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

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

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.

Gaussian process - Introduction

- A non-parametric and statistical model based on normal distribution
- A conditional of a Gaussian is Gaussian!
- A framework that has existed for more than a half century
- Why is it so popular?

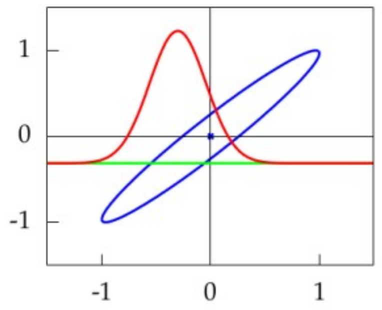


Photo courtesy from Neil Lawrence (2016).

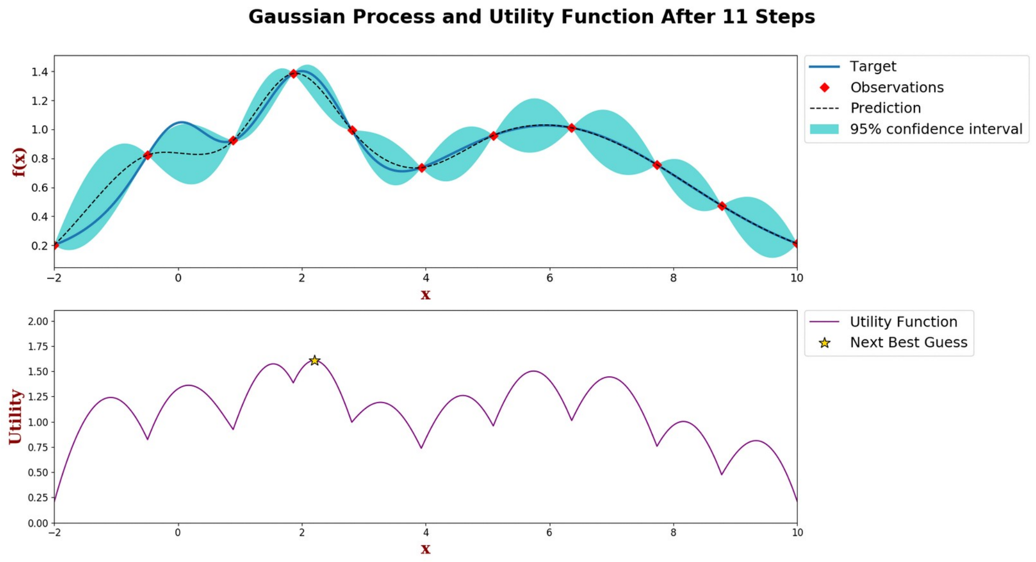
If $P(\mathbf{x},\mathbf{y}) = \mathcal{N}\left(\begin{bmatrix}\mu_{\mathbf{x}} \\ \mu_{\mathbf{y}}\end{bmatrix}, \begin{bmatrix}A & C \\ C^{\top} & B\end{bmatrix}\right)$ then

$$P(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mu_{\mathbf{x}} + CB^{-1}(y-\mu_{\mathbf{y}}), A - CB^{-1}C^{\top})$$

1. Deringer, V. L., Bartók, A. P., Bernstein, N., Wilkins, D. M., Ceriotti, M., & Csányi, G. (2021). Gaussian process regression for materials and molecules. Chemical Reviews, 121(16), 10073-10141.
2. Rasmussen, C.E., 2003. Gaussian processes in machine learning. In Summer school on machine learning (pp. 63-71). Springer, Berlin, Heidelberg.
3. B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, Proc. IEEE 104, 148 (2016).

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
  - Etc.
- 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 and $\mathbf{x}$ 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.

Gaussian process - Introduction

- Kernels:
  - Stationary (default), i.e. only depends on distance
  - Anisotropic vs. isotropic
  - Symmetric positive semi-definite matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$
  - Many good choices
  - Affect smoothness of GP
    (hint: first derivative at origin)

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

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

$$k_{\text{Matérn1}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-r)$$

$$k_{\text{Matérn3}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp(-\sqrt{3}r)(1 + \sqrt{3}r)$$

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

$$k_{\text{sq-exp}}(\mathbf{x}, \mathbf{x}') = \theta_0^2 \exp\left(-\frac{1}{2}r^2\right)$$

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
  - 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.

