# Squirrel: A Switching Hyperparameter Optimizer

## Description of the entry by AutoML.org & IOHprofiler to the NeurIPS 2020 BBO challenge

**Noor Awad** [1] **Gresa Shala** [1] **Difan Deng** [2] **Neeratyoy Mallik** [1] **Matthias Feurer** [1]
**Katharina Eggensperger** [1] **André Biedenkapp** [1] **Diederick Vermetten** [3] **Hao Wang** [3]
**Carola Doerr** [4] **Marius Lindauer** [2] **Frank Hutter** [1,5]
[1]University of Freiburg [2]Leibniz University Hannover [3]Leiden University
[4]Sorbonne Université, CNRS, LIP6 [5]Bosch Center for Artificial Intelligence

In this short note, we describe our submission to the NeurIPS 2020 BBO challenge. Motivated by the fact that different optimizers work well on different problems, our approach *switches* between different optimizers.[1] Since the team names on the competition's leaderboard were randomly generated "alliteration nicknames", consisting of an adjective and an animal with the same initial letter, we called our approach the *Switching Squirrel*, or here, short, *Squirrel*.

The challenge mandated to suggest 16 successive batches of 8 hyperparameter configurations at a time. We chose to only use one optimizer for a given batch, warmstarted with all previous observations. **In our Squirrel framework, we switched between the following components**:

1. An initial design (for known hyperparameter spaces: found by meta-learning; otherwise: selected by differential evolution) (3 batches);
2. Optimization using Bayesian optimization by integrating the SMAC optimizer [6, 10] with a portfolio of different triplets of surrogate model, acquisition function, and output space transformation (8 batches); and
3. Optimization using Differential Evolution (DE) [15] with parameter adaptation (5 batches).

We now discuss these components in turn.

**Meta-learning the initial design** Since the challenge was a blackbox challenge, it was a priori unknown which algorithms should be optimized using which configuration spaces. In order to mimic what one would do in a real-world optimization service based on the bayesmark [16] environment, for each of the configuration spaces provided by bayesmark we applied meta-learning to select an initial design of strong configurations to start from [12, 4]. For each of these spaces, we performed extensive offline optimization on 11 datasets (the 6 datasets built into bayesmark and 5 additional OpenML [17] datasets[2]), both for optimizing accuracy and negative log-likelihood. This resulted in 22 configurations[3] for each of the configuration spaces in bayesmark. When facing one of these known spaces (defined as an exact match of dimensions, names, and search ranges), we warm-started Squirrel with the corresponding 22 configurations, plus 2 randomly sampled ones, using 3 batches of 8 configurations.

When facing an unknown configuration space, Squirrel initialized the optimization procedure by employing DE with a random initial design, using DE for 3 batches of 8 configurations.

**Bayesian optimization with SMAC** Squirrel exploits that the SMAC3 package implements different surrogate models (including GPs [11] & random forests [2] (RF)) and different acquisition functions (Expected Improvement (EI) [7], LogEI [6], probability of improvement (PI) [8], and lower

[1]Switching between algorithms also relates to work on *chaining* or *algorithm schedules*.

[2]We only considered 5 additional datasets for lack of time and compute power; better performance could be obtained using more datasets for meta-learning.

[3]Extracted from the offline performance data using IOHanalyzer [18].

confidence bound (LCB) [14]) that can be chosen from, making it easier to adapt to different loss landscapes and combat model mismatch. In view of the batch setting of the competition, Squirrel employs a portfolio of different triplets of surrogate model, acquisition function, and output space transformation to sample a batch of configurations in each iteration. This choice exploits that different acquisition functions implement different trade-off strategies between exploitation and exploration, while different surrogate models allow to fit different kinds of functions. Furthermore, Squirrel applies different transformations of the output space before fitting the surrogate model, incl. `log` and `copula` [13] transformations. By doing so, we warp the loss function landscape, which further promotes the diversity of the suggestions. For the portfolio, we made the following choices. GPs usually describe low-dimensional response functions better and EI is a very robust acquisition function; thus, we overall favoured GPs and EI. In total, we used 8 different triplets as follows: GP + EI + no_transform, GP + PI + no_transform, GP + LCB + no_transform, GP + EI + copula_transform, GP + logEI + log_transform, RF + EI + no_transform, RF + logEI + logTransform and RF + EI + copula_transform. These choices could likely be optimized further. To further promote diversity among the proposals we use the Kriging Believer [5] and randomly reorder the triplets for each batch before sequentially querying them for suggestions.

**Optimization with DE** To complement the global search strategy of Bayesian optimization, Squirrel runs differential evolution (DE) as a more greedy and final stage for the last 5 batches. Our version of DE takes the observations from the previous stages as an input to initialize a population of size 8 (the required batch size) by taking the best configuration observed so far, and choosing the other configurations of the population randomly from all previously-evaluated configurations. It then starts the evolutionary search by iterating three operations: mutation, crossover, and selection. For the mutation operation, we generated a child/offspring for each configuration in the population by the greedy $best/2$ mutation strategy [15]. The scaling factor $F$ which controls the difference vector in the mutation operation is adapted using a decreasing sinusoidal equation [3, 1]. This adaptation leads to more exploration of the search space when DE starts and more exploitation around the best-found region in later phases of the search. When the mutation phase is completed, the binomial (also known as uniform) crossover [15] is applied to each target configuration $X_{i,g}$ and its corresponding mutant configuration $V_{i,g}$ to generate an offspring $U_{i,g}$. The crossover rate $CR_{i,g}$ is sampled from the normal distribution $\mathcal{N}(0.5, 0.01)$, determining the probability by which a hyperparameter value is copied from the mutant configuration (the hyperparameter value is taken from the parent otherwise). After the final offspring is evaluated, truncation selection is applied: the trial configuration $U_{i,g}$ replaces its parent if it has a better function value and is discarded otherwise (in this case the parent configuration $X_{i,g}$ remains in the population).

