Optimisation Avancée

Black-Box Optimization Strategies for Wind Farm Layout Expansion

Sarah ALI
Camille PASCAL
Wilia CHEKLI

Professeurs :
Dr. Regis DUVIGNEAU
Dr. Michael BINOIS

26 Novembre 2025

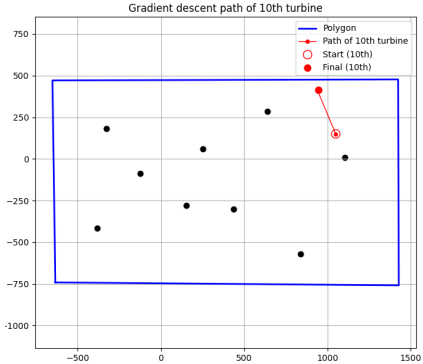
Problem

- Penalized function minimization

$$-f(x) = -(EAP - \lambda \cdot (Spacing + Placing)) \quad for \quad \lambda = 10$$

- Optimization of the position of the $(n+1)^{th}$ turbine , for instance 2.

Gradient Descent

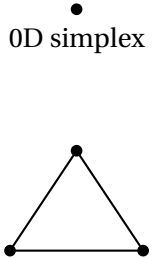


ALGORITHM SETTINGS		10th TURBINE & RUNTIME	
Finite diff step h	12.000	Start position [x, y]	[1050.0, 150.0]
Initial step size alpha	1000.000	Final position [x, y]	[946.0726008656645, 412.87856262444313]
Final step size alpha	31.250		
Tolerance	1.00e-03	Runtime (s)	347.3268
Max iterations	500	Iterations used	199
Penalty coefficient λ	10.000		
Reached max_iter?	False		
EAP at start (BB)	19.094		
EAP at optimum (BB)	19.277		

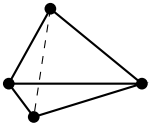
Algorithm

Simplex

- Smallest complex set containing $n+1$ affinely independent points in $\mathbb{R}^n$.(all of the points DO NOT lie on the same line)
- It is the generalization of the triangle in higher dimensions.



2D simplex



3D simplex

0D simplex

1D simplex

Algorithm

Steps

- Simplex Ordering
- Centroid
- Reflection
- Expansion
- Outside Contraction
- Inside Contraction
- Shrink

$$f(x_1) \leq \cdots \leq f(x_{n+1})$$

$$x_c = \frac{1}{n}\sum_{i=1}^{n} x_i$$

$$x_r = x_c + \alpha(x_c - x_{n+1})$$

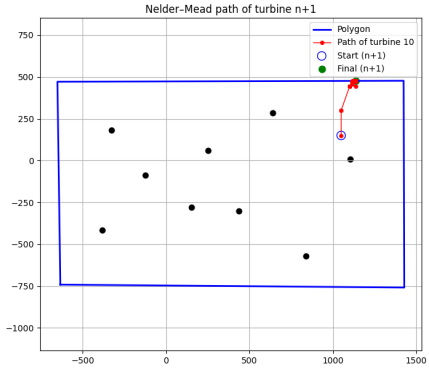
$$x_e = x_c + \gamma(x_r - x_c)$$

$$x_{oc} = x_c + \rho(x_r - x_c)$$

$$x_{ic} = x_c - \rho(x_c - x_{n+1})$$

$$x_i \leftarrow x_1 + \sigma(x_i - x_1)$$

Results




```
================================================================================
NELDER-MEAD SETTINGS                    TURBINE (n+1) & RUNTIME
--------------------------------------------------------------------------------
Alpha (reflection)          1.500       Start [x, y]            [1050.0, 150.0]
Gamma (expansion)           3.000       Final [x, y]            [1136.020000489699, 476.2056384949842]
Rho (contraction)           0.500       Runtime (s)             78.2064
Sigma (shrink)              0.500       Iterations used         37
Tolerance                   1.00e-06    Reached max_iter?       False
Max iterations              100         Penalty λ               10.000
EAP at start (BB)           19.094
EAP at optimum (BB)         19.383
================================================================================
```

Neural Network - Attempt 1

As our blackbox function that calculates the EAP is very expensive to run, we opt to approximate it , we used neural networks to do it.