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

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
  - 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

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

- Log marginal likelihood

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

- Fidelity level to query

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

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 back ML prediction with DFT when it's necessary.

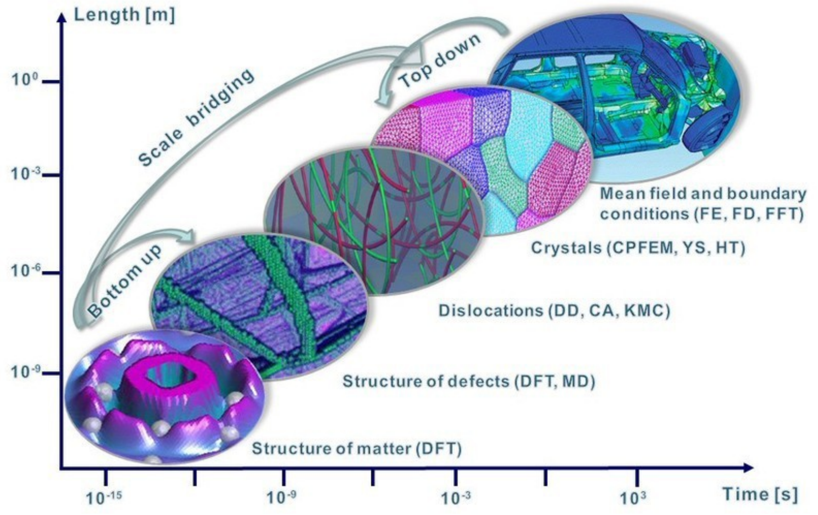
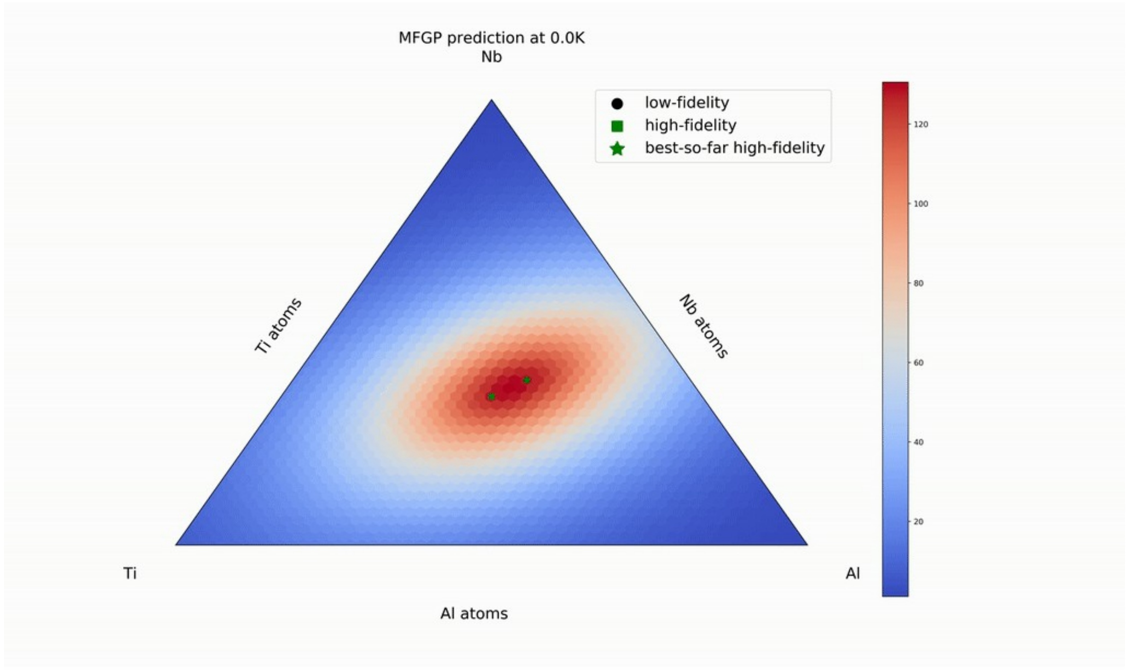

