2022/9/9

第一回　ブラックボックス最適化

物質・材料研究機構/東京大学　田村 亮

自己紹介

- 1984年　目黒生まれ，代々木小学校 → 原宿外苑中学校 → 新宿高校

- 大学：埼玉大学理学部物理学科（2003年4月ー2007年3月）統計力学

- 大学院：東京大学理学系研究科物理学専攻 (2007年4月ー2012年3月)
  物性理論

- ポスドク研究員：（独）物質・材料研究機構 若手国際研究センター
  (2012年4月ー2015年３月) 磁気冷凍

- 研究員(2015) → 主任研究員(2018) → 主幹研究員(2022)：
  国立研究開発法人 物質・材料研究機構 (2015年4月ー)

- 講師：東京大学大学院新領域創成科学研究科
  メディカル情報生命専攻(2017年4月ー)
  機械学習
  マテインフォ

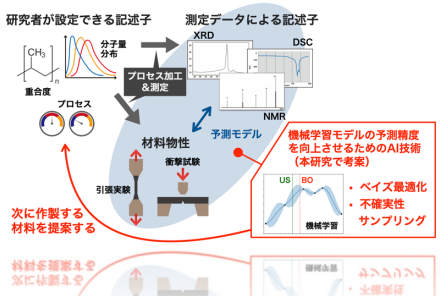
- PM：IPA未踏ターゲット事業(2018年９月ー)　アニーリングマシン

マテリアルズ・インフォマティクス研究

機械学習の予測を利用した物質設計

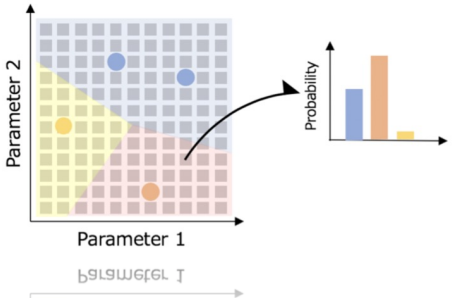
レビュー論文
Acc. Chem. Res. (2021).

材料提案手法



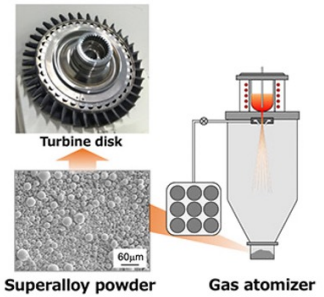
New J. Phys. (2020)
Phys. Rev. Res. (2020)
STAM Methods (2021)

相図構築



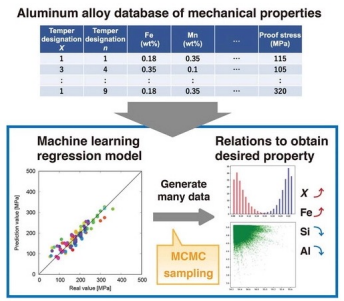
Phys. Rev. Mat. (2019)
ACS Mater. Lett. (2020)
Scr. Mater. (2022)

ガスアトマイズ



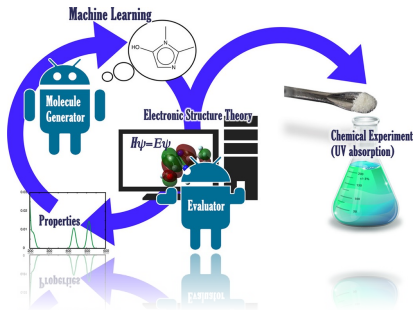
JJAP (2019)
Mater. & Des. (2021)

合金設計



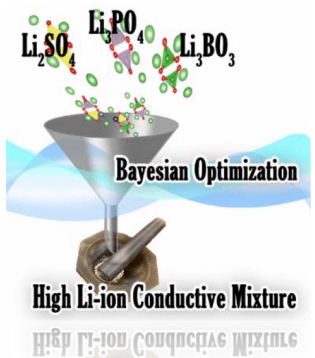
STAM (2020)

有機化合物



ACS Cent. Sci. (2018)
Chem. Sci. (2020)
Sci. Adv. (2022)

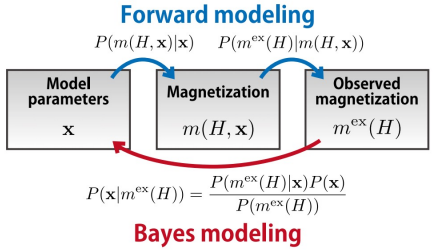
リチウムイオン伝導



BCSJ (2019)
JPCC (2020)

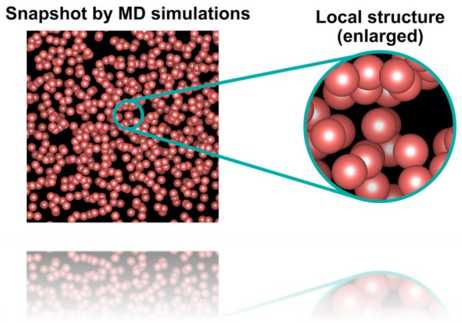
計算技術高度化

ハミルトニアン推定



Phys. Rev. B (2017)
PLoS ONE (2018)
Phys. Rev. B (2020)

MD解析



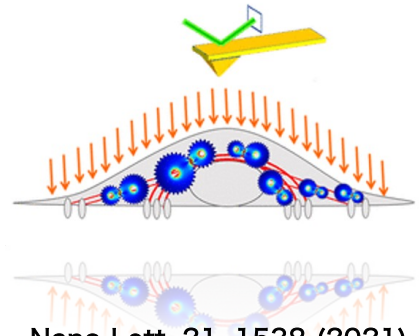
JPSJ. (2019)
Phys. Rev. B (2022)

測定技術高度化

嗅覚センサー

ACS Sensors (2018)
Sci. Rep. (2021)
Sensors (2021).

原子間力顕微鏡

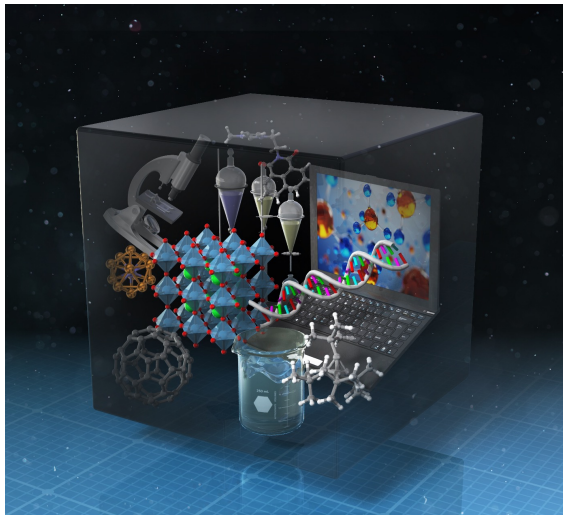


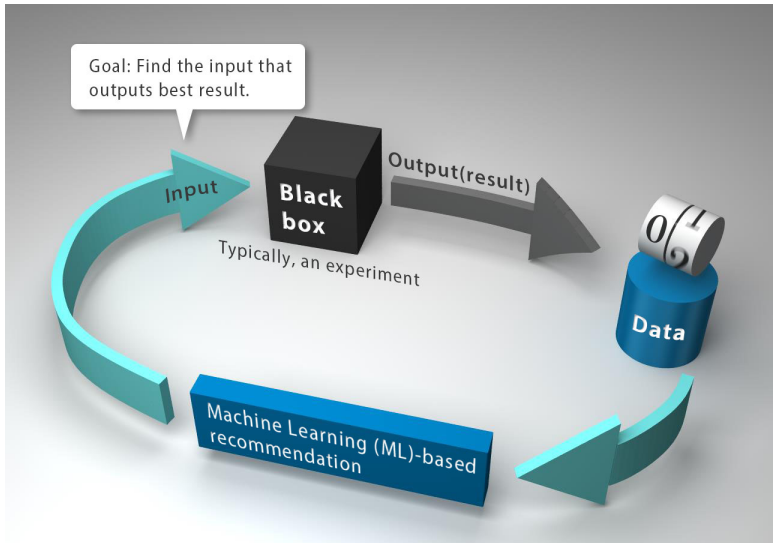
Nano Lett. 21, 1538 (2021)

ブラックボックス最適化
良い物性・特性を持つ材料を見つける

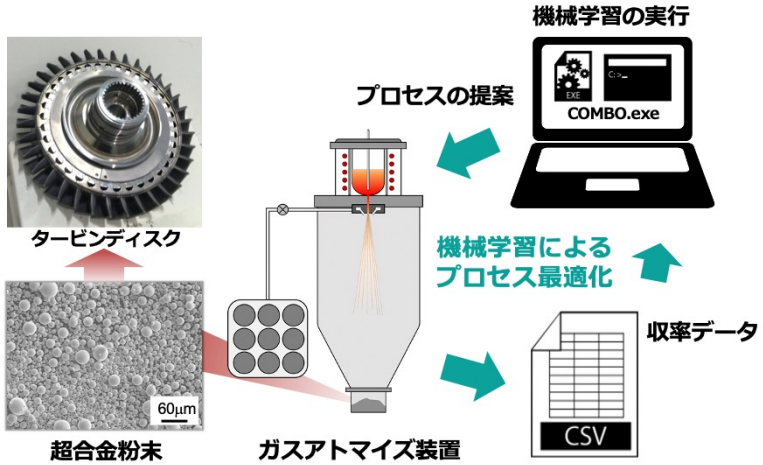
ブラックボックス最適化

機械学習の予測を使って最適化を高速化する！

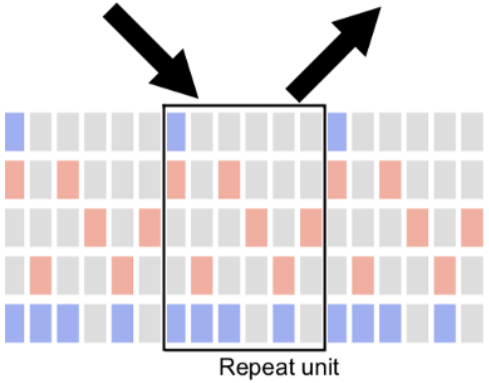




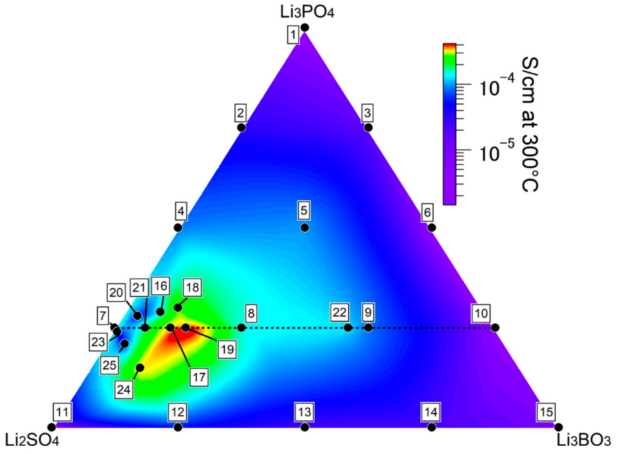
プロセス最適化



構造最適化



材料組成最適化



有名な手法はベイズ最適化

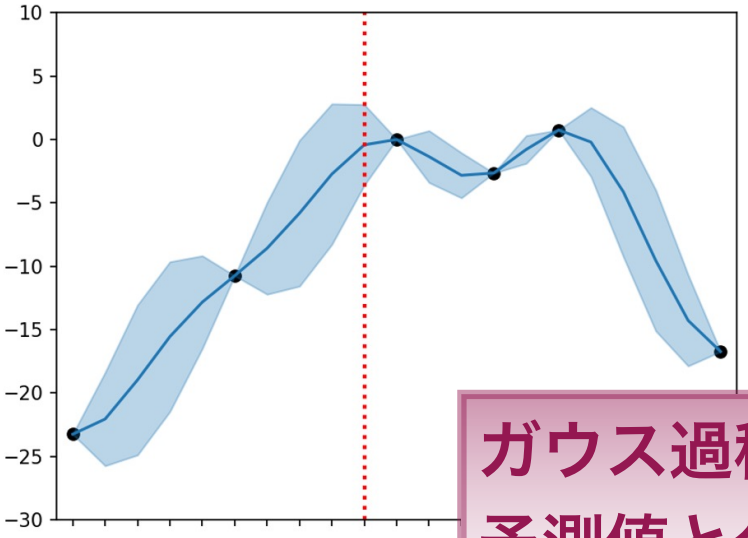
ベイズ最適化とは？

回帰 $y = f(\mathbf{x})$ fを機械学習で置き換える

目的変数
(材料特性)

説明変数
(組成，構造，プロセス)

- N個の候補点があり，この中から最大の観測値を持つものを探したい.
  $\mathbf{x}_i \quad (i = 1, 2, ..., N)$
- できるだけ実験数を少なくしたい.
- M個の候補点に対する実験が終わった.
- 次のM+1個目の候補点を最適に選びたい.
- M個のサンプル点から予測モデルを学習し，それを用いて，
  残りの候補点をスコアリングし実際に観測するサンプルを選ぶ.

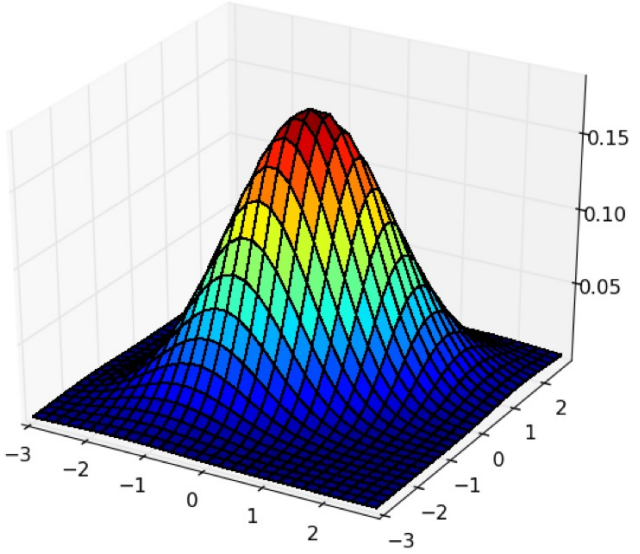


ガウス過程回帰
予測値と分散から，
次の候補を選ぶ

ガウス過程回帰

$y(\mathbf{x})$ の同時分布が多次元ガウス分布に従う

$$P(\mathbf{x}|\boldsymbol{\mu},\Sigma)=\frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\top}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$



線形回帰：$y(\mathbf{x})=\mathbf{w}^{\top}\phi(\mathbf{x})$

係数はガウス分布：$P(\mathbf{w})=\mathcal{N}(\mathbf{w}|\mathbf{0},\alpha^{-1}\mathbf{I})$

$y(\mathbf{x})$ が従うガウス分布

$$\mathrm{E}[\mathbf{y}]=\boldsymbol{\Phi}\mathrm{E}[\mathbf{w}]=\mathbf{0}$$

$$\Sigma=\mathrm{E}[\mathbf{y}\mathbf{y}^{\top}]=\boldsymbol{\Phi}\mathrm{E}[\mathbf{w}\mathbf{w}^{\top}]\boldsymbol{\Phi}^{\top}=\alpha^{-1}\boldsymbol{\Phi}\boldsymbol{\Phi}^{\top}$$

$$\Sigma_{ij}=\alpha^{-1}\phi(\mathbf{x}_i)\phi(\mathbf{x}_j)^{\top}=k(\mathbf{x}_i,\mathbf{x}_j)$$

$$k(\mathbf{x}_i,\mathbf{x}_j)=\exp(-\|\mathbf{x}_i-\mathbf{x}_j\|^2/\eta)$$

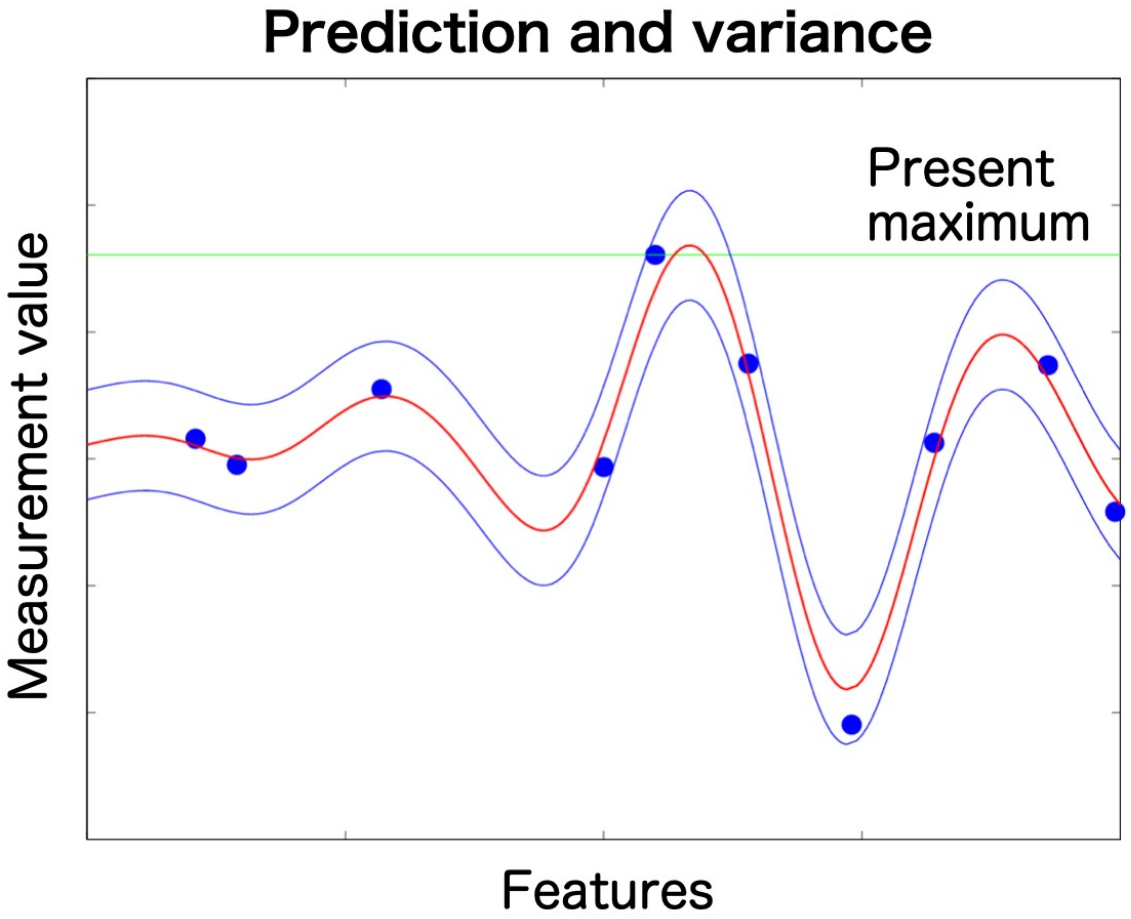
ノイズがある場合の未知の点における推定値

$$\mathrm{E}[y^*]=\mathbf{k}^{*\top}K^{-1}\mathbf{y}\qquad \mathrm{V}[y^*]=k(\mathbf{x}^*,\mathbf{x}^*)-\mathbf{k}^{*\top}K^{-1}\mathbf{k}^*$$

