Setup Of Fuzzy Hybrid Particle Swarms

A Heuristic Approach

UNIVERSITÉ DE NAMUR

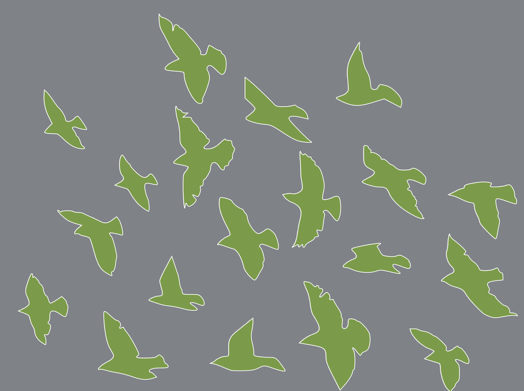
Nicolas Roy, Charlotte Beauthier, Anthony Hendrickx, Timotéo Carletti and Alexandre Mayer

Poster

Contact

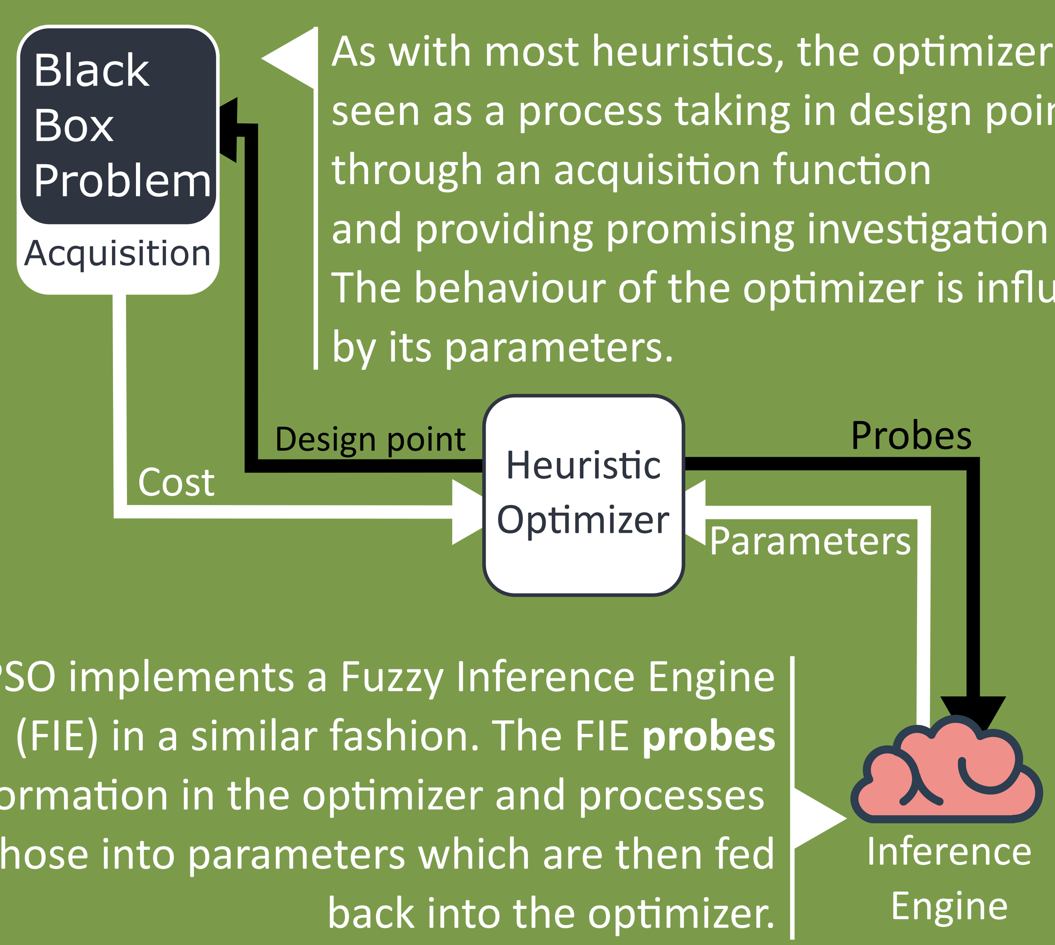
Motivation

The design of Adaptive Fuzzy PSO (AFPSO) requires the implementation to provide rules linking the current state of the optimizer to tuned parameters. We aim to help the user in **designing the rules** given a **target benchmark** we call the **training set**.
A systematic process is proposed for designing and integrating fuzzy rulesets in **any population heuristic**.
The framework is inspired by the **methodology** surrounding **neural networks**, using both training and validation benchmarks.
The system aims to automatically taylor an optimizer to a specific class of problems. Even if we seek specialization we also investigate generalization of produced optimizers to quantify overfitting using the validation benchmark.



