GECCO 2020 Industrial Challenge: Optimizing Expensive Computational Fluid Dynamics Simulations

Frederik Rehbach, Margarita Rebolledo, Thomas Bartz-Beielstein [1]

January, 2020

The goal of the GECCO 2020 Industrial Challenge is to develop an optimizer for this year's industrial application in a programming language of your choice. The optimizer has to efficiently use every objective function evaluation as each evaluation involves a computationally intensive fluid dynamics simulation. As usual, the task from this industrial challenge stems directly from ongoing cooperation with industry.

The challenge is split into two tracks. In the first track, the amount of allowed objective function evaluations is only limited by the computing power you are willing to invest on your own machine. In the second track, only 100 evaluations are allowed per optimization run. In both cases, the algorithm with the best-found objective function value wins.

This document provides a set of rules and regulations for the GECCO IC, a detailed problem description, as well as contact and submission information.

[1] Cologne University of Applied Sciences, 51643 Gummersbach, Germany
frederik.rehbach@th-koeln.de,
thomas.bartz-beielstein@th-koeln.de

Technology
Arts Sciences
TH Köln

1 *Introduction*

The Goal of the GECCO 2020 Industrial Challenge is to develop an optimization algorithm for the Gas Distribution System (GDS) of an Electrostatic Precipitator (ESP). The Electrostatic Precipitator Problem stems from a corporation with Steinmüller Babcock Environment GmbH. It was first published to GECCO in [2].

The ESP Problem is a 49-dimensional discrete-valued problem. Each objective function evaluation involves the automatic setup of a new Computational Fluid Dynamics (CFD) simulation, running that given simulation, and measuring the final obtained gas distribution in the precipitator.

The aim of the challenge was to allow for an open optimization process. The competitors are allowed to use any programming language and optimization technique of their choice.

[2] Frederik Rehbach, Martin Zaefferer, Jörg Stork, and Thomas Bartz-Beielstein. Comparison of parallel surrogate-assisted optimization approaches. In Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '18. ACM Press, 2018

Highlights of the GECCO IC include:

- Interesting Problem Domain: Reduction of emissions and environmental pollution is more important than ever before. Efficient filters help reduce global CO_2 emissions.
- Real-world Problems: Test your algorithms and methods, directly on a real industry problem.
- Easy Access: Easily Participate through our online platform, no installations required.

- Fair Submission Assessment: Winners are determined automatically through our online portal, fully objectively, only based on the final result quality.
- Publication Options: Participants can submit a 2-page extended abstract describing the algorithm they are applying in the challenge. Accepted abstracts are published together with GECCO poster-papers in the GECCO-companion.

The remainder of this document specifies the information needed to take part in this competition. Section 2 gives more detail about the ESP as well as the CFD-Test-Problem-Suite it is implemented in. Section 3 summarizes how to participate in the challenge and which rules have to be followed.

2 *Problem Description*

2.1 *The Electrostatic Precipitator Problem*

The ESP is one of the main components of gas cleaning systems. They are used in large scale combustion power plants or other industries where solid particles have to be removed from a gas stream. ESPs are large devices with dimensions of around 30m $\times$ 30m $\times$ 50m, resulting in multiple millions of euros just in building cost. The main task of an ESP is to separate and extract particles from exhaust gases in order to reduce environmental pollution. Figure 1 illustrates this system.

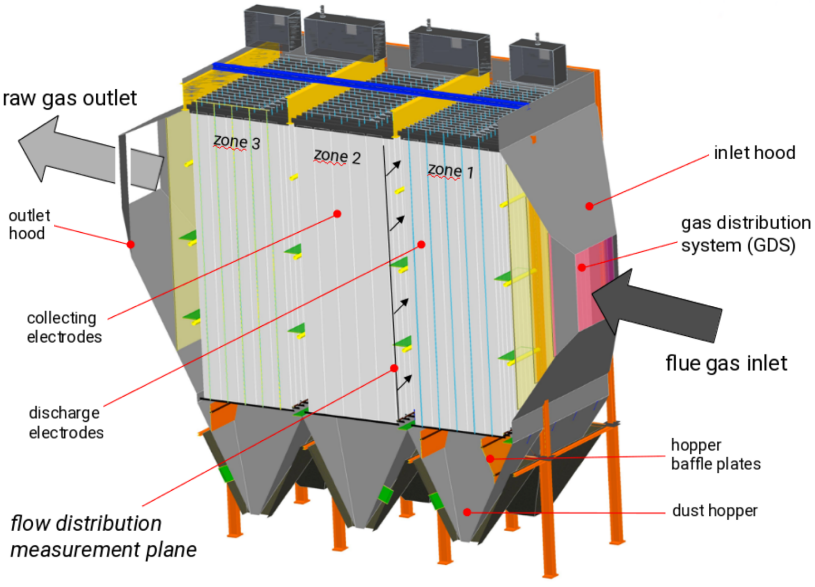

Figure 1: Electrostatic precipitator with 3 separation zones. This figure was kindly provided by Steinmüller Babcock Environment GmbH.

