UNIVERSITÄT BIELEFELD

Faculty of Business Administration and Economics

DECISION ANALYTICS

Combining OR and Data Science

Summer Term 2022

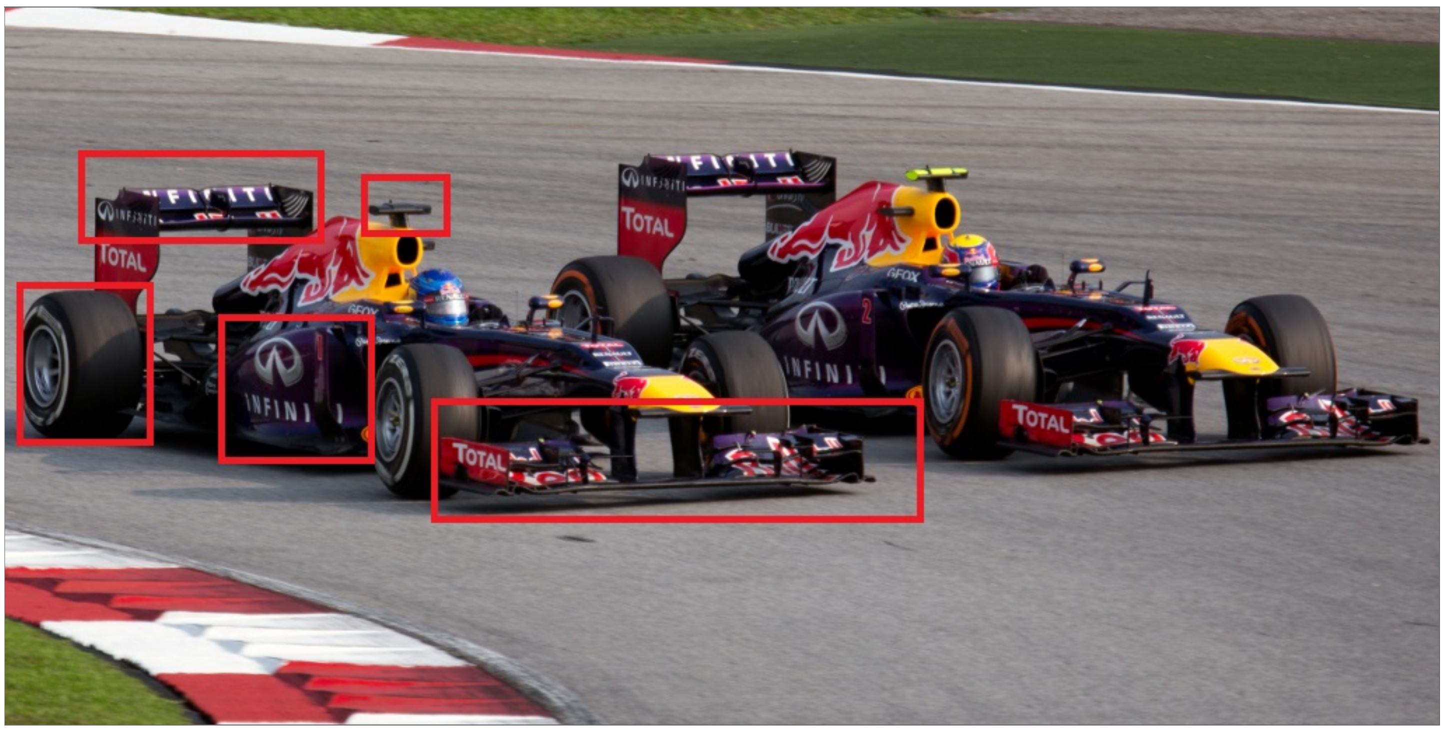
8. Algorithm Configuration

J-Prof. Dr. Michael Römer, Till Porrmann
Decision Analytics Group | Bielefeld University

What should be the Deadline for Homework 3?

- mid-July (end of lecturing phase)
- beginning of August (after the exams)
- end of August
- end of September?

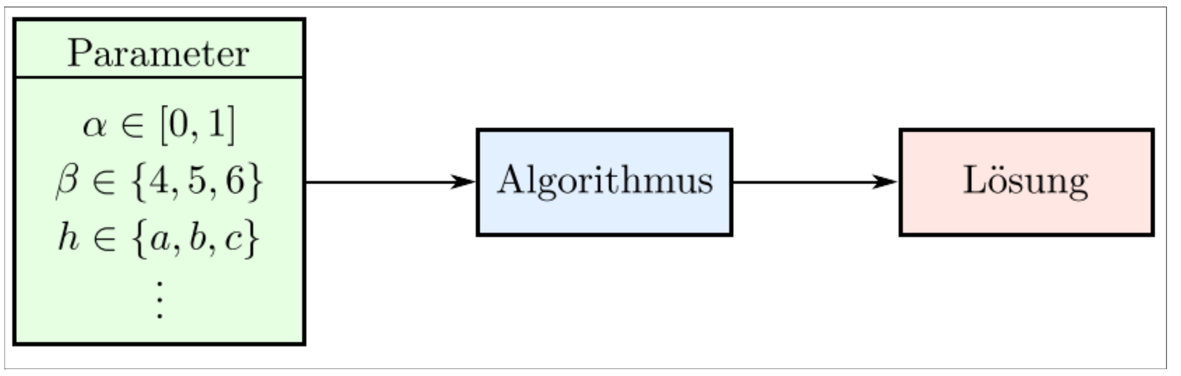
Motivation: Consider a Race Car..



- a race car has lots of design parameters / settings to be decided upon
- there is not a single best car configuration
    - for each race track
    - for all weather conditions

Tuning Algorithm Parameters

Like race cars, algorithms have many parameters that influence their performance:



- with good parameter settings, problems can be solved faster (or better)

This Meeting: Algorithm Configuration

In this meeting, we deal with **algorithm configuration**

- understanding the relevance of algorithm configuration
- basic ideas and concepts
- introducing a case study
- automating algorithm configuration using black box (Bayesian) optimization
- addressing key challenges in algorithm configuration:
    - avoiding overfitting
    - reducing configuration time

Case Study: Solving Chance-Constrained Programs Faster

- imagine a consulting project with a small belt manufacturing company where we want to use **joint chance constrained programming** (CCP) (see part 6 for the implementation)
- CCP models with binary variables can be very hard to solve
- commercial solvers are very powerful, but only free for academic use
    - for small commercial projects, they tend to be too expensive: **a single Gurobi license is about $ 10 000**
- we will instead use the **open source** solver **CBC**

How can we improve CBC's performace?

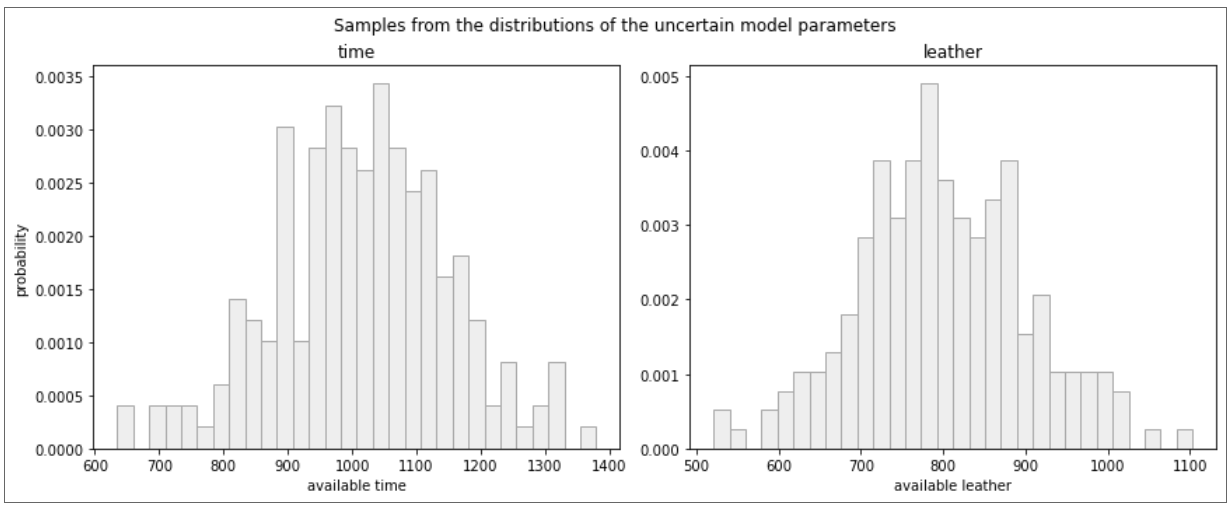
- use algorithm configuration to improve CBC's performance for CCP models with 200 scenarios
- we would like to achieve a similar performance as Gurobi
    - in my tests, Gurobi was about 3x as fast as CBC

Case Study: Uncertain Availability of Time and Leather

- in the case study, we assume that both time and leather availability is uncertain
- there is no possibility to resort to extra hours
- we want to find a production plan that is feasible with probability of 95\%

We create 200 samples for both time and leather:

In [3]:
```
n_scenarios = 200

np.random.seed(seed=1)
time_available_dist = stats.norm(1000,150)
leather_available_dist = stats.norm(800,100)

samples_time_available = time_available_dist.rvs(n_scenarios)
samples_leather_available = leather_available_dist.rvs(n_scenarios)


fig, (ax1,ax2) = plt.subplots(1, 2,figsize=(12, 4.8), constrained_layout=True)

fig.suptitle("Samples from the distributions of the uncertain model parameters")
ax1.set_ylabel("probability")


ax1.set_title("time")
count, bins, ignored = ax1.hist( samples_time_available , bins=30, density=True, color='#EEEEEE', edgecolor="#AAAAAA")
ax1.set_xlabel("available time")

ax2.set_title("leather")
count, bins, ignored = ax2.hist( samples_leather_available , bins=30, density=True, color='#EEEEEE', edgecolor="#AAAAAA")
ax2.set_xlabel("available leather")

plt.show()
```



Case Study: Python Implementation of the Model Creation

Below, the creation of the mathematical is wrapped into a function that takes

- the time and leather samples
- and the solver as parameters

In [4]:

```python
def build_model_with_samples(samples_time_available, samples_leather_available, solver):

    n_scenarios = len(samples_time_available)
    scenarios = np.arange(n_scenarios)

    #sets
    belt_types = np.arange(2)

    #deterministic parameters
    profit_contribution = [2, 1.5]
    time_consumption = [2, 1]
    bucks_available = [400, 700]

    #probability of each scenario
    prob = np.full((n_scenarios), 1.0/n_scenarios)

    alpha = 0.95
    #bigM is used for switching off constraints by adding it to the resource capacity in a scenario
    #if it is chosen as difference between the max and min capacity, it is definitely large enough to switch off the constraint
    bigM = [max(samples_time_available)-min(samples_time_available),max(samples_leather_available)-min(samples_leather_available)]

    m = Model(solver_name=solver)
    production = [ m.add_var(var_type=INTEGER) for i in belt_types ] # production variable
    is_scenario_violated =  [ m.add_var(var_type=BINARY) for s in scenarios ] # binary variable indicating if scenario is vilated or not

    m.objective =  maximize( xsum(profit_contribution[b] * production[b] for b in belt_types) ) # objective function

    for s in scenarios:
        m += sum (time_consumption[b]*production[b] for b in belt_types) <= samples_time_available[s] + bigM[0]*(is_scenario_violated[s]) # chance constraint for time
        m += sum (production[b] for b in belt_types) <= samples_leather_available[s] + bigM[1]*(is_scenario_violated[s]) # chance constraint for leather

    m += sum ( prob[s]*is_scenario_violated[s] for s in scenarios) <= 1-alpha   # enforcing the maximium allowed violation probability

    return m
```