The principle

Although the framework is heuristic-agnostic, we will discuss results and implementation on a modern modification of **Particle Swarms** by J. Kennedy & R. Eberhart. The modifications include among others **inertia weight** and **random adaptive topology**.
In PSO, efficient heuristic emerges from simple agent's behaviours. Tuning parameters of PSO during the optimization process using fuzzy logics was explored in Yuhui Shi et al.



As with most heuristics, the optimizer can be seen as a process taking in design points through an acquisition function and providing promising investigation points. The behaviour of the optimizer is influenced by its parameters.

AFPSO implements a Fuzzy Inference Engine (FIE) in a similar fashion. The FIE **probes** information in the optimizer and processes those into parameters which are then fed back into the optimizer.

The implementation

Agent's probes

An agent **i** knows the **proximity to its best friend**,

$$\delta_i = \frac{\|\vec{x}_i - \vec{b}_i\|}{\|\vec{u} - \vec{l}\|},$$

Self and best friend's position

Space diagonal

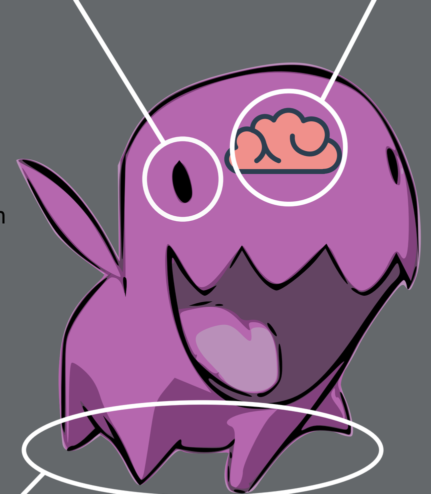
its rank modification or **improvement**

$$\phi_i = 4\frac{\min(\max(r|_i^t - r|_i^{t-1}, -S/4), S/4)}{S}$$

Rank at iteration t in the swarm

Number of agents

and the **objective evaluation counter.**



Search method

The search procedure is mostly unaltered except for parameters exposed to the FIE. In our case, PSO has a **weakly emergent** search procedure from the agents' simple behaviour. Each agent follows two attractors: its best solution and its neighbourhood's best solution. The neighbourhood is defined in the swarm's social network: the topology.
Two exposed parameters are, among **five others**:

- ω ▶ The tendency to stay on its track: **inertia**
- h ▶ Probability to use Quantum PSO update: **hybridization**

Rule Processing

Each controlled **parameter** has an associated fuzzy rules triplet which could read