ベイズ最適化の考え方

ノイズがある場合の未知の点における推定値

$$\mathrm{E}[y^*] = \mathbf{k}^{*\top} K^{-1} \mathbf{y} \qquad \mathrm{V}[y^*] = k(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}^{*\top} K^{-1} \mathbf{k}^*$$



探索（exploration）
まだデータがない場所を探索しよう
分散値が重要！

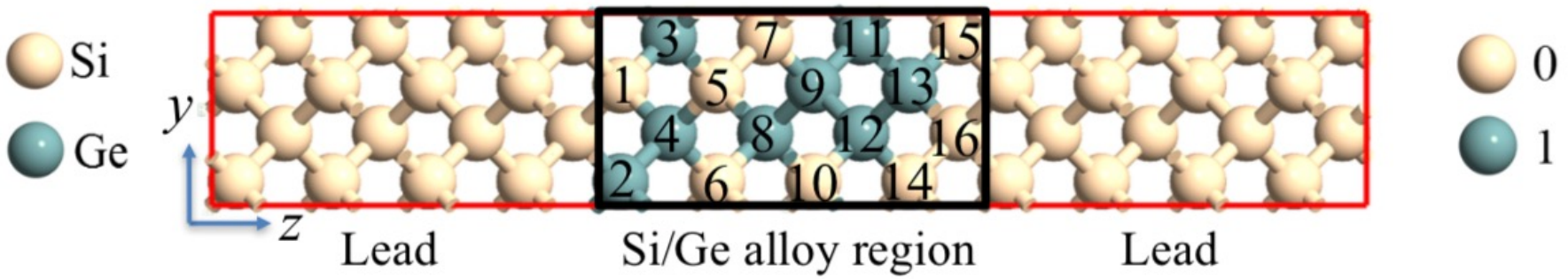
活用（exploitation）
良さそうなところを掘り下げていこう
予測値が重要！

ベイズ最適化でどんなことができるか？

応用事例１：熱伝導率の最適化

構造
最適化

Question: How to organize 16 alloy atoms (Si: 8, Ge: 8) to obtain the largest and smallest interfacial thermal conductance?



Descriptors: $C_{16}^{8} = 12{,}870$

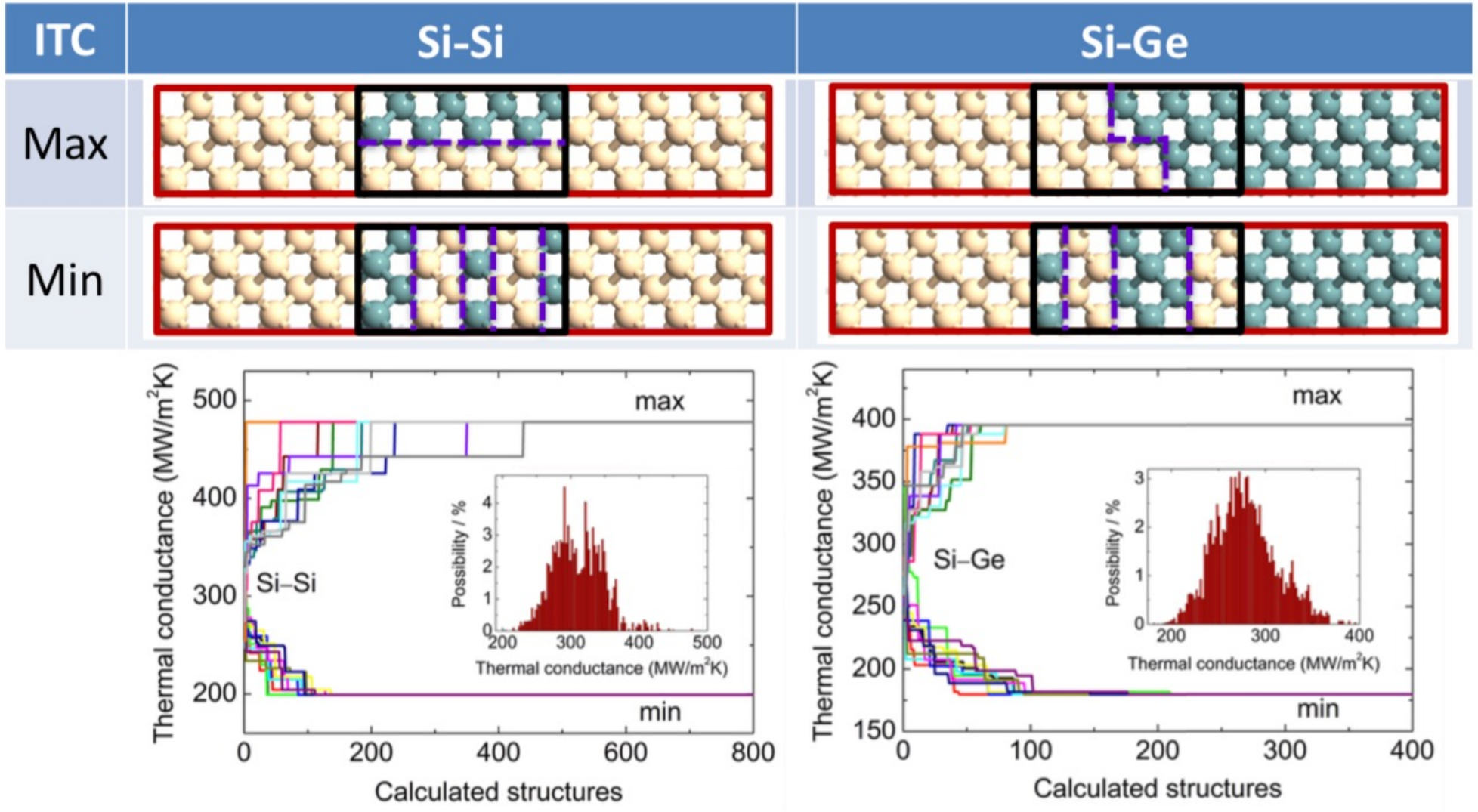
Case	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
2	1	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0
3	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

Calculator: Atomistic Green's Function (AGF): Phonon transmission

Evaluator: Interfacial Thermal Conductance (ITC)

Optimization method: Thompson Sampling (Bayesian Optimization)

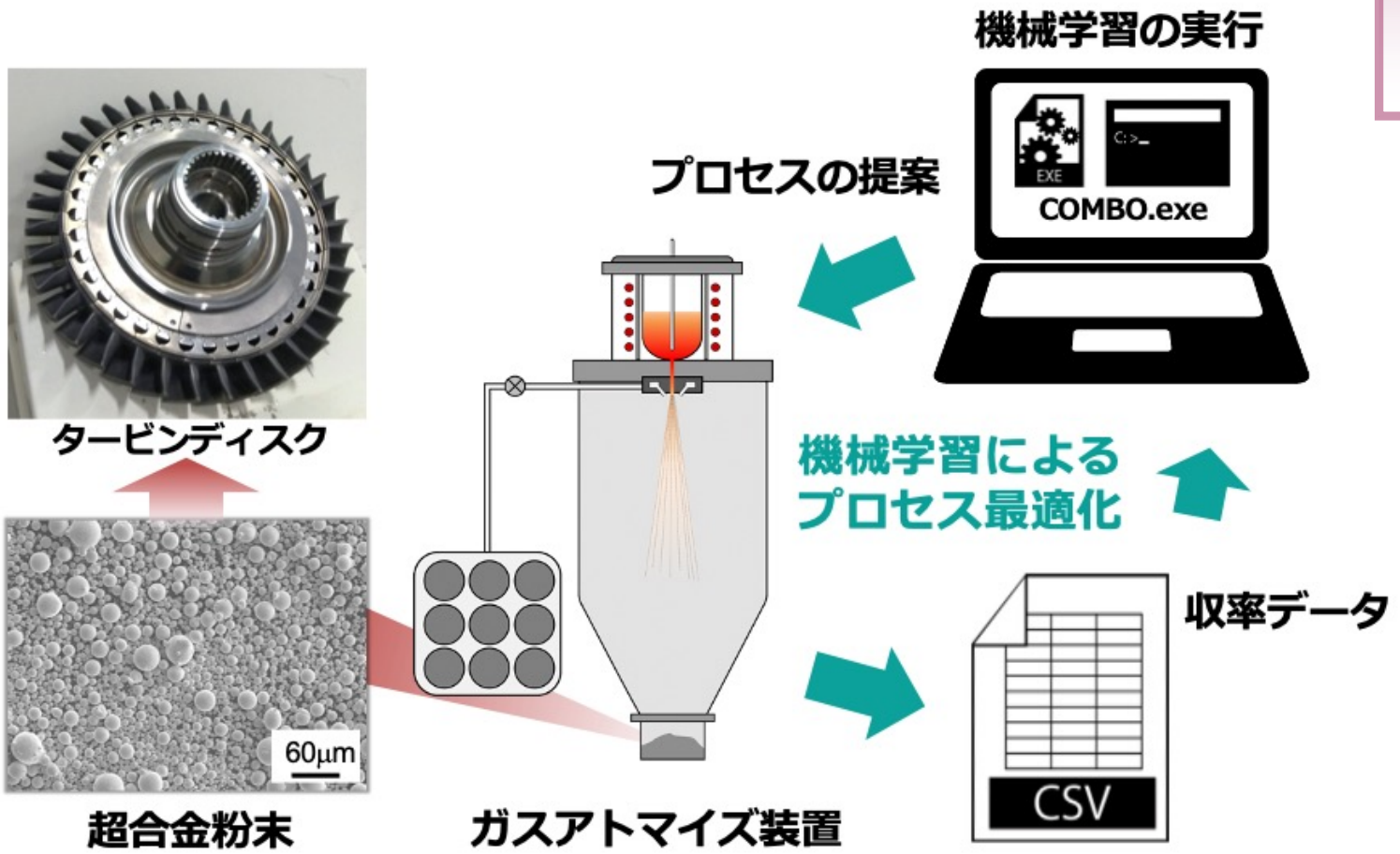
応用事例１：熱伝導率の最適化



S. Ju, K. Tsuda, J. Shiomi, et al, Phys. Rev. X **7**, 021024 (2017).

応用事例２：ガスアトマイズ最適化

プロセス
最適化



R. Tamura, T. Osada, K. Minagawa, T. Kohata, M. Hirosawa, K. Tsuda, and K. Kawagishi, Materials & Design 198, 109290 (2020).

最適化対象超合金

航空機エンジン用材料として有望なNi-Co基超合金

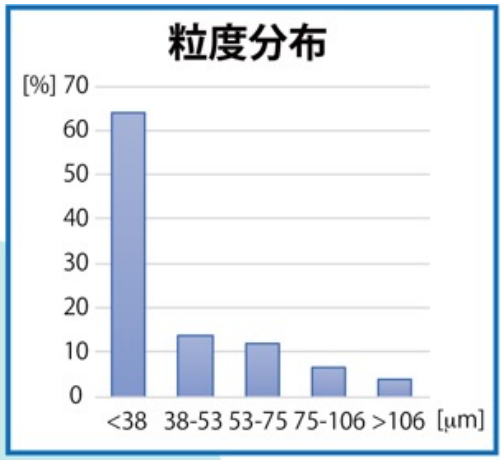
	Ni	Co	Cr	Mo	W	Al	Ti	Nb	Ta	Hf	Zr	C	B	O	γ'-solvus Temp.	Solidus Temp.	Liquidus Temp.
							wt.%							ppm	℃	℃	℃
Nominal	Bal.	27.0	11.7	3.4	1.9	3.2	4.4	0.5	2.2	0.35	0.05	0.03	0.02	-	1178	1232	1339
Actual (< 53 μm)	Bal.	27.8	11.6	3.39	1.95	3.22	4.53	0.49	2.21	0.33	0.046	0.03	0.016	150	1182	1231	1337

最適化手順

現状の収率データセット

溶解温度 [°C]	ガス圧力 [MPa]	53 μm以下収率 [%]
1600	6.0	64.63658
1500	9.0	76.64815
1650	7.0	71.50594
⋮	⋮	⋮