Parameters

Parameter	Value
Architecture	Linear $\rightarrow$ ReLU $\rightarrow$ Linear $\rightarrow$ Output
Hidden Neurons	16
Optimizer	Adam
Loss Function	MSELoss
Training Set	800 realizations
Testing Set	200 realizations
Epochs	700
Script runtime	341.70 seconds

Table 1: Neural Network 1 Configuration Summary

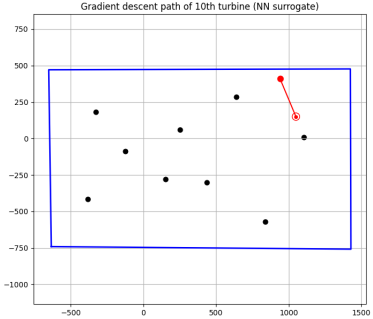
Comparison

```
=== Penalized Objective Comparison ===
EAP_true            = 20.339394
Spacing_true        = 0.000000
Placing_true        = 232.878556
Penalized_true      = 2308.446161

Penalized_surrogate = 2308.397786
Difference          = 0.048375
Relative difference = 0.002%
```

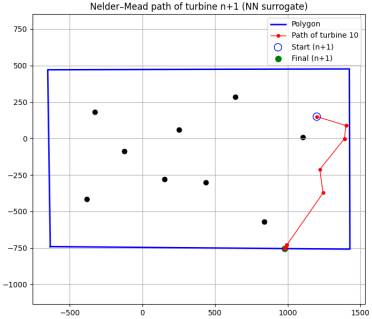
Figure 1: Penalized Objective Comparison: True vs Surrogate 1 Evaluation

Gradient Descent



ALGORITHM SETTINGS		10th TURBINE & RUNTIME	
Finite diff step h	12.000	Start [x, y]	[1050.0, 150.0]
Initial alpha	1000.000	Final [x, y]	[940.6293361980656, 409.7967649561053]
Final alpha	15.625	Runtime (s)	0.0211
Tolerance	1.00e-03	Iterations used	7
Max iterations	100	Reached max_iter?	False
Penalty λ	10.000		
EAP at start (NN)	20.387		
EAP at optimum (NN)	20.388		

Nelder Mead



NELDER–MEAD (NN) SETTINGS		TURBINE (n+1) & RUNTIME	
Alpha (reflection)	1.500	Start [x, y]	[1050.0, 150.0]
Gamma (expansion)	3.000	Final [x, y]	[980.3437869195792, -754.4630401170864]
Rho (contraction)	0.500	Runtime (s)	0.2175
Sigma (shrink)	0.500	Iterations used	39
Tolerance	1.00e-06	Reached max_iter?	False
Max iterations	100	Penalty λ	10.000

Neural Network - Attempt 2

Parameters

Parameter	Value
Architecture	Linear $\rightarrow$ ReLU $\rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ Linear $\rightarrow$ ReLU $\rightarrow$ Output
Hidden Neurons	$64 + 64 + 32$
Optimizer	Adam
Loss Function	MSELoss
Training Set	4000 realizations
Testing Set	1000 realizations
Epochs	130
Script runtime	1763.65 seconds

Table 2: Neural Network 2 Configuration Summary

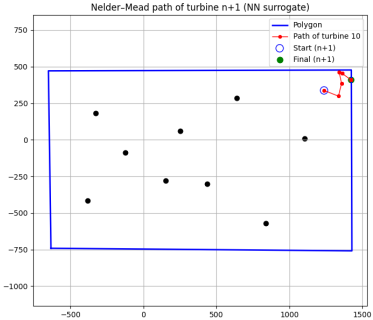
Comparison

```
=== Penalized Objective Comparison ===
EAP_true            = 20.339394
Spacing_true        = 0.000000
Placing_true        = 232.878556
Penalized_true      = 2308.446161

Penalized_surrogate = 2308.405082
Difference          = 0.041080
Relative difference = 0.002%
```

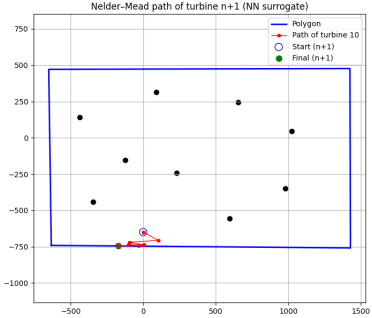
Figure 2: Penalized Objective Comparison: True vs Surrogate 2 Evaluation

Nelder Mead



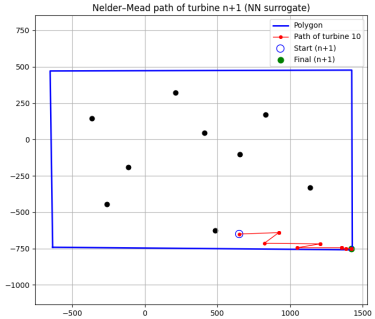
NELDER-MEAD (NN) SETTINGS		TURBINE (n+1) & RUNTIME	
Alpha (reflection)	1.500	Start [x, y]	[1050.0, 150.0]
Gamma (expansion)	3.000	Final [x, y]	[1422.889206558466, 408.7753303349018]
Rho (contraction)	0.500	Runtime (s)	0.1162
Sigma (shrink)	0.500	Iterations used	29
Tolerance	1.00e-06	Reached max_iter?	False
Max iterations	100	Penalty λ	10.000
EAP at start (NN)	20.387		
EAP at optimum (NN)	20.390		

