Surrogate Model-Based Multi-Objective Optimization Using Desirability Functions

Thomas Bartz-Beielstein

2025-07-15

1. Introduction & Motivation

Motivation from Industry

- **The Dichotomy:** Significant gap between industrial adoption and academic use
  - Desirability functions (Harrington, 1965): Established in industrial optimization
  - Seldom used in academic multi-objective optimization (MOO)
  - (Never?) used in ML/hyperparameter tuning (HPT)
- **The Problem in ML/DL HPT:** Manual, irreproducible trial-and-error processes
  - Balancing model accuracy, training time, complexity
  - Lack of systematic multi-objective approaches
- **Background:** This work is motivated by requests from industrial partners:
  - "Confused" by the Pareto-front concepts
- **Our Aim:** Providing easy to use tools

Core Research Questions

1. **Application:** How can desirability functions be methodically used for:
    - Classical multi-objective optimization
    - Contemporary hyperparameter tuning
2. **Long-term Goal:** What are the concrete advantages and disadvantages compared to other MOO methods?
3. **Enhancement:** How can the desirability framework be improved to overcome known limitations?

Materials

Jupyter Notebook

- Updates and Jupyter Notebook of this Presentation:
  - Research Cluster “Technische Hochschule Köln - Artificial Intelligence”
  - https://thk-ai.de

2. Theoretical Foundation

The Desirability Function: Fundamentals

Core Concept

- Transform multiple **incommensurable objectives** $f_r(x)$
- Single **dimensionless scale** $d_r \in [0, 1]$
- 0 = completely unacceptable
- 1 = perfectly desirable

Three Function Types

- **Larger-is-Better ($d_{\max}$):** Maximization (accuracy, yield)
- **Smaller-is-Better ($d_{\min}$):** Minimization (error, cost)
- **Target-is-Best (d_{target}):** Specific target value

Maximization Formula

Maximization

- For maximization of $f_r(\vec{x})$ ("larger-is-better"), the following function is used:

$$d_r^{\text{max}} = \begin{cases} 0 & \text{if } f_r(\vec{x}) < A \\ \left(\frac{f_r(\vec{x})-A}{B-A}\right)^s & \text{if } A \leq f_r(\vec{x}) \leq B \\ 1 & \text{if } f_r(\vec{x}) > B \end{cases}$$

- Parameters A ("acceptable"), B ("ideal"), and s ("scale") are chosen by the investigator.
- Similar in the minimization case ("smaller-is-better").
- Scale parameter s can be adjusted to make the desirability criterion easier or harder to satisfy.

Target Optimization

- Target value: t_0.
- In "target-is-best" situations, the following function is used:

$$d_r^{\text{target}} = \begin{cases} \left(\frac{f_r(\vec{x})-A}{t_0-A}\right)^{s_1} & \text{if } A \leq f_r(\vec{x}) \leq t_0 \\ \left(\frac{f_r(\vec{x})-B}{t_0-B}\right)^{s_2} & \text{if } t_0 \leq f_r(\vec{x}) \leq B \\ 0 & \text{otherwise.} \end{cases}$$

Visualization of Desirability Functions

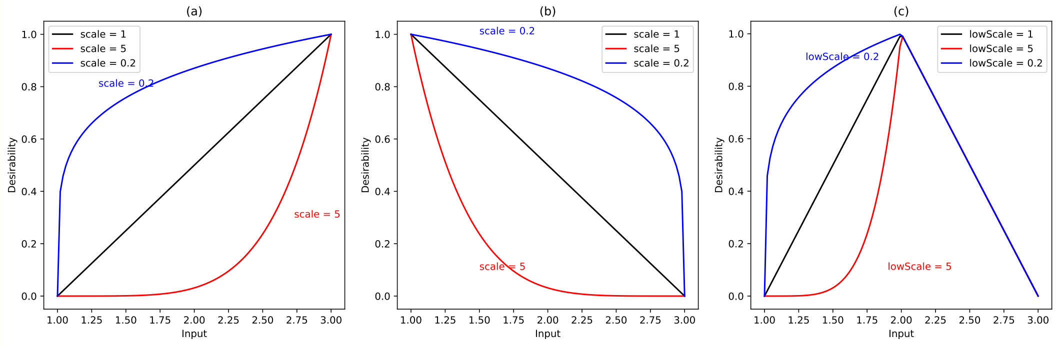


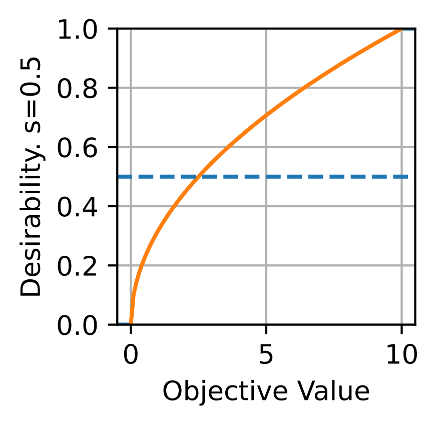
Figure 1: Examples of the three primary desirability functions. Panel (a) Larger–is–better function, panel (b) Smaller–is–better desirability function and panel (c) target value.