データの追加



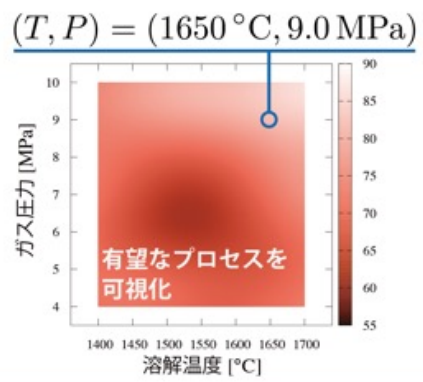
ふるい分級

機械学習予測
モデルの学習

機械学習を用いた粉末製造プロセス最適化

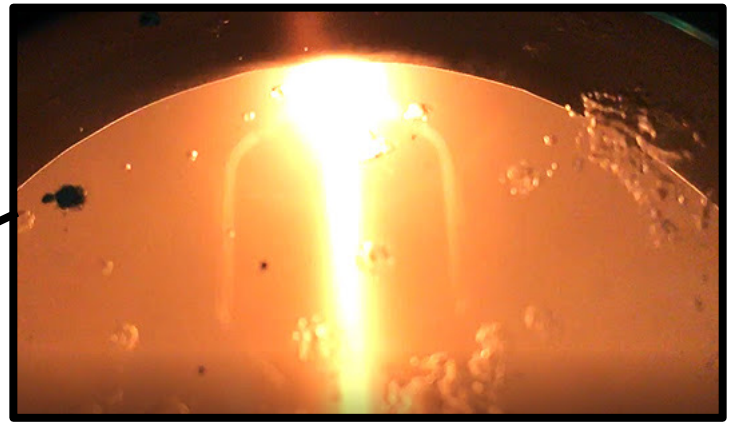
ベイズ最適化

候補プロセスの選定



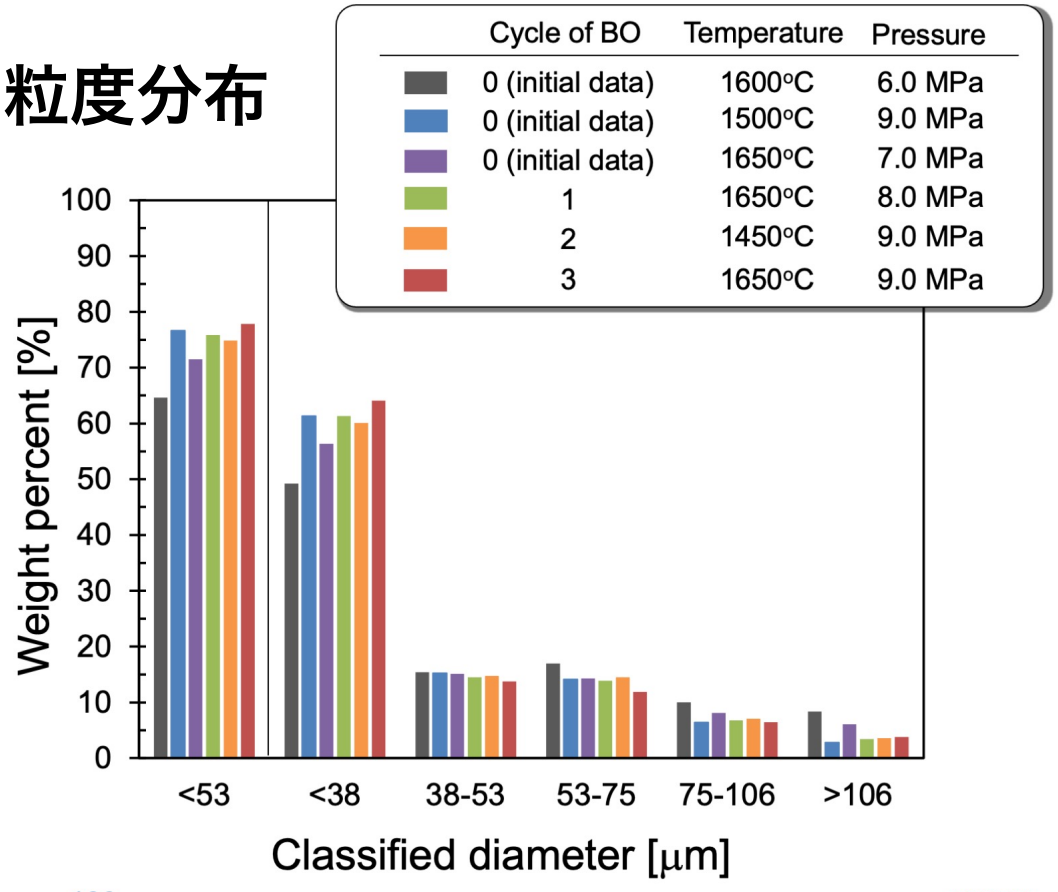
プロセスの提案

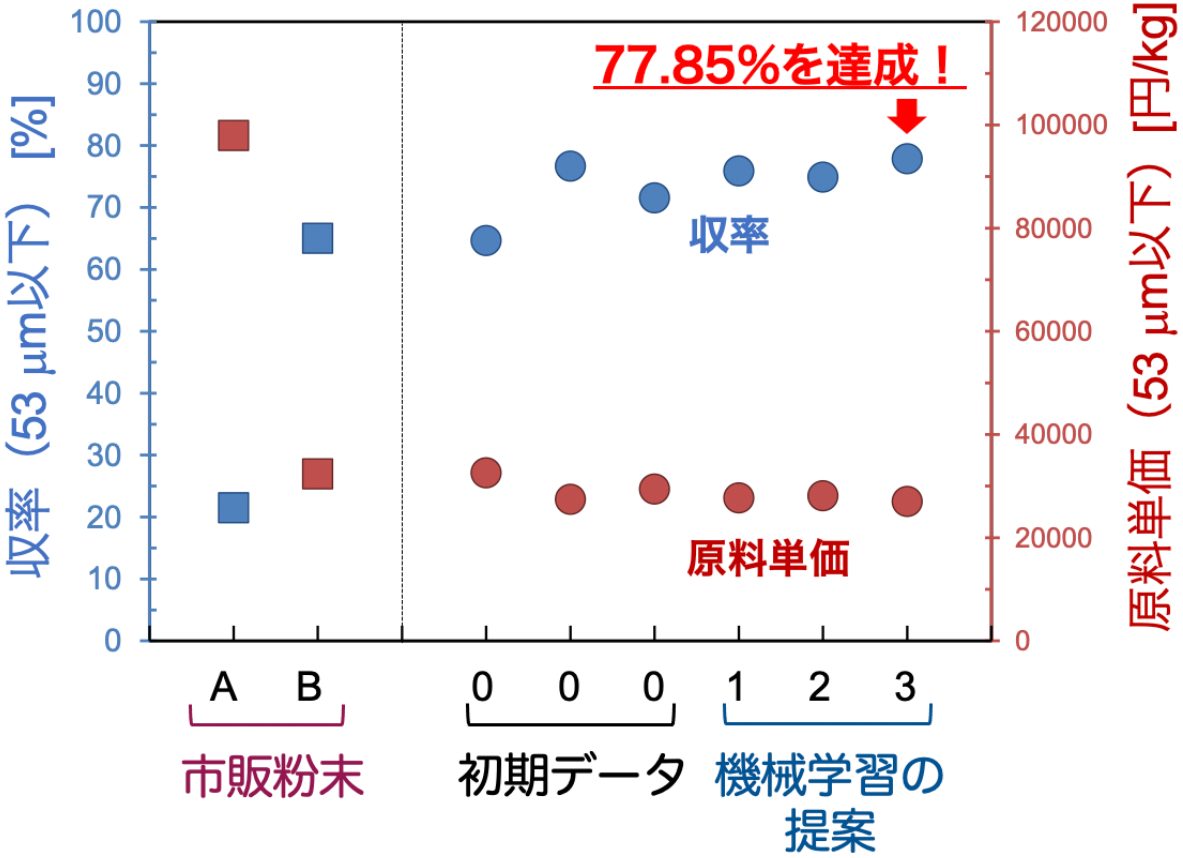
ガスアトマイズ



最適化結果-試行数６回-

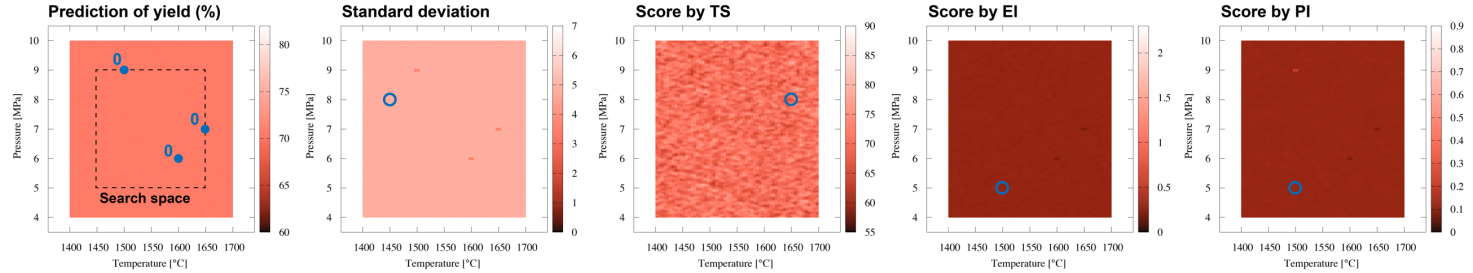
粒度分布



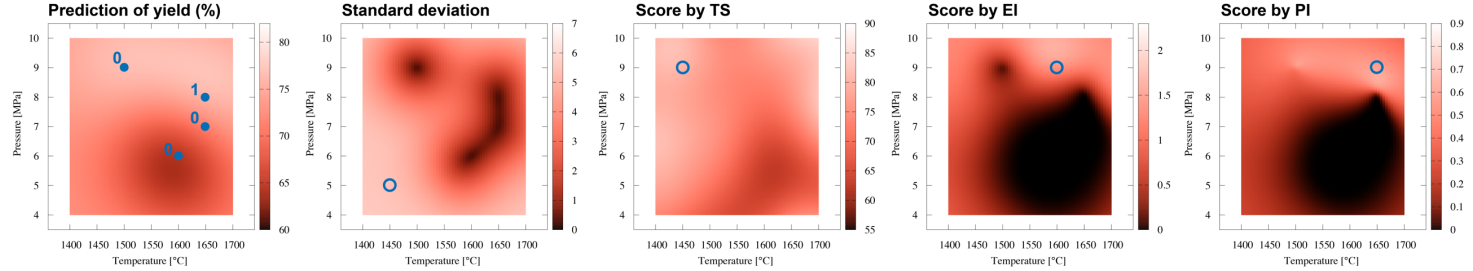


機械学習による予測結果

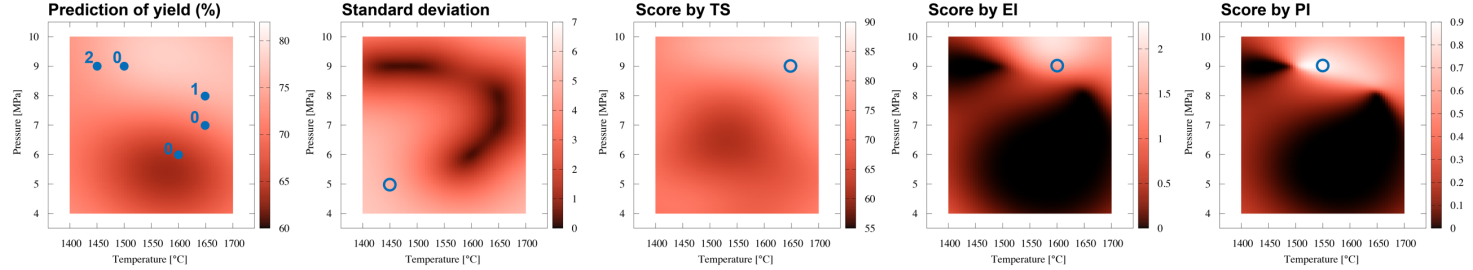
(a) 1st cycle (number of training data is 3)



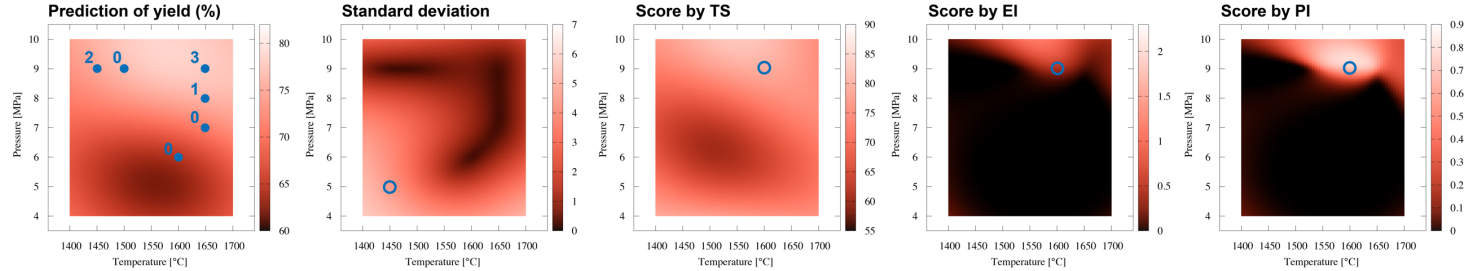
(b) 2nd cycle (number of training data is 4)



(c) 3rd cycle (number of training data is 5)



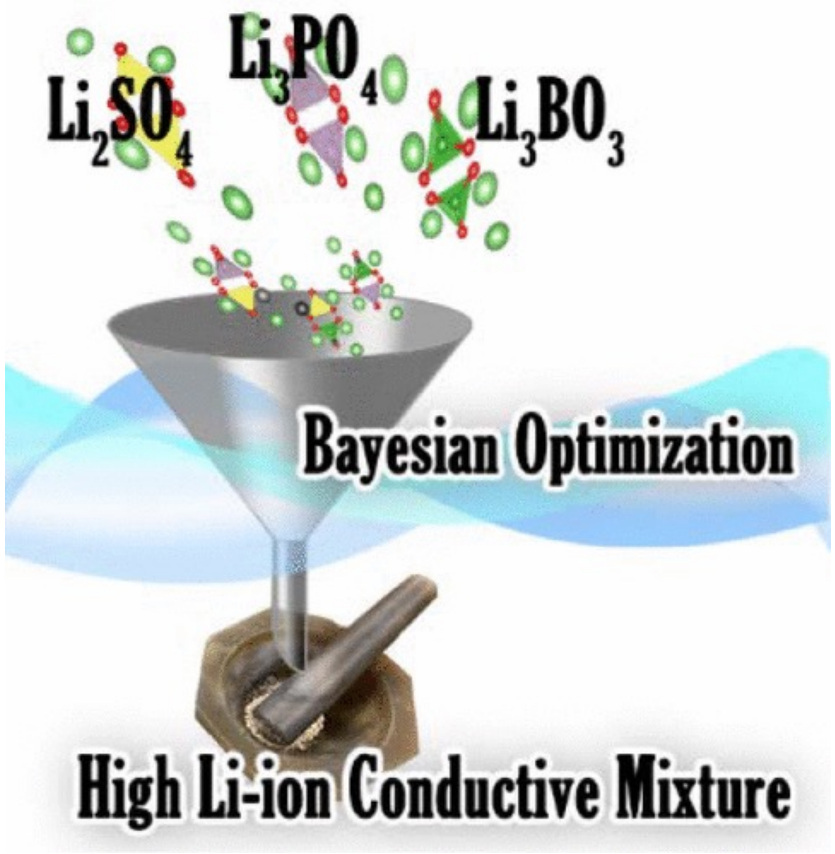
(d) 4th cycle (number of training data is 6)



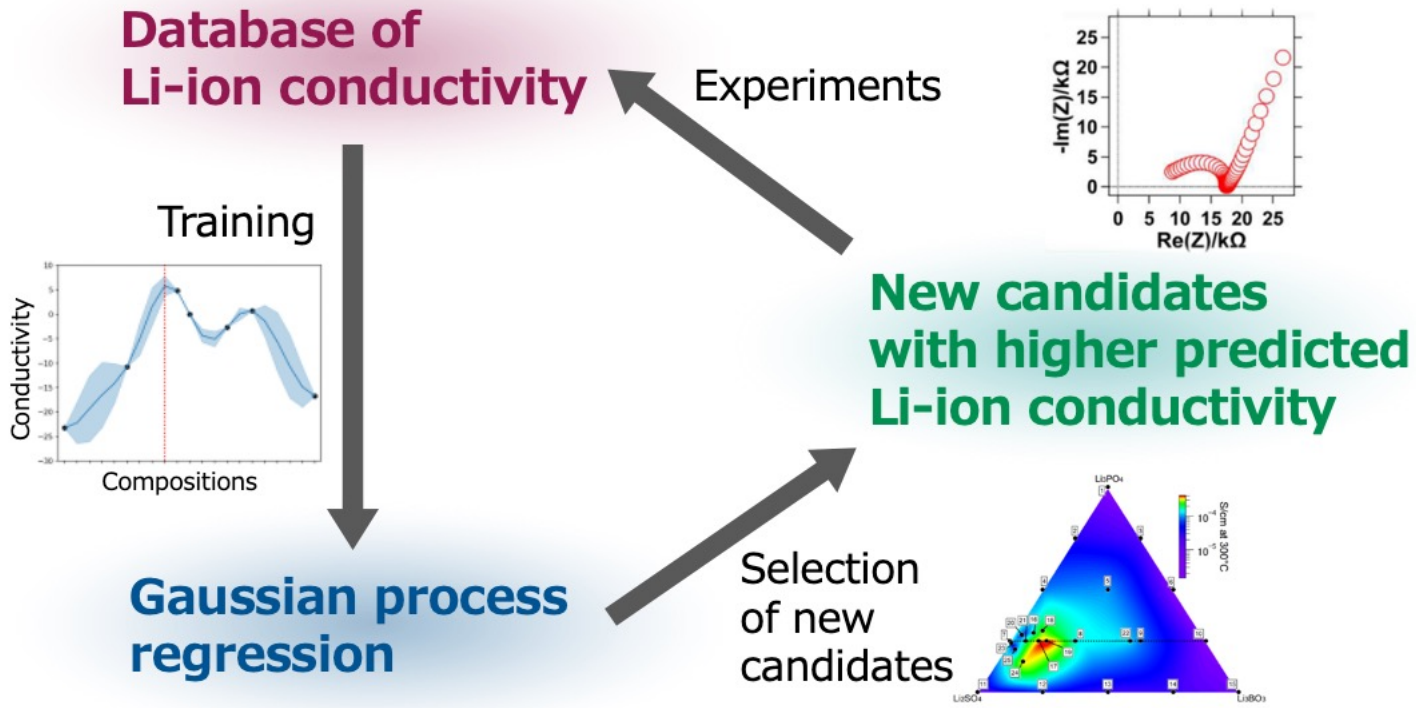
53μm以下の
粉末収率を
向上させる
・溶解温度
・ガス圧力
を探索する.

応用事例３：Liイオン伝導度最適化

材料組成最適化



最適化手順



K. Homma, Y. Liu, M. Sumita, R. Tamura, N. Fushimi, J. Iwata, K, Tsuda, and C. Kaneta,
The Journal of Physical Chemistry C 124, 12865 (2020).