Let us Try the Different Solvers

- CBC

In [5]:
```
model_instance_cbc = build_model_with_samples(samples_time_available, samples_leather_available, solver=CBC)
start = time.time()
model_instance_cbc.optimize()
print (f'Solution time with CBC: {time.time()-start:0.2f} seconds' )
```

```
Solution time with CBC: 3.18 seconds
```

- Gurobi

In [6]:
```
model_instance_gurobi = build_model_with_samples(samples_time_available, samples_leather_available, solver=GRB)
start = time.time()
model_instance_gurobi.optimize()
print (f'Solution time with Gurobi: {time.time()-start:0.2f} seconds' )
```

```
Solution time with Gurobi: 0.87 seconds
```

Question: Can we tune the (free) solver CBC to become competitive with Gurobi for this problem?

Algorithm Configuration Basics

What are Parameters?

Parameters are settings of an algorithm (or solver) that change the way an algorithm solves a problem.

What are Parameters?

Parameters are settings of an algorithm (or solver) that change the way an algorithm solves a problem.
There are configurable and non-configurable parameters

Configurable

- simulated annealing start temperature
- population size in a genetic algorithm (GA)
- branching heuristic in Branch-and-Bound
- probability of a GA mutation

Non-Configurable

- random seed
- instance to be solved
- numerical accuracy in an exact solver
- maximum solution time

Parameter Types

- Continuous: e.g. $[1.0, 5.0]$
- Discrete: e.g. $\{1, \ldots, 10\}$
- Categorical: e.g. $\{a, b, c, d\}$
- Ordinal: e.g. $\{low, medium, high\}$ (Ordered set)

Parameter Examples: Genetic Algorithm

A genetic algorithm could have the following parameters:

Parameter	Description	Considered values
population	Population size	1..infinity
recombProb	Recombination probability	0-1.0
mutProb	Mutation probability	0-1.0
select	Selection heuristic	'Roulette', 'Tournament', 'Random'
fitness-scaling	Use fitness scaling?	'yes', 'no'

discrete , continuous, categorical

MIP Solver Parameters Used in our Case Study

In our case study, we will configure the MIP solver *CBC*

- CBC has many parameters, but we will only use a subset
- we will use the following parameters for configuration:

Parameter	Description	Considered values
nodeStrategy	Strategy for selecting nodes to branch on	'depth', 'fewest','hybrid'
strongBranching	# vars. to consider in strong branching	0-10
trustPseudoCosts	str. br. evals before trusting pseudo costs	0-10
preprocess	Switch for model preprocessing	'on', 'off'
presolve	Switch for presolve reductions	'on', 'off','sos'
cutsOnOff	Switch for cut generation (all cuts)	'on', 'off'
heuristicsOnOff	Turn (all) heuristics on/off	'on', 'off'

discrete , continuous, categorical

Let us Start Playing with CBC Parameters in our Case Study

- first, we write a function for setting parameters

In [7]:
```
def set_cbc_parameter(model_instance, parameter, value):
    cbc_set_parameter(model_instance.solver,parameter,str(value)) # the function cbc_set parameter only takes string, we thus convert numbers to strings here
```

- now, we try out the parameter 'node_strategy' to 'depth'
- observe: we first make a copy to avoid that CBC uses a *warm start*, that is, starts from the previous state

In [8]:
```
model_instance_cbc = model_instance_cbc.copy() # reset the model to a fresh state (to avoid warm start!)

set_cbc_parameter(model_instance_cbc, 'node_strategy','depth') # as an example, let us try depth-first-search

start = time.time()
model_instance_cbc.optimize()
print (f'Solution time: {time.time()-start:0.2f} seconds' )
```

```
Solution time: 2.50 seconds
```

Making Experimentation Easier: An Evaluation Function

- we write a function that takes a model instance and a set of named parameter-value pairs
- and performs the complete evaluation based on a threshold of 30 seconds (returning the PAR10-performance, that is 10*30 in case of timeout

In [9]:

```python
def evaluate_parameters_on_instance(model_instance, **params):

    model = model_instance.copy() # we make a copy of the model here since
    model.seed = 42 # set the seed to make multiple runs comparable

    for param, value in params.items(): # iterate over the names and values of the parameters
        cbc_set_parameter(model.solver,param,str(value)) # the function cbc_set parameter takes only strings

    time_limit_seconds = 30
    start = time.time()
    status = model.optimize(max_seconds=time_limit_seconds) # we set the runtime limit to 30

    if status == OptimizationStatus.OPTIMAL:
        return (time.time()-start)
    else: #  if the limit is exceeded, we return the time_limit *10, this is the PAR10 logic
        return time_limit_seconds * 10
```

- let's see this in action:

In [10]:

```python
performance = evaluate_parameters_on_instance(model_instance_cbc, nodeStrategy='hybrid', heuristicsOnOff='off', preprocess='off')
print (f'Solution time: {performance:0.2f} seconds' )
performance = evaluate_parameters_on_instance(model_instance_cbc, nodeStrategy='hybrid', heuristicsOnOff='off', strongBranching =0.4, presolve='off', preprocess='off',trustPseudoCosts=0)
print (f'Solution time: {performance:0.2f} seconds' )
```

```
Solution time: 2.24 seconds
Solution time: 2.08 seconds
```

Exercise: Play with the Parameters!

Play with the following parameters! What is your best parameter combination?

Parameter	Description	Considered values
nodeStrategy	Strategy for selecting nodes to branch on	'depth', 'fewest','hybrid'
strongBranching	# vars. to consider in strong branching	0-10
trustPseudoCosts	str. br. evals before trusting pseudo costs	0-10
preprocess	Switch for model preprocessing	'on', 'off'
presolve	Switch for presolve reductions	'on', 'off','sos'
cutsOnOff	Switch for cut generation (all cuts)	'on', 'off'
heuristicsOnOff	Turn (all) heuristics on/off	'on', 'off'

discrete , continuous, categorical

```
In [ ]:
```

Why Bothering with Algorithm Configuration?

Why would we care about algorithm configuration?

- in many cases, algorithms are not only used once, but are part of a repeating planning workflow
    - e.g. monthly staff scheduling, daily power plant dispatch
- in these cases, the instance to solve are very similar
- sometimes, runtime is critical to allow testing multiple scenarios
- in many cases, configuration can lead to big runtime reductions
    - often, only a fraction (say, 1/10) of original runtime is needed after configuration

Goals of Algorithm Configuration

There are two main types of algorithm configuration:

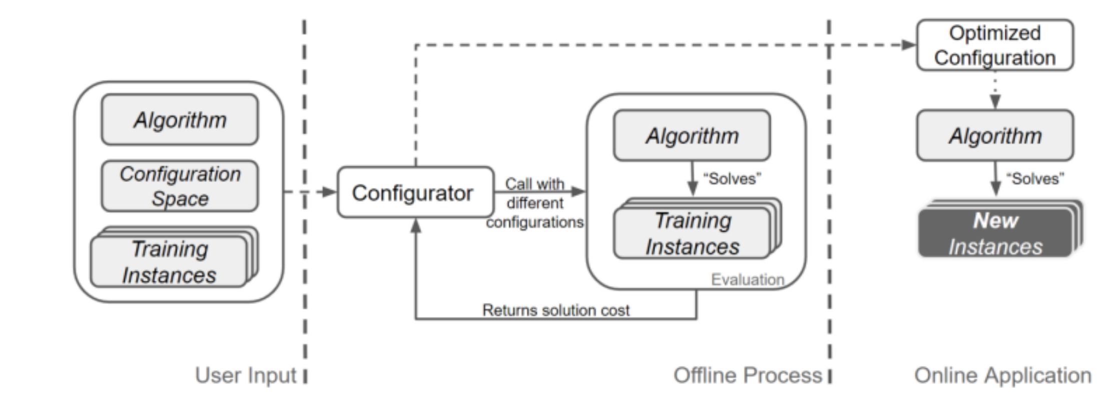
Runtime configuration

- Concerned with minimizing the runtime of a target algorithm.

Quality configuration

- Concerned with minimizing an objective function seen only through the evaluation of an algorithm.
- This is generally the fitness/objective function of an optimization algorithm or the solution of a simulation.

Offline Configuration and Online Application



Source: Lindauer, Biedenkapp(2020): Algorithm Configuration Tutorial

Homogeneous vs Heterogeneous Instance Sets

Homogeneous datasets

Homogeneous instance sets contain instances with similar structures, allowing algorithms to use similar parameters on all of the instances to solve them.

Heterogeneous instance sets

Heterogeneous instance sets consist of two or more homogeneous sub-sets, but which instance belongs to which sub-aset is unclear. Different parameter values are required for different instances.
→ This case may require a combination of algorithm selection (e.g. based on clustering) and algorithm configuration!

Manual Configuration?

Can't we just **manually** tune parameters?
We can, but it is

- time-consuming
- error-prone and
- tedious

In [11]:

```
performance = evaluate_parameters_on_instance(model_instance_cbc, nodeStrategy='hybrid', heuristicsOnOff='off')
print (f'Solution time: {performance:0.2f} seconds' )

performance = evaluate_parameters_on_instance(model_instance_cbc, nodeStrategy='hybrid', heuristicsOnOff='on')
print (f'Solution time: {performance:0.2f} seconds' )

performance = evaluate_parameters_on_instance(model_instance_cbc, nodeStrategy='fewest', heuristicsOnOff='on')
print (f'Solution time: {performance:0.2f} seconds' )
```

```
Solution time: 2.14 seconds
Solution time: 2.63 seconds
Solution time: 2.62 seconds
```

In many cases, it is more effective to **automate** algorithm configuration!