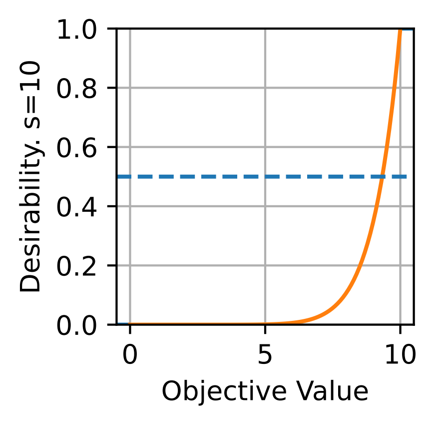
Larger is Harder

The values of s, s_1, or s_2 can be chosen so that the desirability criterion is easier or more difficult to satisfy (examples on the next slides):

- Values of s **greater than 1** will make d_r^{max} **harder to satisfy** in terms of desirability.
  - If s is chosen to be less than 1 in d_r^{max}, d_r^{max} is near 1 even if $f_r(\vec{x})$ is not low.
  - As values of s move closer to 0, the desirability reflected by d_r^{max} becomes higher.
- Scaling factors are useful when one equation holds more importance than others.
- Any function can reflect model desirability.

Maximization for s=0.5 and s=5

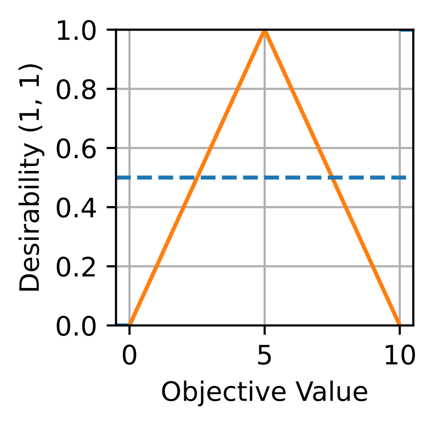


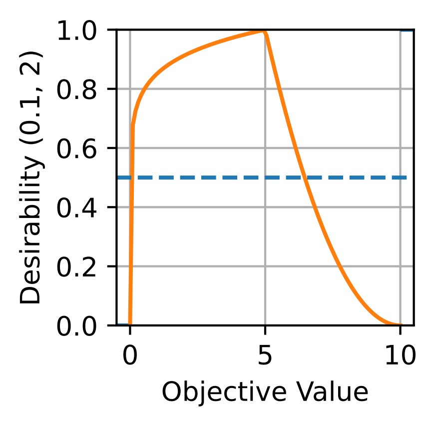


- Dotted blue lines: desirability where objective value cannot be computed (**NA**s)

Target Desirability

Target-is-Best





Zero-Desirability Problem

- In high-dimensional MOO outcomes, finding feasible solutions where every desirability value is acceptable can be challenging.
- Each desirability R function has a **`tol`** argument, which can be set between [0, 1] (default is **`NULL`**).
- If not null, zero desirability values are replaced by **`tol`**.
- Research Question:
  - Using a default, small value for **`tol`** can be usefull (similar to the λ nugget in Kriging).

The **dArb** function (**Arb** stands for “Arbitary”) can be used to create a custom desirability function.

Example: Logistic Desirability Function

- The logistic function defined as

$$d(\vec{x}) = \frac{1}{1 + \exp(-\vec{x})}.$$

- Inputs in-between these grid points are linearly interpolated.

```
def logistic(u):
    return 1 / (1 + np.exp(-u))
x = np.linspace(-5, 5, 20)
logistic_d = DArb(x, logistic(x))
```

Plotting the Logistic Desirability Function

- Figure 2 displays a **plot** of the **logisticD** object.

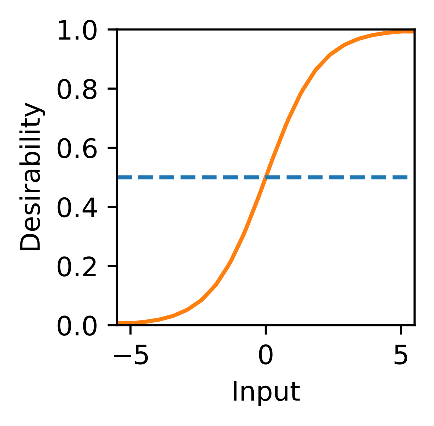


Figure 2: Using the **DArb** function. The desirability function is a logistic curve.

Overall Desirability and “Veto” Property

Aggregation Formula

$$D = \left(\prod_{r=1}^{R} d_r \right)^{1/R}$$

- **Geometric mean** ensures the “veto” property:
  - If any $d_r = 0$, then $D = 0$
- All objectives must meet minimal acceptability
- Translates specification limits into **hard constraints**

3. Case Study 1: Validation and Verification

How to Validate and Verify the Software Package?

Testing the Package

- **`pytest`**

Comparison with Existing R Packages

- Validate **`spotdesirability`** on classic RSM problem (quadratic response surface model) (Myers, Montgomery, and Anderson-Cook 2016).
- R package **`desirability`** is used for comparison.

Methods Compared

- Direct Search (Nelder-Mead):
  - R: **`desirability`** with **`optim`** with **`method="Nelder-Mead"`**
  - Python: **`spotdesirability`** with **`scipy.optimize.minimize`** with **`method="Nelder-Mead"`**

In addition: **Surrogate-Model Based Optimization (SMBO)**

Chemical Reaction Optimization

- Well-studied optimization problem from Myers, Montgomery, and Anderson-Cook (2016).
- Study based on the work of Kuhn (2016).
- Three input variables normalized to $[-1, 1]$:
  - x_1: Time (hours),
  - x_2: Temperature (°C), and
  - x_3: Catalyst concentration (g/L)