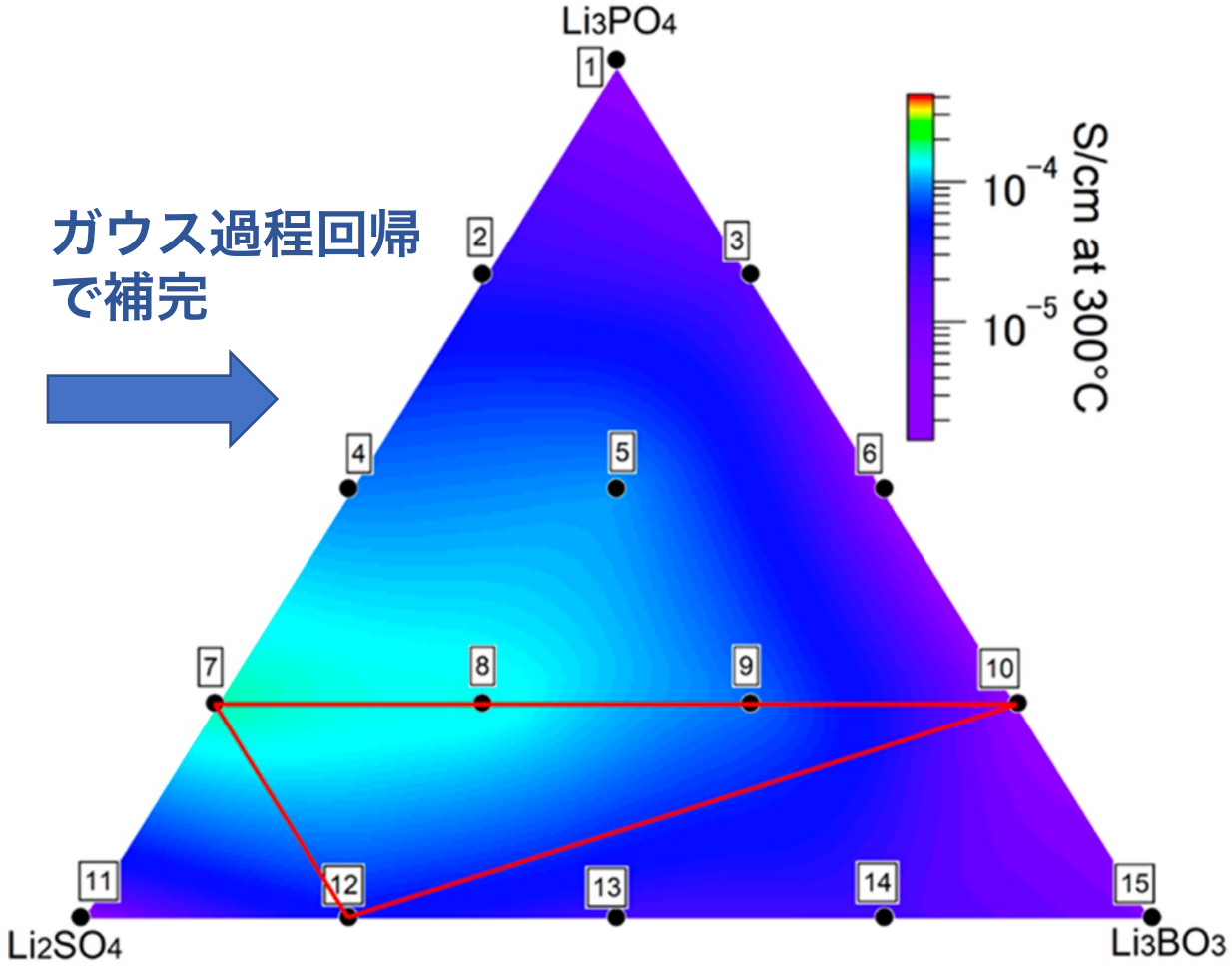
初期データの準備

Li_2SO_4, Li_3PO_4, Li_3BO_3の混合

15点の初期データ

sample no.	ratio (Li_3PO_4, Li_3BO_3, Li_2SO_4)	ionic conductivity (S/cm)
1	(100, 0, 0)	3.9×10^{-9}
2	(75, 0, 25)	3.3×10^{-5}
3	(75, 25, 0)	1.7×10^{-5}
4	(50, 0, 50)	9.9×10^{-5}
5	(50, 25, 25)	9.7×10^{-5}
6	(50, 50, 0)	5.6×10^{-7}
7	(25, 0, 75)	1.6×10^{-4}
8	(25, 25, 50)	1.7×10^{-4}
9	(25, 50, 25)	8.3×10^{-5}
10	(25, 75, 0)	1.2×10^{-6}
11	(0, 0, 100)	1.4×10^{-7}
12	(0, 25, 75)	4.9×10^{-5}
13	(0, 50, 50)	2.3×10^{-5}
14	(0, 75, 25)	2.3×10^{-5}
15	(0, 100, 0)	9.1×10^{-6}

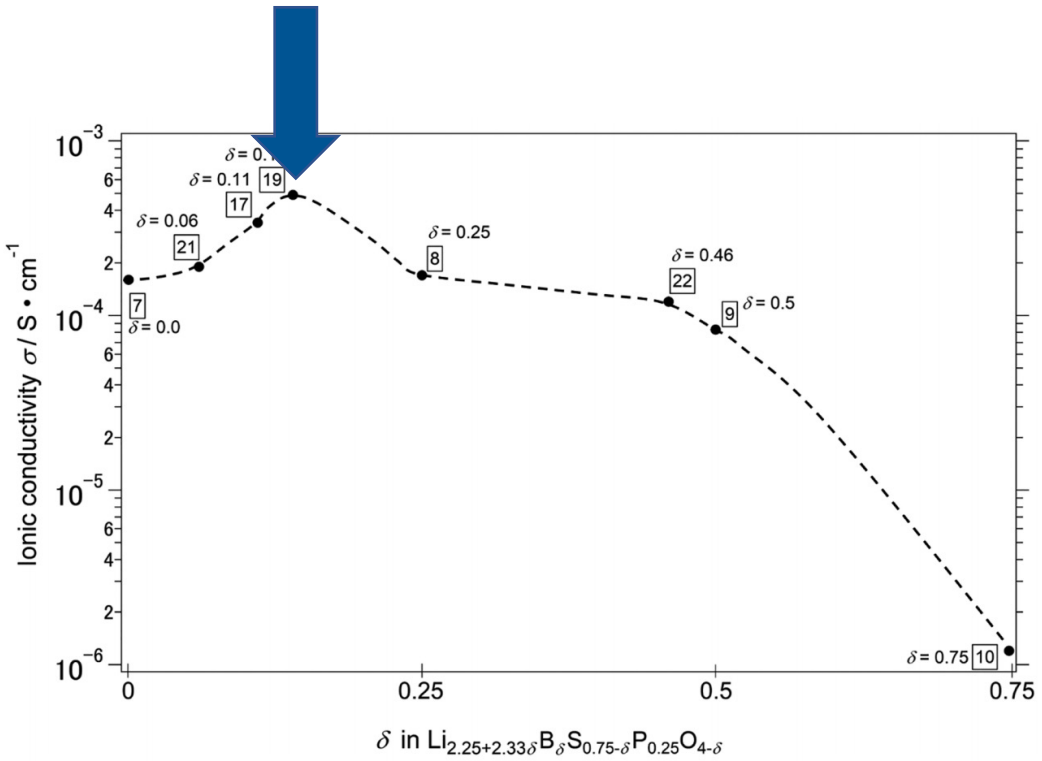
ガウス過程回帰で補完



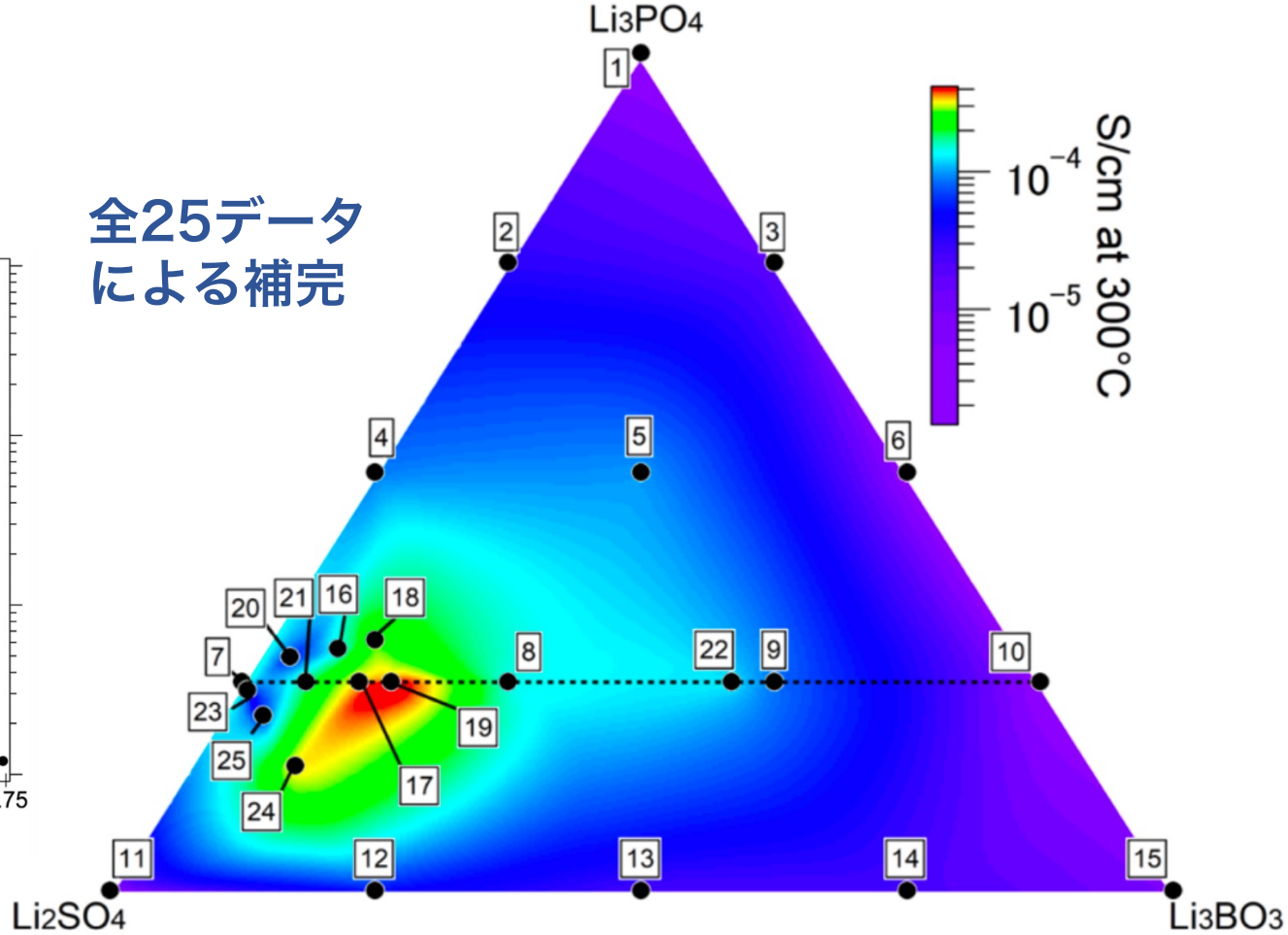
ベイズ最適化による最適組成

Li_2SO_4, Li_3PO_4, Li_3BO_3の混合

4.9 ×10⁻⁴ S/cm (300 °C)

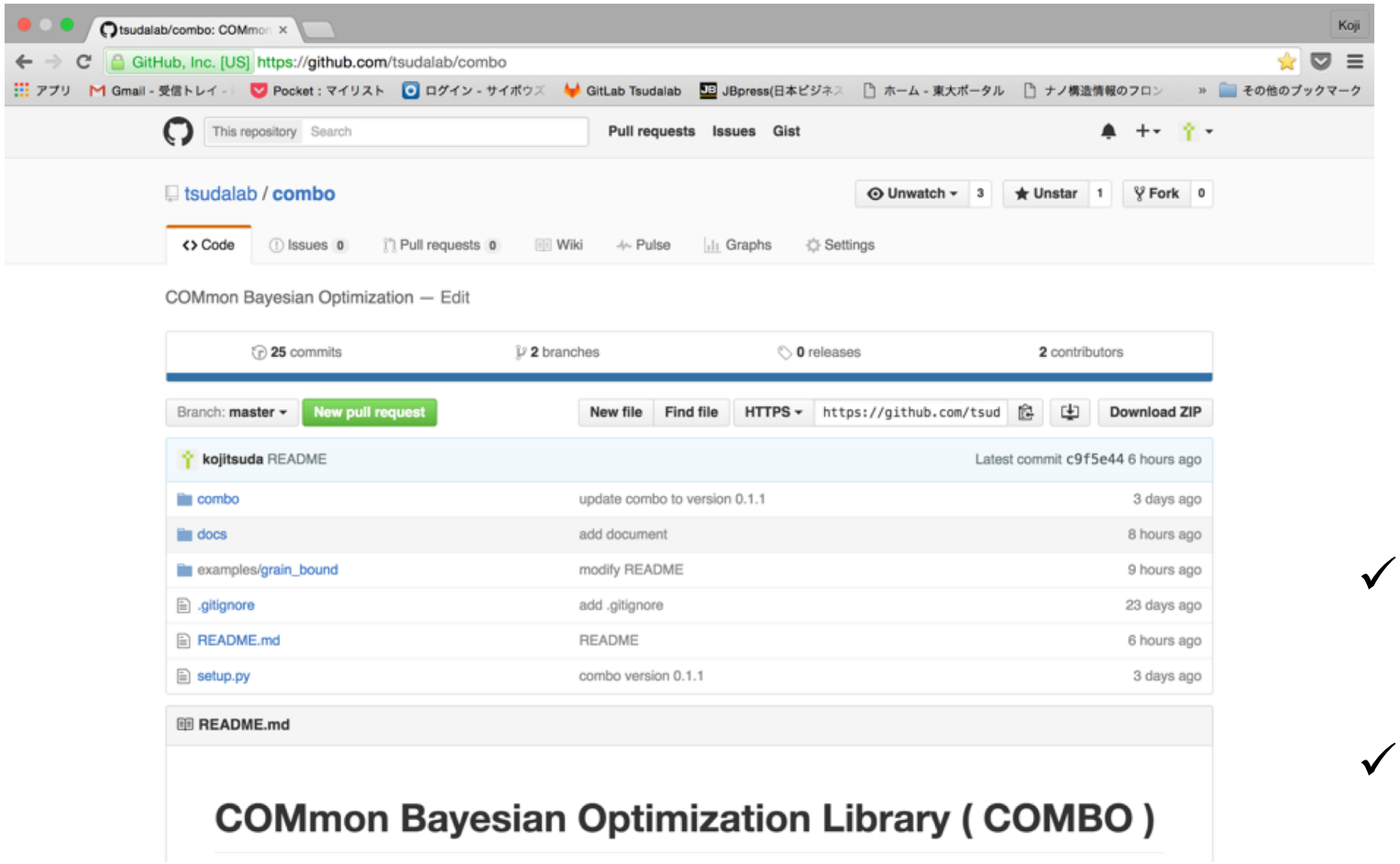


全25データによる補完



ベイズ最適化を実行するには？

ベイズ最適化パッケージCOMBO



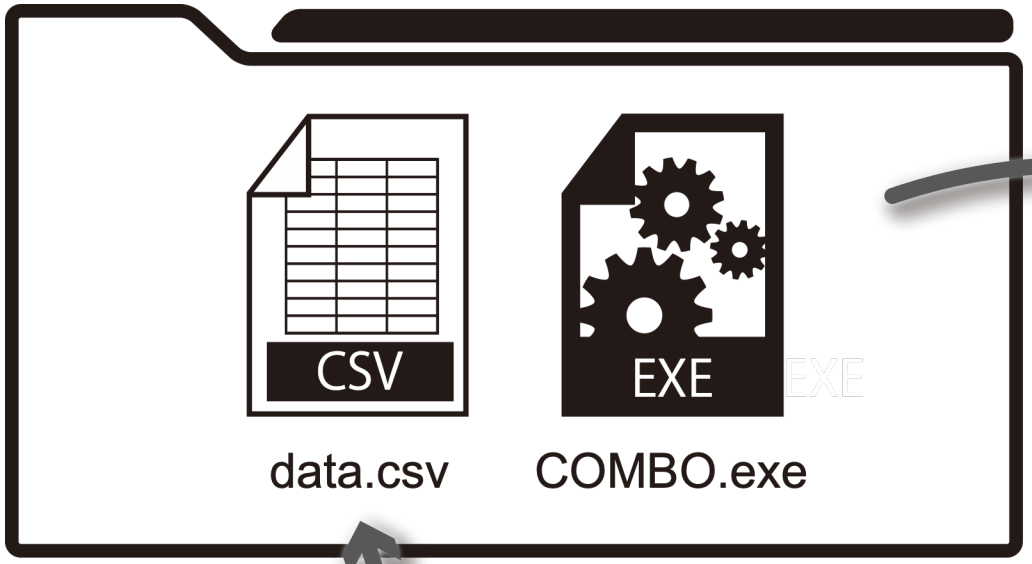
- ✓ ハイパーパラメタの学習を自動で実行
- ✓ トレーニングデータに対して線形計算可能

https://github.com/tsudalab/combo

Python2.7
MITライセンス

Windowsでベイズ最適化

COMBOをPythonのインストールなしにWindowsで実行



ドラッグ＆ドロップ

Enterキー
を押す

次の候補が出力される

Next Point: [0.74 0.45]
Row Number: 274

結果をデータファイルに
書き込む

材料特性の評価

K. Terayama, K. Tsuda, and R. Tamura,
Jpn. J. Appl. Phys. 58, 098001 (2019).

データファイルの用意

data.csv

1	収率	溶解温度	ガス圧力	…
2		0.0	10	…
3	8.2	0.2	9.7	…
4		0.4	9.4	…
5		0.6	9.1	…
6		0.8	8.8	…
7	4.5	1.0	8.5	…
8		1.2	8.2	…
9	2.0	1.4	7.9	…
10	3.7	1.6	7.6	…
11		1.8	7.3	…
12		2.0	7.0	…
13	10.1	2.2	6.7	…
14	⋮	⋮	⋮	

目的変数　説明変数(何次元でもOK)

この空欄から,
次の候補が選択
される.