Automating the Configuration of Algorithms

The goal of the following part is to get an introduction to the topic of **algorithm configuration**
In particular, will discuss

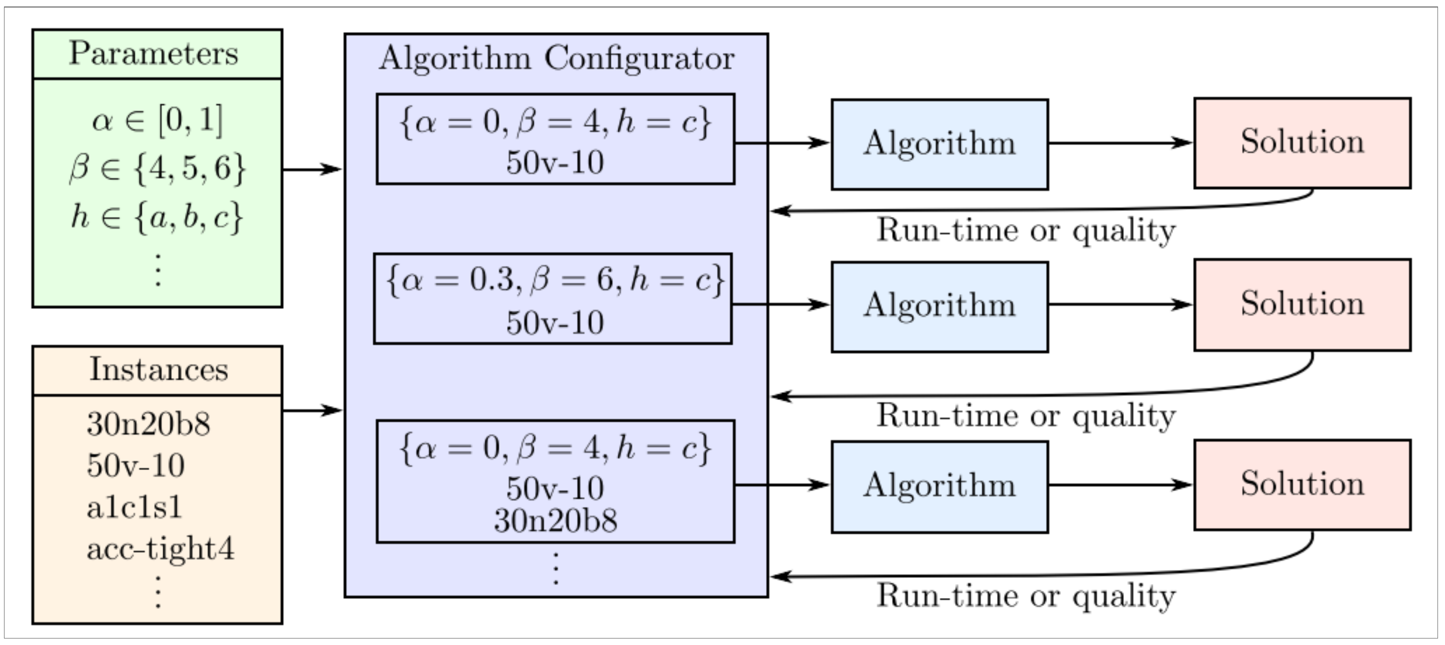
- the idea and the challenges of automatic algorithm configuration
- how to use black box (Bayesian) optimization for automatic algorithm configuration
- how to address the challenge of overfitting
- how to speed up algorithm selection by using early stopping

As in the last weeks, we will

- use a case study (see above) to illustrate the concepts learned
- provide implementations of an algorithm configuration approach for this case study in this Jupyter Notebook

Automatic Algorithm Configuration

- automatic algorithm configurators **automatically** try different parameter settings on selected instances and measure the performance of the target algorithm.
- they perform the search in a somewhat intelligent way



Challenges

Automated algorithm configurators have to overcome the following challenges (among many others):

- Which instances to evaluate parameterizations on
- Which parameterizations to test
- Testing parameterizations is **expensive**
- Big risk of overfitting

Why is algorithm configuration different from other optimization problems?

- The objective function in most metaheuristics is easy to evaluate (that is: fast)
- Algorithm configurators must execute the target algorithm with a parameterization **on multiple instances** to evaluate it.

Addressing the Challenges of Algorithm Configuration: Case Study

In the remainder of this meeting, we will see how we can address some of the challenges fusing "standard" tools in our case study:

- using **Bayesian Optimization** to search the space of configurations
- optimize using **multiple training instances** to avoid overfitting
- use a simple **early stopping strategy** to reduce the time spent evaluating bad configurations

Black Box Optimization using Bayesian Optimization

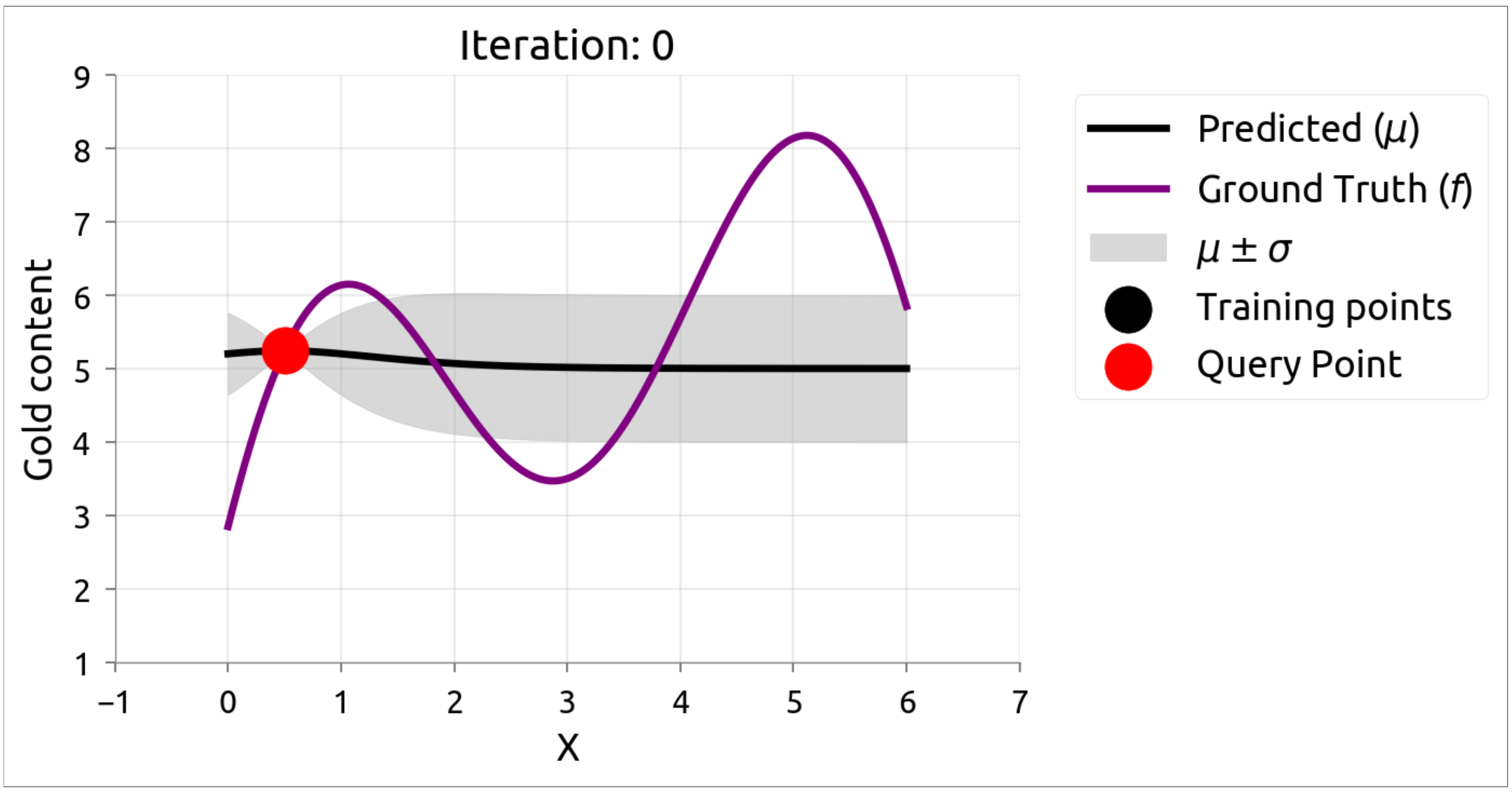
A **black box optimizer** treats the optimization problem as "black box"
- it does not make any assumption regarding problem structure (e.g. linearity)
- it only requires a mechanism that returns the value of a solution / configuration

Key Idea of Bayesian Optimization
- use a machine learning model (mostly: Gaussian Processes) as a surrogate model for the configuration space
- initialize the model using few (e.g. random) configurations
- update the surrogate model after each new evaluation

Illustrating Bayesian Optimization

Key Idea of Bayesian Optimization

- use a machine learning model (mostly: Gaussian Processes) as a surrogate model for the configuration space
- initialize the model using few (e.g. random) configurations
- update the surrogate model after each new evaluation