Length [m]
10⁰
10⁻³
10⁻⁶
10⁻⁹
Scale bridging
Top down
Bottom up
Mean field and boundary conditions (FE, FD, FFT)
Crystals (CPFEM, YS, HT)
Dislocations (DD, CA, KMC)
Structure of defects (DFT, MD)
Structure of matter (DFT)
10⁻¹⁵
10⁻⁹
10⁻³
10³
Time [s]

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**

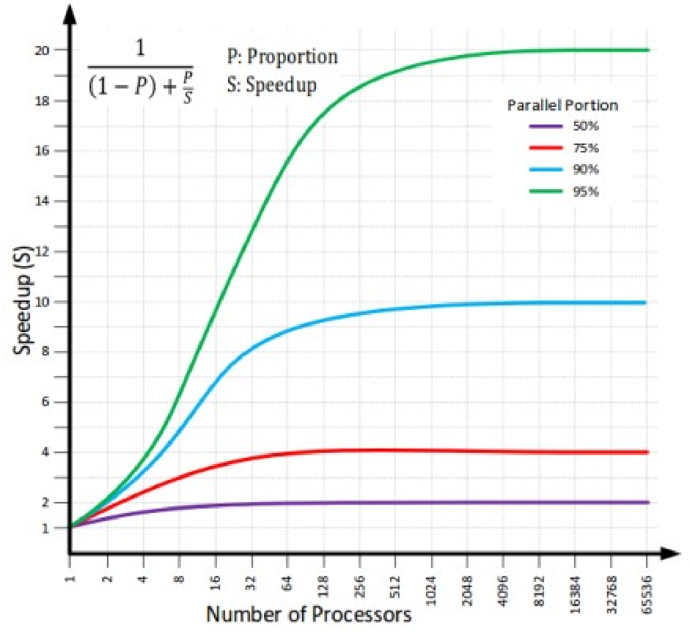


Figure 19: Amdahl's law for parallelization.

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?