COMBO.exe:
最適化したい
材料特性値を入力

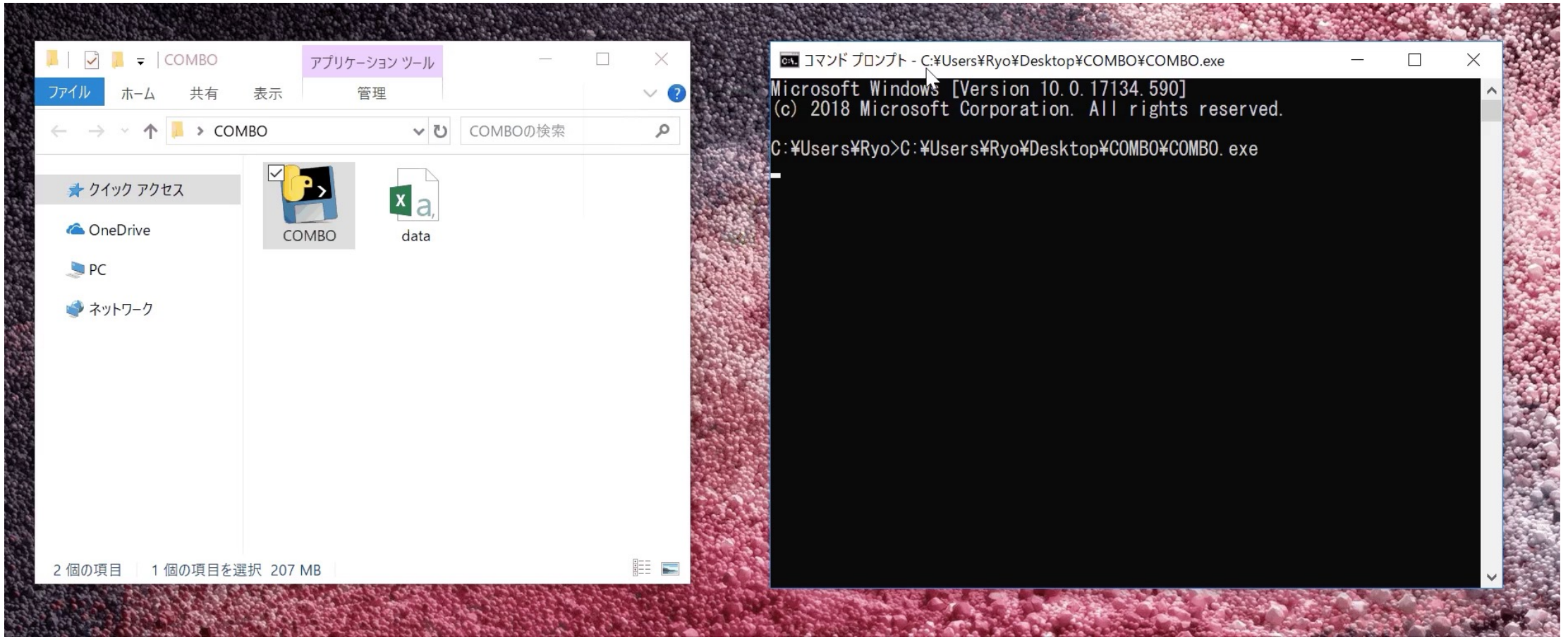
候補の
羅列

組成の違い

プロセスの違い

構造の違い

Windowsでの実行例



次に検討すべき候補が１つ出力される

Windowsアプリケーション

MItools for everyone

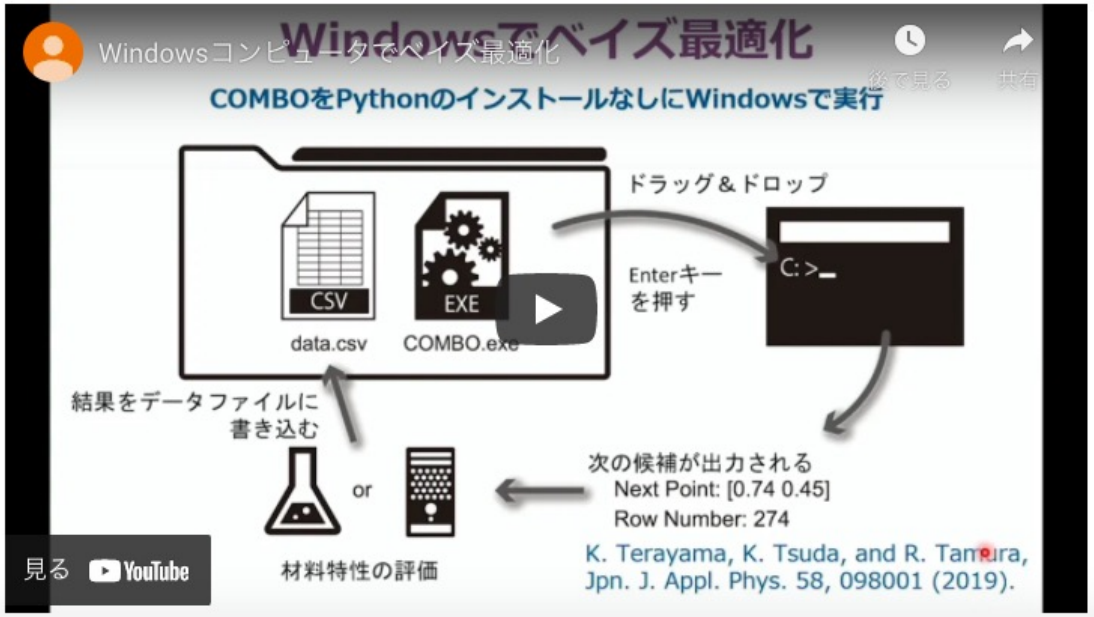
Kei Terayama, Koji Tsuda, Ryo Tamura

2019/10/10

改変はできないということを理解した上でダウンロードして使ってください。

作者または著作権者は、契約行為、不法行為、またはそれ以外であろうと、ソフトウェアに起因または関連し、あるいはソフトウェアの使用またはその他の扱いによって生じる一切の請求、損害、その他の義務について何らの責任も負わないものとします。

- COMBO.exe
- PDC.exe



https://www.tsudalab.org/project/mitools/

新ベイズ最適化ライブラリPHYSBO

COMBOの
アップグレード版
！GPLライセンス！

https://www.pasums.issp.u-tokyo.ac.jp/physbo/

- **pipでインストール**
- **Python3対応**
- **計算スピード高速化**
- **複数候補提案**
- **多目的最適化**
- **インタラクティブな実行**

開発者

- ver. 1.0-
  - 田村 亮 (物質・材料研究機構 国際ナノアーキテクトニクス研究拠点)
  - 寺山 慧 (横浜市立大学大学院 生命医科学研究科)
  - 津田 宏治 (東京大学大学院 新領域創成科学研究科)
  - 植野 剛 (東京大学大学院 新領域創成科学研究科)
  - 本山 裕一 (東京大学 物性研究所)
  - 吉見 一慶 (東京大学 物性研究所)
  - 川島 直輝 (東京大学 物性研究所)







インストール

https://issp-center-dev.github.io/PHYSBO/manual/master/ja/index.html

実行環境・必要なパッケージ

PHYSBOの実行環境・必要なパッケージは以下の通りです。

- Python >= 3.6
- numpy
- scipy

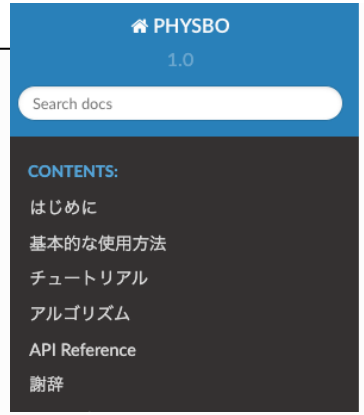
ダウンロード・インストール

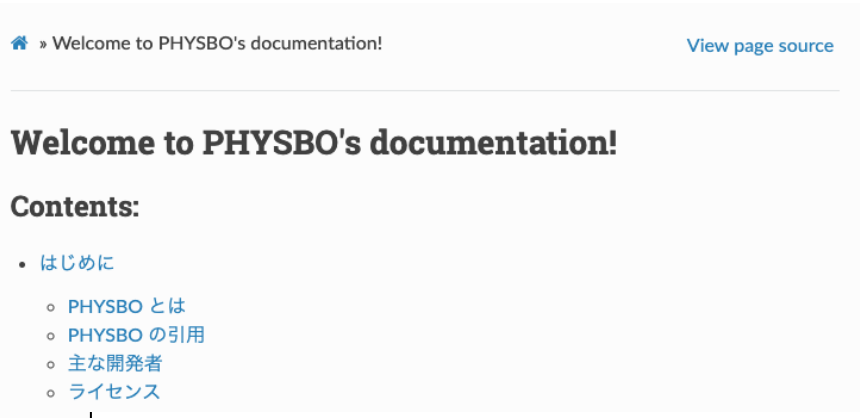
- `PyPI` からのインストール（推奨）

```
$ pip3 install physbo
```

  - NumPy などの依存パッケージも同時にインストールされます。
  - `--user` オプションを追加するとユーザのホームディレクトリ以下にインストールされます。

```
$ pip3 install --user physbo
```





オンラインマニュアル

基本的な使い方

探索する候補データを準備

```
import numpy as np
import scipy
import physbo
import itertools


#候補データを作成
window_num=10001
x_max = 2.0
x_min = -2.0

X = np.linspace(x_min,x_max,window_num).reshape(window_num, 1)
```

↑ 外部データを読み込むように設定してもOK
任意の次元の候補データを利用可能

基本的な使い方

シミュレータを設定（xからyを見積もる部分）

テスト問題：以下の関数を最小とするxを求める

$$f(x) = 3x^4 + 4x^3 + 1$$

```
#シミュレータを設定
class simulator:

    def __call__(self, action):
        action_idx = action[0]  ← actionが候補データのインデックス
        x = X[action_idx][0]
        fx = 3.0*x**4 + 4.0*x**3 + 1.0
        fx_list.append(fx)
        x_list.append(X[action_idx][0])

        print ("**********************")
        print ("Present optimum interactions")

        print ("x_opt=", x_list[np.argmin(np.array(fx_list))])

        return -fx  ← PHYSBOは最大化を目的として設定
```

基本的な使い方

Policyのセットとランダムサンプリングの実行

```
# policy のセット
policy = physbo.search.discrete.policy(test_X=X)

# シード値のセット
policy.set_seed(0)


fx_list=[]
x_list = []

#ランダムサンプリングを実行
res = policy.random_search(max_num_probes=5, simulator=simulator())
```

↑ 最適化対象をセット

サンプリングを行う回数↑

↑yを見積もるシミュレータ

```
**********
Present optimum interactions
x_opt= 1.758
0001-th step: f(x) = -51.387604 (action=9395)
   current best f(x) = -51.387604 (best action=9395)

**********
Present optimum interactions
x_opt= -0.5668
0002-th step: f(x) = -0.581263 (action=3583)
   current best f(x) = -0.581263 (best action=3583)

**********
Present optimum interactions
x_opt= -0.5668
0003-th step: f(x) = -0.827643 (action=4015)
   current best f(x) = -0.581263 (best action=3583)

**********
Present optimum interactions
x_opt= -0.5668
0004-th step: f(x) = -14.220707 (action=154)
   current best f(x) = -0.581263 (best action=3583)
```

基本的な使い方

ベイズ最適化を実行

```
#ベイズ最適化を実行
res = policy.bayes_search(max_num_probes=25, simulator=simulator(), score='TS',
                                                interval=0, num_rand_basis=500)
```

max_num_probes: ベイズ最適化でサンプリングする回数

score: ベイズ最適化で用いるスコア（TS, EI, PIが設定可能）

interval: ハイパーパラメタを学習する頻度（0は最初だけ）

num_rand_basis: random feature mapの数

```
Start the initial hyper parameter searching ...
Done

Start the hyper parameter learning ...
0 -th epoch marginal likelihood 20.97382285870551
50 -th epoch marginal likelihood 20.912127433547955
100 -th epoch marginal likelihood 20.87096704197894
150 -th epoch marginal likelihood 20.83711886237566
200 -th epoch marginal likelihood 20.80955386449423
250 -th epoch marginal likelihood 20.78762706057598
300 -th epoch marginal likelihood 20.770561465402302
350 -th epoch marginal likelihood 20.757535174493253
400 -th epoch marginal likelihood 20.747743310952192
450 -th epoch marginal likelihood 20.74044551012996
500 -th epoch marginal likelihood 20.73499825017711
Done

**********
Present optimum interactions
x_opt= -0.5668
0006-th step: f(x) = -0.716884 (action=3800)
   current best f(x) = -0.581263 (best action=3583)

**********
Present optimum interactions
x_opt= -0.5668
0007-th step: f(x) = -1.008253 (action=5309)
   current best f(x) = -0.581263 (best action=3583)
```

基本的な使い方

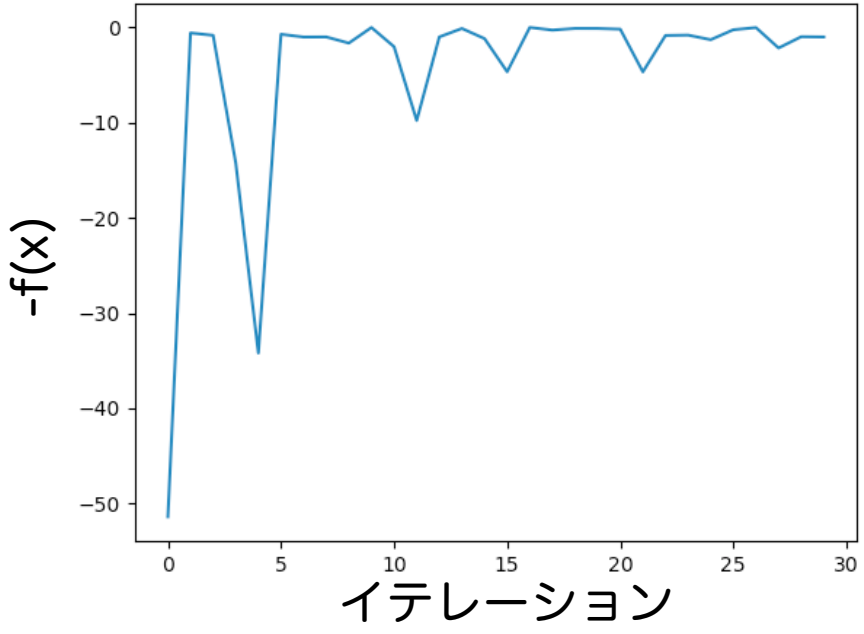
結果を出力

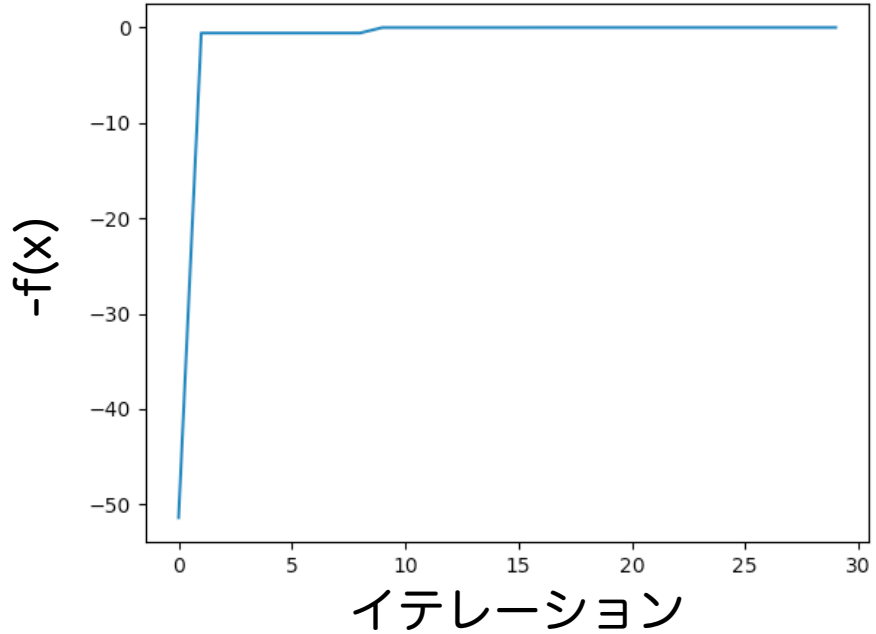
f(x)の履歴をプロット

```
plt.plot(res.fx[0:res.total_num_search])
```

f(x)の最大値をプロット

```
best_fx, best_action = res.export_all_sequence_best_fx()
plt.plot(best_fx)
```





PHYSBOでのBO高速化

ガウシアンカーネルをRandom feature mapで近似

$$k(\mathbf{x}, \mathbf{x}') = \exp\left[-\frac{1}{2\eta^2}\|\mathbf{x}-\mathbf{x}'\|\right]^2 \simeq \phi(\mathbf{x})^\top \phi(\mathbf{x}')$$

$$\phi(\mathbf{x}) = (z_{\boldsymbol{\omega}_1, b_1}(\mathbf{x}/\eta), \ldots, z_{\boldsymbol{\omega}_l, b_l}(\mathbf{x}/\eta))^\top$$

← ベクトルの次元がnum_rand_basisに対応

$$z_{\omega, b}(\mathbf{x}) = \sqrt{2}\cos(\boldsymbol{\omega}^\top \mathbf{x} + b)$$

$l \to \infty$ ：近似が厳密に成立

通常のガウス過程回帰：学習データNに対してO(N^3)計算が必要

num_rand_basis=0で計算可能

Random feature mapによる近似：学習データNに対してO(N)計算が可能

PHYSBOで使えるスコア

-Thompson Sampling (TS)

Random feature mapの数lに対してO(l)

$$\mathrm{TS}(\mathbf{x}) = \mathbf{w}^{*\top}\phi(\mathbf{x})$$

-Maximum Probability of Improvement (PI)

Random feature mapの数lに対してO(l^2)

$$\mathrm{PI}(\mathbf{x}) = \Phi(z(\mathbf{x})), \quad z(\mathbf{x}) = \frac{\mu_c(\mathbf{x}) - y_{\max}}{\sigma_c(\mathbf{x})}$$

-Maximum Expected Improvement (EI)

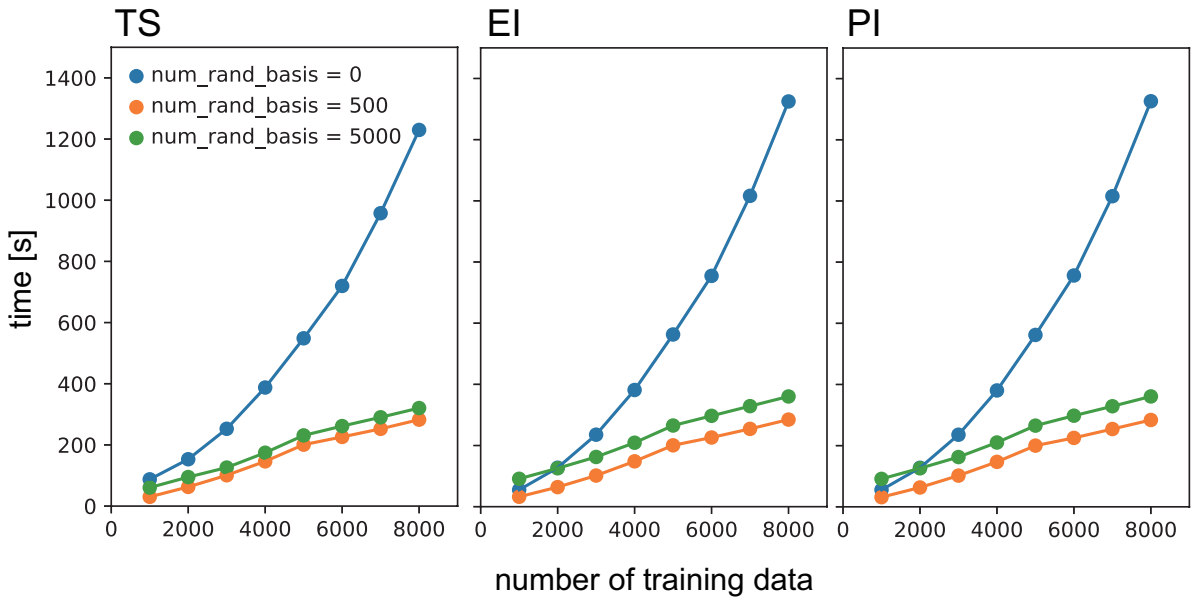
Random feature mapの数lに対してO(l^2)

$$\mathrm{EI}(\mathbf{x}) = [\mu_c(\mathbf{x}) - y_{\max}]\Phi(z(\mathbf{x})) + \sigma_c(\mathbf{x})\phi(z(\mathbf{x}))$$

