Anh Tran (SNL); Julien Tranchida (CEA Cadarache)

Multi-fidelity Gaussian process and Bayesian optimization for materials design: Application to ternary random alloys

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. The views expressed in the article do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

Tran, A., Tranchida, J., Wildey, T., & Thompson, A. P. (2020). Multi-fidelity machine-learning with uncertainty quantification and Bayesian optimization for materials design: Application to ternary random alloys. The Journal of Chemical Physics, 153(7), 074705.

The problem we are trying to solve

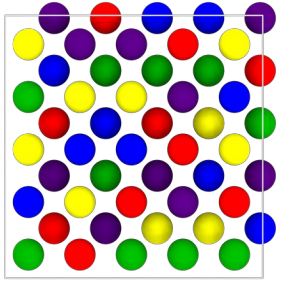
High entropy alloys

Random alloys with four or more principal components.

HEAs are promising materials (**refractory** and aerospace applications, **nuclear degradation-resistant** materials, …).

At Sandia, HEAs are manufactured by **additive manufacturing** techniques.

Prototypical compositions: FeCoCrMnNi, MoNbTaW.



Materials design problem

Materials design can be seen as an inverse problem in the structure-property relationship.

In the context of modern random alloys, optimizing functional performances requires the exploration of vast composition spaces.

The developed ML approaches will be applied to exploring functional properties of different alloy compositions: AlNbTi, MoNbTaW, ...

Comparisons to experimental measurements of additive manufacturing based HEAs performed at Sandia will allow to probe our methods.

The problem we are trying to solve

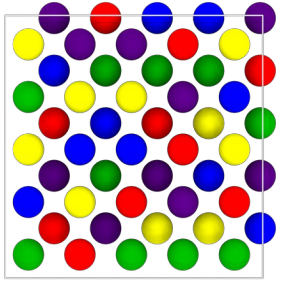
High entropy alloys

Random alloys with four or more principal components.

HEAs are promising materials (**refractory** and aerospace applications, **nuclear degradation-resistant** materials, …).

At Sandia, HEAs are manufactured by **additive manufacturing** techniques.

Prototypical compositions: FeCoCrMnNi, MoNbTaW.



Machine-learning problem

The predicting capabilities of high-accuracy first-principles methods, such as Density Functional Theory (DFT) is limited by their computational cost.

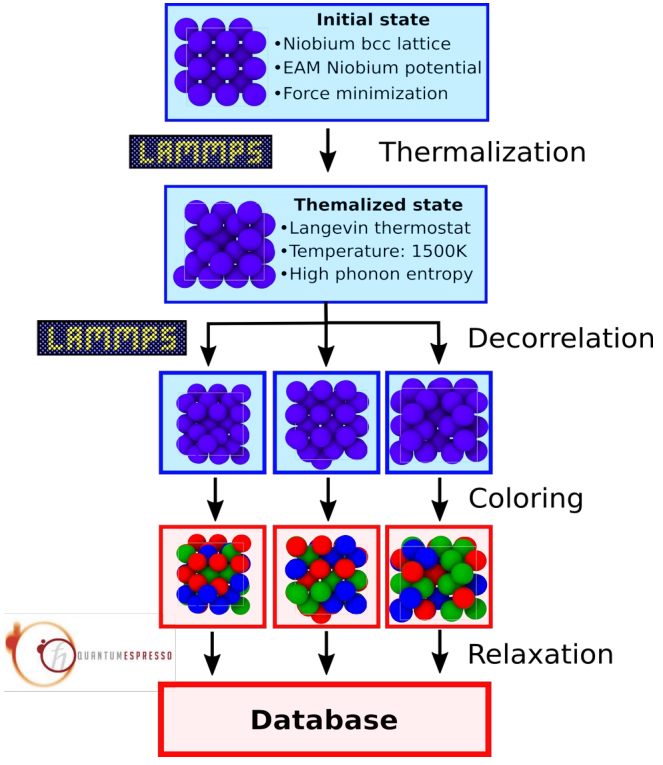
Machine-Learning interatomic potentials (ML-IAP) are emerging as a promising solution to reduce this cost while preserving accuracy.

However, the use of DFT and ML-IAP remains mostly segregated.

We leverage Gaussian process and Bayesian optimization to fuse the predictions of DFT and ML-IAP, and develop a multi-fidelity framework for random alloys. We use it to accurately and efficiently predict materials properties.

SNAP potential for ternary random alloy

A framework was developed to generate a DB for ternary composition.



Generated SNAP training set:

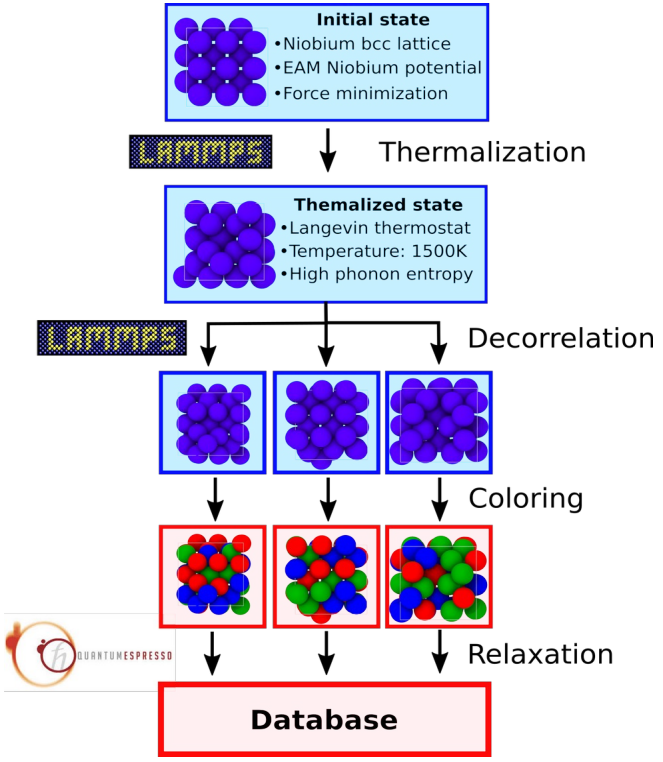
Elements	Type of struct.	No. of structures	No. of atoms
AlNbTi	Compression	2100	113400
	Damping - MD	3000	162000
Al (bcc)	Compression	210	11340
	Damping - MD	500	27000
Nb (bcc)	Compression	210	11340
	Damping - MD	500	27000
Ti (bcc)	Compression	210	11340
	Damping - MD	500	27000
Total:		7230	390420

“Large” but unbalanced dataset: a lot of atomic configurations, but mainly at equicomposition.

