# CBX: Python and Julia Packages for Consensus-Based Interacting Particle Methods

Rafael Bailo [1], **Alethea Barbaro** [2], **Susana N. Gomes** [3], **Konstantin Riedl** [4,5], **Tim Roith** [6], **Claudia Totzeck** [7], **and Urbain Vaes** [8,9]

1 Mathematical Institute, University of Oxford, United Kingdom **2** Delft University of Technology, The Netherlands **3** Mathematics Institute, University of Warwick, United Kingdom **4** Technical University of Munich, Germany **5** Munich Center for Machine Learning, Germany **6** Helmholtz Imaging, Deutsches Elektronen-Synchrotron DESY, Hamburg, Germany **7** University of Wuppertal, Germany **8** MATHERIALS team, Inria Paris, France **9** École des Ponts ParisTech, Marne-la-Vallée, France

Summary

We introduce CBXPy and ConsensusBasedX.jl, Python and Julia implementations of consensus-based interacting particle systems (CBX), which generalise consensus-based optimization methods (CBO) for global, derivative-free optimisation. The *raison d'être* of our libraries is twofold: on the one hand, to offer high-performance implementations of CBX methods that the community can use directly, while on the other, providing a general interface that can accommodate and be extended to further variations of the CBX family. Python and Julia were selected as the leading high-level languages in terms of usage and performance, as well as for their popularity among the scientific computing community. Both libraries have been developed with a common *ethos*, ensuring a similar API and core functionality, while leveraging the strengths of each language and writing idiomatic code.

Mathematical background

Consensus-based optimisation (CBO) is an approach to solve, for a given (continuous) *objective function* $f : \mathbb{R}^d \to \mathbb{R}$, the *global minimisation problem*

$$x^* = \operatorname*{argmin}_{x \in \mathbb{R}^d} f(x),$$

i.e., the task of finding the point x^* where f attains its lowest value. Such problems arise in a variety of disciplines including engineering, where x might represent a vector of design parameters for a structure and f a function related to its cost and structural integrity, or machine learning, where x could comprise the parameters of a neural network and f the empirical loss, which measures the discrepancy of the neural network prediction with the observed data.

In some cases, so-called *gradient-based methods* (those that involve updating a guess of x^* by evaluating the gradient ∇f) achieve state-of-the-art performance in the global minimisation problem. However, in scenarios where f is *non-convex* (when f could have many *local minima*), where f is *non-smooth* (when ∇f is not well-defined), or where the evaluation of ∇f is impractical due to cost or complexity, *derivative-free* methods are needed. Numerous techniques exist for derivative-free optimisation, such as *random* or *pattern search* (Friedman & Savage, 1947; Hooke & Jeeves, 1961; Rastrigin, 1963), *Bayesian optimisation* (Močkus, 1975) or *simulated annealing* (Henderson et al., 2003). Here, we focus on *particle-based methods*,

specifically, consensus-based optimisation (CBO), as proposed by Pinnau et al. (2017), and the consensus-based taxonomy of related techniques, which we term *CBX*.

CBO uses a finite number N of *agents* (particles), $x_t = (x_t^1, ..., x_t^N)$, dependent on time t, to explore the landscape of f without evaluating any of its derivatives (as do other CBX methods). The agents evaluate the objective function at their current position, $f(x_t^i)$, and define a *consensus point* c_α. This point is an approximation of the global minimiser x^*, and is constructed by weighing each agent's position against the *Gibbs-like distribution* $\exp(-\alpha f)$ (Boltzmann, 1868). More rigorously,

$$c_\alpha(x_t) = \frac{1}{\sum_{i=1}^N \omega_\alpha(x_t^i)} \sum_{i=1}^N x_t^i \, \omega_\alpha(x_t^i), \quad \text{where} \quad \omega_\alpha(\,\cdot\,) = \exp(-\alpha f(\,\cdot\,)),$$

for some $\alpha > 0$. The exponential weights in the definition favour those points x_t^i where $f(x_t^i)$ is lowest, and comparatively ignore the rest, particularly for larger α. If all the found values of the objective function are approximately the same, $c_\alpha(x_t)$ is roughly an arithmetic mean. Instead, if one particle is much better than the rest, $c_\alpha(x_t)$ will be very close to its position.