Two Objectives

- **Maximize Percent Conversion:** d_{max} (80% min, 97% target)
- **Target Thermal Activity:** d_{target} (55-60 range, 57.5 ideal)

$$
\begin{aligned}
f_{\text{con}}(x) =& 81.09 + 1.0284 \cdot x_1 + 4.043 \cdot x_2 + 6.2037 \cdot x_3 + 1.8366 \cdot x_1^2 + 2.9382 \cdot x_2^2 \\
&+ 5.1915 \cdot x_3^2 + 2.2150 \cdot x_1 \cdot x_2 + 11.375 \cdot x_1 \cdot x_3 + 3.875 \cdot x_2 \cdot x_3 \\
f_{\text{act}}(x) =& 59.85 + 3.583 \cdot x_1 + 0.2546 \cdot x_2 + 2.2298 \cdot x_3 + 0.83479 \cdot x_1^2 + 0.07484 \cdot x_2^2 \\
&+ 0.05716 \cdot x_3^2 + 0.3875 \cdot x_1 \cdot x_2 + 0.375 \cdot x_1 \cdot x_3 + 0.3125 \cdot x_2 \cdot x_3.
\end{aligned}
$$

Objective 1: Maximize Percent Conversion

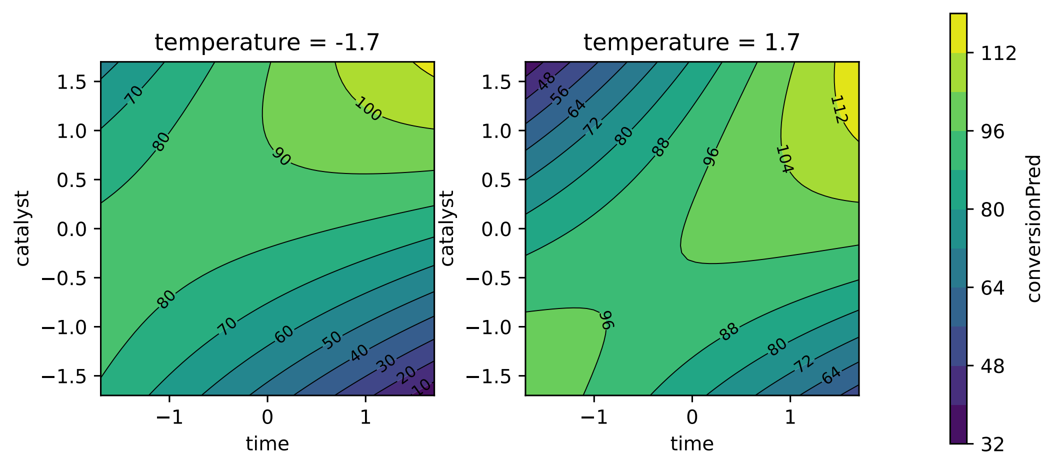
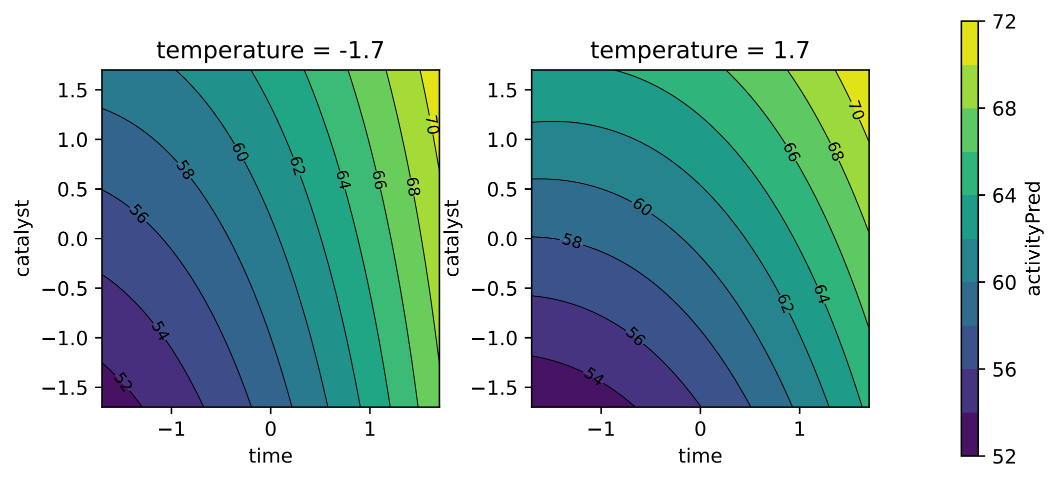


Figure 3

Case Study 1: The response surface for the thermal activity model

Objective 2: Target Thermal Activity



Package Installation

`spotdesirability` Python Package

- Available on GitHub https://github.com/sequential-parameter-optimization/spotdesirability and
- on PyPi https://pypi.org/project/spotdesirability
- It can be installed via **`pip install spotdesirability`**

Larger-is-Better and Target Desirability

- A larger-is-better function (d_r^{max}) is used for percent conversion with values $A = 80$ and $B = 97$.
- A target-oriented desirability function (d_r^{target}) was used for thermal activity with $t_0 = 57.5$, $A = 55$, and $B = 60$.