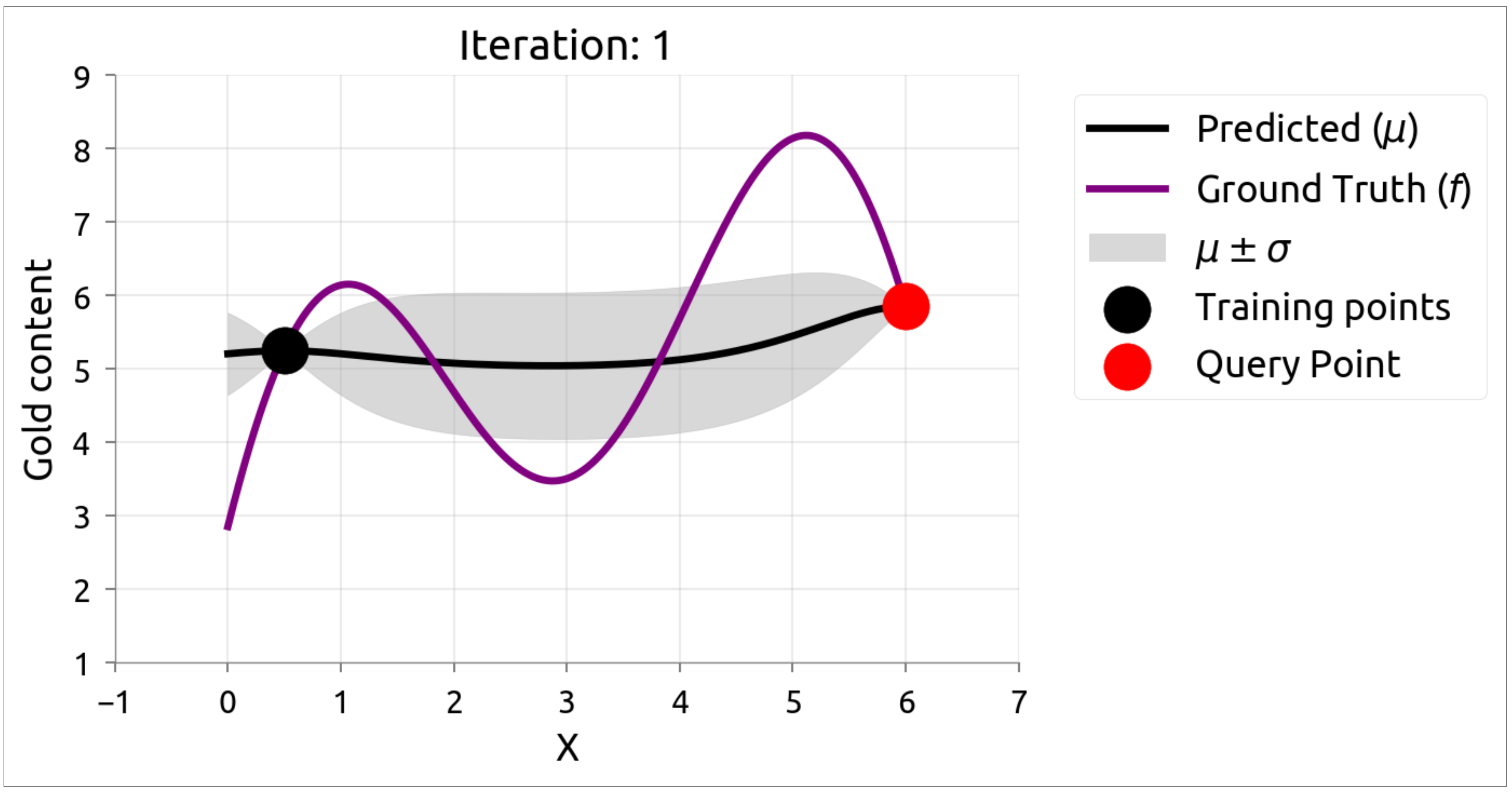


Source: **https://distill.pub/2020/bayesian-optimization/**

Illustrating Bayesian Optimization

Key Idea of Bayesian Optimization

- use a machine learning model (mostly: Gaussian Processes) as a surrogate model for the configuration space
- initialize the model using few (e.g. random) configurations
- update the surrogate model after each new evaluation

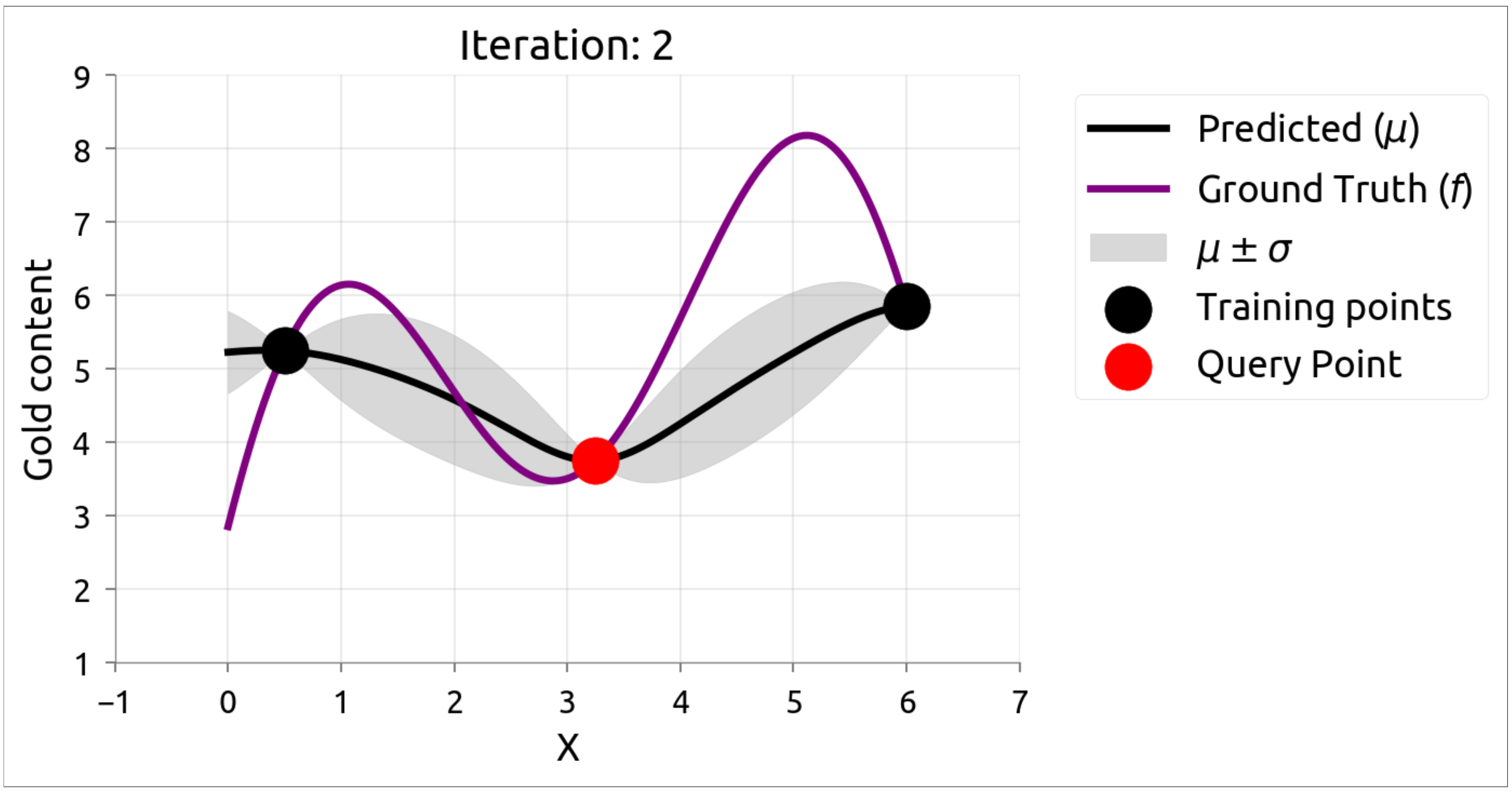


Source: **https://distill.pub/2020/bayesian-optimization/**

Illustrating Bayesian Optimization

Key Idea of Bayesian Optimization

- use a machine learning model (mostly: Gaussian Processes) as a surrogate model for the configuration space
- initialize the model using few (e.g. random) configurations
- update the surrogate model after each new evaluation

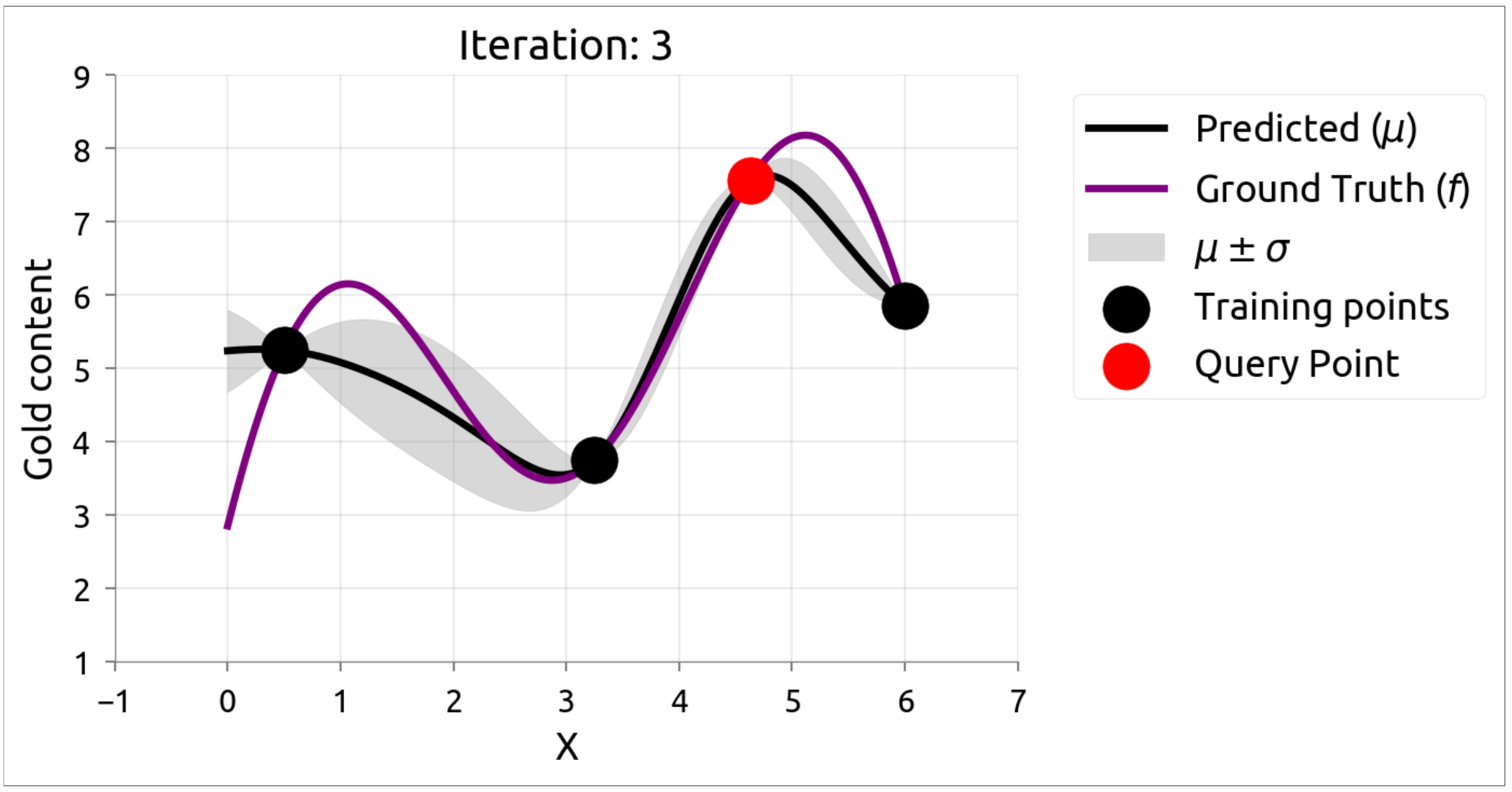


Source: **https://distill.pub/2020/bayesian-optimization/**

Illustrating Bayesian Optimization

Key Idea of Bayesian Optimization

- use a machine learning model (mostly: Gaussian Processes) as a surrogate model for the configuration space
- initialize the model using few (e.g. random) configurations
- update the surrogate model after each new evaluation

