve them

Universiteit Utrecht

The Gurobi Python interface

- Gurobi is for solving your optimization problem
- Python is a language interface for modelling your problem
- Python interface allows you to build models using a more mathematical syntax

http://www.gurobi.com/documentation/7.5/quickstart_windows/py_python_interface.html#section:Python

Universiteit Utrecht

Gurobi Python installation

Gurobi supports Python 2.7 and 3.6 for Windows.

1. Install Gurobi into Anaconda:

From an Anaconda terminal, run the following command to add the Gurobi channel to your default search list:

```
conda config --add channels http://conda.anaconda.org/gurobi
```

Install the Gurobi package using the command:

```
conda install gurobi
```

2. Install a Gurobi license

You are now ready to use Gurobi from within Anaconda. Your next step is to launch either the Spyder IDE or Jupyter Notebook.

Universiteit Utrecht

Example 1:

- You want to decide about three activities (do or don’t) and aim for a maximum value
- The total time limit is 4 hours
  - Activity 1 takes 1 hour
  - Activity 2 takes 2 hours
  - Activity 3 takes 3 hours
- You need to choose at least activity 1 or 2 (or both)
- Activity 3 is worth twice as much as 1 and 2

Universiteit Utrecht

Example 1:

- You want to decide about three activities (**do or don’t**) and aim for a maximum value
- The total time limit is 4 hours
  - Activity 1 takes 1 hour
  - Activity 2 takes 2 hours
  - Activity 3 takes 3 hours
- You need to choose at least activity 1 or 2 (or both)
- Activity 3 is worth twice as much as 1 and 2

maximize

$$x1 + x2 + 2*x3$$

Subject to

$$Xi \in \{0,1\}, \text{ for all } i$$

$$x1+2*x2+3*x3 <= 4$$

$$x1+x2 \qquad >= 1$$

Universiteit Utrecht

Example 1:

#Step 0: Import functions and classes from the Gurobipy module:

```
import gurobipy as gp
```

# Step 1: Create a model:

```
m = gp.Model(“example1")
```

# Step 2: Create variables:

```
x1 = m.addVar(vtype=gp.GRB.BINARY, name="x1")
x2 =
..
```

# Step 3: Add constraints:

```
cons1 = m.addConstr(x1 + 2 * x2 + 3 * x3 <= 4)
..
```

Universiteit Utrecht

Example 1:

```
# Step 4: Set objective:
 m.setObjective(x1 + x2 + 2 * x3, gp.GRB.MAXIMIZE)

# Step 5: Solve the model:
 m.optimize()

# Step 6: Print variable values for optimal solution:
 for v in m.getVars():
     print(v.varName, v.x)
# Or:
 m.printAttr('X')
```

Universiteit Utrecht

Example 2: optimal generators dispatch

- Schedule the electrical output power of a set of generators over some time interval
- Objective: to minimize the total cost of generation while meeting electrical demand.

One challenge in this problem is that the generators have dynamic constraints, which couple their output powers over time. For example, every generator has a maximum rate at which its power can be increased or decreased.

Universiteit Utrecht

Example 2: problem formulation

- We label the generators **i = 1, …, n**, and the time periods **t = 1, …, T**.
- We let **p$_{i,t}$** denote the (nonnegative) power output of generator i at time interval t.

Constraints:

1. The total generated power in each period should satisfy the demand at that time period:

$$\sum_{i=1}^{N} p_{i,t} = d_t \;\forall t$$

2. Each generator has a minimum and maximum allowed output power:

$$P_i^{\min} \leq p_{i,t} \leq P_i^{\max}$$

Universiteit Utrecht

Example 2: problem formulation

The total cost of operating the generators is:

$$C = \sum_{i=1}^{n} \sum_{t=1}^{T} \phi_i(p_{i,t})$$

where **Ø(pi,t)** is the cost of operating generator **i** at power output **pi,t.**

We will assume these cost functions are quadratic:

$$\phi_i(u) = \alpha_i u$$

Universiteit Utrecht

Example 2:

#Step 0: Import functions and classes from the Gurobipy module:

- **import gurobipy as gp**

# Step 1: Create a model:

- **m = gp.Model(“example2")**

# Step 2: Create variables:

- *Use **m.addVars( )**: define a vector (M*1) or an array (M*N) of variables in one statement.*
- **p = m.addVars(N,T,name='p')**

Universiteit Utrecht

Example 2:

# Step 3: Add constraints:

- *Use **m. addConstrs( )**: define a vector (M*1) or an array (M*N) of constraints in one statemet.*
- **Cons1= m.addConstrs(p[n,t] <= Pmax[n] for t in range(T) for n in range(N))**
- **Cons2=m.addConstrs(p[n,t] >= Pmin[n] for t in range(T) for n in range(N))**
- **Cons3=m.addConstrs(gp.quicksum(p[n,t] for n in range(N)) >= d[t] for t in range(T))**

# Step 4: Set objective:

- **obj = gp.quicksum(alpha[n]*p[n,t] for n in range(N) for t in range(T))**
- **m.setObjective(obj, gp.GRB.MINIMIZE)**

# Step 5: Solve the model:

- **m.optimize()**

Universiteit Utrecht

Example 2:

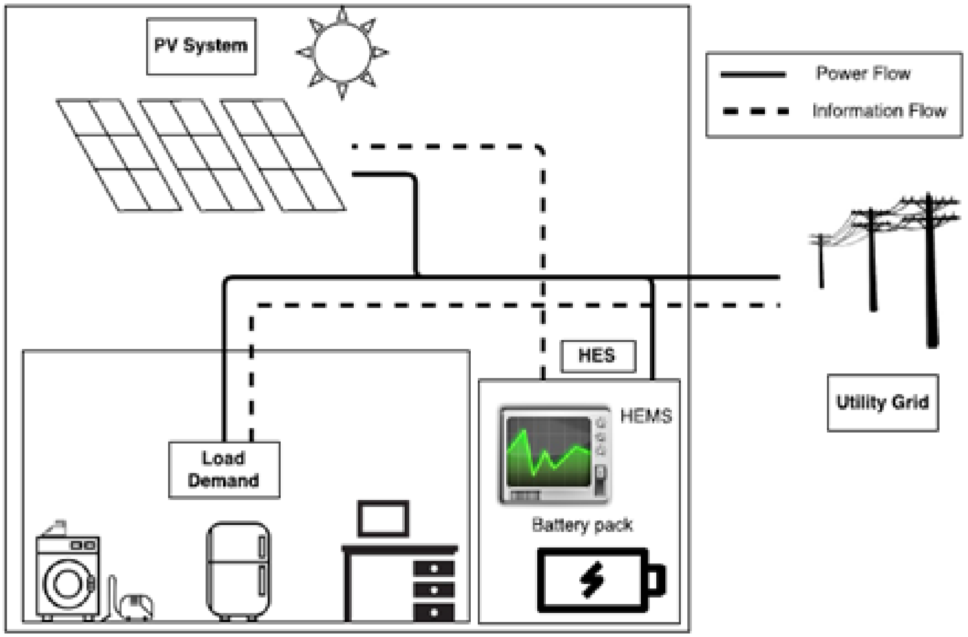
# Step 6: Print variable values for optimal solution:

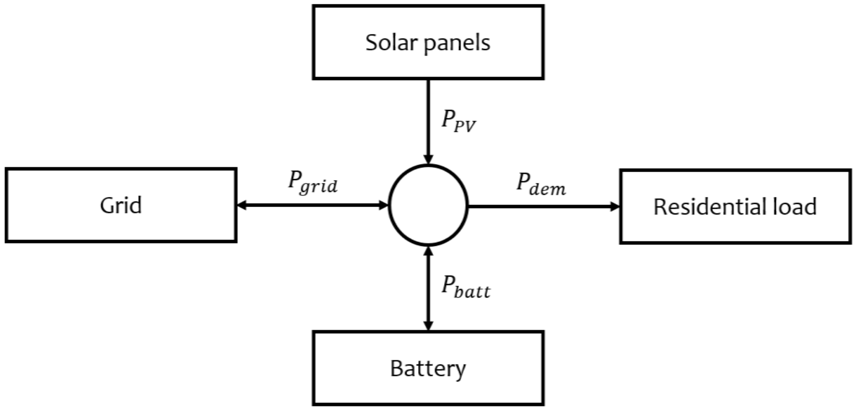
- **P=m.getAttr('X’)**
- **P=np.reshape(P, (N,T))**
- **p_gen = np.asmatrix(P)**