**Result in the official BBO competition** Our submission ranked 3rd on the public leaderboard of the NeurIPS 2020 BBO challenge. For the final evaluation, it would have also ranked 3rd, had it not had a minor bug regarding the type of its output object, which let it crash.

In detail, the API specifications asked for the optimizers to return a list of dictionaries. Squirrel is built around a custom library for handling hyperparameter configurations [9], and we returned a list of its `Configuration` objects (a `Configuration` is a dict-like object implementing many methods of standard Python dictionaries, including all methods used by bayesmark). This worked with the challenge starter kit, the public bayesmark code and the testing server throughout the entire 3-months submission period without causing an error, but failed in the final evaluation, possibly because it does not fully implement the specifications of a Python dictionary. This indicates that the final evaluation used benchmarks with stricter type checks or different evaluation code than the code in the starter kit/bayesmark/test server. After the official rankings, the organizers reran our code with a 1-line fix (which replaced our `Configuration` output object with its corresponding `dict` object); this fixed version yielded a score of 92.551495, which would have placed it on rank 3.

**Result in the BBO competition's warmstarting-friendly leaderboard** While the final evaluation used configuration spaces with altered hyperparameter names, the organizers also computed an alternate "warmstarting-friendly leaderboard" that uses un-altered hyperparameter names and thereby provides full information about the identity of the algorithm being optimized. Our submission won 1st place in this leaderboard (with a score of 94.845476 and the organizers' bootstrap analysis showing a 100% confidence in this 1st place ranking), winning a prize of 3 000 USD generously sponsored by 4Paradigm. This success underlines the power of even simple meta-learning approaches.

## References

[1] N. Awad, M. Ali, P. Suganthan, and R. Reynolds. An ensemble sinusoidal parameter adaptation incorporated with l-shade for solving cec2014 benchmark problems. In *Proceedings of IEEE congress on evolutionary computation (CEC'16)*. IEEE, 2016.

[2] L. Breimann. Random forests. *Machine Learning Journal*, 45:5–32, 2001.

[3] A. Draa, S. Bouzoubia, and I. Boukhalfa. A sinusoidal differential evolution algorithm for numerical optimisation. *Applied Soft Computing*, 27:99 – 126, 2015.

[4] M. Feurer, J. Springenberg, and F. Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In B. Bonet and S. Koenig, editors, *Proceedings of the Twenty-nineth National Conference on Artificial Intelligence (AAAI'15)*, pages 1128–1135. AAAI Press, 2015.

[5] D. Ginsbourger, R. Le Riche, and L. Carraro. Kriging is well-suited to parallelize optimization. In *Computational Intelligence in Expensive Optimization Problems*, pages 131–162. Springer, 2010.

[6] F. Hutter, H. Hoos, and K. Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In C. Coello, editor, *Proceedings of the Fifth International Conference on Learning and Intelligent Optimization (LION'11)*, volume 6683 of *Lecture Notes in Computer Science*, pages 507–523. Springer, 2011.

[7] D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black box functions. *Journal of Global Optimization*, 13:455–492, 1998.

[8] H. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86(1):97–106, 1964.

[9] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, J. Marben, P. Müller, and F. Hutter. BOAH: a tool suite for multi-fidelity bayesian optimization & analysis of hyperparameters. *arXiv:1908.06756 [cs.LG]*, 2019.

[10] M. Lindauer, K. Eggensperger, M. Feurer, S. Falkner, A. Biedenkapp, and F. Hutter. SMAC v3: Algorithm configuration in Python. `https://github.com/automl/SMAC3`, 2017.

[11] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[12] M. Reif, F. Shafait, and A. Dengel. Meta-learning for evolutionary parameter optimization of classifiers. *Machine Learning*, 87:357–380, 2012.

[13] D. Salinas, H. Shen, and V. Perrone. A quantile-based approach for hyperparameter transfer learning. In H. Daumé and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*, volume 82, pages 8438–8448. Proceedings of Machine Learning Research, 2020.

[14] N. Srinivas, A. Krause, S. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*, pages 1015–1022. Omnipress, 2010.

[15] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[16] Uber. Bayesopt benchmark. `https://bayesmark.readthedocs.io/en/latest/`.

[17] J. Vanschoren, J. van Rijn, B. Bischl, and L. Torgo. OpenML: Networked science in machine learning. *SIGKDD Explor. Newsl.*, 15(2):49–60, 2014.

[18] H. Wang, D. Vermetten, F. Ye, C. Doerr, and T. Bäck. IOHanalyzer: Performance analysis for iterative optimization heuristic. *arXiv:2007.03953 [cs.NE]*, 2020.