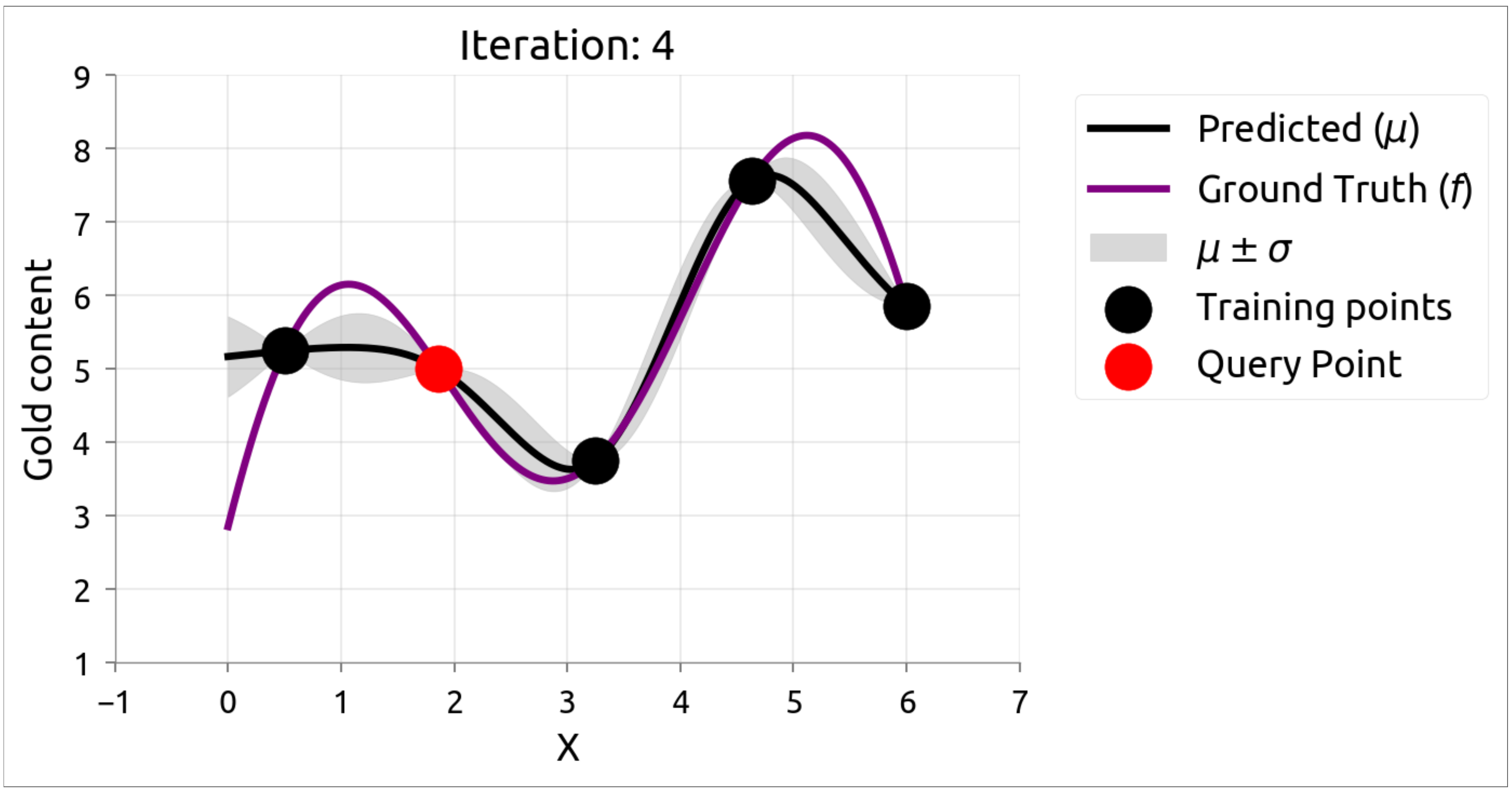


Source: **https://distill.pub/2020/bayesian-optimization/**

Illustrating Bayesian Optimization

Key Idea of Bayesian Optimization

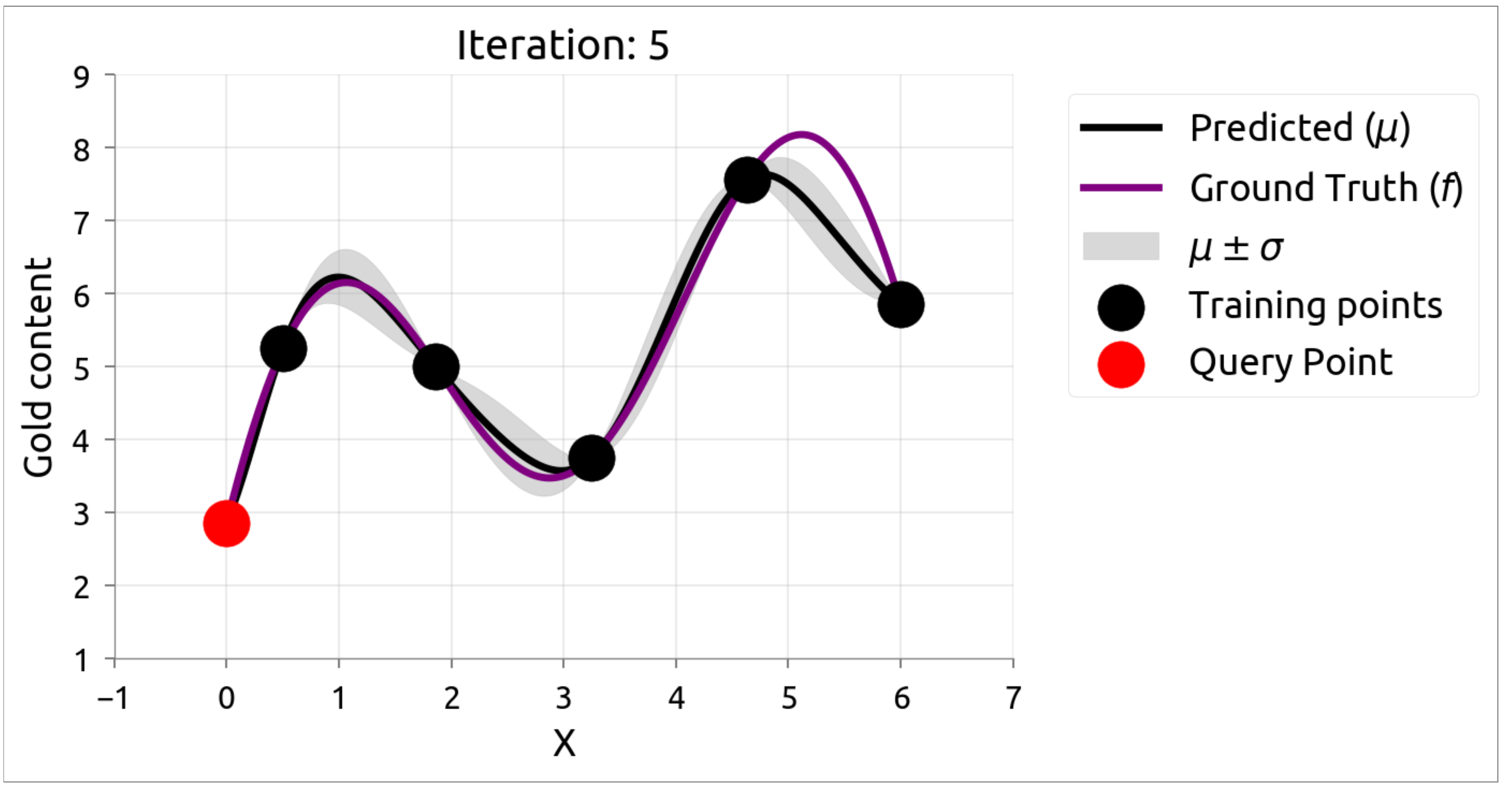
- use a machine learning model (mostly: Gaussian Processes) as a surrogate model for the configuration space
- initialize the model using few (e.g. random) configurations
- update the surrogate model after each new evaluation



Source: **https://distill.pub/2020/bayesian-optimization/**

Key Idea of Bayesian Optimization

- use a machine learning model (mostly: Gaussian Processes) as a surrogate model for the configuration space
- initialize the model using few (e.g. random) configurations
- update the surrogate model after each new evaluation



Source: **https://distill.pub/2020/bayesian-optimization/**

Scikit-Optimize: A Package for Black Box Optimization

A **black box optimizer** treats the optimization problem as "black box"

- it does not make any assumption regarding problem structure (e.g. linearity)
- it only requires a mechanism that returns the value of a solution / configuration

Scikit-Optimize

- is a package for black box optimization with different methods, in particular:
    - Random Search
    - **Bayesian Optimization**
- it can also be used for hyperparameter tuning of machine learning models

Scikit-Optimize: A Package for Black Box Optimization

A **black box optimizer** treats the optimization problem as "black box"

- it does not make any assumption regarding problem structure (e.g. linearity)
- it only requires a mechanism that returns the value of a solution / configuration

Scikit-Optimize

- is a package for black box optimization with different methods, in particular:
    - random search
    - **Bayesian Optimization**
- it can also be used for hyperparameter tuning of machine learning models

Recall: MIP Solver Parameters Used in our Case Study

In our case study, we will configure the MIP solver *CBC*

- CBC has many parameters, but we will only use a subset
- we will use the following parameters for configuration:

Parameter	Description	Considered values
nodeStrategy	Strategy for selecting nodes to branch on	'depth', 'fewest','hybrid'
strongBranching	# vars. to consider in strong branching	0-10
trustPseudoCosts	str. br. evals before trusting pseudo costs	0-10
preprocess	Switch for model preprocessing	'on', 'off'
presolve	Switch for presolve reductions	'on', 'off','sos'
cutsOnOff	Switch for cut generation (all cuts)	'on', 'off'
heuristicsOnOff	Turn (all) heuristics on/off	'on', 'off'

discrete , continuous, categorical

Defining the Parameter Space and the Evaluation Function

- using data structures from `scikit-optimize`, we define the parameter space

In [12]:
```
from skopt.space import Real, Integer, Categorical
from skopt.utils import use_named_args
from skopt.space import Space

space  = [Categorical(['hybrid','fewest','depth'], name='nodeStrategy'),
          Integer(0,10,name='strongBranching'),
          Integer(0,10,name='trustPseudoCosts'),
          Categorical(['on','off','sos'], name='preprocess'),
          Categorical(['on','off'], name='presolve'),
          Categorical(['on','off'], name='cutsOnOff'),
          Categorical(['on','off'], name='heuristicsOnOff')]
```

- then, we define the objective / evaluation function. For now, we only use a single instance for evaluation

In [13]:
```
model_instance_for_objective = model_instance_cbc

@use_named_args(space)
def performance_objective_single_instance(**params):
    return evaluate_parameters_on_instance(model_instance_for_objective,**params)
```

Running the Optimization

- let us now run the optimization
- note that by using the parameter `n_calls`, we limit the number of configurations evaluated to 50