Once the consensus point is computed, the particles evolve in time following the *stochastic differential equation* (SDE)

$$\mathrm{d}x_t^i = -\lambda \underbrace{(x_t^i - c_\alpha(x_t)) \, \mathrm{d}t}_{\text{consensus drift}} + \sigma \underbrace{\|x_t^i - c_\alpha(x_t)\| \, \mathrm{d}B_t^i}_{\text{scaled diffusion}},$$

where λ and σ are positive parameters, and where B_t^i are independent Brownian motions in d dimensions. The *consensus drift* is a deterministic term that drives each agent towards the consensus point, with rate λ. Meanwhile, the *scaled diffusion* is a stochastic term that encourages exploration of the landscape. The scaling factor of the diffusion is proportional to the distance of the particle to the consensus point. Hence, whenever the position of a particle and the location of the weighted mean coincide, the particle stops moving. On the other hand, if the particle is far away from the consensus, its evolution has a stronger exploratory behaviour. While both the agents' positions and the consensus point evolve in time, it has been proven that all agents eventually reach the same position and that the consensus point $c_\alpha(x_t)$ is a good approximation of x^* (Carrillo et al., 2018; Fornasier, Klock, et al., 2021). Other variations of the method, such as CBO with anisotropic noise (Carrillo et al., 2021), *polarised CBO* (Bungert et al., 2024), or *consensus-based sampling* (CBS) (Carrillo et al., 2022) have also been proposed.
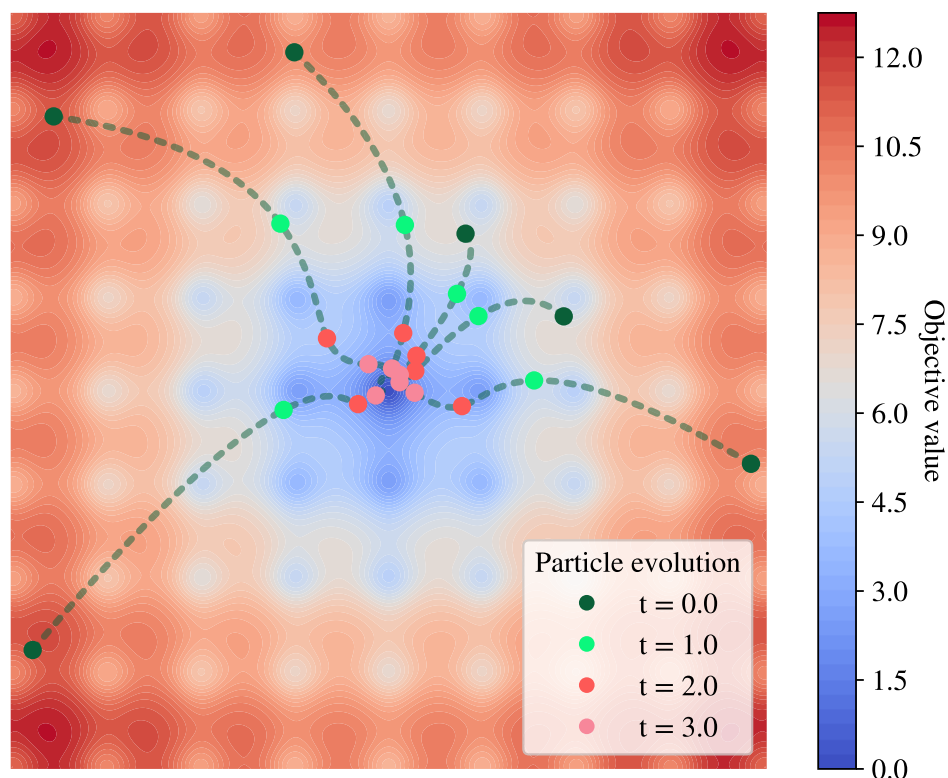
In practice, the solution to the SDE above cannot be found exactly. Instead, an *Euler–Maruyama scheme* (Kloeden & Platen, 1992) is used to update the position of the agents. The update is given by