Creating Desirability Objects

- The two desirability objects can be created as follows:

```
conversionD = DMax(80, 97)
activityD = DTarget(55, 57.5, 60)
overallD = DOverall(conversionD, activityD)
```

Predicting Desirability Values

Computing Desirability at the Center Point

- Predict the desirability for the center point of the experimental design.
- Overall desirability computed using the **`DOverall`** class.
- Based on these desirability predictions, contour plots can be generated to visualize the desirability surfaces.

```
Conversion Desirability: [0.06411765]
Activity Desirability: [0.06]
Overall Desirability (geom. mean): [0.06202466]
```

First Objective: Individual Desirability Surface

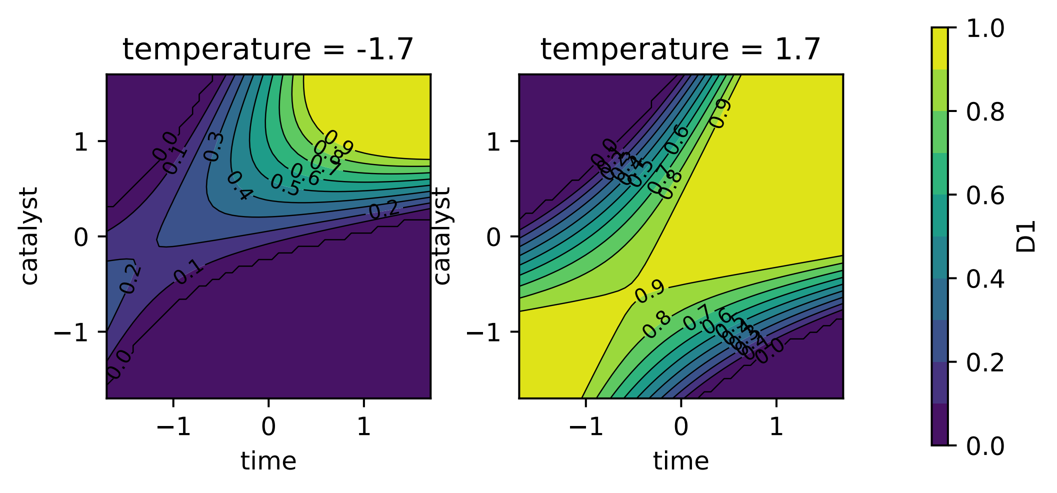


Figure 5

Second Objective: Individual Desirability Surface

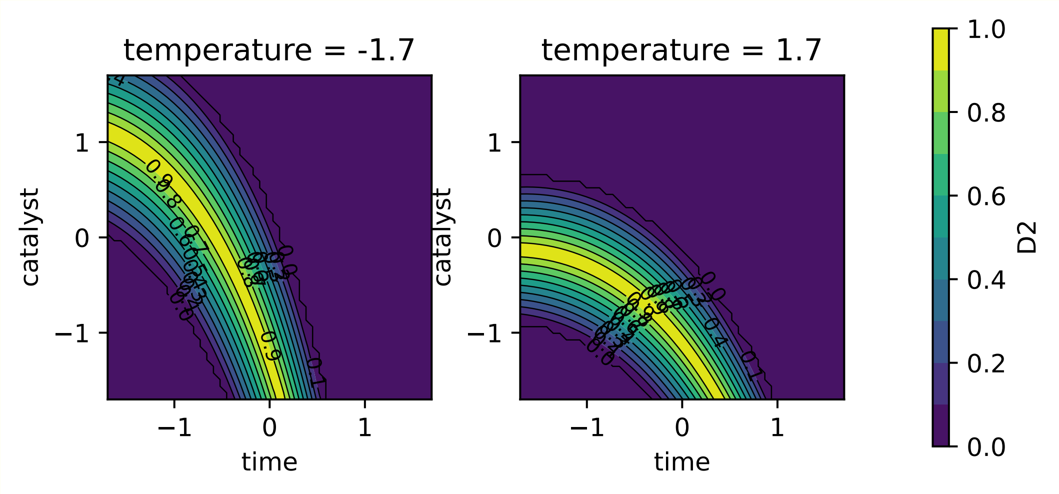


Figure 6

Overall Desirability Surface

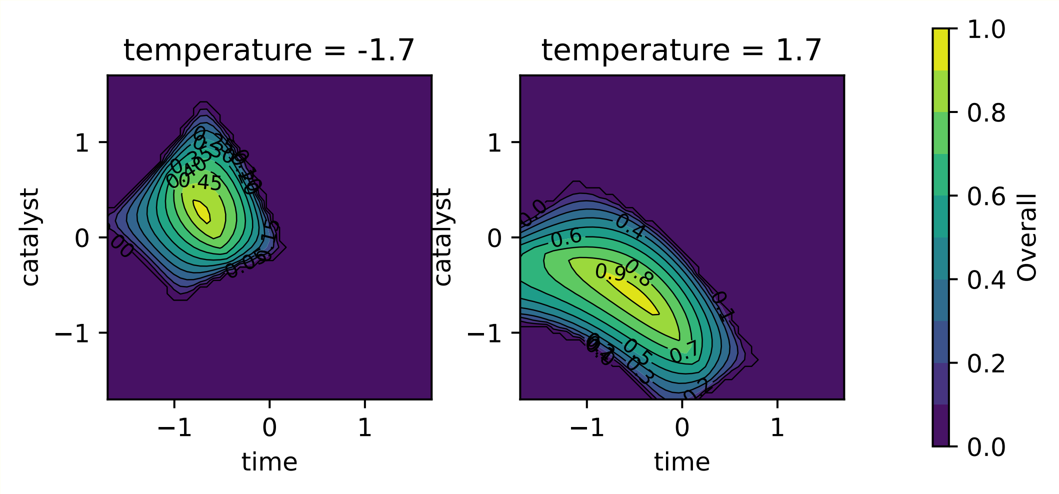


Figure 7

Optimization Function and `minimize` Call

```python
def rsm_opt(x, d_object, prediction_funcs) -> float:
    predictions = [func(x) for func in prediction_funcs]
    desirability = d_object.predict(np.array([predictions]))
    return -desirability
```

```python
    result = minimize(
        rsm_opt,
        initial_guess,
        args=(overallD, prediction_funcs),
        method="Nelder-Mead",
        options={"maxiter": 1000, "disp": False}
    )
```