In [14]:
```python
from skopt import gp_minimize

results_single_instance = gp_minimize(performance_objective_single_instance, space, verbose=True, n_calls=50, random_state=0)

print(f"Best configuration: {results_single_instance.x}")
print(f"Best performance in seconds: {results_single_instance.fun:0.2f}")
```

```
Iteration No: 1 started. Evaluating function at random point.
Iteration No: 1 ended. Evaluation done at random point.
Time taken: 1.6930
Function value obtained: 1.6530
Current minimum: 1.6530
Iteration No: 2 started. Evaluating function at random point.
Iteration No: 2 ended. Evaluation done at random point.
Time taken: 1.7757
Function value obtained: 1.7447
Current minimum: 1.6530
Iteration No: 3 started. Evaluating function at random point.
Iteration No: 3 ended. Evaluation done at random point.
Time taken: 2.1628
Function value obtained: 2.1408
Current minimum: 1.6530
Iteration No: 4 started. Evaluating function at random point.
Iteration No: 4 ended. Evaluation done at random point.
Time taken: 2.7101
Function value obtained: 2.6870
Current minimum: 1.6530
Iteration No: 5 started. Evaluating function at random point.
Iteration No: 5 ended. Evaluation done at random point.
Time taken: 0.6513
Function value obtained: 0.6273
Current minimum: 0.6273
Iteration No: 6 started. Evaluating function at random point.
Iteration No: 6 ended. Evaluation done at random point.
Time taken: 2.6595
Function value obtained: 2.6325
Current minimum: 0.6273
Iteration No: 7 started. Evaluating function at random point.
Iteration No: 7 ended. Evaluation done at random point.
Time taken: 3.8184
Function value obtained: 3.7964
Current minimum: 0.6273
Iteration No: 8 started. Evaluating function at random point.
Iteration No: 8 ended. Evaluation done at random point.
Time taken: 3.0262
Function value obtained: 2.9932
Current minimum: 0.6273
Iteration No: 9 started. Evaluating function at random point.
Iteration No: 9 ended. Evaluation done at random point.
Time taken: 3.0877
Function value obtained: 3.0617
Current minimum: 0.6273
Iteration No: 10 started. Evaluating function at random point.
Iteration No: 10 ended. Evaluation done at random point.
Time taken: 4.3278
Function value obtained: 3.1297
Current minimum: 0.6273
Iteration No: 11 started. Searching for the next optimal point.
Iteration No: 11 ended. Search finished for the next optimal point.
Time taken: 2.4063
Function value obtained: 1.0544
Current minimum: 0.6273
Iteration No: 12 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 12 ended. Search finished for the next optimal point.
Time taken: 2.0750
Function value obtained: 0.6210
Current minimum: 0.6210
Iteration No: 13 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 13 ended. Search finished for the next optimal point.
Time taken: 2.1908
Function value obtained: 0.6328
Current minimum: 0.6210
Iteration No: 14 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 14 ended. Search finished for the next optimal point.
Time taken: 2.1504
Function value obtained: 0.6484
Current minimum: 0.6210
Iteration No: 15 started. Searching for the next optimal point.
Iteration No: 15 ended. Search finished for the next optimal point.
Time taken: 2.0896
Function value obtained: 0.5646
Current minimum: 0.5646
Iteration No: 16 started. Searching for the next optimal point.
Iteration No: 16 ended. Search finished for the next optimal point.
Time taken: 1.9601
Function value obtained: 0.5691
Current minimum: 0.5646
Iteration No: 17 started. Searching for the next optimal point.
Iteration No: 17 ended. Search finished for the next optimal point.
Time taken: 2.6457
Function value obtained: 1.2797
Current minimum: 0.5646
Iteration No: 18 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 18 ended. Search finished for the next optimal point.
Time taken: 2.0551
Function value obtained: 0.5641
Current minimum: 0.5641
Iteration No: 19 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 19 ended. Search finished for the next optimal point.
Time taken: 1.8700
Function value obtained: 0.5740
Current minimum: 0.5641
Iteration No: 20 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 20 ended. Search finished for the next optimal point.
Time taken: 2.0268
Function value obtained: 0.5388
Current minimum: 0.5388
Iteration No: 21 started. Searching for the next optimal point.
Iteration No: 21 ended. Search finished for the next optimal point.
Time taken: 1.8407
Function value obtained: 0.5327
Current minimum: 0.5327
Iteration No: 22 started. Searching for the next optimal point.
Iteration No: 22 ended. Search finished for the next optimal point.
Time taken: 6.7599
```

```
Function value obtained: 5.3799
Current minimum: 0.5327
Iteration No: 23 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 23 ended. Search finished for the next optimal point.
Time taken: 2.0425
Function value obtained: 0.5615
Current minimum: 0.5327
Iteration No: 24 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 24 ended. Search finished for the next optimal point.
Time taken: 2.0892
Function value obtained: 0.5832
Current minimum: 0.5327
Iteration No: 25 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 25 ended. Search finished for the next optimal point.
Time taken: 2.1630
Function value obtained: 0.5470
Current minimum: 0.5327
Iteration No: 26 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 26 ended. Search finished for the next optimal point.
Time taken: 2.2179
Function value obtained: 0.5480
Current minimum: 0.5327
Iteration No: 27 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 27 ended. Search finished for the next optimal point.
Time taken: 2.0248
Function value obtained: 0.5578
Current minimum: 0.5327
Iteration No: 28 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 28 ended. Search finished for the next optimal point.
Time taken: 2.2240
Function value obtained: 0.5410
Current minimum: 0.5327
Iteration No: 29 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
```

```
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 29 ended. Search finished for the next optimal point.
Time taken: 2.2340
Function value obtained: 0.5830
Current minimum: 0.5327
Iteration No: 30 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 30 ended. Search finished for the next optimal point.
Time taken: 1.9897
Function value obtained: 0.5617
Current minimum: 0.5327
Iteration No: 31 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 31 ended. Search finished for the next optimal point.
Time taken: 1.8956
Function value obtained: 0.5496
Current minimum: 0.5327
Iteration No: 32 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 32 ended. Search finished for the next optimal point.
Time taken: 1.8190
Function value obtained: 0.5380
Current minimum: 0.5327
Iteration No: 33 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 33 ended. Search finished for the next optimal point.
Time taken: 2.2156
Function value obtained: 0.5766
Current minimum: 0.5327
Iteration No: 34 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 34 ended. Search finished for the next optimal point.
Time taken: 2.3990
Function value obtained: 0.5720
Current minimum: 0.5327
Iteration No: 35 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 35 ended. Search finished for the next optimal point.
Time taken: 2.2693
Function value obtained: 0.5693
Current minimum: 0.5327
```

```
Iteration No: 36 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 36 ended. Search finished for the next optimal point.
Time taken: 2.1992
Function value obtained: 0.5702
Current minimum: 0.5327
Iteration No: 37 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 37 ended. Search finished for the next optimal point.
Time taken: 2.0840
Function value obtained: 0.5360
Current minimum: 0.5327
Iteration No: 38 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 38 ended. Search finished for the next optimal point.
Time taken: 2.2519
Function value obtained: 0.5349
Current minimum: 0.5327
Iteration No: 39 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 39 ended. Search finished for the next optimal point.
Time taken: 2.2791
Function value obtained: 0.5491
Current minimum: 0.5327
Iteration No: 40 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 40 ended. Search finished for the next optimal point.
Time taken: 2.3051
Function value obtained: 0.5461
Current minimum: 0.5327
Iteration No: 41 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 41 ended. Search finished for the next optimal point.
Time taken: 2.1771
Function value obtained: 0.5591
Current minimum: 0.5327
Iteration No: 42 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 42 ended. Search finished for the next optimal point.
Time taken: 2.2852
Function value obtained: 0.5342
Current minimum: 0.5327
Iteration No: 43 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 43 ended. Search finished for the next optimal point.
Time taken: 2.2612
Function value obtained: 0.5671
Current minimum: 0.5327
Iteration No: 44 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 44 ended. Search finished for the next optimal point.
Time taken: 2.3192
Function value obtained: 0.5832
Current minimum: 0.5327
Iteration No: 45 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 45 ended. Search finished for the next optimal point.
Time taken: 2.2846
Function value obtained: 0.5655
Current minimum: 0.5327
Iteration No: 46 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 46 ended. Search finished for the next optimal point.
Time taken: 2.3815
Function value obtained: 0.5595
Current minimum: 0.5327
Iteration No: 47 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 47 ended. Search finished for the next optimal point.
Time taken: 2.3186
Function value obtained: 0.5326
Current minimum: 0.5326
Iteration No: 48 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 48 ended. Search finished for the next optimal point.
Time taken: 2.3190
Function value obtained: 0.5420
Current minimum: 0.5326
Iteration No: 49 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 49 ended. Search finished for the next optimal point.
Time taken: 2.3276
Function value obtained: 0.5586
Current minimum: 0.5326
Iteration No: 50 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 50 ended. Search finished for the next optimal point.
Time taken: 2.3170
Function value obtained: 0.5740
Current minimum: 0.5326
Best configuration: ['depth', 4, 1, 'on', 'off', 'off', 'off']
Best performance in seconds: 0.53
```

- let's perform some pretty-printing of the best solution:

```
In [15]:
def print_best_solution(result_object):
    df = pd.DataFrame([result_object.x])
    df.columns = Space(space).dimension_names
    df["seconds"]=(result_object.fun)
    display(HTML(df.to_html(index=False)))

print_best_solution(results_single_instance)
```

nodeStrategy	strongBranching	trustPseudoCosts	preprocess	presolve	cutsOnOff	heuristicsOnOff	seconds
depth	4	1	on	off	off	off	0.532567

Showing all Evaluations

- it may also be interesting to see all evaluations

In [16]:
```python
def all_evaluations_as_df(result_object):
    df = pd.DataFrame(result_object.x_iters)
    df.columns = Space(space).dimension_names
    df["time"]=(result_object.func_vals)
    return df

df_evaluations = all_evaluations_as_df(results_single_instance)
df_evaluations.head()
```

Out[16]:

	nodeStrategy	strongBranching	trustPseudoCosts	preprocess	presolve	cutsOnOff	heuristicsOnOff	time
0	fewest	8	9	sos	on	off	off	1.653041
1	depth	3	5	sos	off	off	on	1.744692
2	fewest	6	4	sos	off	on	off	2.140805
3	hybrid	5	7	on	on	on	on	2.687033
4	depth	5	2	on	off	off	off	0.627334

- we can then even filter these results, e.g. to show all runs taking less than one second:

In [17]:
```python
df_evaluations[df_evaluations["time"] <= 1]
```

Out[17]:

	nodeStrategy	strongBranching	trustPseudoCosts	preprocess	presolve	cutsOnOff	heuristicsOnOff	time
4	depth	5	2	on	off	off	off	0.627334
11	depth	5	2	on	off	off	off	0.620972
12	depth	5	2	on	off	off	off	0.632798
13	depth	5	2	on	off	off	off	0.648449
14	depth	5	1	on	off	off	off	0.564583
15	depth	5	0	on	off	off	off	0.569099
17	depth	5	0	on	off	off	off	0.564064
18	depth	5	0	on	off	off	off	0.574008
19	depth	5	1	on	off	off	off	0.538828
20	depth	4	1	on	off	off	off	0.532681
22	depth	4	1	on	off	off	off	0.561487
23	depth	4	1	on	off	off	off	0.583198
24	depth	4	1	on	off	off	off	0.546971
25	depth	4	1	on	off	off	off	0.547991
26	depth	4	1	on	off	off	off	0.557836
27	depth	4	1	on	off	off	off	0.540973
28	depth	4	1	on	off	off	off	0.582996
29	depth	4	1	on	off	off	off	0.561710
30	depth	4	1	on	off	off	off	0.549632
31	depth	4	1	on	off	off	off	0.537979
32	depth	4	1	on	off	off	off	0.576582
33	depth	4	1	on	off	off	off	0.572005
34	depth	4	1	on	off	off	off	0.569268
35	depth	4	1	on	off	off	off	0.570194
36	depth	4	1	on	off	off	off	0.535994
37	depth	4	1	on	off	off	off	0.534872
38	depth	4	1	on	off	off	off	0.549127
39	depth	4	1	on	off	off	off	0.546070
40	depth	4	1	on	off	off	off	0.559119
41	depth	4	1	on	off	off	off	0.534242

	nodeStrategy	strongBranching	trustPseudoCosts	preprocess	presolve	cutsOnOff	heuristicsOnOff	time
42	depth	4	1	on	off	off	off	0.567139
43	depth	4	1	on	off	off	off	0.583189
44	depth	4	1	on	off	off	off	0.565511
45	depth	4	1	on	off	off	off	0.559535
46	depth	4	1	on	off	off	off	0.532567
47	depth	4	1	on	off	off	off	0.542022
48	depth	4	1	on	off	off	off	0.558597
49	depth	4	1	on	off	off	off	0.574002

Avoiding Overfitting

Avoiding Overfitting

- even in case of homogeneous instances, algorithm configuration is very prone to overfitting
- in general, in machine learning, we would use cross validation for
- in algorithm configuration, however, **cross validation is not practical** since it requires too many time-consuming evaluations
- thus, in algorithm configuration one mostly uses
    - a single training set
    - and a single test set

Evaluating the Configuration on a Test Set

Let us see how our configuration (selected for a single instance) performs on average for a test set of 10 instances

Question:

- How can we create a test set (a set of test instances) for our CCP case study?

Exercise:

- Create a test set consisting of 10 instances
- Write an function for evaluating a configuration on multiple instances (using runtime as result measure)
- Evaluate the best configuration found above using that function and compare the result to the single-instance evaluation used for configuration!

Hint: See the following function for how to evaluate an sk_opt solution on a single instance!

In [21]:

```
def evaluate_skopt_parameters_on_instance(instance, sk_opt_solution):
    param_dict = dict(zip(Space(space).dimension_names, sk_opt_solution))
    return evaluate_parameters_on_instance(instance, **param_dict)

evaluate_skopt_parameters_on_instance(model_instance_cbc, results_single_instance.x)
```

Out[21]:

```
0.8564021587371826
```

Evaluating the Configuration on a Test Set: Results

Let's create a data frame to collect the results

In [21]:
```
df_results = pd.DataFrame(columns=['train_perf','test_perf'])
```

- we start with an evaluation of the CBC standard settings:

In [22]:
```
df_results.loc['Standard CBC','test_perf'] = evaluate_parameters_on_instances(test_instances)
```

- and then add those from training with a single instance (note: train performance is based on a single instance, test performance is evaluated using 10 instances

In [23]:
```
df_results.loc['Configured Single Instance','train_perf'] = results_single_instance.fun
df_results.loc['Configured Single Instance','test_perf']=  evaluate_skopt_parameters_on_instances(test_instances,results_single_instance.x )
```

In [24]:
```
df_results
```

Out[24]:

	train_perf	test_perf
Standard CBC	NaN	3.208156
Configured Single Instance	0.404865	0.7003

Addressing Overfitting: Configuration using Multiple Training Instances

Let us see if we can do better by evaluating each configuration on multiple instances **during training**!

- we thus create 10 training instances:

In [25]:
```
np.random.seed(seed=42)
number_of_training_instances = 10
training_instances = []
for i in range(number_of_training_instances):
    time_available = time_available_dist.rvs(n_scenarios)
    leather_available = leather_available_dist.rvs(n_scenarios)

    training_instances.append( build_model_with_samples(time_available, leather_available, CBC) )
```

- then, we create an objective function for scikit-optimize that uses these 10 instances:

In [26]:
```
instances_to_evaluate = training_instances
@use_named_args(space)
def evaluate_parameters_on_multiple_instances(**params):
    return evaluate_parameters_on_instances(instances_to_evaluate,**params)
```

Automatic Configuration Using Multiple Instances

In [27]:

```python
start_time_configuration_multiple_instances = time.time()

results_multiple_instances = gp_minimize(evaluate_parameters_on_multiple_instances, space, verbose=True, n_calls=50, random_state=0)

duration_configuration_multiple_instances = time.time() - start_time_configuration_multiple_instances

print (f"The configuration took {duration_configuration_multiple_instances:0.2f} seconds")
```

```
Iteration No: 1 started. Evaluating function at random point.
Iteration No: 1 ended. Evaluation done at random point.
Time taken: 10.3334
Function value obtained: 1.0089
Current minimum: 1.0089
Iteration No: 2 started. Evaluating function at random point.
Iteration No: 2 ended. Evaluation done at random point.
Time taken: 23.8370
Function value obtained: 2.3527
Current minimum: 1.0089
Iteration No: 3 started. Evaluating function at random point.
Iteration No: 3 ended. Evaluation done at random point.
Time taken: 28.7291
Function value obtained: 2.8476
Current minimum: 1.0089
Iteration No: 4 started. Evaluating function at random point.
Iteration No: 4 ended. Evaluation done at random point.
Time taken: 47.6773
Function value obtained: 4.7306
Current minimum: 1.0089
Iteration No: 5 started. Evaluating function at random point.
Iteration No: 5 ended. Evaluation done at random point.
Time taken: 8.6584
Function value obtained: 0.8323
Current minimum: 0.8323
Iteration No: 6 started. Evaluating function at random point.
Iteration No: 6 ended. Evaluation done at random point.
Time taken: 45.9470
Function value obtained: 4.5604
Current minimum: 0.8323
Iteration No: 7 started. Evaluating function at random point.
Iteration No: 7 ended. Evaluation done at random point.
Time taken: 42.3580
Function value obtained: 4.2041
Current minimum: 0.8323
Iteration No: 8 started. Evaluating function at random point.
Iteration No: 8 ended. Evaluation done at random point.
Time taken: 50.1222
Function value obtained: 4.9818
Current minimum: 0.8323
Iteration No: 9 started. Evaluating function at random point.
Iteration No: 9 ended. Evaluation done at random point.
Time taken: 28.2812
Function value obtained: 2.7972
Current minimum: 0.8323
Iteration No: 10 started. Evaluating function at random point.
Iteration No: 10 ended. Evaluation done at random point.
Time taken: 47.3275
Function value obtained: 4.4717
Current minimum: 0.8323
Iteration No: 11 started. Searching for the next optimal point.
Iteration No: 11 ended. Search finished for the next optimal point.
Time taken: 110.8081
Function value obtained: 91.8341
Current minimum: 0.8323
Iteration No: 12 started. Searching for the next optimal point.
Iteration No: 12 ended. Search finished for the next optimal point.
Time taken: 40.6410
Function value obtained: 3.8849
Current minimum: 0.8323
Iteration No: 13 started. Searching for the next optimal point.
Iteration No: 13 ended. Search finished for the next optimal point.
Time taken: 25.1850
Function value obtained: 2.3639
Current minimum: 0.8323
Iteration No: 14 started. Searching for the next optimal point.
Iteration No: 14 ended. Search finished for the next optimal point.
Time taken: 41.8350
Function value obtained: 3.9873
Current minimum: 0.8323
Iteration No: 15 started. Searching for the next optimal point.
Iteration No: 15 ended. Search finished for the next optimal point.
Time taken: 16.6500
Function value obtained: 1.4881
```

```
Current minimum: 0.8323
Iteration No: 16 started. Searching for the next optimal point.
Iteration No: 16 ended. Search finished for the next optimal point.
Time taken: 10.3287
Function value obtained: 0.8796
Current minimum: 0.8323
Iteration No: 17 started. Searching for the next optimal point.
Iteration No: 17 ended. Search finished for the next optimal point.
Time taken: 63.5073
Function value obtained: 33.1688
Current minimum: 0.8323
Iteration No: 18 started. Searching for the next optimal point.
Iteration No: 18 ended. Search finished for the next optimal point.
Time taken: 20.8812
Function value obtained: 1.8943
Current minimum: 0.8323
Iteration No: 19 started. Searching for the next optimal point.
Iteration No: 19 ended. Search finished for the next optimal point.
Time taken: 146.9921
Function value obtained: 94.9285
Current minimum: 0.8323
Iteration No: 20 started. Searching for the next optimal point.
Iteration No: 20 ended. Search finished for the next optimal point.
Time taken: 26.1017
Function value obtained: 2.4284
Current minimum: 0.8323
Iteration No: 21 started. Searching for the next optimal point.
Iteration No: 21 ended. Search finished for the next optimal point.
Time taken: 61.0755
Function value obtained: 5.8122
Current minimum: 0.8323
Iteration No: 22 started. Searching for the next optimal point.
Iteration No: 22 ended. Search finished for the next optimal point.
Time taken: 15.5248
Function value obtained: 1.2522
Current minimum: 0.8323
Iteration No: 23 started. Searching for the next optimal point.
Iteration No: 23 ended. Search finished for the next optimal point.
Time taken: 27.3244
Function value obtained: 2.5048
Current minimum: 0.8323
Iteration No: 24 started. Searching for the next optimal point.
Iteration No: 24 ended. Search finished for the next optimal point.
Time taken: 15.4008
Function value obtained: 1.2653
Current minimum: 0.8323
Iteration No: 25 started. Searching for the next optimal point.
Iteration No: 25 ended. Search finished for the next optimal point.
Time taken: 39.6969
Function value obtained: 3.7544
Current minimum: 0.8323
Iteration No: 26 started. Searching for the next optimal point.
Iteration No: 26 ended. Search finished for the next optimal point.
Time taken: 30.5879
Function value obtained: 2.8339
Current minimum: 0.8323
Iteration No: 27 started. Searching for the next optimal point.
Iteration No: 27 ended. Search finished for the next optimal point.
Time taken: 14.0822
Function value obtained: 1.1520
Current minimum: 0.8323
Iteration No: 28 started. Searching for the next optimal point.
Iteration No: 28 ended. Search finished for the next optimal point.
Time taken: 19.5771
Function value obtained: 1.7491
Current minimum: 0.8323
Iteration No: 29 started. Searching for the next optimal point.
Iteration No: 29 ended. Search finished for the next optimal point.
Time taken: 15.0367
Function value obtained: 1.3284
Current minimum: 0.8323
Iteration No: 30 started. Searching for the next optimal point.
Iteration No: 30 ended. Search finished for the next optimal point.
Time taken: 32.1523
Function value obtained: 2.9835
Current minimum: 0.8323
Iteration No: 31 started. Searching for the next optimal point.
Iteration No: 31 ended. Search finished for the next optimal point.
Time taken: 11.2725
Function value obtained: 0.9065
Current minimum: 0.8323
Iteration No: 32 started. Searching for the next optimal point.
Iteration No: 32 ended. Search finished for the next optimal point.
Time taken: 27.2680
Function value obtained: 2.4883
Current minimum: 0.8323
Iteration No: 33 started. Searching for the next optimal point.
Iteration No: 33 ended. Search finished for the next optimal point.
```

```
Time taken: 22.6088
Function value obtained: 1.9535
Current minimum: 0.8323
Iteration No: 34 started. Searching for the next optimal point.
Iteration No: 34 ended. Search finished for the next optimal point.
Time taken: 38.3285
Function value obtained: 3.5784
Current minimum: 0.8323
Iteration No: 35 started. Searching for the next optimal point.
Iteration No: 35 ended. Search finished for the next optimal point.
Time taken: 28927.5465
Function value obtained: 33.0818
Current minimum: 0.8323
Iteration No: 36 started. Searching for the next optimal point.
Iteration No: 36 ended. Search finished for the next optimal point.
Time taken: 15.7970
Function value obtained: 1.2512
Current minimum: 0.8323
Iteration No: 37 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 37 ended. Search finished for the next optimal point.
Time taken: 11.8160
Function value obtained: 0.9177
Current minimum: 0.8323
Iteration No: 38 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 38 ended. Search finished for the next optimal point.
Time taken: 9.5402
Function value obtained: 0.7552
Current minimum: 0.7552
Iteration No: 39 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 39 ended. Search finished for the next optimal point.
Time taken: 9.1369
Function value obtained: 0.7169
Current minimum: 0.7169
Iteration No: 40 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 40 ended. Search finished for the next optimal point.
Time taken: 9.1785
Function value obtained: 0.7185
Current minimum: 0.7169
Iteration No: 41 started. Searching for the next optimal point.
Iteration No: 41 ended. Search finished for the next optimal point.
Time taken: 10.6262
Function value obtained: 0.8482
Current minimum: 0.7169
Iteration No: 42 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 42 ended. Search finished for the next optimal point.
Time taken: 9.1648
Function value obtained: 0.7083
Current minimum: 0.7083
Iteration No: 43 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 43 ended. Search finished for the next optimal point.
Time taken: 8.9203
Function value obtained: 0.6852
Current minimum: 0.6852
Iteration No: 44 started. Searching for the next optimal point.
Iteration No: 44 ended. Search finished for the next optimal point.
Time taken: 9.6267
Function value obtained: 0.7103
Current minimum: 0.6852
Iteration No: 45 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 45 ended. Search finished for the next optimal point.
Time taken: 9.3964
Function value obtained: 0.7131
Current minimum: 0.6852
Iteration No: 46 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 46 ended. Search finished for the next optimal point.
Time taken: 10.5682
Function value obtained: 0.8299
Current minimum: 0.6852
Iteration No: 47 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 47 ended. Search finished for the next optimal point.
Time taken: 10.1307
Function value obtained: 0.7682
Current minimum: 0.6852
Iteration No: 48 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 48 ended. Search finished for the next optimal point.
Time taken: 14.4987
Function value obtained: 1.1028
Current minimum: 0.6852
Iteration No: 49 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 49 ended. Search finished for the next optimal point.
Time taken: 9.6291
```

```
Function value obtained: 0.7589
Current minimum: 0.6852
Iteration No: 50 started. Searching for the next optimal point.
```

```
C:\Users\Michael\miniconda3\envs\cords2022\lib\site-packages\skopt\optimizer\optimizer.py:449: U
serWarning: The objective has been evaluated at this point before.
  warnings.warn("The objective has been evaluated "
```

```
Iteration No: 50 ended. Search finished for the next optimal point.
Time taken: 9.5523
Function value obtained: 0.7583
Current minimum: 0.6852
The configuration took 30291.32 seconds
```