$$x^i \leftarrow x^i - \lambda \, \Delta t \, (x^i - c_\alpha(x)) + \sigma \sqrt{\Delta t} \, \|x^i - c_\alpha(x)\| \, \xi^i,$$

where $\Delta t > 0$ is the *step size* and $\xi^i \sim \mathcal{N}(0, \mathrm{Id})$ are independent, identically distributed, standard normal random vectors.

As a particle-based family of methods, CBX is conceptually related to other optimisation approaches which take inspiration from biology, like *particle-swarm optimisation* (PSO) (Kennedy & Eberhart, 1995), from physics, like *simulated annealing* (SA) (Henderson et al., 2003), or from other heuristics (Bayraktar et al., 2013; Chandra Mohan & Baskaran, 2012; Karaboga et al., 2012; Yang, 2009). However, unlike many such methods, CBX has been designed to be compatible with rigorous convergence analysis at the mean-field level (the infinite-particle limit, see Huang & Qiu, 2022). Many convergence results have been shown, whether in the original formulation (Carrillo et al., 2018; Fornasier, Klock, et al., 2021), for CBO with

anisotropic noise (Carrillo et al., 2021; Fornasier et al., 2022), with memory effects (Riedl, 2023), with truncated noise (Fornasier et al., 2024), for polarised CBO (Bungert et al., 2024), and PSO (Huang et al., 2023). The relation between CBO and *stochastic gradient descent* has been recently established by Riedl et al. (2023), which suggests a previously unknown yet fundamental connection between derivative-free and gradient-based approaches.

Figure 1: Typical evolution of a CBO method minimising the Ackley function (Ackley, 1987).

CBX methods have been successfully applied and extended to several different settings, such as constrained optimisation problems (Borghi et al., 2023b; Fornasier, Huang, et al., 2021), multi-objective optimisation (Borghi et al., 2023a; Klamroth et al., 2024), saddle-point problems (Huang et al., 2024), federated learning tasks (Carrillo et al., 2023), uncertainty quantification (Althaus et al., 2023), or sampling (Carrillo et al., 2022).

Statement of need

In general, very few implementations of CBO already exist, and none have been designed with the generality of other CBX methods in mind. We summarise here the related software:

Regarding Python, we refer to PyPop7 (Duan et al., 2022) and scikit-opt (Guo, 2021) for a collection of various derivative-free optimisation strategies. For packages connected to Bayesian optimisation, we refer to BayesO (Kim & Choi, 2023), bayesian-optimization (Nogueira, 2014–), GPyOpt (The GPyOpt authors, 2016), GPflowOpt (Knudde et al., 2017), pyGPGO (Jiménez & Ginebra, 2017), PyBADS (Singh & Acerbi, 2024) and BoTorch (Balandat et al., 2020). Furthermore, CMA-ES (Hansen & Ostermeier, 1996) was implemented in pycma (Hansen et al., 2019). To the best of our knowledge the connection between consensus-based methods and evolution strategies is not fully understood, and is therefore an interesting future direction. PSO and SA implementations are already available in PySwarms (Miranda, 2018), scikit-opt (Guo, 2021), DEAP (Fortin et al., 2012) and pagmo (Biscani et al., 2017). They

are widely used by the community and provide a rich framework for the respective methods. However, adjusting these implementations to CBO is not straightforward. The first publicly available Python packages implementing CBX algorithms were given by some of the authors together with collaborators. Tukh & Riedl (2022) implement standard CBO (Pinnau et al., 2017), and the package `PolarCBO` (Roith et al., 2023) provides an implementation of polarised CBO (Bungert et al., 2024). CBXPy is a significant extension of the latter, which was tailored to the polarised variant. The code architecture was generalised, which allowed the implementation of the whole CBX family within a common framework.