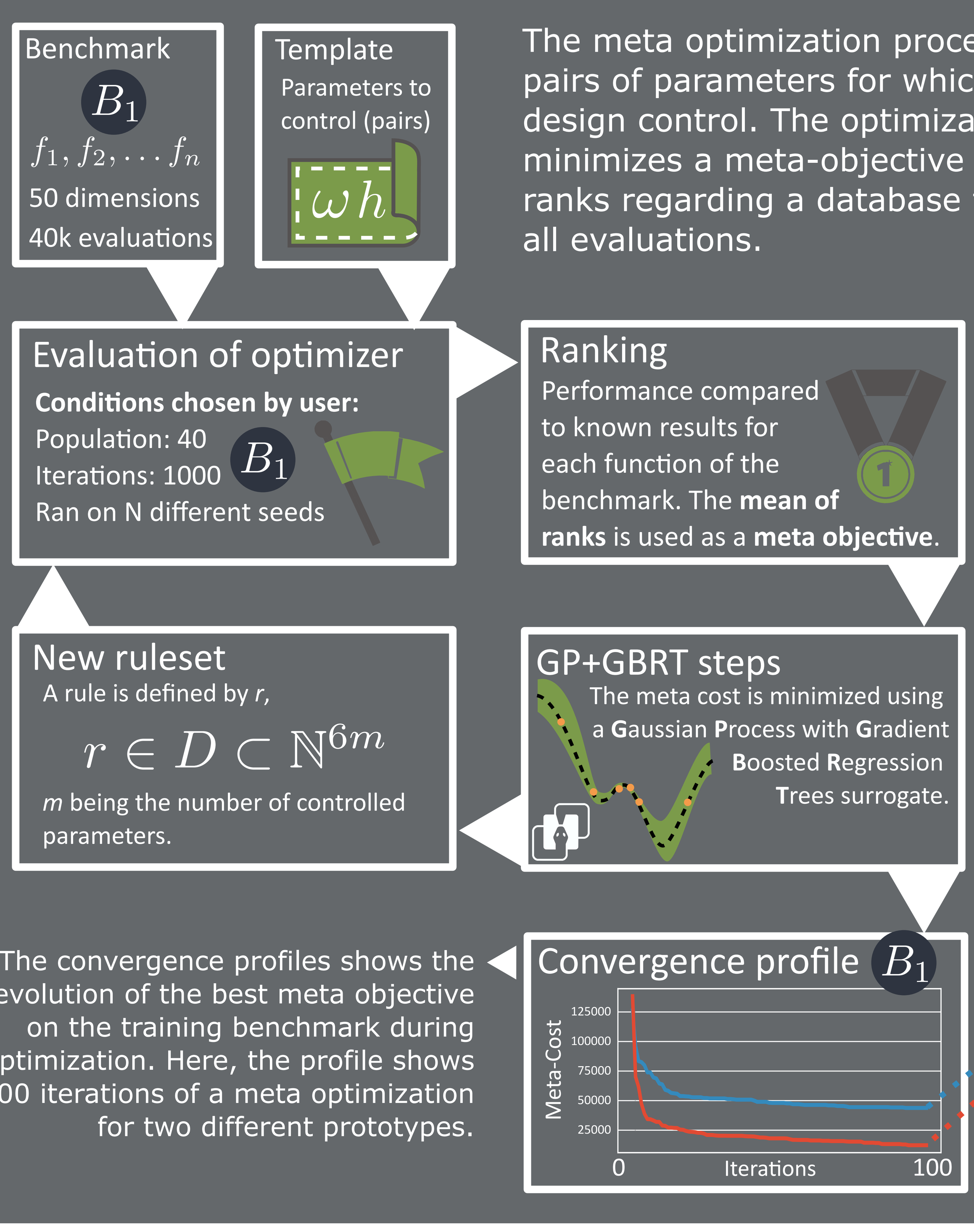
C is LOW if A is LOW and B is HIGH
C is MED if A is MED and B is MED
C is HIGH if A is HIGH and B is LOW

In this example, a *higher* **probe** A reinforces **parameter** C while *high* B implies *low* C.

The **ruleset** gets compiled and is processed by the agent at each iteration to produce new parameters from the probes.

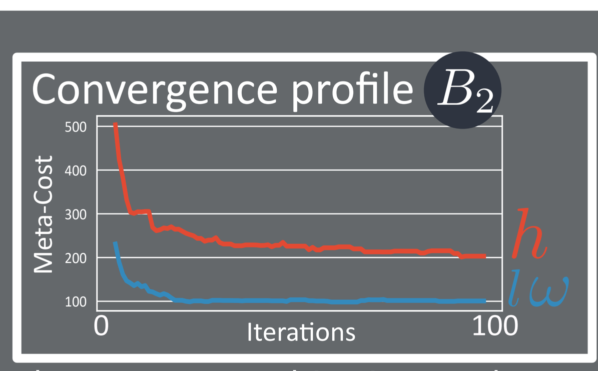
All agents in the swarm share the same FIE but they process their own probes.
The concrete details of rules application are not explained here. We will however indicate critical choices: we used the **Zadeh operators** and **Sugeno inference** to implement our engine.