# Step 7: Plotting results:

- **plt.figure(1)**
- **plt.subplot(2,1,1)**
- **plt.plot(t, d);**
- **plt.title('demand')**
- **plt.subplot(2,1,2)**
- **plt.plot(t, p_gen.T);**
- **plt.title('generator powers')**

Assignment B





Q1+2 – Minimize electricity costs

Considering max. grid capacity, max charging/discharging rates of battery, battery capacity, residential load

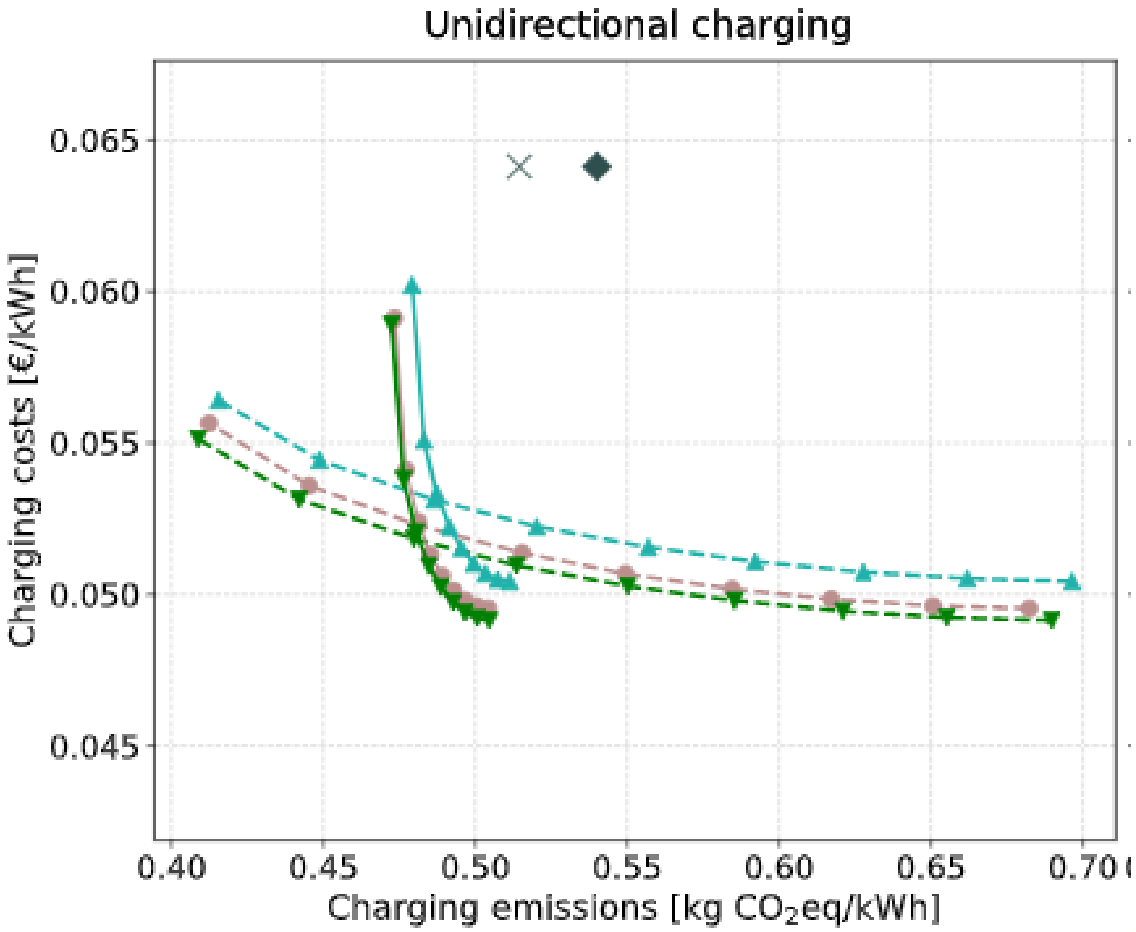
Assignment B

Question 3:

- Multi-objective optimization using ϵ-constraint method
- Find trade-off between cost and emission minimization.
- Present Pareto frontier.