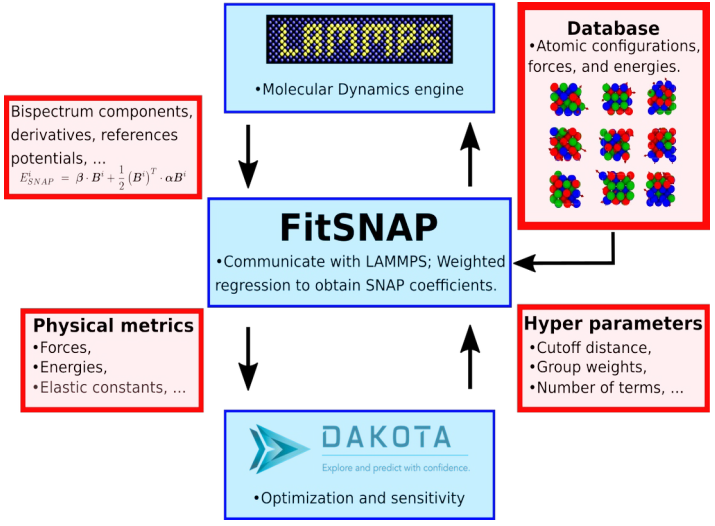
We only work on bcc structures.

SNAP potential for ternary random alloy

A framework was developed to generate a DB for ternary composition.



SNAP - Dakota machinery:



A SNAP IAP for AlNbTi was trained on the generated training set.

Thompson, A. P., *et al*.. (2015). J. Comp. Phys., 285, 316-330.

Focus on one property: the bulk modulus

The data consists of **High-** and **Low-fidelity bulk modulus** calculations accros the AlNbTi composition space.

We leveraged two atomistic approaches:

Ab initio calculations

We used density-functional theory as implemented in Quantum Espresso.

Slow calculations, but accurate for materials design.

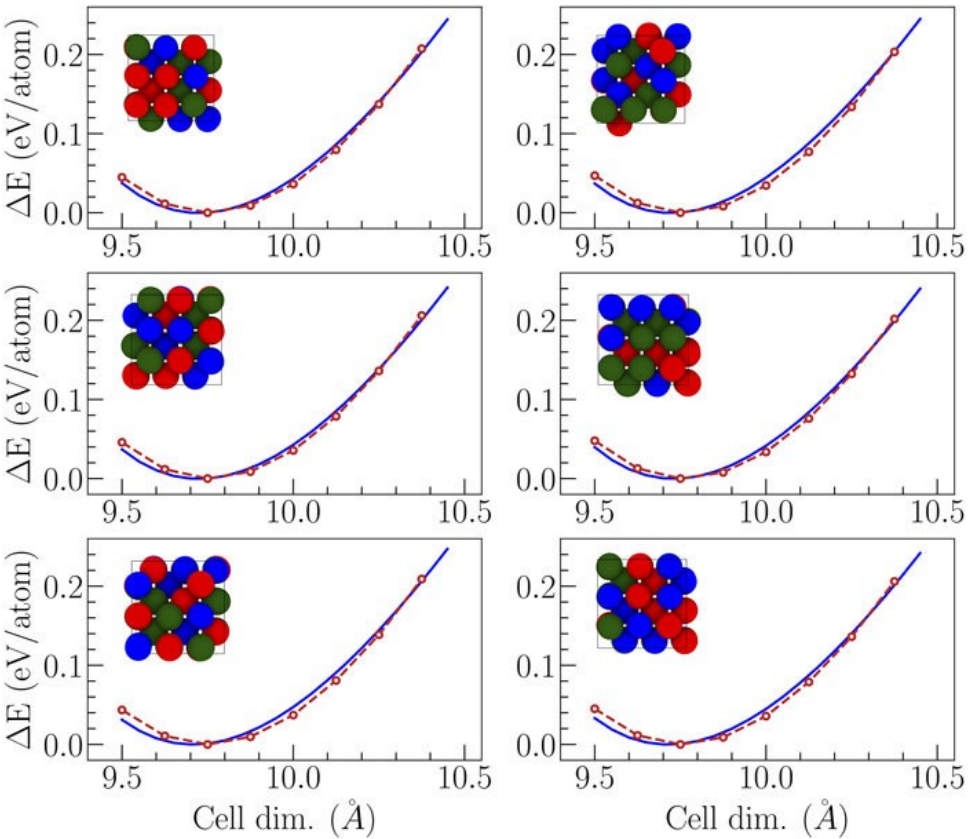
High-Fidelity (HF) Data

Classical potential calculations

We used LAMMPS and a Machine Learning SNAP interatomic potential.

Orders of magnitude faster than HF, but less accurate for materials design.

Low-Fidelity (LF) Data

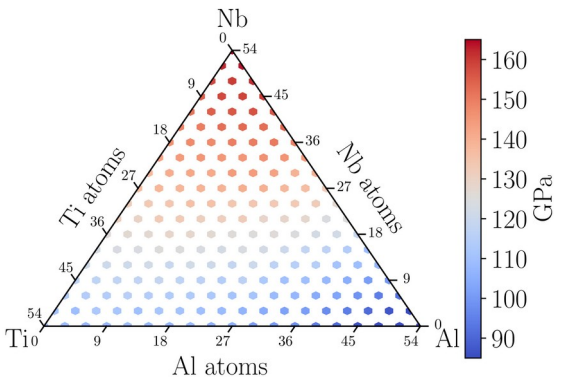


Focus on one property: the bulk modulus

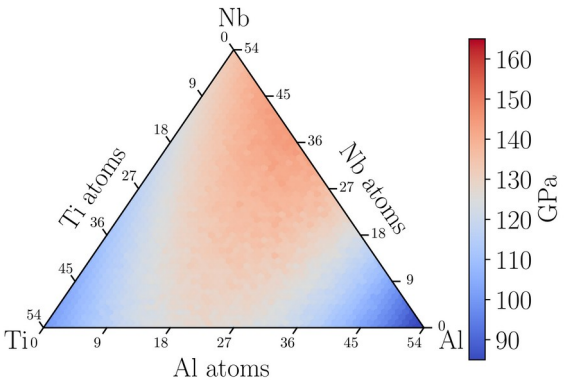
For each composition across the AlNbTi diagram, Equation of State (EOS) calculations are performed. Their fitting (Birch-Murnaghan polynomials) allows to extract the bulk modulus:

$$E(V) = E_0 + \frac{9V_0B_0}{16}\left\{B_0'\left[\left(\frac{V_0}{V}\right)^{\frac{3}{2}} - 1\right]^3 + \left[\left(\frac{V_0}{V}\right)^{\frac{3}{2}} - 1\right]^2\left[6 - 4\left(\frac{V_0}{V}\right)^{\frac{3}{2}}\right]\right\}$$

HF bulk modulus predictions



LF bulk modulus predictions



This HF and LF data is used as a training and testing set for the Gaussian process model.