Training & Validation

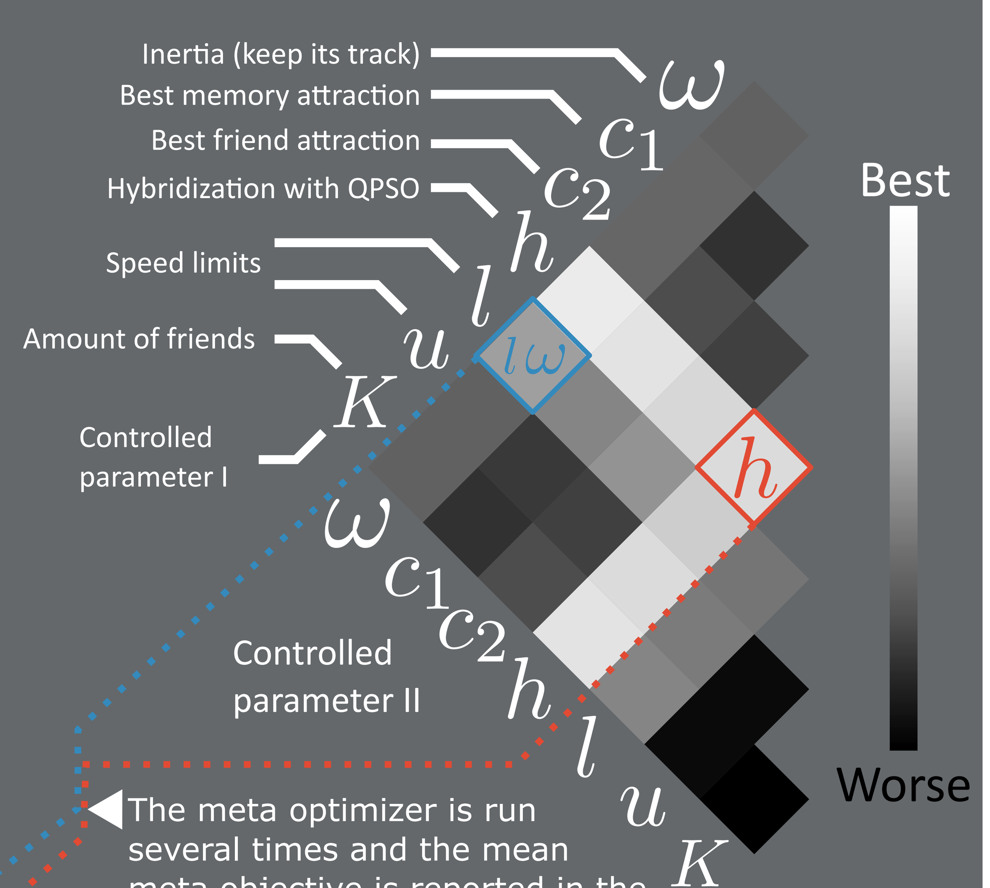


The meta optimization process takes in pairs of parameters for which we want to design control. The optimization loop minimizes a meta-objective based on ranks regarding a database that collects all evaluations.

The convergence profiles shows the evolution of the best meta objective on the training benchmark during optimization. Here, the profile shows 100 iterations of a meta optimization for two different prototypes.



The same meta objective can be computed on another benchmark for the same design points that form the training profile. This leads to a validation profile which shows that both profiles converge!

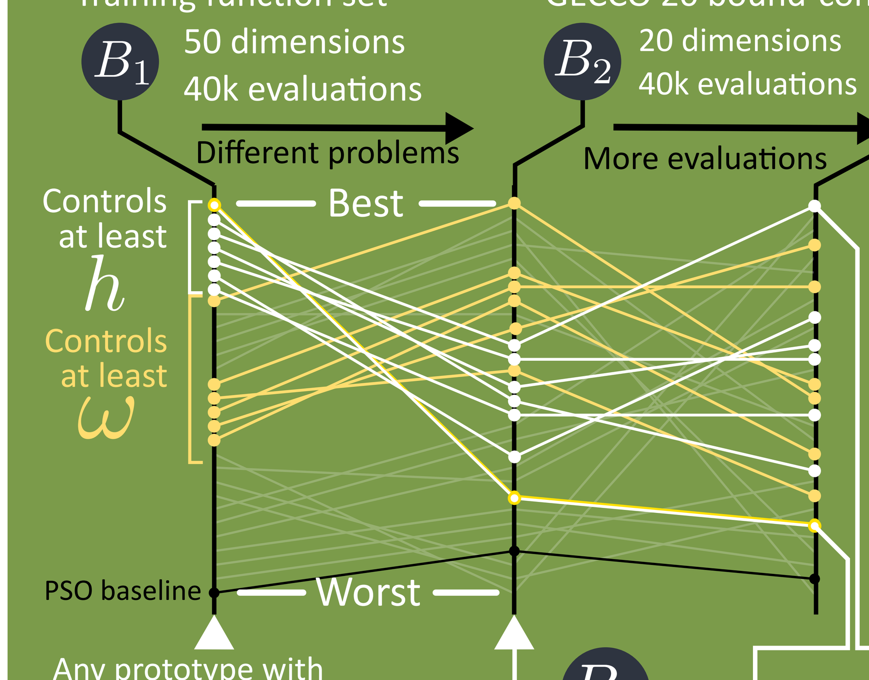


The meta optimizer is run several times and the mean meta objective is reported in the map above for B_1. Configurations controlling hybdridization showed great performance on the training functions.

Benchmark

Training function set
B_1 50 dimensions
40k evaluations

GECCO'20 bound-contrained competition
B_2 20 dimensions
40k evaluations
B_3 20 dimensions
1e8 evaluations



We take our 28 different controllers (trained instance of a prototype) and evaluate their mean meta objective on those benchmarks.
Most of the prototypes showed better performance than standard PSO on all benchmarks.
Absolute performance remained high even for worst cases like $\omega\&h$

Any prototype with hybridization or inertia controls are good fitters for B_1

Switching to B_2, inertia controls take the lead but ranks remain stable.