PHYSBOの計算時間

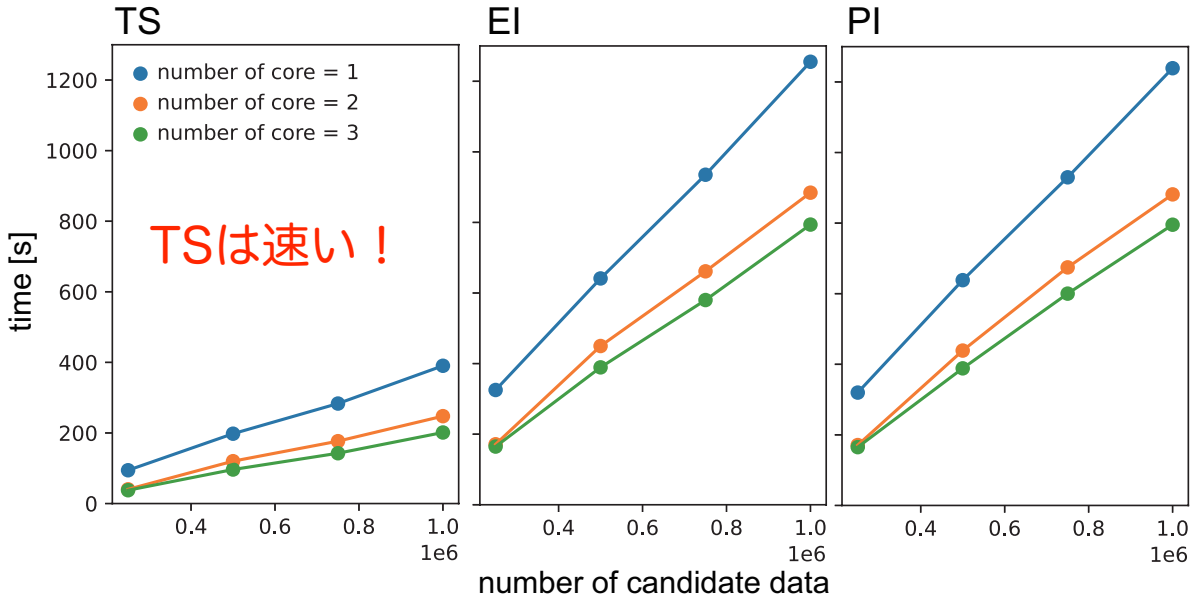
-ガウス過程回帰の学習時間依存性

1000個の候補から200個を選ぶ時間
の初期データ数依存性
(200回の学習が行われる)
ハイパーパラメータの学習２回を含む



-候補選択時間

初期データ数1000個とし，200個を
選ぶ時間の候補データ数依存性
(200回の学習が行われる)
ハイパーパラメータの学習２回を含む



材料スクリーニングの実行例

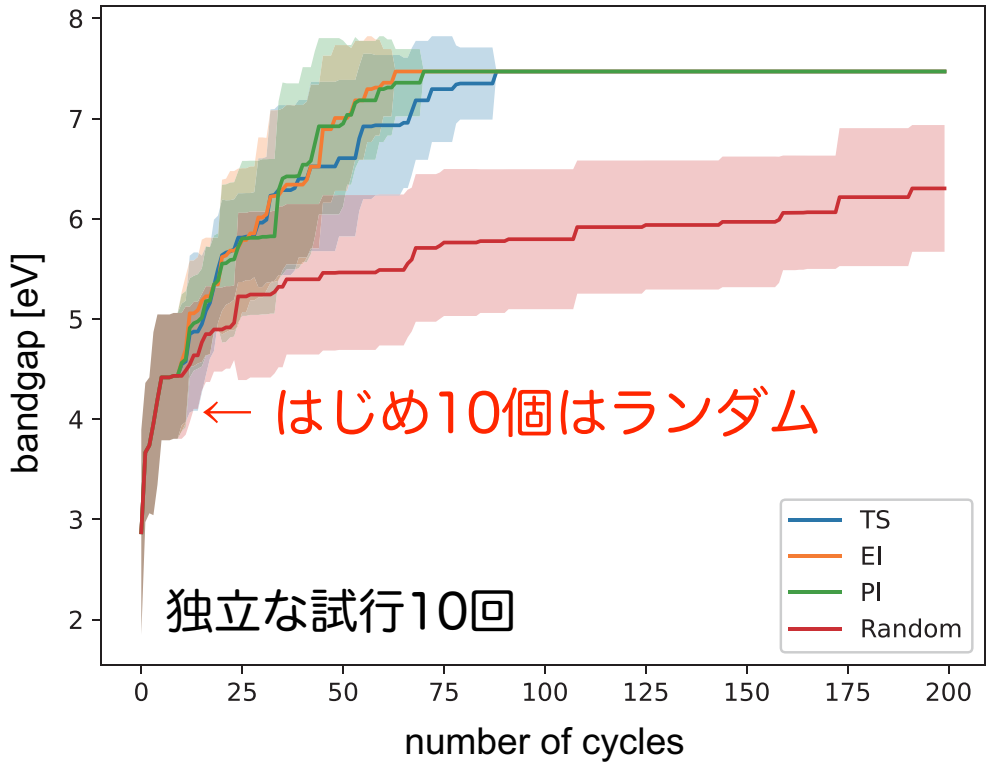
バンドギャップの大きな材料を見つける問題に適用

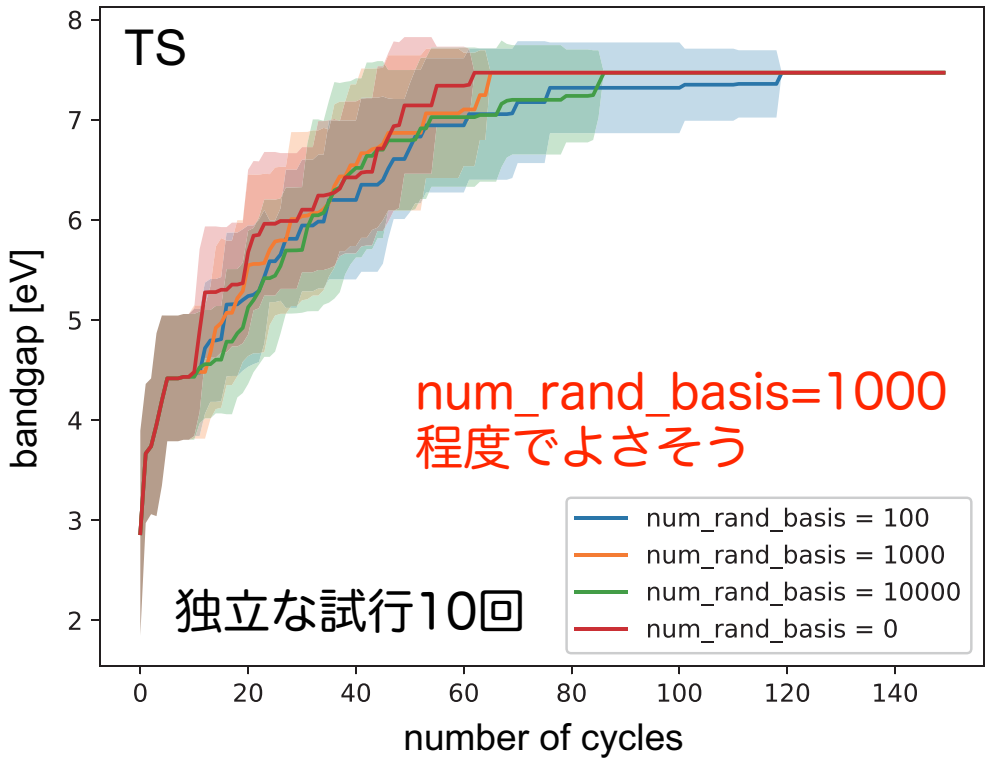
材料データ数：1277個
説明変数x: magpie descriptor

Open Access

Machine learning models for predicting the dielectric constants of oxides based on high-throughput first-principles calculations

Akira Takahashi, Yu Kumagai, Jun Miyamoto, Yasuhide Mochizuki, and Fumiyasu Oba
Phys. Rev. Materials **4**, 103801 – Published 9 October 2020





複数提案の場合

```
#シミュレータを設定
class simulator:

    def __call__(self, action):

        fx_list = []

        for i in range(len(action)):

            action_idx = action[i]
            x = X[action_idx][0]
            fx = 3.0*x**4 + 4.0*x**3 + 1.0

            fx_list.append(-fx)

        return np.array(fx_list)
```

←複数の提案がactionに来るので,
numpyのリストで返す

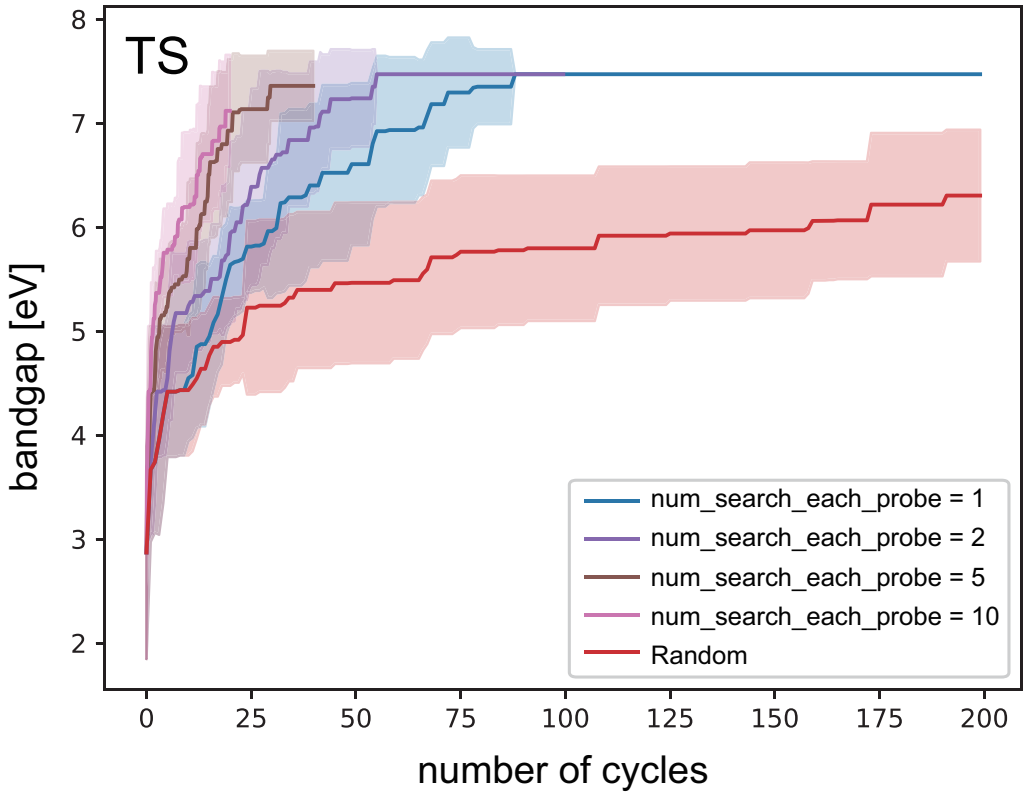
num_search_each_probeに提案数を記載
↓

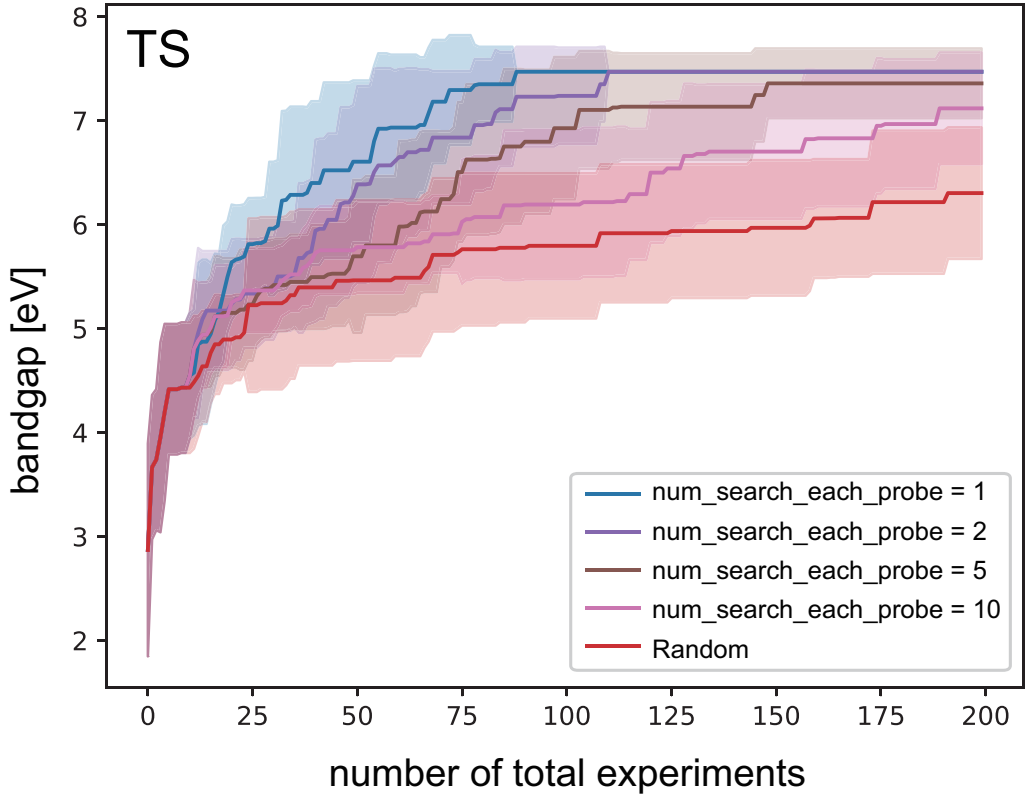
```
#ベイズ最適化を実行
res = policy.bayes_search(max_num_probes=8, num_search_each_probe=10, simulator=simulator(), score='TS',
                          interval=0, num_rand_basis=500)
```

```
0031-th multiple probe search (TS)

current best f(x) = -0.001735 (best action = 2543)
list of simulation results
f(x)=-0.015494 (action = 2377)
f(x)=-0.030324 (action = 2330)
f(x)=-0.005170 (action = 2428)
f(x)=-0.013488 (action = 2385)
f(x)=-0.029953 (action = 2331)
f(x)=-0.017371 (action = 2370)
f(x)=-0.006574 (action = 2419)
f(x)=-0.017096 (action = 2371)
f(x)=-0.019952 (action = 2361)
f(x)=-0.029218 (action = 2333)


0032-th multiple probe search (TS)

current best f(x) = -0.001735 (best action = 2543)
list of simulation results
f(x)=-0.054117 (action = 2276)
f(x)=-0.009913 (action = 2401)
f(x)=-0.007970 (action = 2411)
f(x)=-0.014223 (action = 2382)
f(x)=-0.016285 (action = 2374)
f(x)=-0.028493 (action = 2335)
f(x)=-0.033381 (action = 2322)
f(x)=-0.009507 (action = 2403)
f(x)=-0.010120 (action = 2400)
f(x)=-0.021779 (action = 2355)
```

複数提案の場合

利点：ベイズ最適化のサイクル数を少なくできる.
欠点：yの評価回数（実験回数・計算回数に対応）は増える.