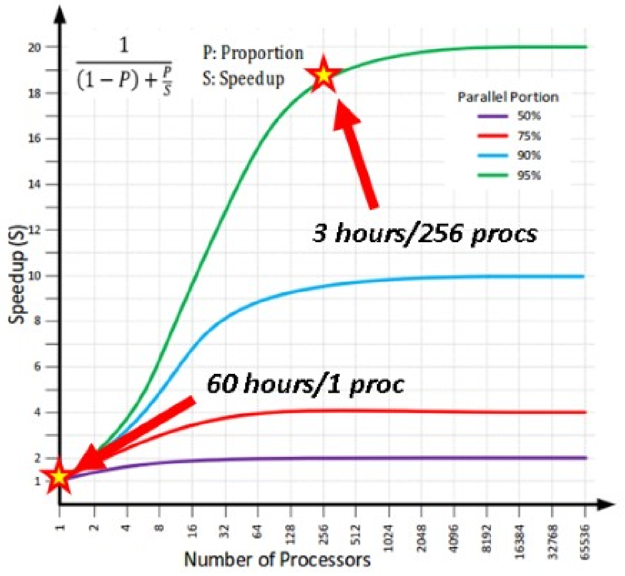
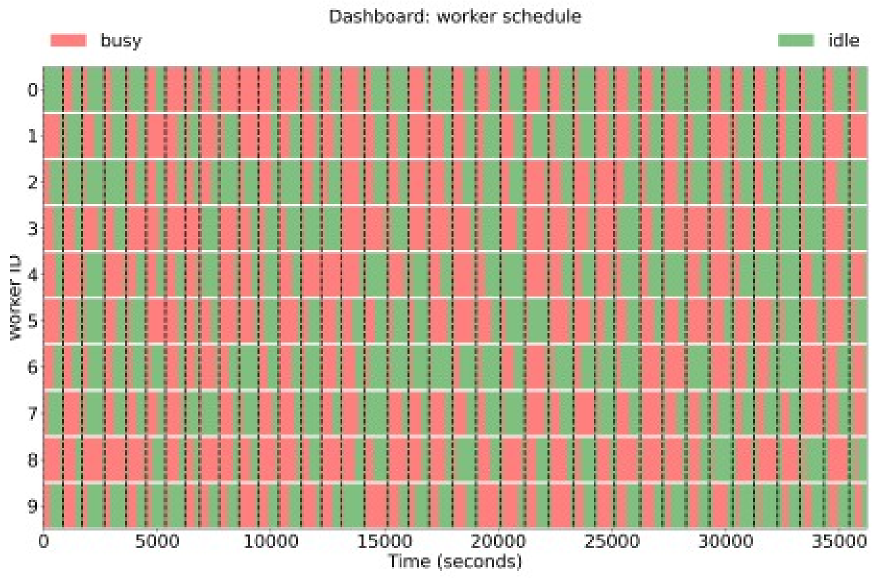
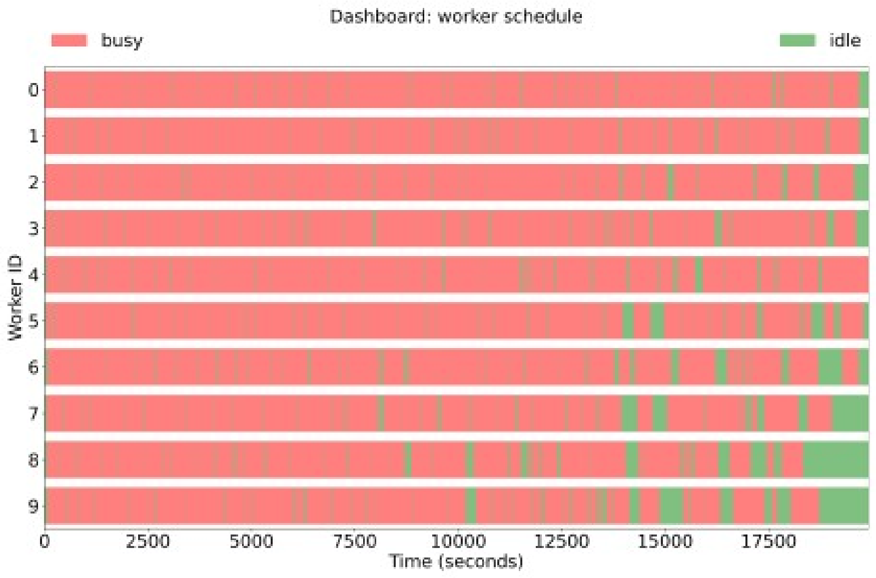


Figure 20: Amdahl's law for parallelization.

Bayesian Optimization Parallelism

- A MPI perspective
  - Why asynchronous parallel?





The code, the data, and the tutorials

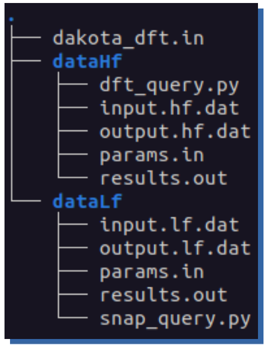
- https://dakota.sandia.gov/
- Source code available: https://dakota.sandia.gov/download.html
- A Sandia National Laboratories’s flagship software in uncertainty quantification and optimization
- How to compile Dakota on Ubuntu 20.04 LTS: https://dakota.sandia.gov/content/linux-ubuntu-2004

```
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
- Dakota input file: dakota_dft.in



```
.
├── dakota_dft.in
├── dataHf
│   ├── dft_query.py
│   ├── input.hf.dat
│   ├── output.hf.dat
│   ├── params.in
│   └── results.out
└── dataLf
    ├── input.lf.dat
    ├── output.lf.dat
    ├── params.in
    ├── results.out
    └── snap_query.py
```