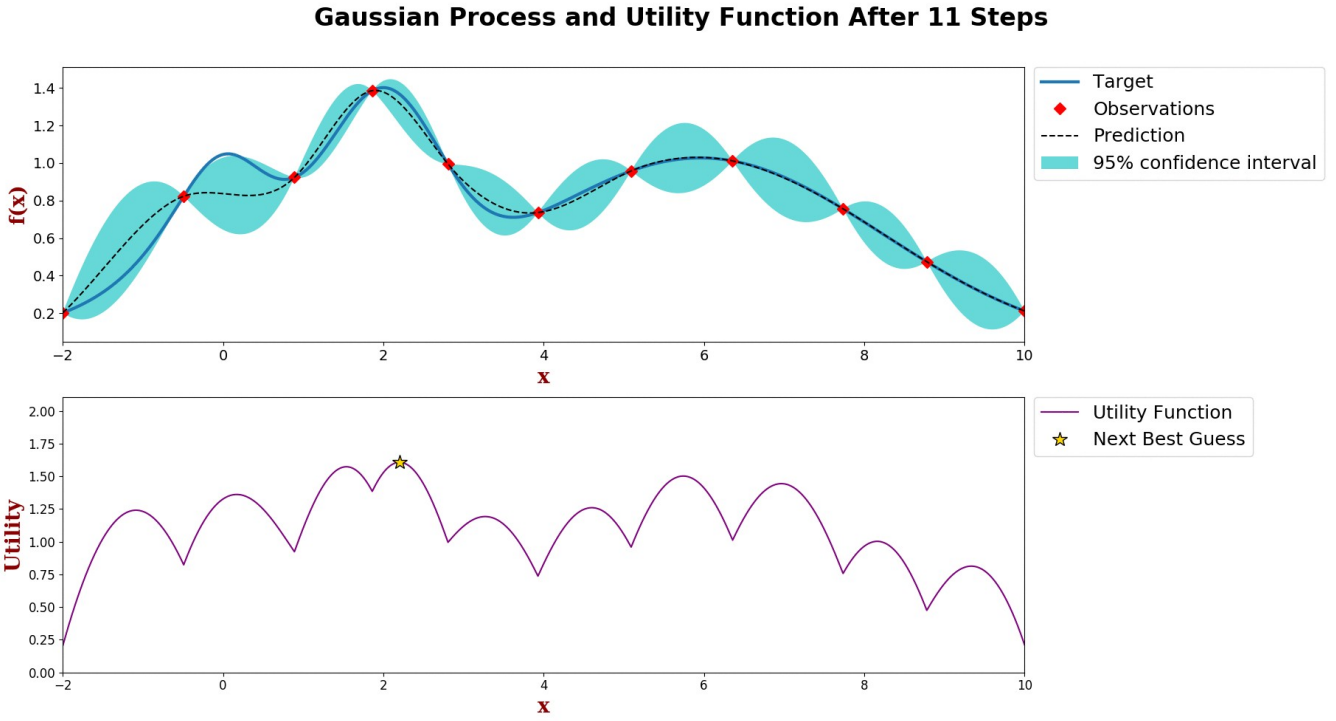
Gaussian process - Introduction

A powerful and flexible framework

And versatile, but in what sense?

- Multi-objective
- Multi-task
- Multi-fidelity
- Mixed-integer
- Scalable for Big Data
- High-dimensional
- Asynchronous parallel
- And more

An animation of Bayesian optimization



Gaussian process - Introduction

Let $\mathcal{D}_n = \{\mathbf{x}_i, y_i\}_{i=1}^n$ denote the set of observations

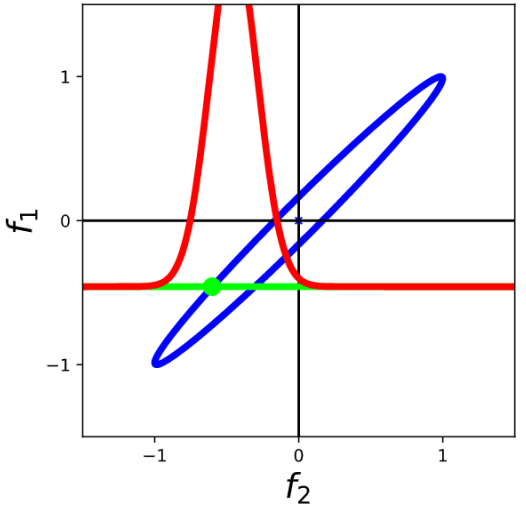
$\mathbf{x}$ denote an arbitrary test points

Prediction is a Gaussian distribution

$$\mu_n(\mathbf{x}) = \mu_0(\mathbf{x}) + \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m})$$

$$\sigma_n^2(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x})^T(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{k}(\mathbf{x})$$

$\mathbf{k}(\mathbf{x})$ is a vector of covariance terms.