Regarding Julia, PSO and SA methods are, among others, implemented in `Optim.jl` (Mogensen & Riseth, 2018), `Metaheuristics.jl` (Mejía-de-Dios & Mezura-Montes, 2022), and `Manopt.jl` (Bergmann, 2022). PSO and SA are also included in the meta-library `Optimization.jl` (Dixit & Rackauckas, 2023), as well as Nelder–Mead, which is a direct search method. The latter is also implemented in `Manopt.jl` (Bergmann, 2022), which further provides a manifold variant of CMA-ES (Colutto et al., 2009). One of the authors gave the first specific Julia implementation of standard CBO `Consensus.jl` (Bailo, 2023). That package has now been deprecated in favour of ConsensusBasedX.jl, which improves the performance of the CBO implementation with a type-stable and allocation-free implementation. The package also adds a CBS implementation, and overall presents a more general interface that accomodates the wider CBX class of methods.

Features

CBXPy and ConsensusBasedX.jl provide a lightweight and high-level interface. An existing function can be optimised with just one call. Method selection, parameters, different approaches to particle initialisation, and termination criteria can be specified directly through this interface, offering a flexible point of entry for the casual user. Some of the methods provided are standard CBO (Pinnau et al., 2017), CBO with mini-batching (Carrillo et al., 2021), polarised CBO (Bungert et al., 2024), CBO with memory effects (Grassi & Pareschi, 2021; Riedl, 2023), and consensus-based sampling (CBS) (Carrillo et al., 2022). Parallelisation tools are available.

A more proficient user will benefit from the fully documented interface, which allows the specification of advanced options (e.g., debug output, the noise model, or the numerical approach to the matrix square root of the weighted ensemble covariance matrix). Both libraries offer performance evaluation methods as well as visualisation tools.

Ultimately, a low-level interface (including documentation and full-code examples) is provided. Both libraries have been designed to express common abstractions in the CBX family while allowing customisation. Users can easily implement new CBX methods or modify the behaviour of the existing implementation by strategically overriding certain hooks. The stepping of the methods can also be controlled manually.

CBXPy

Figure 2: CBXPy logo.

Most of the CBXPy implementation uses basic Python functionality, and the agents are handled as an array-like structure. For certain specific features, like broadcasting-behaviour, array copying, and index selection, we fall back to the numpy implementation (Harris

et al., 2020). However, it should be noted that an adaptation to other array or tensor libraries like PyTorch (Paszke et al., 2019) is straightforward. Compatibility with the latter enables gradient-free deep learning directly on the GPU, as demonstrated in the documentation.

The library is available on GitHub and can be installed via `pip`. It is licensed under the MIT license. Below, we provide a short example on how to optimise a function with CBXPy.

```python
from cbx.dynamics import CBO        # import the CBO class
f = lambda x: x[0]**2 + x[1]**2    # define the function to minimise
x = CBO(f, d=2).optimize()         # run the optimisation
```

More examples and details on the implementation are available in the documentation.

ConsensusBasedX.jl

Figure 3: ConsensusBasedX.jl logo.

ConsensusBasedX.jl has been almost entirely written in native Julia (with the exception of a single call to LAPACK). The code has been developed with performance in mind, thus the critical routines are fully type-stable and allocation-free. A specific tool is provided to benchmark a typical method iteration, which can be used to detect allocations. Through this tool, unit tests are in place to ensure zero allocations in all the provided methods. The benchmarking tool is also available to users, who can use it to test their implementations of f, as well as any new CBX methods.