Automatic Configuration Using Multiple Instances: Results

How did it work?

In [28]:

```
df_results.loc['Configured Multiple Instances','train_perf'] = results_multiple_instances.fun
df_results.loc['Configured Multiple Instances','test_perf']=  evaluate_skopt_parameters_on_instances(test_instances,results_multiple_instances.x )
df_results
```

Out[28]:

	train_perf	test_perf
Standard CBC	NaN	3.208156
Configured Single Instance	0.404865	0.7003
Configured Multiple Instances	0.685224	0.677522

- it worked better, that is, using multiple training instances is advantageous.

Making Configuration Faster: Early Stopping

Challenge: Long Configuration Time
Long configuration times

- using multiple training instances will improve the robustness of the results,
- but it will also drastically increase configuration time

Question: Do you have any idea how to reduce configuration time?

We can reduce configuration time by

- quickly recognizing bad configurations and evaluating them on only a few instances
- we may even use statistical tests to detect bad configurations early
- in our case study, we will use a simple early stopping approach

A Simple Early Stopping Approach for our Case Study

Goal:
Detect "bad" configs early and stop their evaluation after few instances
Assumption:

- we have a *very homogeneous* set of instances
- the run times of the instances are relatively similar

Evaluation of a configuration c with early stopping:
Let $g(i)$ be a factor ≥ 1 that decreases with an increasing instance index i
and f^* be the current best average performance
For the set of instances from $i = 1$ to n:

- evaluate the performance $f_c(i)$ of configuration
- update the average performance $\hat{f}_c$ with $f_c(i)$
- if $\hat{f}_c > g(i)f^*$, stop early and return $\hat{f}_c$

return $\hat{f}_c$
Intuitive rationale of using the decreasing factor g(i)

- at the beginning, the "bar" should be higher in order to stop evaluation based on one or two underperforming runs (bad luck)

Implementing the Early Stopping Approach

Scikit-Optimize's "ask-and-tell" interface allows to interact with the Optimizer solver via two functions:

- **Ask** (the optimizer) for the next configuration to evaluate
- **Tell** the objective function value to the optimizer $\rightarrow$ This gives us control on how to compute the objective value

For our purposes, we use it as follows:
For a given number of iterations:

- **Ask** the optimizer for the next configuration
- perform the evaluation of the configuration
    - in our case, using the early-stopping mechanism
- **Tell** the optimizer the performance of the configuration

The Implementation of the Early Stopping Approach

- first we write a function returning the exceedance factor $g(i)$ based on the total number of instances and the number of instances evaluated so far (i)
- this implementation of the function is pretty simple, the key idea is that the factor decreases with the number of instances evaluated so far

In [29]:

```
def get_exceedance_factor (total_number_of_instances, number_of_instances_evaluated_so_far):
    if number_of_instances_evaluated_so_far <= 1:
        return 1000

    fraction_instances_remaining = (total_number_of_instances - number_of_instances_evaluated_so_far) / total_number_of_instances
    return 1.1 + fraction_instances_remaining * 0.5

#example
get_exceedance_factor(10,8)
```

Out[29]:

```
1.2000000000000002
```

- then, we write a function that uses the early stopping idea for evaluation

In [30]:

```
def evaluate_with_early_stopping(instances, best_performance, sk_opt_params):
    evaluated_instances = 0
    total_performance  = 0

    total_number_of_instances = len(instances)
    for instance in instances:
        evaluated_instances  += 1
        param_dict = dict(zip(Space(space).dimension_names, sk_opt_params))
        total_performance += evaluate_parameters_on_instance(instance, **param_dict)
        average_performance = total_performance / evaluated_instances
        # early stopping condition.
        if average_performance >  best_performance * get_exceedance_factor(total_number_of_instances, evaluated_instances):
            print(f" Stopping early after { evaluated_instances } instances")
            return average_performance
    return average_performance
```

Running the Algorithm Configuration with the Early Stopping Approach

- now, we run the early stopping-based algorithm configuration
- the following implementation uses the sickit-optimizes ask-and-tell interface explained above

In [ ]:

```python
from skopt import Optimizer

# let us take the standard configuration as best-known configuration in the beginning
best_performance = evaluate_parameters_on_instances(training_instances)

number_of_total_iterations = 50

opt = Optimizer(space, random_state=0)
start_time_early_stopping = time.time()
for n in range(number_of_total_iterations):

    next_params = opt.ask() # ask the optizer for the next configuration

    # call the early stopping evaluation
    performance = evaluate_with_early_stopping(training_instances, best_performance, next_params)

    print (f"Iteration {n}: performance: {performance:0.2f}")

    if performance < best_performance:
        best_performance = performance

    print (f" After Iteration {n}: best performance: {best_performance:0.2f}\n")

    results_early_stopping = opt.tell(next_params, performance)  # tell the optizer the performance of the evaulation

duration_early_stopping = time.time() - start_time_early_stopping
print (f"The configuration using early stopping took {duration_early_stopping:0.2f} seconds")
print_best_solution(results_early_stopping)
```

```
Iteration 0: performance: 0.88
 After Iteration 0: best performance: 0.88

 Stopping early after 2 instances
Iteration 1: performance: 1.85
 After Iteration 1: best performance: 0.88

 Stopping early after 2 instances
Iteration 2: performance: 2.64
 After Iteration 2: best performance: 0.88

 Stopping early after 2 instances
Iteration 3: performance: 3.28
 After Iteration 3: best performance: 0.88

Iteration 4: performance: 0.69
 After Iteration 4: best performance: 0.69

 Stopping early after 2 instances
Iteration 5: performance: 3.39
 After Iteration 5: best performance: 0.69

 Stopping early after 2 instances
Iteration 6: performance: 4.14
 After Iteration 6: best performance: 0.69

 Stopping early after 2 instances
Iteration 7: performance: 4.98
 After Iteration 7: best performance: 0.69

 Stopping early after 2 instances
Iteration 8: performance: 2.96
 After Iteration 8: best performance: 0.69

 Stopping early after 2 instances
Iteration 9: performance: 3.85
 After Iteration 9: best performance: 0.69
```

Let us see the results

- let's add the results from the early stopping approach

In [ ]:

```
#add to results
df_results.loc['Configured Multi Instance Early Stop','train_perf'] = results_early_stopping.fun

df_results.loc['Configured Multi Instance Early Stop','test_perf']=  evaluate_skopt_parameters_on_instances(test_instances,results_early_stopping.x )
```

In [ ]:

```
df_results
```

..we can see that the early stopping was much faster, but we obtained similar (even better) results.

Conclusions

In this meeting, we dealt with algorithm configuration

- we learned about the key concepts of algorithm configuration
- and learned how to address the key challenges in automatic algorithm configuration

This was the last meeting of our course!

- thank you very much for your active participation!