A conditional of a Gaussian is Gaussian.
Photo courtesy of Neil Lawrence.
http://inverseprobability.com/talks/notes/gaussian-processes.html

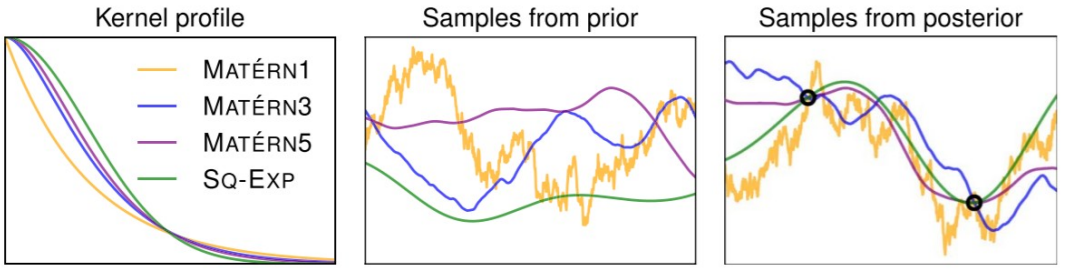
[2] If $P(\mathbf{f},\mathbf{g}) = \mathcal{N}\left(\begin{bmatrix} a \\ b \end{bmatrix}, \begin{bmatrix} A & C \\ C^\top & B \end{bmatrix}\right)$ then $P(\mathbf{f}|\mathbf{g}) = \mathcal{N}(a + CB^{-1}(y - b), A - CB^{-1}C^\top)$ (cf. Appendix A [29]).

Gaussian process - Introduction

Kernels: $\mathbf{K} \in \mathbb{R}^{n \times n}$

Stationary (default)

$$\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i) = \mathbf{K}_{ji}$$



B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, Proc. IEEE 104, 148 (2016).

Anisotropic vs. isotropic

Symmetric positive semi-definite matrix

Many good choices

Affect smoothness of GP

(hint: first derivative at origin)

$$k_{\text{Matérn1}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-r)$$
$$k_{\text{Matérn3}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-\sqrt{3}r)(1 + \sqrt{3}r)$$
$$k_{\text{Matérn5}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-\sqrt{5}r)\left(1 + \sqrt{5}r + \tfrac{5}{3}r^2\right)$$
$$k_{\text{sq-exp}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp\left(-\tfrac{1}{2}r^2\right)$$

Gaussian process - Introduction

How to train a GP

Maximizing the log marginal likelihood function

$$\log p(\mathbf{y}|\mathbf{x}_{1:n}, \theta) = -\frac{n}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{K}^{\theta} + \sigma^2\mathbf{I}| - \frac{1}{2}(\mathbf{y} - \mathbf{m}_{\theta})^T(\mathbf{K}^{\theta} + \sigma^2\mathbf{I})^{-1}(\mathbf{y} - \mathbf{m}_{\theta})$$

Alternatively, can treat hyper-parameters as stochastic parameters and marginalize out using MC

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 25.

Complexity $\mathbf{K}^{-1} \rightarrow \mathcal{O}(n^3)$

More computationally expensive with more data

Approximations are available to reduce complexity at the cost of accuracy, e.g. low-rank approx.

Gaussian process - Introduction

What is the point of optimization?

Everything is about efficiency

Crux: how to select the next input parameters

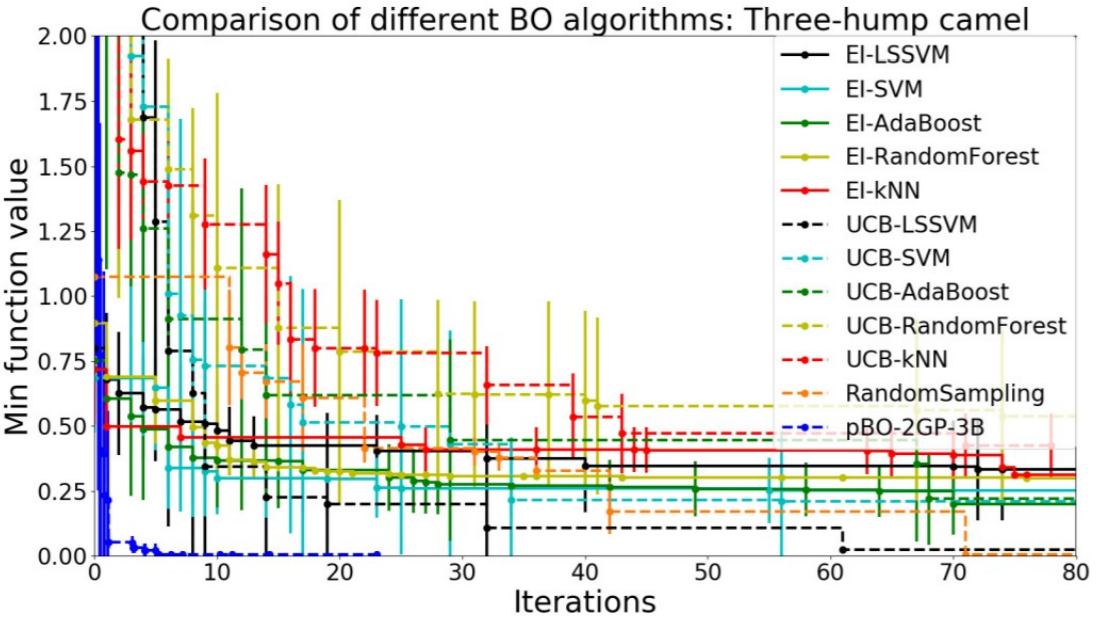
Debate of optimization policies

Bayesian optimization is a gradient-free approach

Next input parameters are found by maximizing acquisition function

Active learning: balancing exploration vs. exploitation

Minimal dataset



Gaussian process - Introduction

Acquisition function

$$\gamma(\mathbf{x}) = \frac{\mu(\mathbf{x}; \{\mathbf{x}_i, y_i\}_{i=1}^n, \theta) - f(\mathbf{x}_{\text{best}})}{\sigma(\mathbf{x}; \{\mathbf{x}_i, y_i\}_{i=1}^n, \theta)}$$

Probability of improvement (PI)

$$a_{\text{PI}}(\mathbf{x}; \{\mathbf{x}_i, y_i\}_{i=1}^n, \theta) = \Phi(\gamma(\mathbf{x}))$$

Mockus, Jonas (1975). “On Bayesian methods for seeking the extremum”. In: Optimization Techniques IFIP Technical Conference. Springer, pp. 400–404.
Mockus, Jonas (1982). “The Bayesian approach to global optimization”. In: System Modeling and Optimization, pp. 473–481

Expected improvement (EI)

$$a_{\text{EI}}(\mathbf{x}; \{\mathbf{x}_i, y_i\}_{i=1}^n, \theta) = \sigma(\mathbf{x}; \{\mathbf{x}_i, y_i\}_{i=1}^n, \theta) \cdot (\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x})) + \phi(\gamma(\mathbf{x}))$$

Upper confidence bound (UCB)

$$a_{\text{UCB}}(\mathbf{x}; \{\mathbf{x}_i, y_i\}_{i=1}^n, \theta) = \mu(\mathbf{x}; \{\mathbf{x}_i, y_i\}_{i=1}^n, \theta) + \kappa\sigma(\mathbf{x}; \{\mathbf{x}_i, y_i\}_{i=1}^n, \theta)$$

Srinivas, Niranjan et al. (2009). “Gaussian process optimization in the bandit setting: No regret and experimental design”. In: arXiv preprint arXiv:0912.3995.
Srinivas, Niranjan et al. (2012). “Information-theoretic regret bounds for Gaussian process optimization in the bandit setting”. In: IEEE Transactions on Information Theory 58.5, pp. 3250–3265.