Input Parameters

- The optimization is performed over a grid of input parameters, and the best result is selected based on the overall desirability:

```
Best Input Parameters: [-0.51207663  1.68199987 -0.58609664]
```

Output Parameters and Desirability

- Using these best parameters, the overall desirability and the predicted values for conversion and activity can be calculated as follows:

```
Best Overall Desirability: 0.9425092694688632
Conversion pred(x): 95.10150374903237
Activity pred(x): 57.49999992427212
```

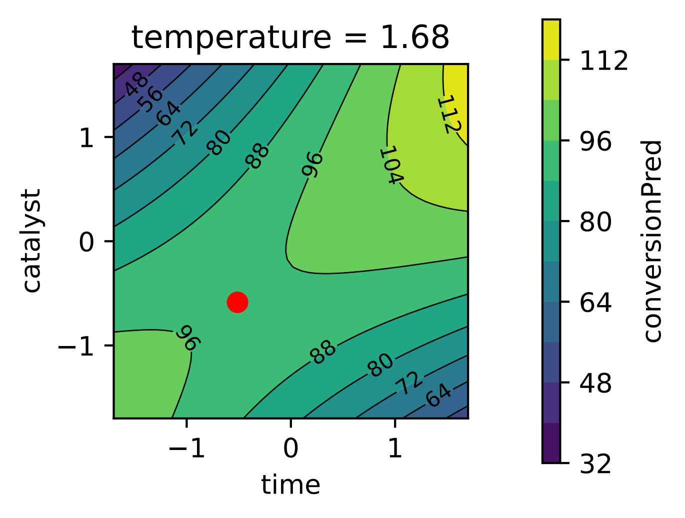


Figure 8: Response surface for percent conversion. Temperature fixed at the best value.

Response Surface for the Thermal Activity Model

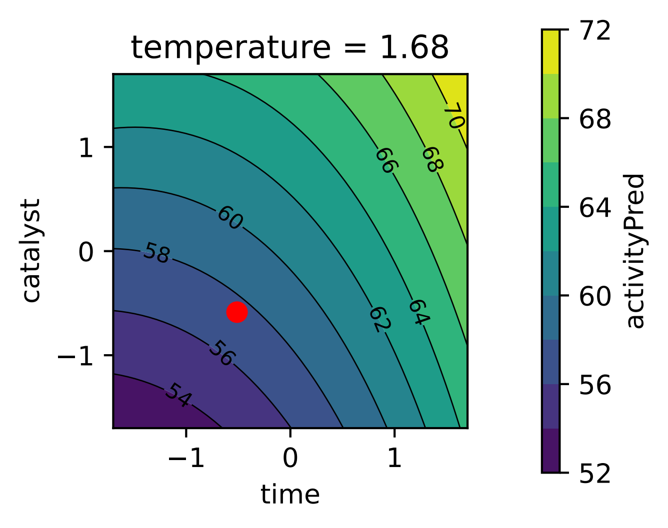


Figure 9: Response surface for thermal activity. Temperature fixed at the best value.

Analysis of the Nelder-Mead Optimization Results

Analysing the Best Values From the Nelder-Mead Optimizer

- Objective function values for the best parameters found by the optimizer are:
  - **`conversion`** = 95.1
  - **`activity`** = 57.5
- Percent conversion should be maximized (**`conversionD = DMax(80, 97)`**).
  - Obtained a value of 95.1, close to the maximum value of 97.
- **`thermal activity`** not maximized, but close to target (**`activityD = DTarget(55, 57.5, 60)`**).
  - Obtained a value of 57.5, exactly the target value.

Using spotpython for Surrogate-Model Based Optimization

- Define the desirability objects (identical to the previous step)
- Setting up the spotpython optimization function:

```
def fun_desirability(X, **kwargs):
    y = fun_myer16a(X)
    conversionD = DMax(80, 97)
    activityD = DTarget(55, 57.5, 60)
    overallD = DOverall(conversionD, activityD)
    overall_desirability = overallD.predict(y, all=False)
    return 1.0 - overall_desirability
```

- **spotpython** uses minimization, but desirability should be maximized, **fun_desirability** is returns **1 - overall_desirability**.

Simple test of the Desirability Function

We can test the function:

```
X = np.array([[0, 0, 0], best.x])
print(f"Objective function values: {fun_desirability(X)}")
```

```
Objective function values: [0.93797534 0.05749073]
```

`spotpython`: Swiss Army Knife for Surrogate-Model Based Optimization

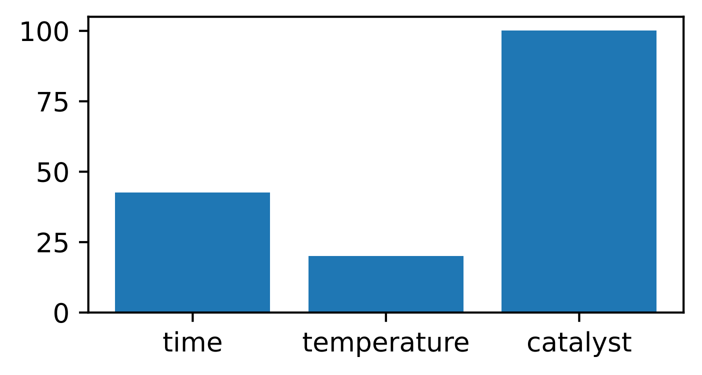
```
fun_control = fun_control_init(
              lower = np.array( [-1.7] * 3),
              upper = np.array([1.7] * 3),
              var_name = ["time", "temperature", "catalyst"],
              fun_evals= 50,
              show_progress=False
)
S = Spot(fun=fun_desirability,
         fun_control=fun_control)
```

```python
S.run()
print(f"Best Desirability: {1.0 - S.min_y}")
```

```
Experiment saved to 000_res.pkl
Best Desirability: 0.949094088557101
```