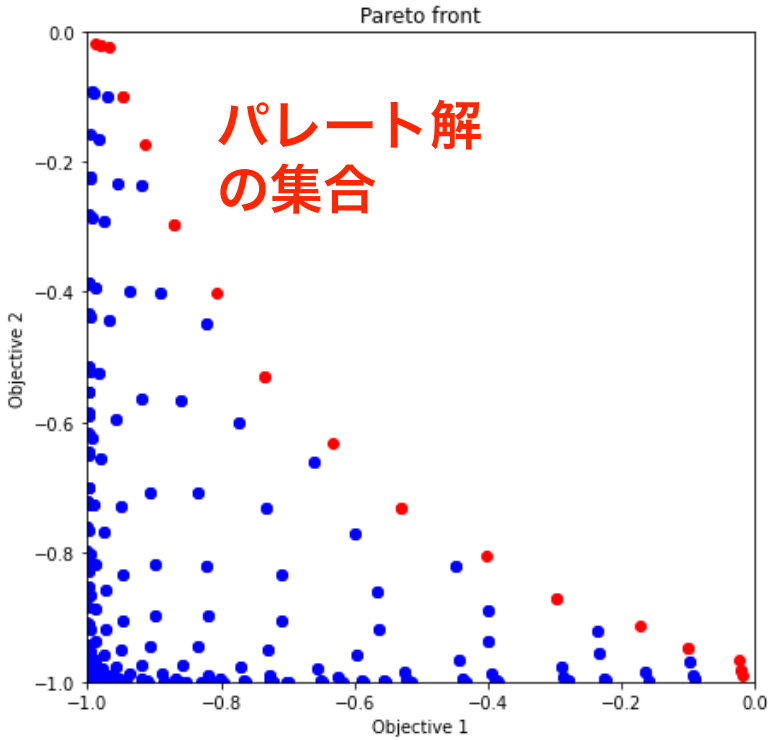


並列実験，並列計算のコストが十分小さければ，複数提案は有効！

多目的最適化

多目的最適化の問題設定： $\min_{\mathbf{x}}[f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})]$

- 複数の最適化したい関数 $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})$ があるが，関数間にはトレードオフの関係があり，多くのパレート解が存在する．
- パレート解を多く求めておき，動機にフィットした解を選択する．

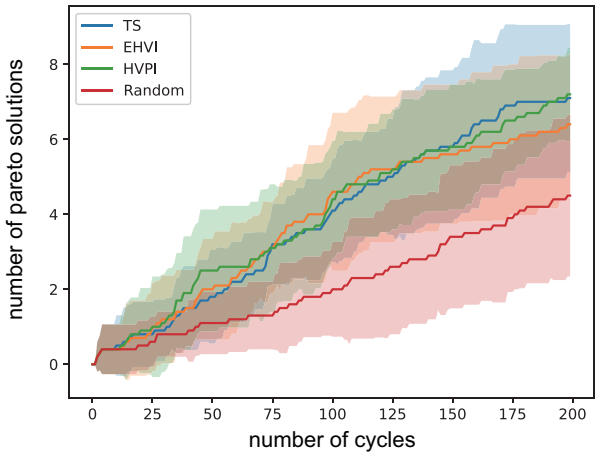


PHYSBOの多目的最適化で利用できる獲得関数

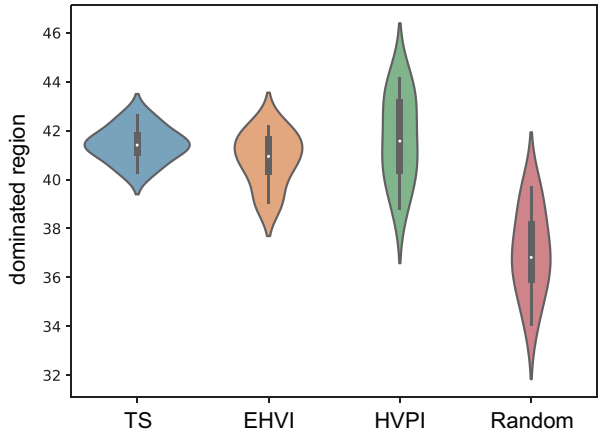
HVPI (Hypervolume-based Probability of Improvement)
EHVI (Expected Hyper-Volume Improvement)
TS (Thompson Sampling)

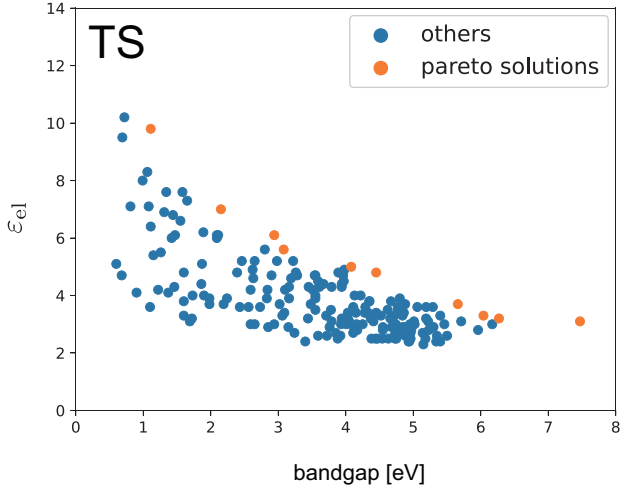
多目的最適化

見つけたパレート解の数



Dominated region






```python
#シミュレータを設定
class simulator:

    def __call__(self, action):

        action_idx = action[0]
        x = X[action_idx][0]
        fx1 = 3.0*x**4 + 4.0*x**3 + 1.0
        fx2 = -3.0*x**4 - 4.0*x**3 - 1.0


        return [[fx1,fx2]]
```

← 2次元配列で返す

```python
# policy のセット
policy = physbo.search.discrete_multi.policy(test_X=test_X, num_objectives=2)
```

↑ 目的関数の数

```python
#ベイズ最適化を実行
res = policy.bayes_search(max_num_probes=40, simulator=simulator(), score='EHVI',
                          interval=0, num_rand_basis=500)
```

↑ HVPI, EHVI, TS

更なる機能

インタラクティブな実行（simulatorをあらかじめ定義しない）

```
#ランダムに選択
actions = policy.random_search(max_num_probes=1, simulator=None)
fx = simulator(actions)
policy.write(actions, fx)
```

←actionだけ提案される

←結果をpolicyに書き込む

```
#ベイズ最適化の実行
actions = policy.bayes_search(max_num_probes=1, simulator=None, score='EI',
                              interval=0,  num_rand_basis = 5000)
fx = simulator(actions)
policy.write(actions, fx)
```

既存の計算・実験結果を利用してベイズ最適化をスタート

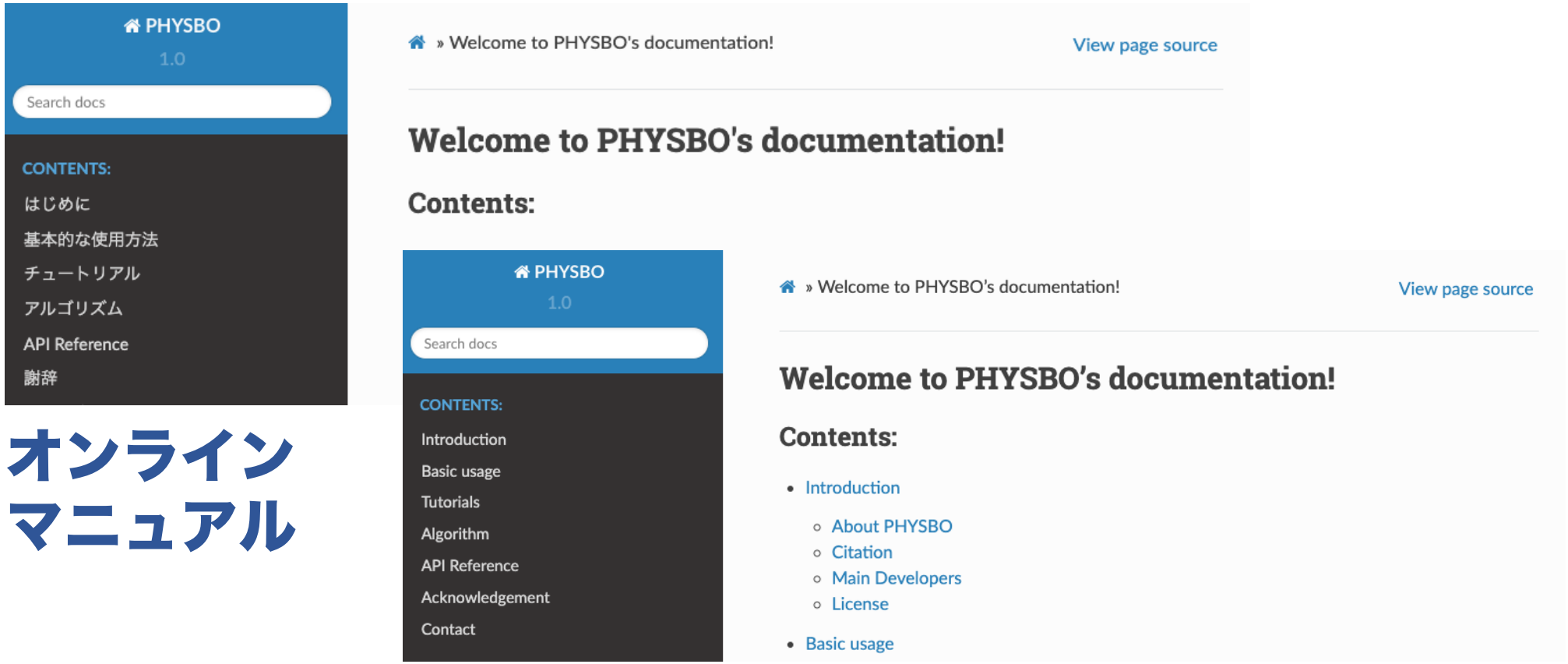
```
# policy のセット
policy = physbo.search.discrete.policy(test_X=X, initial_data=[calculated_ids, fx_initial])
```

calculated_ids: xのうちすでに評価済みのidのリスト
fx_initial: 評価済みのyの値を格納したリスト

```
#ベイズ最適化の実行
actions = policy.bayes_search(max_num_probes=1, simulator=None, score='EI',
                              interval=0,  num_rand_basis = 5000)
```

マニュアル

https://issp-center-dev.github.io/PHYSBO/manual/master/ja/index.html



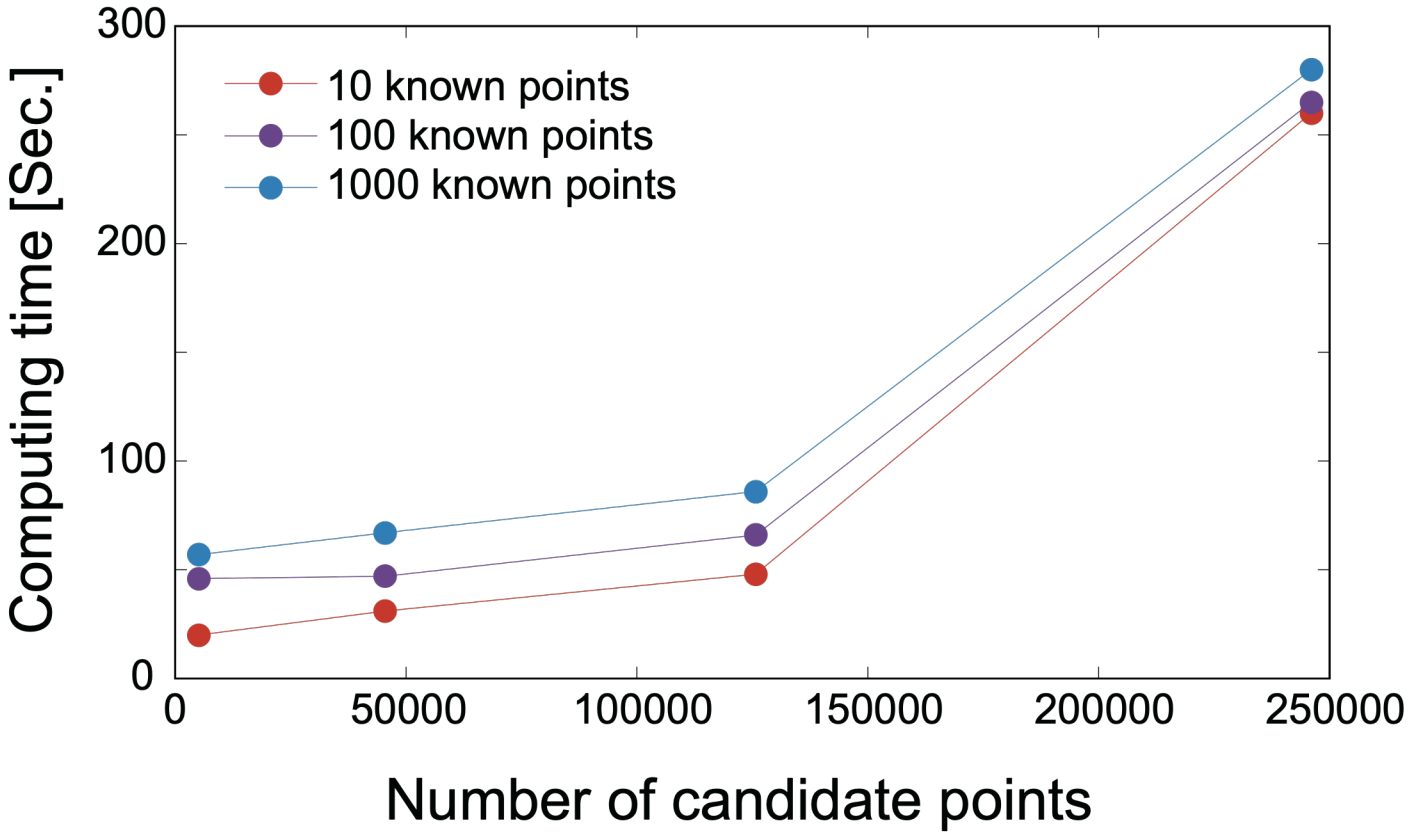
オンライン
マニュアル

Y. Motoyama, R. Tamura, K. Yoshimi, K. Terayama, T. Ueno, and K. Tsuda, Computer Physics Communications 278, 108405 (2022).

ベイズ最適化の問題と
量子アニーリング技術の活用

ベイズ最適化に残された問題

ベイズ最適化では，全ての候補に対して獲得関数を計算している．
候補が多いと計算が大変…

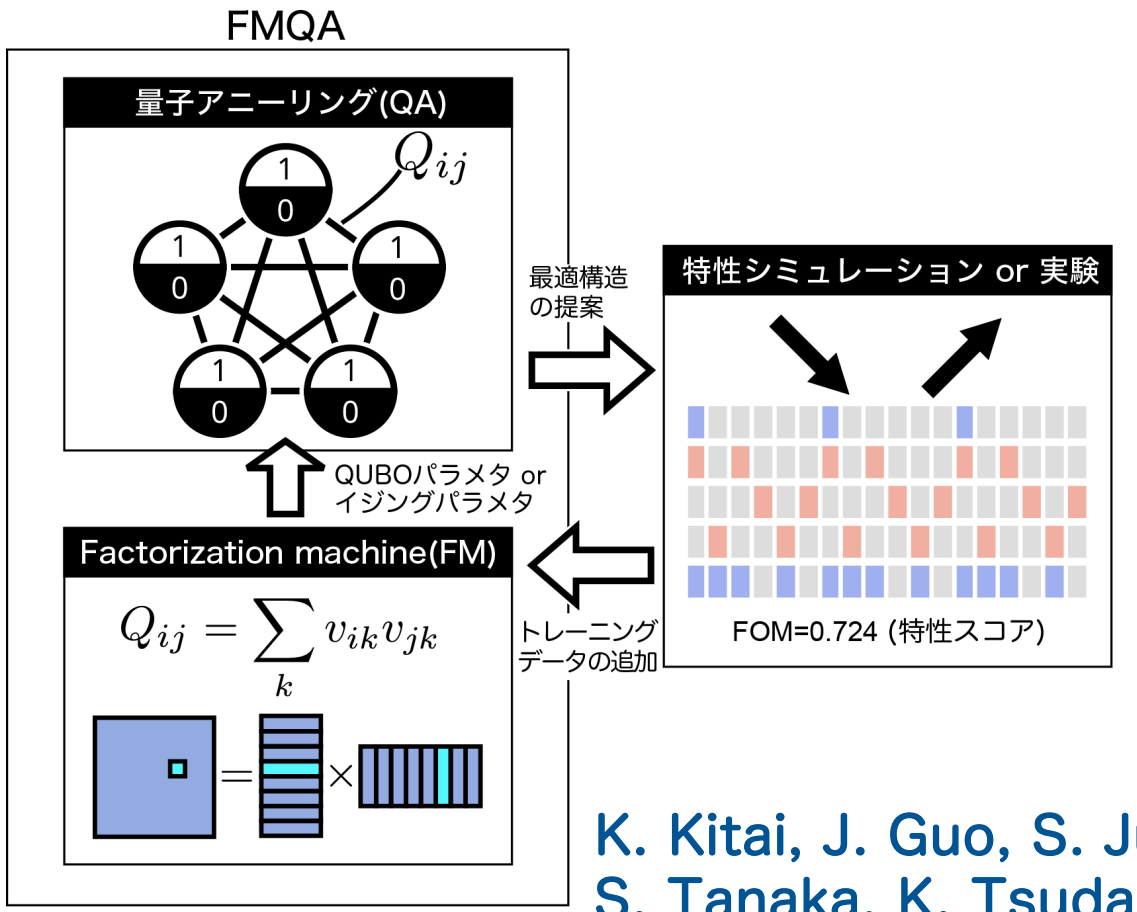


材料科学では，
組成やプロセスの
バラエティが多く，
簡単に組合せ爆発が起こる．

どうにか高速化することはできないか？　→ 次世代計算技術が使える？

量子アニーリング技術を用いて高速化

Factorization machine with quantum annealing (量子アニーリングを用いた新アルゴリズム)



K. Kitai, J. Guo, S. Ju,
S. Tanaka, K. Tsuda, J. Shiomi, and R. Tamura,
Phys. Rev. Research 2, 013319 (2020).

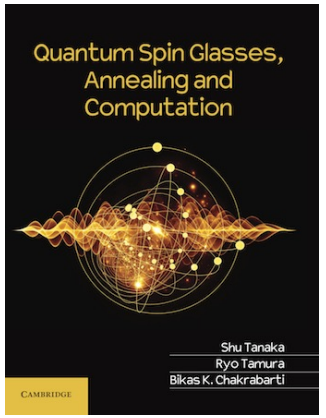
イジングマシン(量子アニーリング)とは？

イジングモデルの基底状態(エネルギーが低い)を高速で解くハードウェア

イジングモデル

$$\mathcal{H} = \sum_{i,j} J_{ij} s_i s_j + \sum_i h_i s_i \quad s_i \in \{-1, +1\}$$