Gaussian process - Multi-fidelity

Auto-regressive formulation

$$f_H(\boldsymbol{x}) = \rho f_L(\boldsymbol{x}) + \delta(\boldsymbol{x})$$

Covariance matrix

$$\tilde{\boldsymbol{K}} = \begin{pmatrix} \sigma_L^2 \boldsymbol{K}_L(\boldsymbol{x}_L, \boldsymbol{x}_L) & \rho\sigma_L^2 \boldsymbol{K}_L(\boldsymbol{x}_L, \boldsymbol{x}_H) \\ \rho\sigma_L^2 \boldsymbol{K}_L(\boldsymbol{x}_H, \boldsymbol{x}_L) & \rho^2\sigma_L^2 \boldsymbol{K}_L(\boldsymbol{x}_H, \boldsymbol{x}_H) + \sigma_d^2 \boldsymbol{K}_D(\boldsymbol{x}_H, \boldsymbol{x}_H) \end{pmatrix}$$

Posterior mean

$$\mu(\boldsymbol{x}) = \mu_0(\boldsymbol{x}) + \tilde{\boldsymbol{k}}(\boldsymbol{x})^T (\tilde{\boldsymbol{K}} + \sigma^2 \boldsymbol{I})^{-1} (\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{m}})$$

Posterior variance

$$\begin{aligned} \sigma^2(\boldsymbol{x}) = {} & \rho^2\sigma_L^2(\boldsymbol{x}) + \sigma_d^2(\boldsymbol{x}) \\ & - \tilde{\boldsymbol{k}}(\boldsymbol{x})(\tilde{\boldsymbol{K}} + \sigma^2 \boldsymbol{I})^{-1} \tilde{\boldsymbol{k}}(\boldsymbol{x}) \end{aligned}$$

Log marginal likelihood

$$\begin{aligned} & \log p(\tilde{\boldsymbol{y}} | \boldsymbol{x}_{1:n_L}, \boldsymbol{x}_{1:n_H}, \tilde{\theta}) \\ = {} & -\frac{1}{2} (\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{m}})^T (\tilde{\boldsymbol{K}}^{\tilde{\theta}} + \sigma^2 \boldsymbol{I})^{-1} (\tilde{\boldsymbol{y}} - \tilde{\boldsymbol{m}}) \\ & -\frac{1}{2} \log |\tilde{\boldsymbol{K}}^{\tilde{\theta}} + \sigma^2 \boldsymbol{I}| - \frac{n_H + n_L}{2} \log (2\pi) \end{aligned}$$

Fidelity level to query

$$t^* = \underset{t}{\operatorname{argmin}} \left(C_t \int_{\mathcal{X}} \sigma^2(\boldsymbol{x}) d\boldsymbol{x} \right)$$

1. Kennedy, Marc C and Anthony O'Hagan (2000). "Predicting the output from a complex computer code when fast approximations are available". In: Biometrika 87.1, pp. 1–13
2. Yang, X., Zhu, X., & Li, J. (2020). When bifidelity meets cokriging: An efficient physics-informed multifidelity method. SIAM Journal on Scientific Computing, 42(1), A220-A249.
3. Xiao, M., Zhang, G., Breitkopf, P., Villon, P., & Zhang, W. (2018). Extended Co-Kriging interpolation method based on multi-fidelity data. Applied Mathematics and Computation, 323, 120-131.
4. Tran, A., Wildey, T., & McCann, S. (2020). sMF-BO-2CoGP: A sequential multi-fidelity constrained Bayesian optimization framework for design applications. Journal of Computing and Information Science in Engineering, 20(3).

A multi-scale perspective

But try to look at this from a multi-fidelity perspective

Machine learning prediction is not very accurate, but computationally cheap

DFT is accurate, but expensive

Propose a multi-fidelity approach to support ML prediction with DFT when it's necessary.