Importance Plot Considering the Overall Desirability

```
S.plot_importance(figsize=(4,2))
```



```
time:   42.47269227441938
temperature:   20.043761981496203
catalyst:   100.0
```

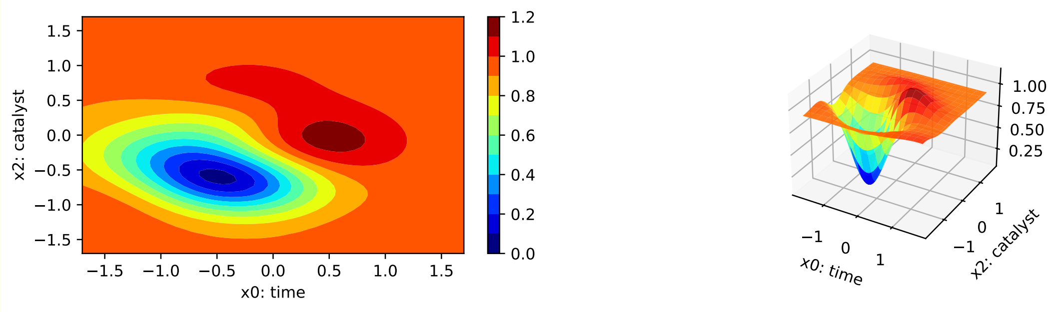


Figure 10: Contour plot of the overall desirability for the important parameters

Case Study 1: Results

Language	Algorithm	Overall Desirability	Conversion	Activity
R	Nelder-Mead	0.9425	95.10	**57.50**
Python	Nelder-Mead	0.9425	95.10	**57.50**
Python	**SMBO**	**0.9449**	**95.37**	**57.50**

- Python implementation matches R package exactly
- All methods achieve perfect target thermal activity
- **SMBO finds superior solution** with slightly higher desirability

4. Case Study 2: ML Application

Neural Network HPO Problem

Application Context

- PyTorch neural network on Diabetes regression dataset
- Feasibility study of MOO in hyperparameter tuning

Competing Objectives

- **Validation Loss** (minimize)
- **Number of Epochs** (minimize)

The Trade-off

- Few epochs → undertrained models
- Many epochs → computational waste

Search Space

11 mixed-type hyperparameters:

- Hidden layer sizes
- Activation functions
- Optimizers, dropout
- Learning rates
- ...

Experimental Comparison

1. **Single-Objective (Baseline)**
   - Optimize validation loss only
   - Ignore number of epochs (computational cost)
2. **Weighted-Sum (Common Practice)**
   - Linear combination: $w_1 \times \text{loss} + w_2 \times \text{epochs}$
   - Problems: Weight sensitivity, scaling issues
3. **Desirability Function (Proposed)**
   - **`lossD = DMin(low=10, high=6000)`**
   - **`epochsD = DMin(low=32, high=64)`**
   - Concrete, interpretable specifications

spotpython for Hyperparameter Tuning (Single-Objective)

- `Spot` object is created.
  - Calling the method `run()` starts the hyperparameter tuning process.

```
module_name: light
submodule_name: regression
model_name: NNLinearRegressor
Result file 0000_no_mo_res.pkl exists. Loading the result.
Loaded experiment from 0000_no_mo_res.pkl
```

```
<spotpython.spot.spot.Spot at 0x3222cb860>
```

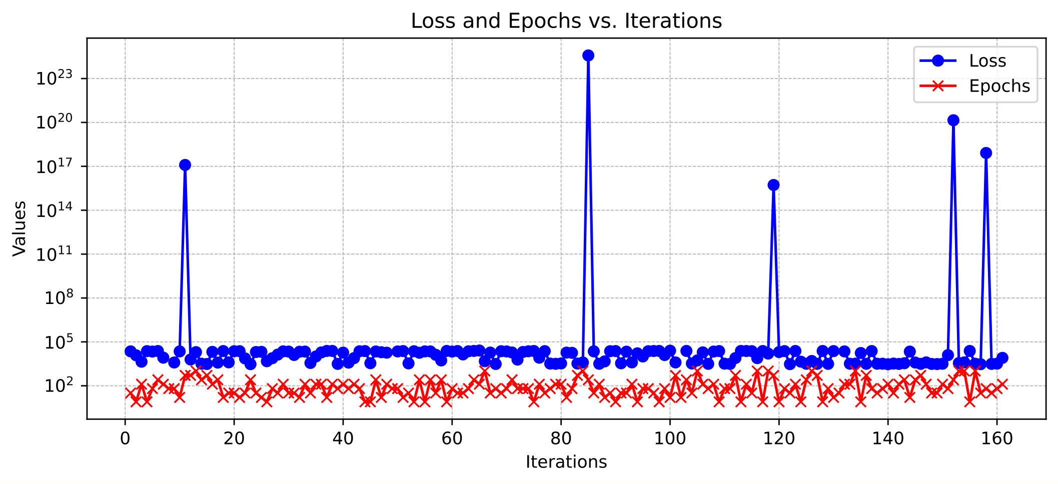


Figure 11: Results of the hyperparameter tuning process. Loss and epochs are plotted versus the function evaluations

```
module_name: light
submodule_name: regression
model_name: NNLinearRegressor
```