Nelder Mead



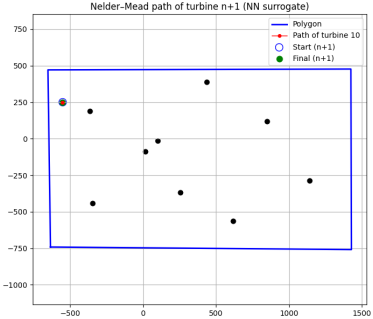

```
================================================================================
NELDER-MEAD (NN) SETTINGS                TURBINE (n+1) & RUNTIME
--------------------------------------------------------------------------------
Alpha (reflection)          1.500        Start [x, y]          [0.0, -650.0]
Gamma (expansion)           3.000        Final [x, y]          [-171.970796585083, -744.9553489685059]
Rho (contraction)           0.500        Runtime (s)           0.0959
Sigma (shrink)              0.500        Iterations used       25
Tolerance                   1.00e-06     Reached max_iter?     False
Max iterations              100          Penalty λ             10.000
EAP at start (NN)           20.390
EAP at optimum (NN)         20.393
================================================================================
```

Nelder Mead



NELDER-MEAD (NN) SETTINGS		TURBINE (n+1) & RUNTIME	
Alpha (reflection)	1.500	Start [x, y]	[500.0, -650.0]
Gamma (expansion)	3.000	Final [x, y]	[1420.2897289083921, -752.7953161785263]
Rho (contraction)	0.500	Runtime (s)	0.0924
Sigma (shrink)	0.500	Iterations used	24
Tolerance	1.00e-06	Reached max_iter?	False
Max iterations	100	Penalty λ	10.000
EAP at start (NN)	20.379		
EAP at optimum (NN)	20.381		

Nelder Mead



NELDER-MEAD (NN) SETTINGS		TURBINE (n+1) & RUNTIME	
Alpha (reflection)	1.500	Start [x, y]	[-550.0, 250.0]
Gamma (expansion)	3.000	Final [x, y]	[-550.0, 250.0]
Rho (contraction)	0.500	Runtime (s)	0.0134
Sigma (shrink)	0.500	Iterations used	2
Tolerance	1.00e-06	Reached max_iter?	False
Max iterations	100	Penalty λ	10.000
EAP at start (NN)	20.378		
EAP at optimum (NN)	20.378		

Pareto Front - Neural Network

In this section we will be optimizing the position of the turbines with respect to 2 different wind directions.
What we did was the following:

- Retrain our Neural Network to both directions
- Find the Pareto Front that will contain the best turbine positions for those 2 wind directions using 2 methods:
  - Weighted Sum
  - Genetic Algorithm

Pareto Front - Neural Network

Weighted Sum

This method is a simple method , it minimizes the following :

$$\omega \cdot f_1(x) + (1-\omega) \cdot f_2(x)$$

Intuitively, what this does is it just basically "chooses" from what you already have, the best layouts for both criteria, and it does not actually do any optimization over the turbine positions.

```
Approximate Pareto points found (unique layouts): 5
[[-21.12888145 -21.55178261]
 [-21.14646339 -21.53886795]
 [-21.1295414  -21.55147362]
 [-21.16372108 -21.51524734]
 [-21.13608932 -21.54761124]]
```

This method was able to get us 5 layouts out of 500!

Pareto Front - Neural Network

NSGA-II (Non-dominated Sorting Genetic Algorithm II)

The algorithm explores the continuous search space and proceeds as follows:

1. Initialize the population ("parents").
2. Evaluate f_1 and f_2 for all individuals.
3. Extract the non-dominated individuals (Pareto Front 1).
4. Repeat non-dominated sorting on remaining individuals (Front 2, Front 3, ...).
5. Compute crowding distance within each front (isolated = priority, crowded = lower priority).

Pareto Front - Neural Network

NSGA-II (Non-dominated Sorting Genetic Algorithm II)

6. Assign each individual a rank (front number) and a crowding distance.
7. Select top individuals and generate children (with small mutations).
8. Merge parents + children and repeat sorting + crowding to form the next generation.
9. Iterate until convergence.