In the flue gas inlet hood of an ESP, a gas distribution system (GDS) (shown in Figure 2) is required to control and guide the gas flow through separation zones in which particles are removed from the exhaust gases. If no GDS is used, or if the system is configured

poorly, the fast inlet gas stream will rush through the separation zones of the ESP. This results in very low separation efficiencies. In case of a well configured GDS, the inflowing gas is nicely distributed across the whole surface of the separation zones, resulting in high efficiency.

Hence, the efficient operation of an ESP requires an optimal configuration of the GDS. The GDS in our example consists of 49 configurable slots. Each of these slots can be configured with baffles, as well as blocking and perforated plates. Baffles are metal plates which are mounted at an angle to the general gas flow. They are used to redirect a gas stream into a new direction. Blocking plates completely block a gas stream. Perforated plates are used to slow down and only partially block gas streams. They are created by punching a grid of holes into metal plates. Smaller holes lead to higher pressure drops and thus a slower gas stream. Larger holes allow for a nearly free gas flow. These plates can be mounted into each of the 49 configurable slots. Optionally, some slots can also be left empty.

of the slots can be configured with eight different types of plates (integer value 0-7), including leaving the slot empty. The vast amount of possible combinations of the GDS reveals a complex discrete optimization problem. For a single evaluation of a given configuration, a computationally expensive CFD simulation is necessary, which results in hours of computation time. Unfortunately, such large computation times make the ESP problem unsuitable for a large number of tests runs which are necessary to derive reasonable conclusions about the performance of several competing algorithms. Therefore, a second model with a largely reduced amount of cells in the simulation mesh was created. By doing so, the runtime of the model was reduced to roughly one minute per evaluation. This speed up comes at the cost of reduced simulation accuracy. However, the reduced model still captures most of the difficulties and complex features of the actual problem, while enabling a detailed experimental study. Reproducing the rugged problem landscape is much more important than the actual accuracy of each sample point. The open source CFD framework OpenFOAM [3] was used to implement our simulations. The original landscape of a real industrial problem is transferred into a function which can be evaluated in reasonable computation time.

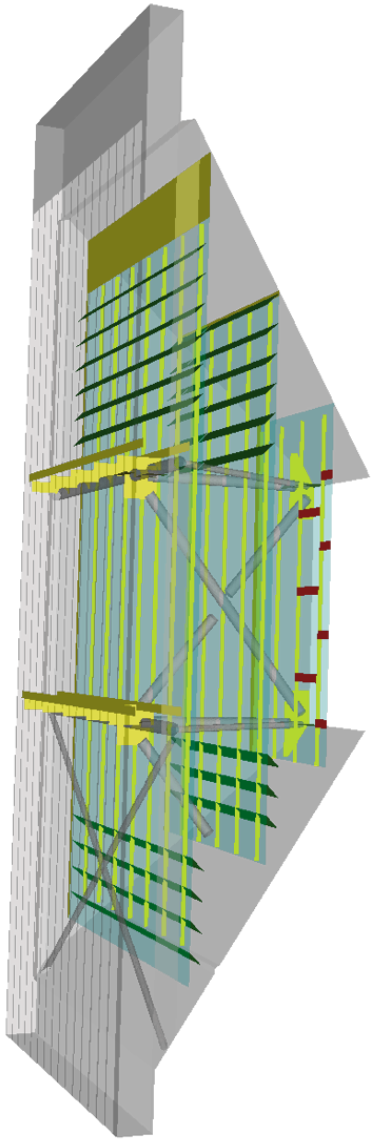

Figure 2: Visualization of a gas distribution system (GDS) mounted in the inlet hood of of an ESP. This figure was kindly provided by Steinmüller Babcock Environment GmbH.