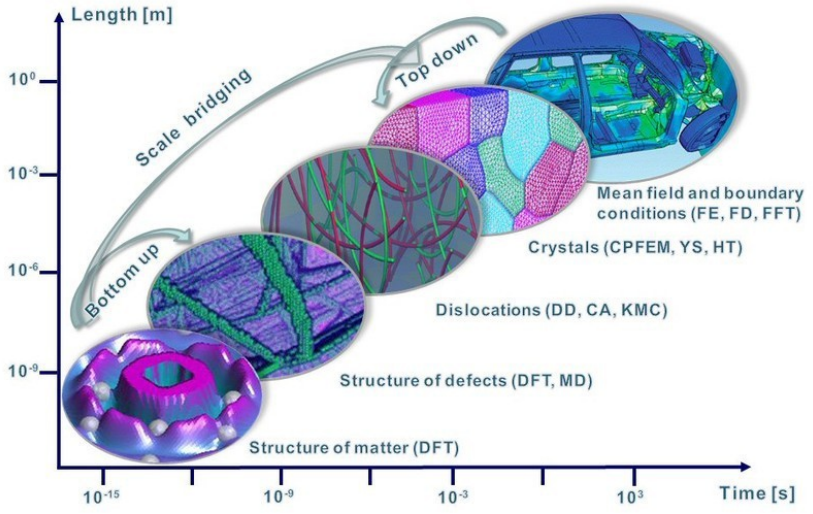
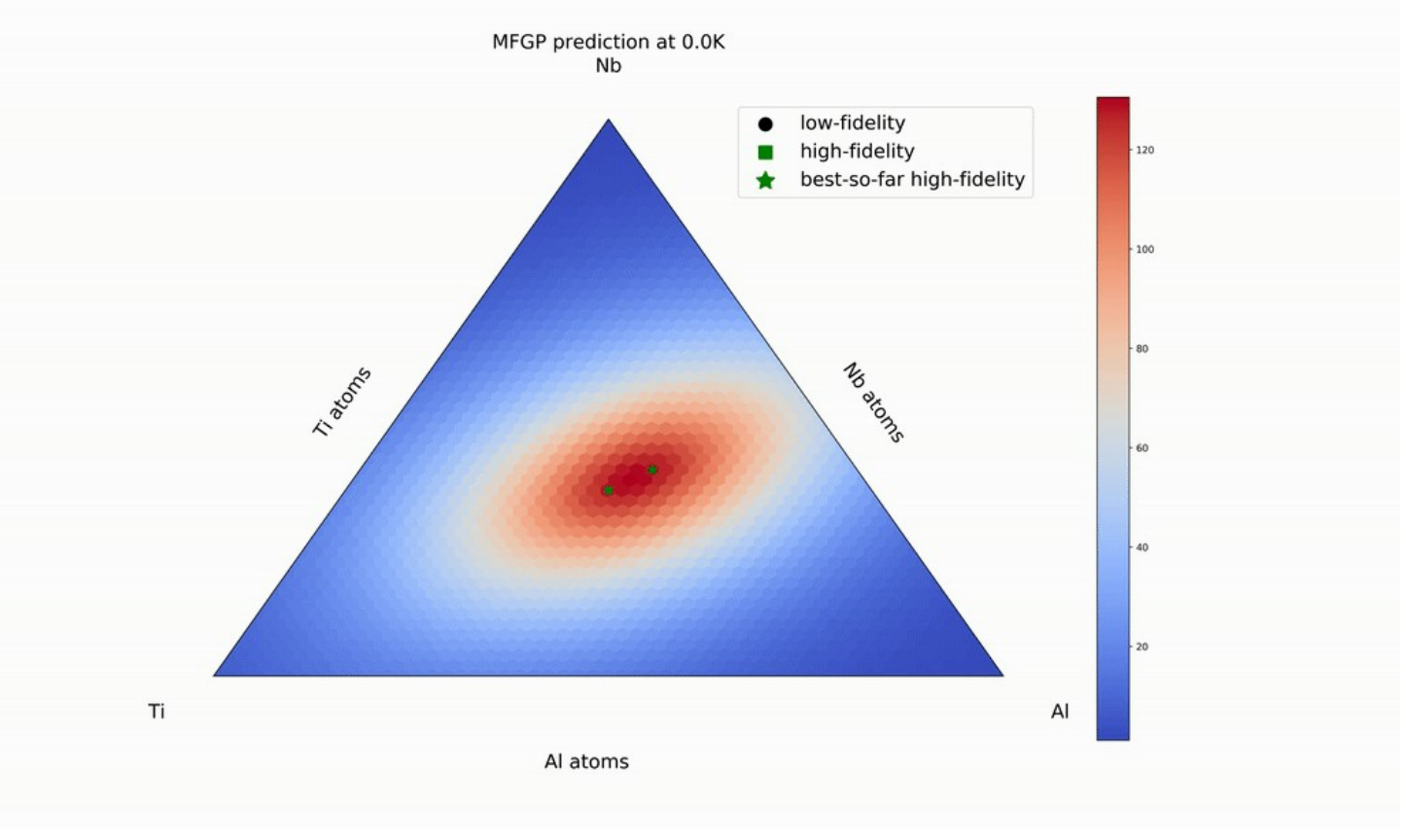


Photo courtesy of Dierk Raabe. http://www.dierk-raabe.com/multiscale-modeling/

Multi-fidelity for multi-scale

Materials design: inverse problem

From chemical composition to materials properties



Bayesian Optimization Parallelism

A MPI perspective: Why parallelize optimization?

arguments:

- focus on multi-core HPC architecture and expensive, high-fidelity simulations
- **Amdahl's law**: diminishing returns, i.e. rewards for parallelizing solvers diminish as # of processors increase
- **motivation**: can we search for the optimal point in **faster** wall-clock time, assuming HPC power is sufficient and/or abundant?
- obviously **beneficial** when computing resource is **sufficient**

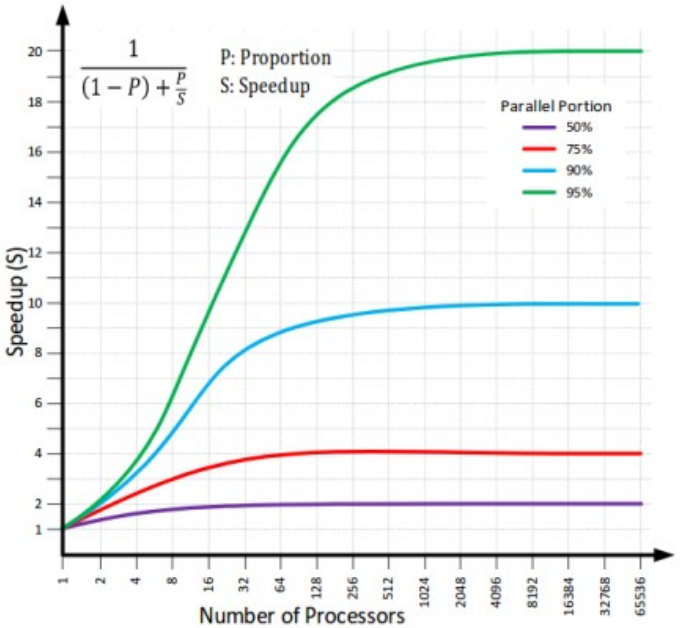


Figure 19: Amdahl's law for parallelization.

Bayesian Optimization Parallelism

A MPI perspective: Why parallelize optimization?

might as well be **beneficial** when computing resource is **insufficient**; examples:

- $P = 0.95 \rightarrow$ SpeedUp $\approx$ 20 times
- MD simulation takes 3 hours to finish with 256 procs $\rightarrow$ **20 cases**/60 hours
- or 60 hours (2.5 days) with 1 proc for 1 case $\rightarrow$ **256 cases**/60 hours
- **fixed** computational budget: 256 $\times$60 CPU hours
- **question:** in the period of **2.5 days**, are we better off with **20 sequential** runs, or with **256 batch-parallel** runs? what about 5 days (**40 vs. 512**)? 10 days (**80 vs. 1024**)? asymptotically?