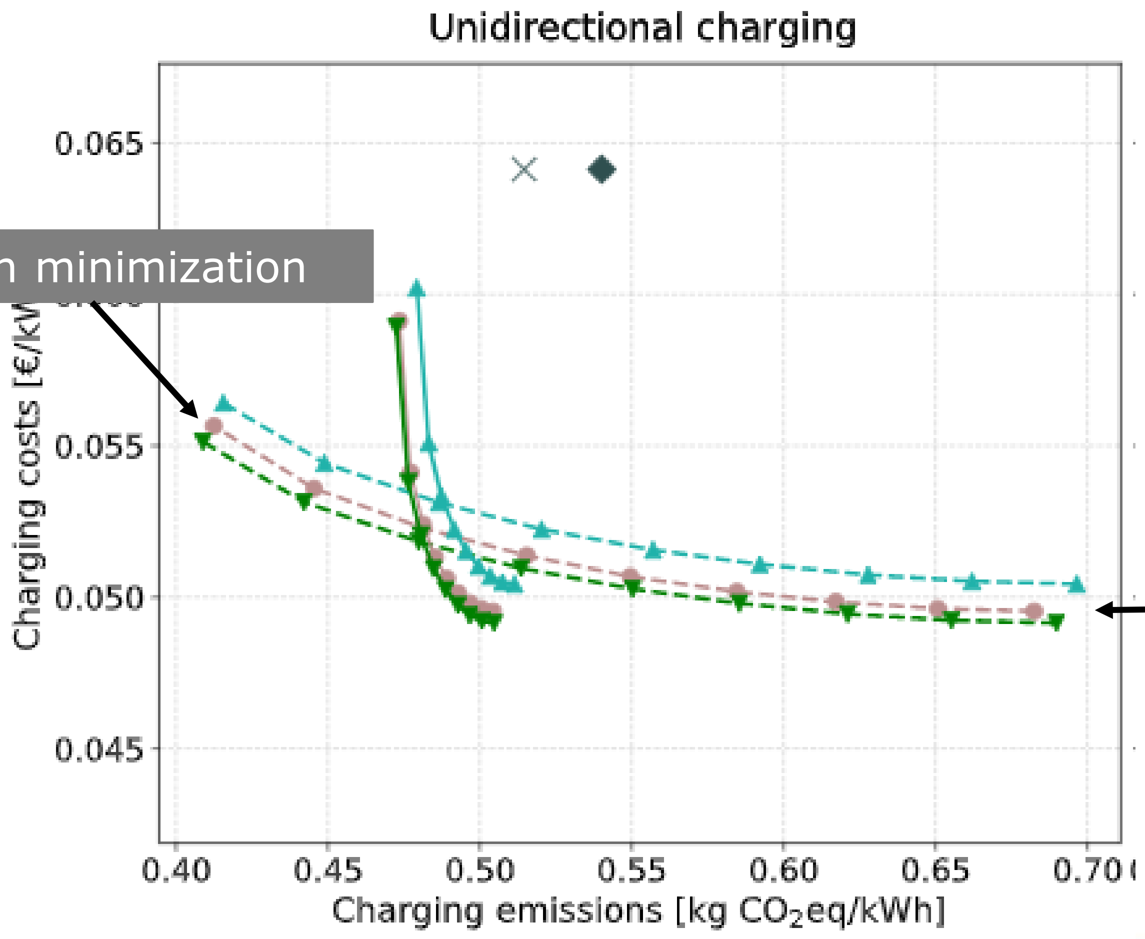
Universiteit Utrecht

Assignment B



Universiteit Utrecht

Assignment B



Emission minimization

Cost minimization

Universiteit Utrecht

Assignment B

Steps to create a Pareto frontier:

1. Determine the costs and emissions for the most extreme points of the Pareto-frontier (i.e., the points with pure cost and emission minimization).
2. To create other points of Pareto frontier, minimize costs or emissions, and add a constraint (ϵ) on the total costs or emissions.

For a more elaborate explanation on the ϵ-constraint method, read the provided papers!