The remaining code is identical to the single-objective approach. The only difference is that the **fun_mo2so** argument is set to the **aggregate** function.

```
Result file 0001_aggregate_res.pkl exists. Loading the result.
Loaded experiment from 0001_aggregate_res.pkl
```

```
<spotpython.spot.spot.Spot at 0x33330fef0>
```

2. Results of the Weighted-Sum Approach

```
min y: 5918.18486328125
l1: 3.0
epochs: 9.0
batch_size: 8.0
act_fn: 2.0
optimizer: 1.0
dropout_prob: 0.013147860245895003
lr_mult: 3.9207231811540493
patience: 4.0
batch_norm: 0.0
initialization: 1.0
```

2. Weighted Multi-Objective Function Approach

- Weighted MOO approach results in a validation loss of **5824** and **64** $(= 2^6)$ epochs.
- Although the number of epochs is smaller than in the single-objective approach, the validation loss is larger.
- Inherent problem of weighted multi-objective approaches, because the deteriination of **"good"** weights is non-trivial.

Setting Up the Desirability Function

- Desirability function is defined as follows:

```python
def desirability(y):
    from spotdesirability.utils.desirability import DOverall, DMin
    lossD = DMin(10, 6000)
    epochsD = DMin(32, 64)
    overallD = DOverall(lossD, epochsD)
    overall_desirability = overallD.predict(y, all=False)
    return 1.0 - overall_desirability
```

Plotting the Desirability Functions

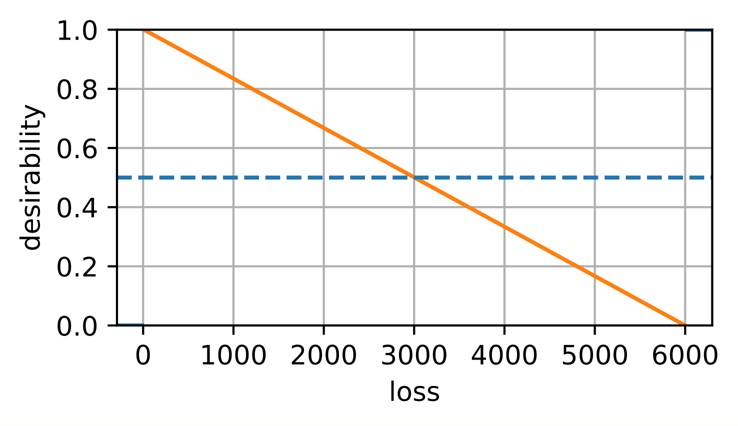


Figure 12: The desirability function for the loss outcome.

Plotting the Desirability Functions for Epochs

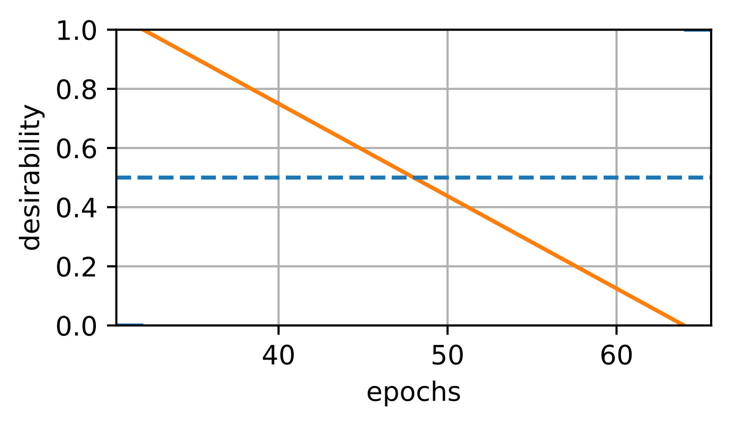


Figure 13: The desirability function for the epochs outcome.

Performing the Hyperparameter Tuning Run

Calling pyspot

```
module_name: light
submodule_name: regression
model_name: NNLinearRegressor
Result file 0002_res.pkl exists. Loading the result.
Loaded experiment from 0002_res.pkl
<spotpython.spot.spot.Spot at 0x3204874a0>
```