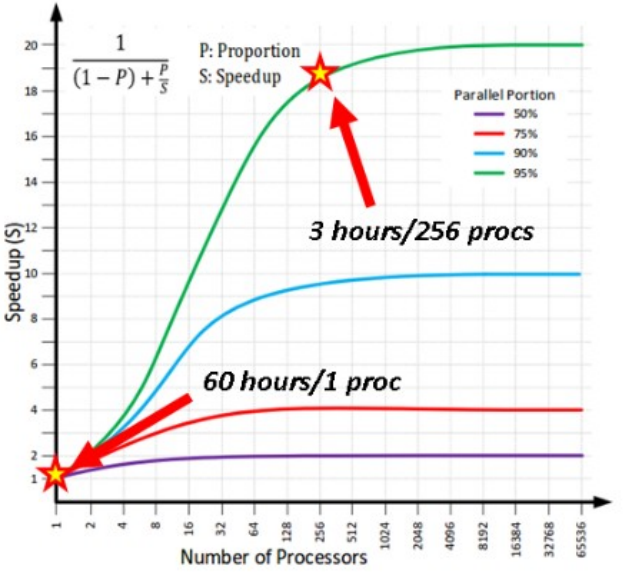
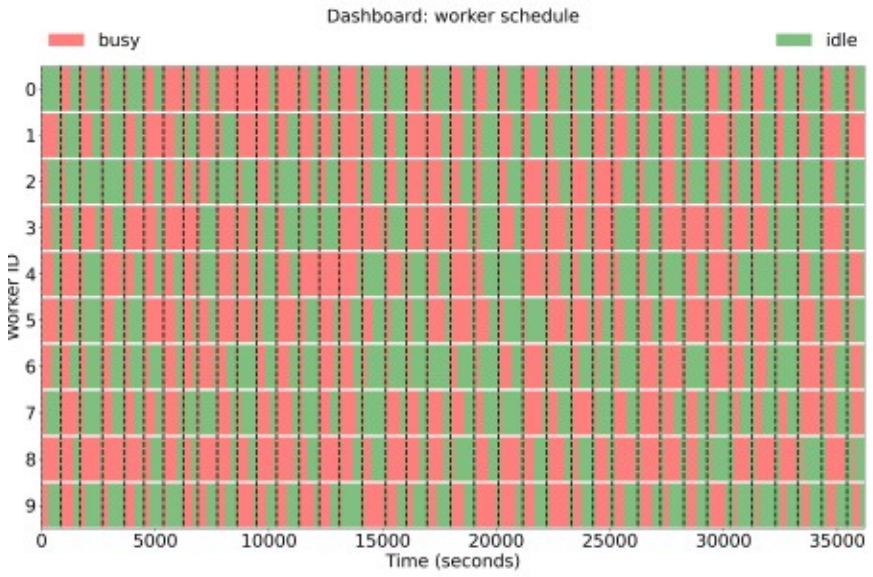


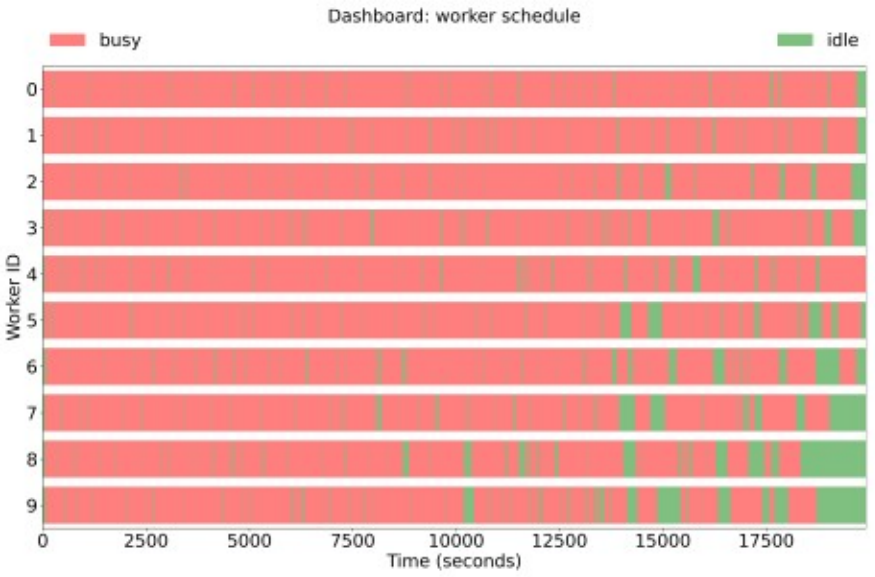
Figure 20: Amdahl's law for parallelization.

Bayesian Optimization Parallelism

A MPI perspective

Why asynchronous parallel?





The code, the data, and the tutorials

https://dakota.sandia.gov/

Source code available: https://dakota.sandia.gov/download.html

A Sandia National Laboratories's flagship software in uncertainty quantification and optimization

How to compile Dakota on Ubuntu 20.04 LTS:

https://dakota.sandia.gov/content/linux-ubuntu-2004

```
apt-get install gcc g++ gfortran cmake libboost-all-dev libblas-dev liblapack-dev libopenmpi-dev openmpi-bin gsl-bin libgsl-dev python perl libhdf5-dev

cmake \
-D CMAKE_C_FLAGS="-O2" \
-D CMAKE_CXX_FLAGS="-O2" \
-D CMAKE_Fortran_FLAGS="-O2“ \
-D DAKOTA_HAVE_GSL:BOOL=TRUE \
-D HAVE_QUESO:BOOL=TRUE \
-D DAKOTA_HAVE_MPI:BOOL=TRUE \
-D DAKOTA_HDF5:BOOL=TRUE \
-D Boost_NO_BOOST_CMAKE:BOOL=TRUE\
 ${DAK_SRC}
```