Basic function minimisation can be performed by running:

```julia
using ConsensusBasedX        # load the ConsensusBasedX package
f(x) = x[1]^2 + x[2]^2       # define the function to minimise
x = minimise(f, D = 2)       # run the minimisation
```

The library is available on GitHub. It has been registered in the general Julia registry, and therefore it can be installed by running `]add ConsensusBasedX`. It is licensed under the MIT license. More examples and full instructions are available in the documentation.

Acknowledgements

References

Ackley, D. H. (1987). A connectionist machine for genetic hillclimbing. In *The Kluwer International Series in Engineering and Computer Science* (Vol. 28). Springer US. https://doi.org/10.1007/978-1-4613-1997-9

Althaus, K., Papaioannou, I., & Ullmann, E. (2023). *Consensus-based rare event estimation*. https://doi.org/10.48550/arXiv.2304.09077

Bailo, R. (2023). *Consensus.jl* (Version 1.0.0). https://doi.org/10.5281/zenodo.7754236

Balandat, M., Karrer, B., Jiang, D. R., Daulton, S., Letham, B., Wilson, A. G., & Bakshy, E. (2020). BoTorch: A framework for efficient monte-carlo Bayesian optimization. *Advances in Neural Information Processing Systems 33*. http://arxiv.org/abs/1910.06403

Bayraktar, Z., Komurcu, M., Bossard, J. A., & Werner, D. H. (2013). The wind driven optimization technique and its application in electromagnetics. *IEEE Transactions on Antennas and Propagation*, *61*(5), 2745–2757. https://doi.org/10.1109/TAP.2013.2238654

Bergmann, R. (2022). Manopt.jl: Optimization on manifolds in Julia. *Journal of Open Source Software*, *7*(70), 3866. https://doi.org/10.21105/joss.03866

Biscani, F., Izzo, D., & Märtens, M. (2017). *Esa/pagmo2: Pagmo 2.6*. https://doi.org/10.5281/zenodo.1054110

Boltzmann, L. (1868). Studien über das Gleichgewicht der lebendigen Kraft zwischen bewegten materiellen Punkten. *Wiener Berichte*, *58*, 517–560.

Borghi, G., Herty, M., & Pareschi, L. (2023a). An adaptive consensus based method for multi-objective optimization with uniform Pareto front approximation. *Applied Mathematics & Optimization*, *88*(2), 1–43. https://doi.org/10.1007/s00245-023-10036-y

Borghi, G., Herty, M., & Pareschi, L. (2023b). Constrained consensus-based optimization. *SIAM Journal on Optimization*, *33*(1), 211–236. https://doi.org/10.1137/22M1471304

Bungert, L., Roith, T., & Wacker, P. (2024). Polarized consensus-based dynamics for optimization and sampling. *Math. Program.* https://doi.org/10.1007/s10107-024-02095-y

Carrillo, J. A., Choi, Y.-P., Totzeck, C., & Tse, O. (2018). An analytical framework for consensus-based global optimization method. *Mathematical Models and Methods in Applied Sciences*, *28*(6), 1037–1066. https://doi.org/10.1142/S0218202518500276

Carrillo, J. A., Garcia Trillos, N., Li, S., & Zhu, Y. (2023). *FedCBO: Reaching group consensus in clustered federated learning through consensus-based optimization*. https://doi.org/10.48550/arXiv.2305.02894

Carrillo, J. A., Hoffmann, F., Stuart, A. M., & Vaes, U. (2022). Consensus-based sampling. *Studies in Applied Mathematics*, *148*(3), 1069–1140. https://doi.org/10.1111/sapm.12470

Carrillo, J. A., Jin, S., Li, L., & Zhu, Y. (2021). A consensus-based global optimization method for high dimensional machine learning problems. *ESAIM: Control, Optimisation and Calculus of Variations*, *27*, S5. https://doi.org/10.1051/cocv/2020046

Chandra Mohan, B., & Baskaran, R. (2012). A survey: Ant colony optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications*, *39*(4), 4618–4627. https://doi.org/10.1016/j.eswa.2011.09.076