B_3

	$\omega\&h$	$l\&h$	PSO	GA	DE
cec_0	1469.0(1719.3)	0.0(0.0)	2.0e+08(2.0e+08)	5857.8(3251.9)	100.9(0.2)
cec_1	729.8(292.3)	267.5(260.8)	534.2(534.2)	1133.0(48.3)	2442.2(538.3)
cec_2	56.5(20.0)	40.1(7.6)	91.5(91.5)	728.2(3.1)	764.3(8.9)
cec_3	3.1(1.3)	1.4(0.4)	20.3(20.3)	1901.7(0.4)	1915.4(0.8)
cec_4	1161.2(1318.3)	148.4(117.2)	3950.3(3950.3)	2.0e+05(1.3e+05)	1967.8(173.6)
cec_5	12.4(12.2)	7.3(0.3)	45.1(45.1)	1601.9(0.7)	1613.0(2.4)
cec_6	244.1(99.8)	9.9(11.7)	3257.2(3257.2)	5.8e+04(3.2e+04)	2120.5(10.8)
cec_7	111.7(12.0)	103.8(2.0)	230.5(230.5)	2967.3(630.0)	2339.5(1.9)
cec_8	433.7(11.4)	417.8(7.8)	419.0(419.0)	2865.8(14.7)	2948.6(10.5)
cec_9	461.9(27.8)	424.8(21.1)	497.2(497.2)	2978.8(19.9)	2936.3(5.4)

B_1 $\mu(\sigma)$

	$\omega\&h$	$l\&h$	PSO	GA	DE
Sphere	0.0(0.0)	0.0(0.0)	0.1(0.1)	0.5(0.1)	0.4(0.1)
Ackley	0.0(0.0)	0.0(0.0)	9.8(9.8)	12.9(0.6)	20.4(0.2)
Rastrigin	0.0(0.0)	0.0(0.0)	209.0(209.0)	191.0(21.2)	483.3(28.8)
Rosenbrock	47.3(0.9)	47.4(0.4)	165.2(165.2)	664.6(126.1)	923.2(220.6)
Styblinski-Tank	181.9(51.3)	269.1(134.5)	392.5(392.5)	239.0(28.6)	699.7(31.4)
Schwefel	8124.9(1035.1)	8245.2(1603.1)	1.2e+04(1.2e+04)	5037.0(323.0)	1.4e+04(2.4e+02)
Griewank	0.0(0.0)	0.0(0.0)	1.2(1.2)	1.7(0.1)	1.8(0.4)
Chung Reynolds	0.0(0.0)	0.0(0.0)	5.6e+09(5.6e+09)	6.3e+10(9.3e+09)	7.2e+10(2.8e+10)
Qing	9232.0(2198.6)	9017.9(2060.7)	4.0e+08(4.0e+08)	5.7e+09(8.3e+08)	6.4e+09(4.1e+09)
HappyCat	0.4(0.0)	0.5(0.1)	0.7(0.7)	0.7(0.0)	0.8(0.1)
Salomon	0.1(0.0)	0.2(0.0)	2.0(2.0)	4.4(0.4)	4.5(0.3)
XinSheYang	0.0(0.0)	0.0(0.0)	0.0(0.0)	0.0(0.0)	0.0(0.0)
Bent Cigar	0.0(0.0)	0.0(0.0)	2.1e+10(2.1e+10)	8.3e+10(7.6e+09)	1.0e+11(4.9e+10)

▲ Absolute performance remains higher even on foreign benchmark B_3.

◀ Outstanding performance is achieved on the training benchmark B_1. With the exception of Schwefel function.

References & Thanks

Yuhui Shi et al., *Fuzzy adaptive particle swarm optimization* doi: 10.1109/CEC.2001.934377.
Kennedy, J., & Eberhart, R. *Particle swarm optimization* doi: 10.1109/ICNN.1995.488968
Sun, J. et al. PSO *with particles having quantum behavior* doi: 10.1109/CEC.2004.1330875.
C. T. Yue et al., *Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization* available: https://github.com/P-N-Suganthan/2020-Bound-Constrained-Opt-Benchmark.

NumPy Scikit-Optimize

Conclusions

We successfully designed and implemented a **framework** for **systematically designing FIE**s for heuristics. Such designed controllers showed **great specialization** of the training function but also **good generalization,** performing well of a past **GECCO competition**, **far from training conditions**.
The work provides the methodology, from there, any implementation should design its own training and validation benchmark, knowing the problem at hand.