Case Study 2: Pareto Front Visualization

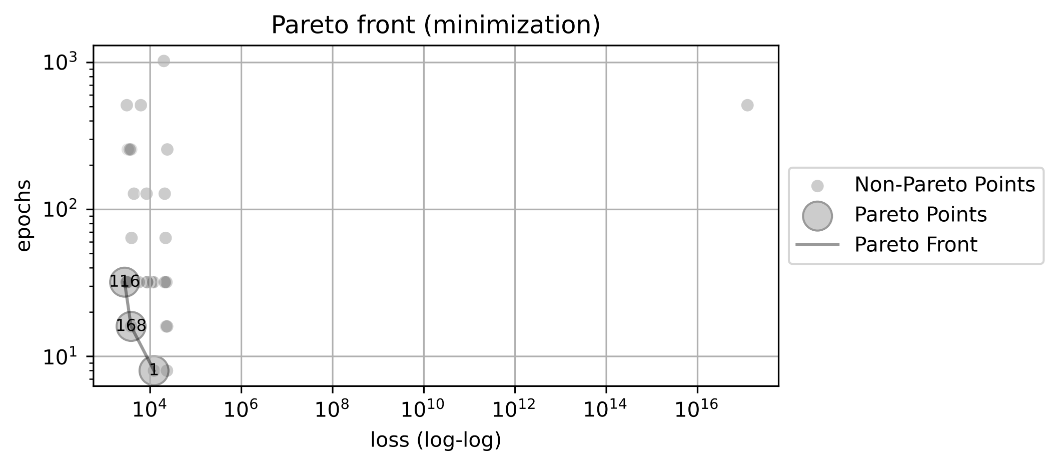


Figure 14

Case Study 2: Pareto Front Visualization

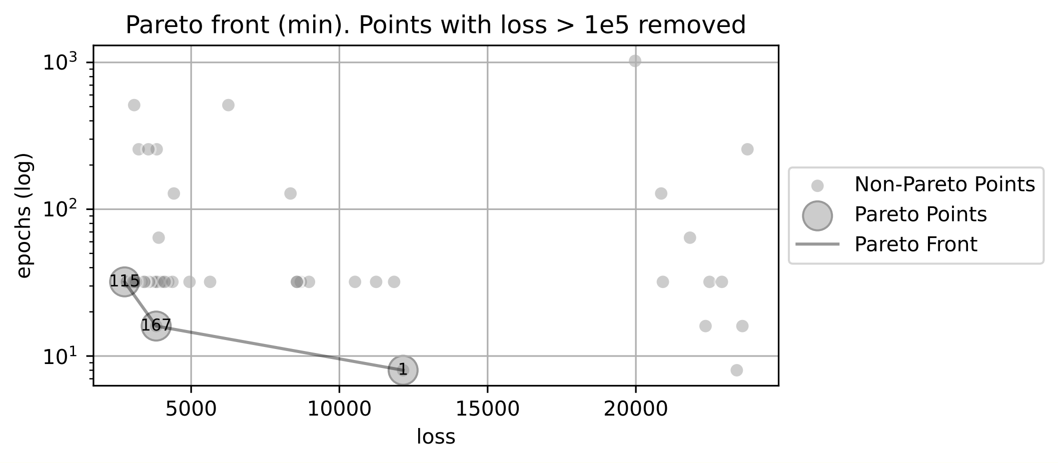


Figure 15

Case Study 2: Results

Algorithm	Validation Loss (min!)	Epochs (min!)	Performance
Single-Objective	**2890**	1024	Best accuracy, huge cost
Weighted-Sum	5824	64	Poor solution
Desirability	**2960**	**32**	**Superior trade-off**

- Desirability achieves **nearly identical accuracy**
- **Reduction** in computational cost
- Acts as intelligent, budget-aware search space pruner

5. Analysis & Discussion

Advantages of Desirability Functions

Intuitive & Practitioner-Focused

- Direct translation of engineering specs
- "Error rate < 5%"
- "Latency < 50ms"
- "Cost < $10/unit"
- No abstract weight assignment

Single, Actionable Solution

- Avoids decision paralysis
- Industry prefers definitive solutions
- Ready for deployment
- Best combined preference satisfaction

Critical Limitations

Scalarization Blind Spots:

- Cannot find solutions in non-convex Pareto regions
- MOEAs superior for complete front mapping

Sensitivity to Outliers:

- Geometric mean sensitive to extreme values
- Single poor objective → overall desirability collapse

The Plateau Problem:

- Zero desirability creates flat landscapes
- No gradient information for optimizers
- Navigation challenges in unacceptable regions

Future Work: Enhancements

1. Synergy with MOEAs
   - Step 1: MOEAs generate complete Pareto front
   - Step 2: Desirability selects best solution
   - Combines exploration power + preference articulation
2. Bio-Inspired Modifications
   - "Leaky" desirability functions (inspired by Leaky ReLU)
   - Small non-zero scores in unacceptable regions
   - Provides optimization signals on plateaus

Future Work: Continued

3. **Rigorous Benchmarking**
    - Compare against advanced scalarization techniques
    - Augmented Tchebycheff functions
    - Test on non-convex Pareto fronts
4. **Broader Model Exploration**
    - Beyond Kriging: Random Forests, Neural Networks
    - Identify optimal surrogate-desirability pairings
    - Leverage **`spotpython`** framework

6. Conclusion

Summary & Impact

- Desirability functions: **underappreciated** in academic ML
- **Practical, powerful method** for multi-objective HPO
- **Key contributions:**
  - **`spotdesirability`** Python package
  - Easier, more intuitive than weighted-sum methods
  - Superior trade-offs in practical applications
  - 32-fold computational savings demonstrated

Impact: Valuable addition to modern multi-objective optimization toolkit

Thank You! Questions?

Contact Information

- **Updates and Jupyter Notebook of this Presentation:** https://thk-ai.de

Key References

- Harington (1965) - Original desirability functions
- Derringer and Suich (1980) - Geometric mean formulation
- Kuhn (2016) - Reference implementation in R
- Bartz-Beielstein (2025) - ArXiv Paper

Software

- **Code: `spotdesirability`** Python package available on GitHub and PyPi:
  - GitHub[https://github.com/sequential-parameter-optimization/spotdesirability]

References

Bartz-Beielstein, Thomas. 2025. "Multi-Objective Optimization and Hyperparameter Tuning With Desirability Functions." *arXiv e-Prints*, March, arXiv:2503.23595. https://doi.org/10.48550/arXiv.2503.23595.

Derringer, G., and R. Suich. 1980. "Simultaneous Optimization of Several Response Variables." *Journal of Quality Technology* 12: 214–19.

Harington, J. 1965. "The Desirability Function." *Industrial Quality Control* 21: 494–98.

Kuhn, M. 2016. "Desirability: Function Optimization and Ranking via Desirability Functions." https://cran.r-project.org/package=desirability.

Myers, R. H., D. C. Montgomery, and C. M. Anderson-Cook. 2016. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments.* John Wiley & Sons.