Colutto, S., Fruhauf, F., Fuchs, M., & Scherzer, O. (2009). The CMA-ES on Riemannian manifolds to reconstruct shapes in 3-d voxel images. *IEEE Transactions on Evolutionary Computation*, *14*(2), 227–245. https://doi.org/10.1109/TEVC.2009.2029567

Dixit, V. K., & Rackauckas, C. (2023). *Optimization.jl: A unified optimization package*. Zenodo. https://doi.org/10.5281/ZENODO.7738525

Duan, Q., Zhou, G., Shao, C., Wang, Z., Feng, M., Yang, Y., Zhao, Q., & Shi, Y. (2022). *PyPop7: A pure-python library for population-based black-box optimization.* https://doi.org/10.48550/arXiv.2212.05652

Fornasier, M., Huang, H., Pareschi, L., & Sünnen, P. (2021). Consensus-based optimization on the sphere: Convergence to global minimizers and machine learning. *Journal of Machine Learning Research (JMLR)*, *22*, Paper No. 237, 55.

Fornasier, M., Klock, T., & Riedl, K. (2022). Convergence of anisotropic consensus-based optimization in mean-field law. In J. L. Jiménez Laredo, J. I. Hidalgo, & K. O. Babaagba (Eds.), *Applications of evolutionary computation* (pp. 738–754). Springer. https://doi.org/10.1007/978-3-031-02462-7_46

Fornasier, M., Klock, T., & Riedl, K. (2021). *Consensus-based optimization methods converge globally.* https://doi.org/10.48550/arXiv.2103.15130

Fornasier, M., Richtárik, P., Riedl, K., & Sun, L. (2024). Consensus-based optimisation with truncated noise. *European Journal of Applied Mathematics*, 1–24. https://doi.org/10.1017/S095679252400007X

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., & Gagné, C. (2012). DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, *13*, 2171–2175.

Friedman, M., & Savage, L. J. (1947). Planning experiments seeking maxima. *Techniques of Statistical Analysis*, 365–372.

Grassi, S., & Pareschi, L. (2021). From particle swarm optimization to consensus based optimization: Stochastic modeling and mean-field limit. *Mathematical Models and Methods in Applied Sciences*, *31*(8), 1625–1657. https://doi.org/10.1142/S0218202521500342

Guo, F. (2021). *scikit-opt.* https://github.com/guofei9987/scikit-opt

Hansen, N., Akimoto, Y., & Baudis, P. (2019). *CMA-ES/pycma on Github.* Zenodo, DOI:10.5281/zenodo.2559634. https://doi.org/10.5281/zenodo.2559634

Hansen, N., & Ostermeier, A. (1996). Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. *Proceedings of IEEE International Conference on Evolutionary Computation*, 312–317. https://doi.org/10.1109/ICEC.1996.542381

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Henderson, D., Jacobson, S. H., & Johnson, A. W. (2003). The theory and practice of simulated annealing. *Handbook of Metaheuristics*, 287–319. https://doi.org/10.1007/0-306-48056-5_10

Hooke, R., & Jeeves, T. A. (1961). "Direct search" solution of numerical and statistical problems. *Journal of the ACM (JACM)*, *8*(2), 212–229. https://doi.org/10.1145/321062.321069

Huang, H., & Qiu, J. (2022). On the mean-field limit for the consensus-based optimization. *Mathematical Methods in the Applied Sciences*, *45*(12), 7814–7831. https://doi.org/10.1002/mma.8279

Huang, H., Qiu, J., & Riedl, K. (2023). On the global convergence of particle swarm optimization methods. *Applied Mathematics & Optimization*, *88*(2), 30. https://doi.org/10.1007/s00245-023-09983-3

Huang, H., Qiu, J., & Riedl, K. (2024). Consensus-based optimization for saddle point

problems. *SIAM Journal on Control and Optimization*, *62*(2), 1093–1121. https://doi.org/10.1137/22M1543367