The code, the data, and the tutorials

https://dakota.sandia.gov/content/manuals

Interface examples: **src/dakota-examples/official/drivers/**

Directory

Two pseudo-simulations

snap_query.py

dft_query.py

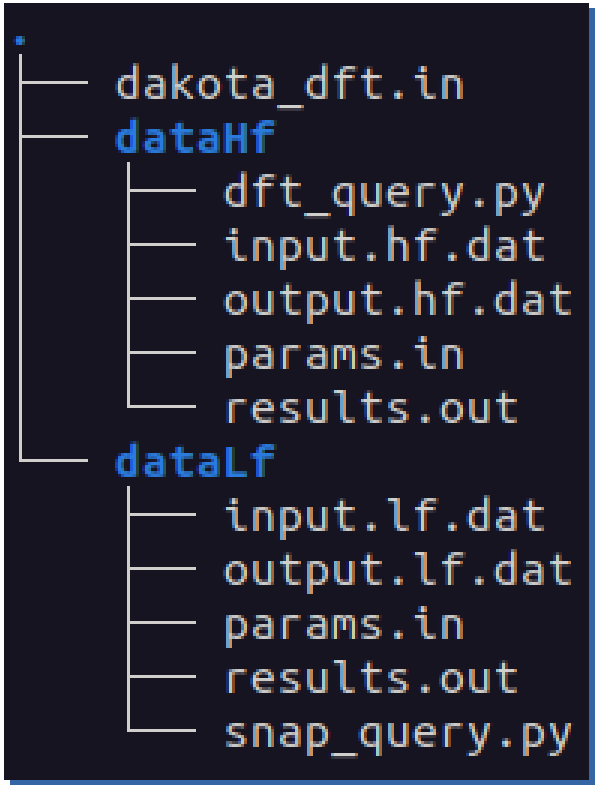
Input: **params.in**

Output: **results.out**

I/O interface (direct interface with Python also available)

Dakota input file: **dakota_dft.in**

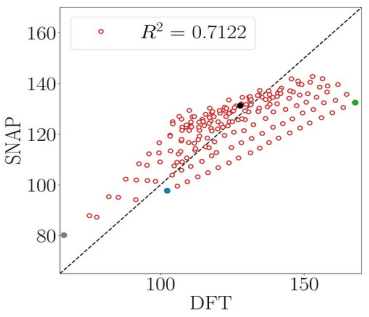
https://github.com/anhvt2/psi-k-tutorials-2021



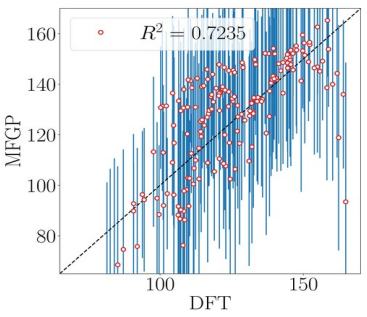
Results: GP-MF model

Convergence of the Multi-Fidelity Gaussian Process (MFGP) model with respect to the HF data, and with an increasing percentage of the training set (bulk modulus values).

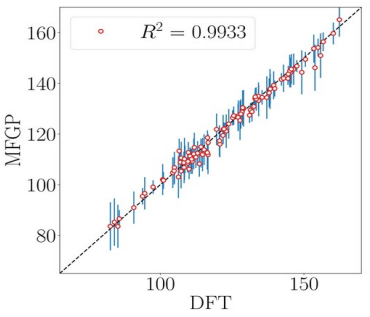
LF vs HF



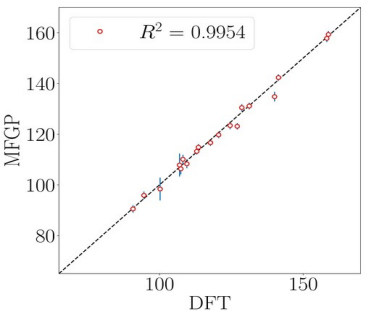
MFGP (10%) vs HF



MFGP (50%) vs HF

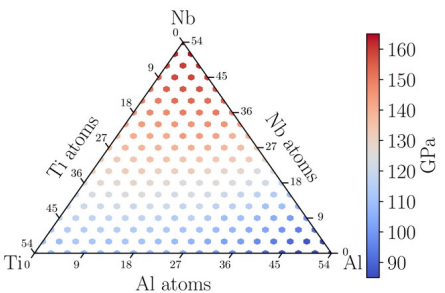


MFGP (90%) vs HF

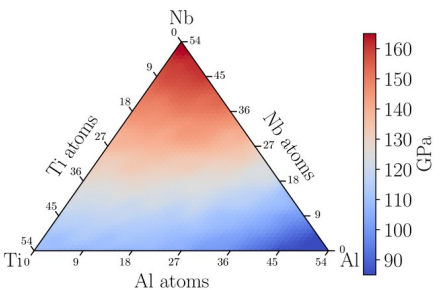


MFGP vs HF bulk modulus diagram:

HF bulk modulus predictions



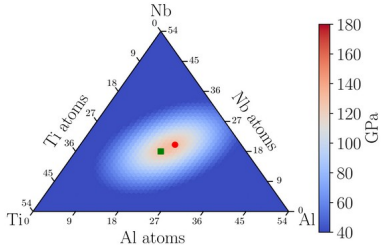
MFGP bulk modulus predictions



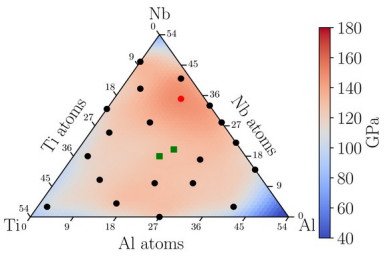
Results: MF Bayesian optimization

We performed a multi-fidelity Bayesian Optimization (MFBO) research of the optimum bulk modulus value across the AlNbTi composition space diagram:

MFBO: Iteration 5

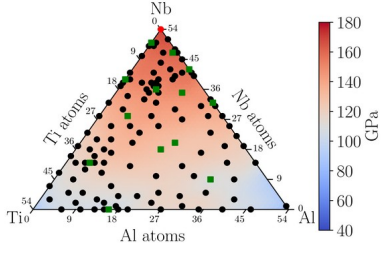


MFBO: Iteration 25

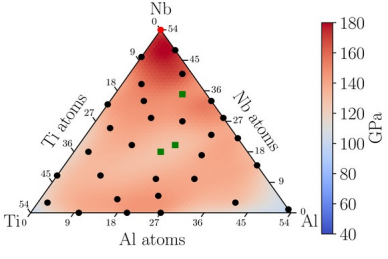


- LF evaluation
- HF evaluation
- Best HF point so far

MFBO: Iteration 131



MFBO: Iteration 36



The optimum bulk modulus value is located at iteration 36, after only 4 expensive HF evaluations.