[3] Henry G Weller, G Tabor, Hrvoje Jasak, and C Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. Computers in physics, 12 (6):620–631, 1998

2.2 *CFD Test Problem Suite*

The CFD-Test-Problem-Suite was presented in last year's GECCO Hot of the Press talk by Daniels et al. [4]. The suite consists of expensive computer simulation-based optimization problems and provides an easy evaluation interface that will be used for the setup of our challenge.

For the purpose of this challenge, the suite was enhanced with a Docker version. Docker is an open-source tool that provides a

[4] Steven J Daniels, Alma AM Rahat, Richard M Everson, Gavin R Tabor, and Jonathan E Fieldsend. A suite of computationally expensive shape optimisation problems using computational fluid dynamics. In International Conference on Parallel Problem Solving from Nature, pages 296–307. Springer, 2018

Linux container-based operating system-level virtualization. [5] Docker is easily installed on any major operating system. It allows for packaging code or experiments into a shareable container image. An image typically contains all required software and prerequisites to run a certain task. Such an image can then be shared and is ready for immediate execution on other machines.

An explanation of how to run the dockerized ESP problem through the test suite is available at: https://bitbucket.org/arahat/cfd-test-problem-suite/src/master/.

[5] Carl Boettiger. An introduction to docker for reproducible research. ACM SIGOPS Operating Systems Review, 49(1):71–79, 2015

3 *How to Participate*

There are two tracks in this competition. Competitors can participate in both tracks, but can also just choose one of them.

Track 1: Unlimited Evaluations

In track one the amount of objective function evaluations is only limited by how much computation time you are willing to give to the optimizer on your machine. In order to participate, you have to follow the Docker instructions presented at: https://bitbucket.org/arahat/cfd-test-problem-suite/. Scroll all the way down, for now you ONLY need section "Setup via Docker". You DO NOT need to install OpenFOAM or any of the described prequisites! Only the described Docker installation is required to run the ESP problem on your machine. Currently, there are only Python and R examples of how to run the ESP Problem through docker given. However, any programming language that can access the command line on your machine should be capable of using the easy interface.

An evaluation of the ESP problem as given in the online example will look something like this (take care, this command should be on one single line!):

```
docker run --rm frehbach/cfd-test-problem-suite ./dockerCall.sh ESP
'1,1,0,1,0,0,0,0,0,4,3,4,6,6,5,6,7,7,6,2,4,7,2,0,4,0,0,1,6,0,0,3,6,7,2,6,7,0,4,1,6,7,2,7,0,4,4,2,7'
```

Once you have setup the evaluation interface on your machine, you can try to use any optimizer of your choice to find the lowest objective function value on the ESP problem. Example scripts for that are also available on the bitbucket link, in the folder "DockerExampleScripts".

Listing 1: 'example.py': sample Python code that creates a random candidate solution and evaluates it on the ESP problem.

```
import subprocess
import random

## evalFun accepts a vector of integers,
## length 49 (dimensionality of the ESP problem)
def evalFun(problemName, candidateSolution):
        evalCommand = "docker run --rm
          frehbach/cfd-test-problem-suite ./dockerCall.sh "
          + problemName + " "
        parsedCandidate = ",".join([str(x) for x in
                candidateSolution])
        return(subprocess.check_output(evalCommand + "'" +
                parsedCandidate + "'", shell=True))

## Create some candidate
candidate = [random.randint(0,7) for i in range(49)]
## Evaluate the candidate, for example on the ESP problem:
print(evalFun("ESP", candidate))
```

Participation is then very easy: You can send your best candidate setup (the 49 integers) along with your measured objective function value to "gecco@gm.fh-koeln.de", mail-subject "GECCO-IC-2020". Each candidate that we receive will be checked at the end of the competition, and the winners will be determined according to the objective function value that we measure.

Finalists selected by the organizers will be invited to present their submission at the competition session, held during the GECCO conference. The winner of the competition will be announced at the SIGEVO meeting ceremony, on July 12, 2020.

Track 2: Severely Limited Evaluations

For the second track, you do not have to install any additional software on your machine. Submissions will be handled through an automated online evaluation tool. You can access the tool via:

`http://owos.gm.fh-koeln.de:3838/GECCO-IC-2020/`

The entrance password to the tool is 'IC2020'. After entering the page you will have to create an account on the registration page. Please remember your password as we have NO mechanisms for resetting a forgotten password! Once you created an account, you can start uploading your submissions as often as you like. The submissions and scores will be saved on our servers.