Pareto Front - Neural Network

NSGA-II Results — Neural Network 1

After running this script, we encountered a major issue: the Pareto front was identical to the initial population, and for the two wind directions, the penalized functions were almost perfectly correlated.

```
Number of Pareto-optimal solutions: 500
F1 range: -21.16379165649414 → -21.128881454467773
F2 range: -21.551782608032227 → -21.515380859375
Number of dominating pairs in final population: 0
Correlation between f1 and f2: -0.9835463671801503
```

Pareto Front - Neural Network

NSGA-II Results — Neural Network 2

We then used our more complex neural network to recompute the Pareto front using the NSGA-II algorithm. While the issue persisted, the results were noticeably less correlated.

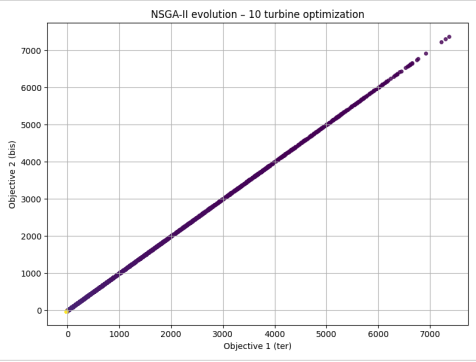
```
Number of Pareto-optimal solutions: 500
F1 range: -21.18744659423828 → -21.165199279785156
F2 range: -21.52583122253418 → -21.508081436157227
Number of dominating pairs in final population: 0
Correlation between f1 and f2: -0.8622652574515238
```

Pareto Front - Neural Network

Problem 2 - move all 10 turbines

We tried to find the pareto front for the same instances of 10 turbines, but moving all the positions

```
Number of Pareto-optimal solutions: 82
F1 range: -21.24669647216797 → -21.18024253845215
F2 range: -21.61009407043457 → -21.524024963378906
Number of dominating pairs in final population: 0
Correlation between f1 and f2: -0.9558289162132404
```



Optimizing the position of 20 turbines

```
=== Nelder-Mead Optimization Finished ===
Best solution vector (X):
[-345.47785904 -346.00349749  895.24611618  187.46170947  277.60502769
 -676.86591763    6.59287788 -279.66414867 -220.99093965  252.47781106
  451.5246612  -625.58027984  575.83809267 -404.30606398 -489.91026969
  350.85311033  111.77695982  277.6599311  1225.45303408 -543.52467583
  141.22554068   24.32807251 -437.06954461 -556.96099939  585.48984939
  354.95003108  763.72625826 -490.46109321 1106.83835541 -218.73071568
 1239.79176752  269.27058777 1354.61819795 -308.82563041  974.28833467
 -332.20283611  630.71382186  176.65916699 -172.2044927  -568.54786525]

Maximized penalized EAP:
36.06633758544922
```

Optimizing the position of 30 turbines

```
=== Nelder-Mead Optimization Finished ===
Best solution vector (X):
[ 1.35539067e+03 -1.26121692e+02  1.28724260e+03  5.01218951e+01
  1.26600501e+03  3.28065641e+02  1.10905131e+03  2.43223134e+02
  6.46637678e+02  2.70833102e+02 -4.19486580e+02 -1.44439742e+01
 -2.35545749e+02 -4.83875327e+02  4.33681695e+00 -4.46389179e+02
  3.57195671e+02 -5.05224099e+02 -1.95383817e+02 -2.72265609e+02
  2.76033596e+02 -2.37346111e+02  7.06822407e+02 -5.84756388e+02
  4.55996271e+02  4.16758413e+02 -6.25970491e+02  1.94775795e+00
  9.66408858e+02 -7.08525356e+02  8.32062828e+01  3.78031488e+02
 -5.23663176e-01 -6.24978622e+02  4.94034128e+02 -7.28755300e+02
 -3.00679858e+01 -1.54547077e+02  4.33683707e+02 -6.30106478e+01
  1.20135560e+03 -7.55799711e+02  1.02049025e+03 -1.16876825e+02
  1.35623162e+03 -5.30090423e+02 -5.70608321e+02 -6.55035975e+02
 -4.69547955e+02  3.43839712e+02  8.78015143e+02  1.37428914e+02
  2.54520574e+02  2.89847110e+02  9.15213510e+02  3.24806959e+02
  1.14126545e+02  8.06776572e+00 -3.72637307e+02 -2.06989890e+02]

Maximized penalized EAP:
48.83151626586914
```

Conclusion

- Neural Networks are a good choice, but you will definitely need a GPU!
- Nelder Mead is not the best algorithm.