Programming Task
Differential Evolution

Continuous Optimization

Continuous Optimization Problems arise in

- Machine Learning (minimize loss function)
- Probability Theory (compute probability bounds)
- Natural Sciences (physical and chemical models)
- Engineering (optimal design and control problems)
- Economics (maximize utility)

Differential evolution is particular interesting if objective function is non-smooth and non-convex

Power Plant Optimization

- suppose, you work for an electric utility service provider
- you can generate energy with different plane types with different costs
- you can sell energy at different markets at different prices (e.g. business, consumer)



- Problem: determine how much energy to generate with which plane type, how much energy to sell to which market and at what price in order to maximize profit

High-Level Profit Model

- Profit = Revenue - Cost

High-Level Profit Model

- Profit = Revenue - Cost
- Revenue = SoldQuantity * Price

High-Level Profit Model

- Profit = Revenue - Cost
- Revenue = SoldQuantity * Price
- Cost = ProductionCost + PurchasingCost

High-Level Profit Model

- Profit = Revenue - Cost
- Revenue = SoldQuantity * Price
- Cost = ProductionCost + PurchasingCost
- Production Cost = GeneratedQuantity * CostFactor

High-Level Profit Model

- Profit = Revenue - Cost
- Revenue = SoldQuantity * Price
- Cost = ProductionCost + PurchasingCost
- Production Cost = GeneratedQuantity * CostFactor
- Purchasing Cost
  = max(SoldQuantity - GeneratedQuantity, 0) * CostPrice

High-Level Profit Model

- Profit = Revenue - Cost
- Revenue = SoldQuantity * Price
- Cost = ProductionCost + PurchasingCost
- Production Cost = GeneratedQuantity * CostFactor
- Purchasing Cost
  = max(SoldQuantity - GeneratedQuantity, 0) * CostPrice

(if you sell more energy than you can provide, you have to buy energy from another provider)

Plant Cost Model

We describe each plant type by three parameters

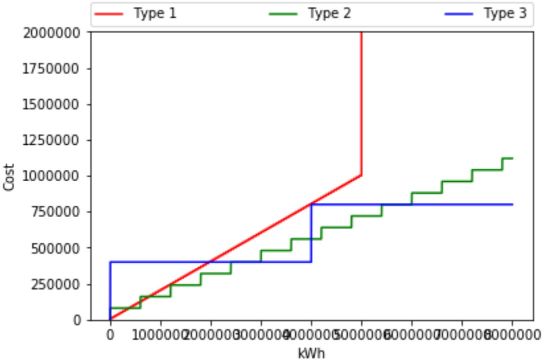
- k - kWh per plant
- c - cost per plant
- m - maximum number of plants that can be used

```
def cost(x, kwhPerPlant, costPerPlant, maxPlants):

    #if x is non-positive, return 0
    if(x <= 0):
        return 0

    #if x is greater than what can be generated return prohibitively large value
    if(x > kwhPerPlant * maxPlants):
        return LARGE

    #otherwise determine number of plants needed to generate x
    plantsNeeded = math.ceil(x / kwhPerPlant)

    #cost is number of plants needed times cost per plant
    return plantsNeeded * costPerPlant
```

Plant Types

We consider three plant types

- Type 1: k = 50,000, c = 10,000, m = 100
- Type 2: k = 600,000, c = 80,000, m = 50
- Type 3: k = 4,000,000, c = 400,000, m = 3



Market Model

We describe each market by two parameters

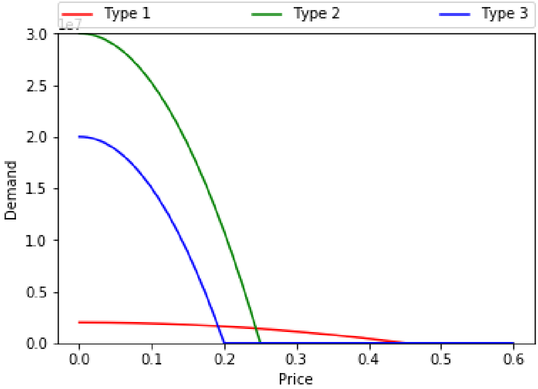
- p - maximum price at which customers buy
- d - maximum demand

```
def demand(price, maxPrice, maxDemand):

    #if price is greater than max price, return 0
    if(price > maxPrice):
        return 0

    #if product is free return maxDemand (ignore negative price)
    if(price <= 0):
        return maxDemand

    #else determine demand based on price
    demand = maxDemand - price**2 * maxDemand / maxPrice**2

    return demand
```

Market Types

We consider three market types

- Type 1: p = 0.45, d = 2,000,000
- Type 2: p = 0.25, d = 30,000,000
- Type 3: p = 0.2, d = 20,000,000



Profit Model

9 Variables $(e_1, e_2, e_3, s_1, s_2, s_3, p_1, p_2, p_3)$

1. e_1, e_2, e_3: energy produced with plants of type i
2. s_1, s_2, s_3: planned amount of energy sold to market of type i
3. p_1, p_2, p_3: price for market of type i

Profit model

- Profit = Revenue - Cost
- Revenue $= \sum_{i=1}^{3} \min\left(demand_i(p_i), s_i\right) \cdot p_i$
- Cost = ProductionCost + PurchasingCost
- Production Cost $= \sum_{i=1}^{3} cost_i(e_i)$
- Purchasing Cost $= \max\left(\left(\sum_{i=1}^{3} s_i\right) - \left(\sum_{i=1}^{3} e_i\right), 0\right) \cdot 0.6$

(cost price is 0.6)

DE Programming Task

- solve Power Plant Problem using Basic DE
- as usual, decompose your implementation
    - Initialization
    - Donor Generation (parameter for scale factor)
    - Trial Generation (parameter for crossover rate)
    - Selection
- document what scheme you use for which component (try several if you have time)
- divide work among group members

DE Programming Task

- make some experiments with different parameter settings
- document your findings and prepare a small presentation (5-10 minutes)
- send me a compressed archive containing
  1. slides (structure findings in table or other visualization)
  2. source files/ notebook
  3. assignment of tasks to group members

Remarks

- there are other (and perhaps better) ways to model this problem equivalently
- if you have a better idea, feel free to try an alternative model
- if you have time, also try different (and more) plant and market types
- note that you are completely flexible in the choice of your objective function (non-linear, non-differentiable, ...)

Some Results

Problem	Plants			Markets		Cost Price	Profit
	k	c	m	p	d		Achieved
P1	50k	10k	100	0.45	2M	0.6	1,514,312
	600k	80k	50	0.25	30M		
	4M	400k	3	0.2	20M		
P2	50k	10k	100	0.45	2M	0.1	1,818,406
	600k	80k	50	0.25	30M		
	4M	400k	3	0.2	20M		
P3	50k	10k	100	0.5	1M	0.6	404,041
	600k	80k	50	0.3	5M		
	4M	400k	3	0.1	5M		