A submission to the online tool will consist of multiple files (You have to mark all your files simultaneously after you pressed the "browse" button). Each upload has to contain one file called "main.sh". The evaluation tool will search for this exact file name and try to run it for your evaluation.

The dockerized environment that your code will be run in contains installations for python 2 & 3 as wells as R. If you want to use other programming languages or install other prerequisites for your optimizers you can do so. You are in a simplified Ubuntu environment. Therefore, commands like "apt-get install" etc. can be used

to add additional software. In the container you are the root user, so using the sudo command is not required. You have to specify each of your prerequisite installation steps as well as lastly the call to your optimizers source code in the "main.sh".

In the following we give a short example of how to setup a very simple optimizer that tries a few random candidates in python. These codes and other examples are included in the ExampleScripts folder of the ressource package. In order to try the base example online, upload both the "main.sh" and "optimRandomSearch.py" to the webpage. After some time (press refresh from time to time), you will see the evaluation result.

Since python is already installed in the docker container, our "main.sh" only contains a single line:

Listing 2: Code inside example 'main.sh'. The script is automatically started on the evaluation server and runs our optimizer.

```
python   optimRandomSearch.py
```

Listing 3: 'optimRandomSearch.py': sample Python code that creates random candidate solutions and evaluates them on the ESP problem.

```
import subprocess
import random

## evalFun accepts a vector of integers,
## length 49 (dimensionality of the ESP problem)
def evalFun(candidateSolution):
        evalCommand = "./evaluate.sh "
        parsedCandidate = ",".join([str(x) for x in
                                    candidateSolution])
        return(subprocess.check_output(evalCommand + "'" +
                        parsedCandidate + "'", shell=True))

for i in range(5):
        ## Generate a bunch of random candidate solutions
        candidate = [random.randint(0,7) for i in range(49)]
        ## Evaluate the candidates
        evalFun(candidate)
```

Listing 3 presents a python example code that can be uploaded to the online submission tool. Similar to the requirements in track 1, the ESP problem is called via the command line. The objective function "evalFun" accepts a vector with 49 integers and passes the, via the command line to 'evaluate.sh'. This same approach has to be used in any other programming language too. 'evaluate.sh' accepts a comma seperated string of intergers e.g. "5,1,2,0,...", evaluates the candidate and returns the objective function value. Your script / optimizer does not have to return anything. The best evaluated candidate is automatically stored on the server and returned as your result.

Each uploaded algorithm should only use 100 evaluations and can not use information gained from previous runs. In order to ensure this, the online platform applies each uploaded algorithm to

a newly created randomized instance of the ESP problem. Since the original task was to optimize the gas distribution in the inlet section of the precipitator, for each run a new target distribution is defined. This means that even if you upload an algorithm that runs with the same seed, your results will vary from upload to upload since the algorithm will face a new harder or easier instance every time. For the final evaluation of the winner one more final instance will be used for every participant to keep the comparison fair. The LAST UPLOADED algorithm of each participant will be applied to that instance. The algorithm with the single best-found candidate wins.

Finalists selected by the organizers will be invited to present their submission at the competition session, held during the GECCO conference. The winner of the competition will be announced at the SIGEVO meeting ceremony, on July 12, 2020. While highly appreciated, it is not required for the participants of the challenge to also participate in the GECCO conference.

3.1 Organizing Committee

If you have questions, feel free to reach out to us at "gecco@gm.fh-koeln.de" , or personally:

- Frederik Rehbach, TH Köln - frederik.rehbach@th-koeln.de
- Margarita Rebolledo, TH Köln - margarita.rebolledo@th-koeln.de
- Thomas Bartz-Beielstein, TH Köln - thomas.bartz-beielstein@th-koeln.de

List of References

Carl Boettiger. An introduction to docker for reproducible research. ACM SIGOPS Operating Systems Review, 49(1):71–79, 2015.

Steven J Daniels, Alma AM Rahat, Richard M Everson, Gavin R Tabor, and Jonathan E Fieldsend. A suite of computationally expensive shape optimisation problems using computational fluid dynamics. In International Conference on Parallel Problem Solving from Nature, pages 296–307. Springer, 2018.

Frederik Rehbach, Martin Zaefferer, Jörg Stork, and Thomas Bartz-Beielstein. Comparison of parallel surrogate-assisted optimization approaches. In Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '18. ACM Press, 2018.

Henry G Weller, G Tabor, Hrvoje Jasak, and C Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. Computers in physics, 12(6):620–631, 1998.