Jiménez, J., & Ginebra, J. (2017). pyGPGO: Bayesian optimization for Python. *Journal of Open Source Software*, *2*(19), 431. https://doi.org/10.21105/joss.00431

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2012). A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review*, *42*(1), 21–57. https://doi.org/10.1007/s10462-012-9328-0

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of ICNN'95-International Conference on Neural Networks*, *4*, 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

Kim, J., & Choi, S. (2023). BayesO: A Bayesian optimization framework in Python. *Journal of Open Source Software*, *8*(90), 5320. https://doi.org/10.21105/joss.05320

Klamroth, K., Stiglmayr, M., & Totzeck, C. (2024). Consensus-based optimization for multi-objective problems: A multi-swarm approach. *Journal of Global Optimization*. https://doi.org/10.1007/s10898-024-01369-1

Kloeden, P. E., & Platen, E. (1992). *Numerical solution of stochastic differential equations*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-12616-5

Knudde, N., van der Herten, J., Dhaene, T., & Couckuyt, I. (2017). *GPflowOpt: A Bayesian Optimization Library using TensorFlow*. https://doi.org/10.48550/arXiv.1711.03845

Mejía-de-Dios, J.-A., & Mezura-Montes, E. (2022). Metaheuristics: A Julia package for single-and multi-objective optimization. *Journal of Open Source Software*, *7*(78), 4723. https://doi.org/10.21105/joss.04723

Miranda, L. J. (2018). PySwarms: A research toolkit for particle swarm optimization in Python. *The Journal of Open Source Software*, *3*(21), 433. https://doi.org/10.21105/joss.00433

Močkus, J. (1975). On Bayesian methods for seeking the extremum. In *Optimization techniques IFIP technical conference* (pp. 400–404). Springer; Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-38527-2_55

Mogensen, P., & Riseth, A. (2018). Optim: A mathematical optimization package for Julia. *Journal of Open Source Software*, *3*(24), 615. https://doi.org/10.21105/joss.00615

Nogueira, F. (2014–). *Bayesian Optimization: Open source constrained global optimization tool for Python*. https://github.com/bayesian-optimization/BayesianOptimization

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., & others. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*.

Pinnau, R., Totzeck, C., Tse, O., & Martin, S. (2017). A consensus-based model for global optimization and its mean-field limit. *Mathematical Models and Methods in Applied Sciences*, *27*(01), 183–204. https://doi.org/10.1142/s0218202517400061

Rastrigin, L. (1963). The convergence of the random search method in the extremal control of a many parameter system. *Automaton & Remote Control*, *24*, 1337–1342.

Riedl, K. (2023). Leveraging memory effects and gradient information in consensus-based optimisation: On global convergence in mean-field law. *European Journal of Applied Mathematics*, 1–32. https://doi.org/10.1017/S0956792523000293

Riedl, K., Klock, T., Geldhauser, C., & Fornasier, M. (2023). *Gradient is all you need?* https://doi.org/10.48550/arXiv.2306.09778

Roith, T., Bungert, L., & Wacker, P. (2023). *polarcbo* (Version 1.0.1). https://github.com/

PdIPS/polarcbo

Singh, G. S., & Acerbi, L. (2024). PyBADS: Fast and robust black-box optimization in Python. *Journal of Open Source Software*, *9*(94), 5694. https://doi.org/10.21105/joss.05694

The GPyOpt authors. (2016). *GPyOpt: A Bayesian optimization framework in Python*. http://github.com/SheffieldML/GPyOpt.

Tukh, I., & Riedl, K. (2022). *CBO-in-python* (Version 1.0). https://github.com/Igor-Tukh/cbo-in-python

Yang, X.-S. (2009). Firefly algorithms for multimodal optimization. In *Lecture notes in computer science* (pp. 169–178). Springer; Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-04944-6_14