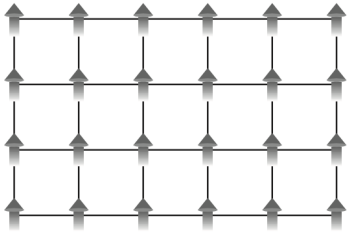
$\sum_{i,j}$: スピン間相互作用　　$\sum_i$: 磁場



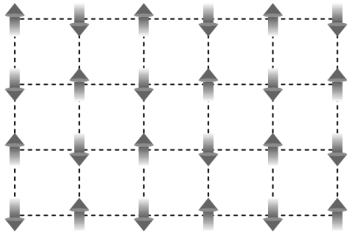
QUBO (Quadratic unconstrained binary optimization)

$$\mathcal{H} = \sum_{i=1}^{N} \sum_{j=1}^{N} Q_{ij} x_i x_j \quad x_i \in \{0, 1\}$$

組合せ最適化問題(NP問題)は
イジングモデルで表現可能

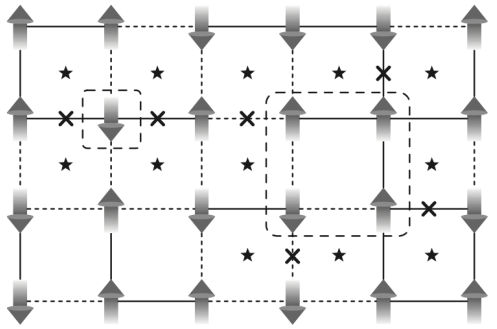


$J < 0$: 強磁性相互作用



$J > 0$: 反強磁性相互作用

全てのエネルギーが
満たされることがなく，
基底状態を見つけるのは
難しい
(スピングラス)



相互作用が混ざったとき

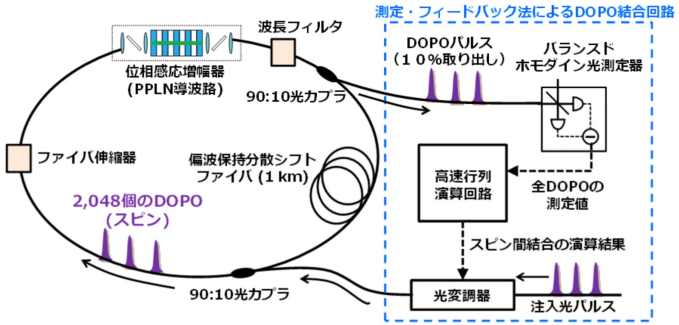
主なイジングマシン

イジングモデルの基底状態を高速で解くハードウェア

D-Wave Systems

Hitachi

NTT

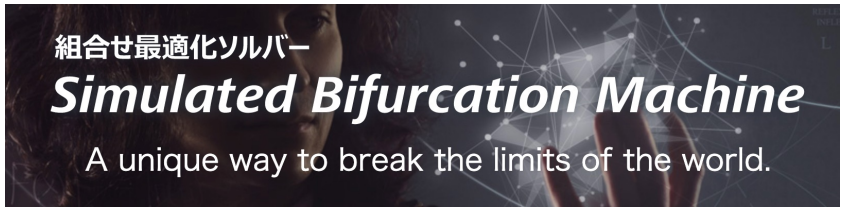


fixstars

Optigan

Fujitsu

Toshiba



NEC

ANCAR

武笠陽介

https://ancar.app/

ANCARによるTSP解説

ANCAR 投稿コンテンツ デモアプリ 典型問題 マシン キーワード検索 ログイン ユーザー登録

- 投稿コンテンツ
- デモアプリケーション
  - 勤務シフト作成ツール
  - 山手線シミュレーター
  - 時間割作成ツール
- 典型問題
  - 巡回セールスマン問題
  - 最大カット問題
  - グラフ彩色問題
- アニーリングマシン
  - D-Wave
  - Amplify Annealing Engine

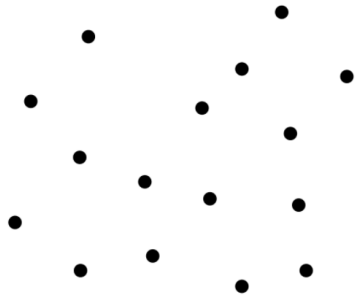
Home / コンテンツ一覧 / 巡回セールスマン問題

巡回セールスマン問題

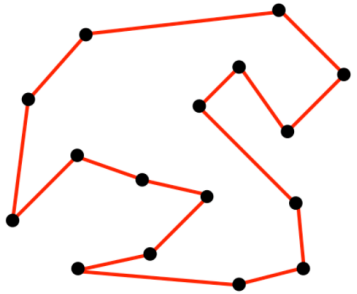
- 概要
- 定式化
- シミュレーション

概要

巡回セールスマン問題(Traveling salesman problem)とは、セールスマンがいくつかの都市を1度ずつすべての都市を訪問して出発点に戻ってくるときに、移動コストが最小になる経路を求める問題です。



入力



出力

都市数 N に対して、経路は $\frac{(N-1)!}{2}$ 通りの候補があり、総当たりで計算すると莫大な計算時間がかかり現実的ではありません。

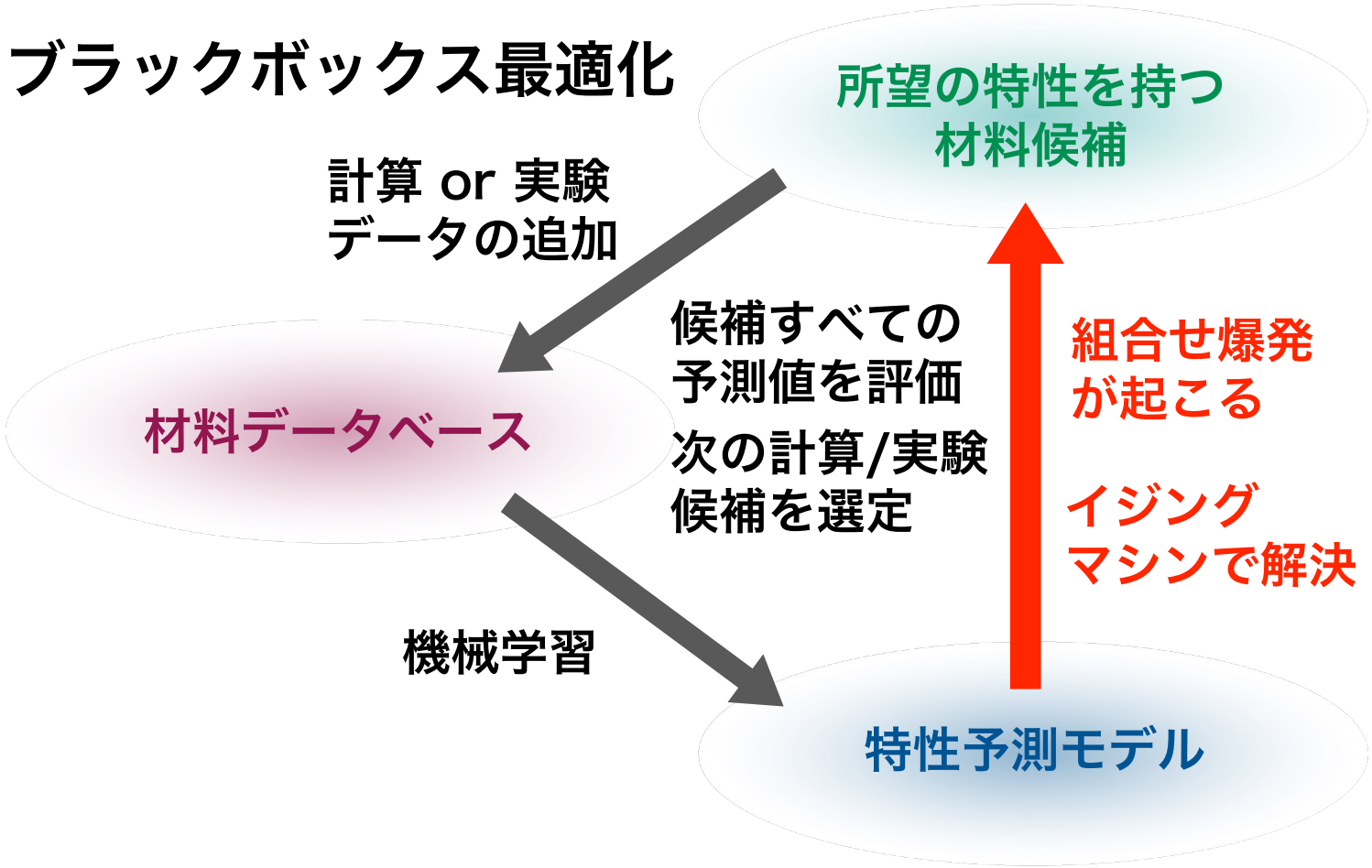
エネルギー関数

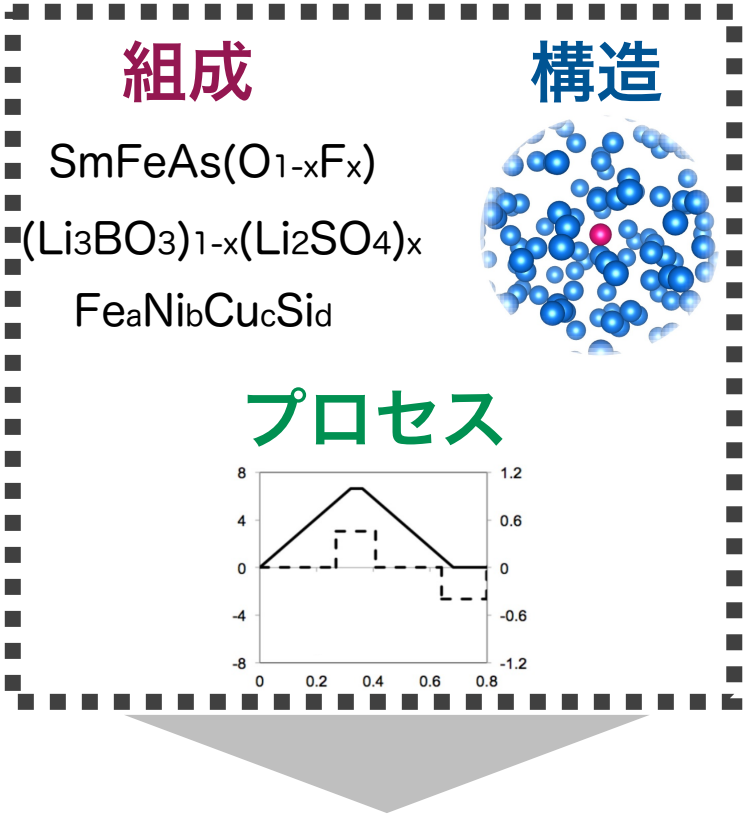
上記の目的関数と制約項を用いて、エネルギー関数は以下の式のように表されます。

$$H = \sum_{t=1}^{N-1}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{i,j}x_{t,i}x_{t+1,j} + A\sum_{t=1}^{N}\left(1-\sum_{i=1}^{N}x_{t,i}\right)^2 + A\sum_{i=1}^{N}\left(1-\sum_{t=1}^{N}x_{t,i}\right)^2$$

ブラックボックス最適化を加速

より多くの候補数を扱うためにイジングマシンを用いた候補選択を行う.





バイナリ問題に特化したブラックボックス最適化手法

0-1で表せる問題に適用可能

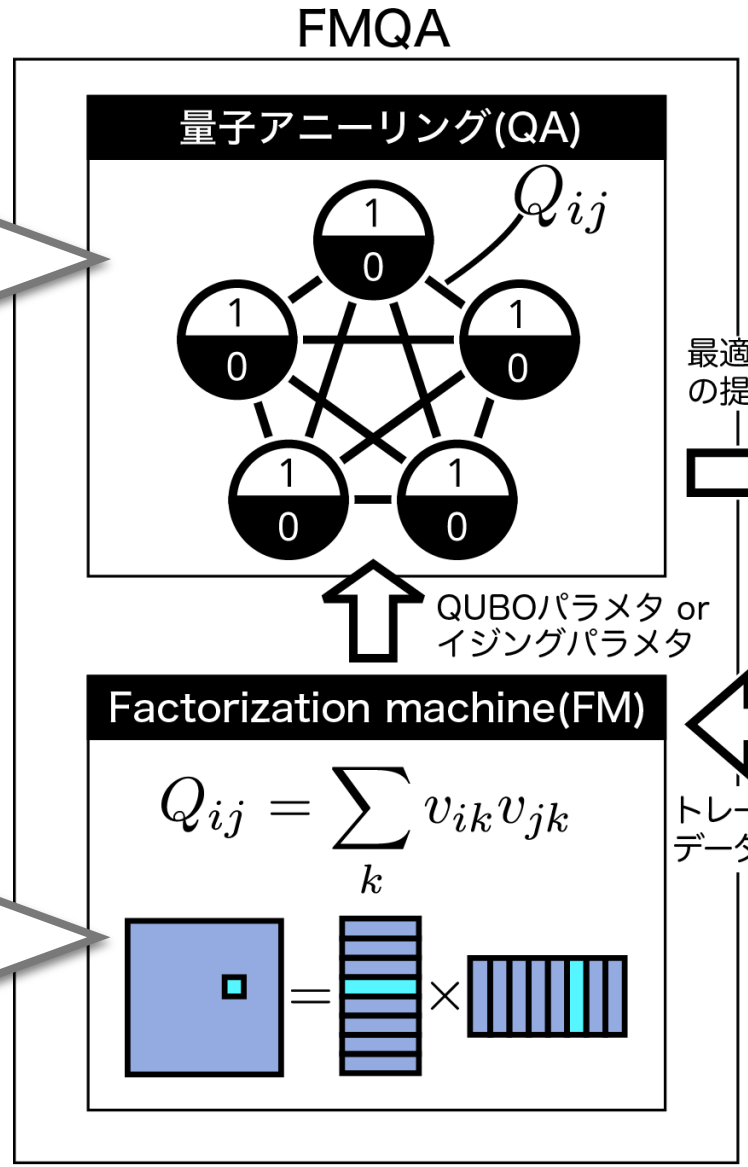
QAを用いたブラックボックス最適化の手順

全ての候補から
予測特性が一番よい
候補をイジング
マシンで選ぶ.

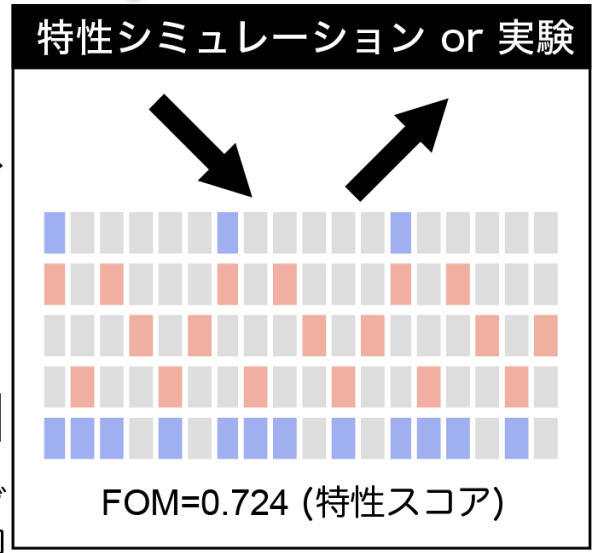
特性予測用
イジングモデル
を学習する.

Factorization machine

イジングモデルの
基底状態が望みの特性を
持つと予測される材料



一番良い候補の特性を
計算or実験する.



既存の材料シミュレータ
に置き換え不要
簡単に導入可能

Factorization machine

$$y(\mathbf{x}) = \sum_{i=1}^{N} w_i x_i + \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{K} v_{ik} v_{jk} x_i x_j$$

目的変数
(材料特性)

：FMの予測モデル(Adamで学習)
(BOでのガウス過程の代わり)

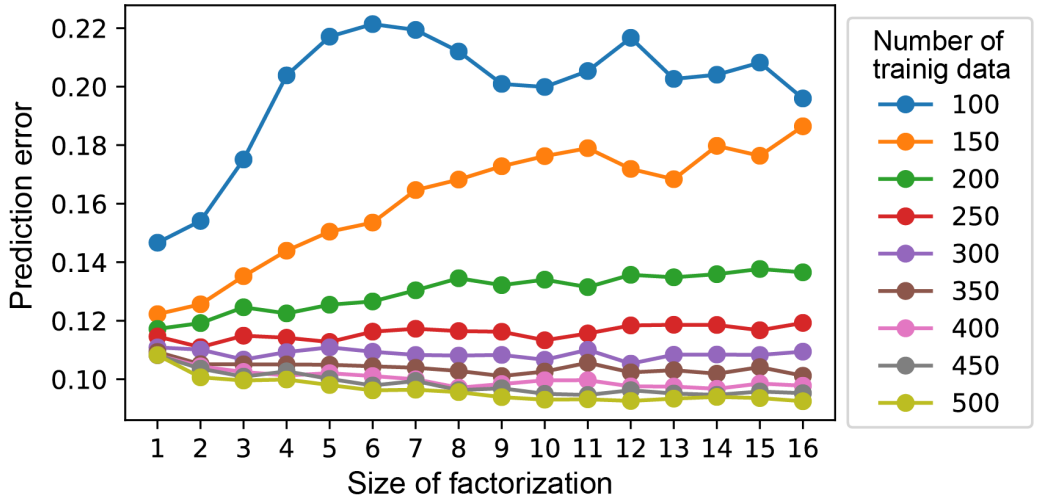
$$\mathcal{H} = \sum_{i=1}^{N}\sum_{j=1}^{N} Q_{ij} x_i x_j$$ ：QUBO

書き換え可能

$x_i \in \{0, 1\}$：説明変数(構造)

FMの利点：フィッティングパラメタがスパース（少なく）になり
過学習が防げる.

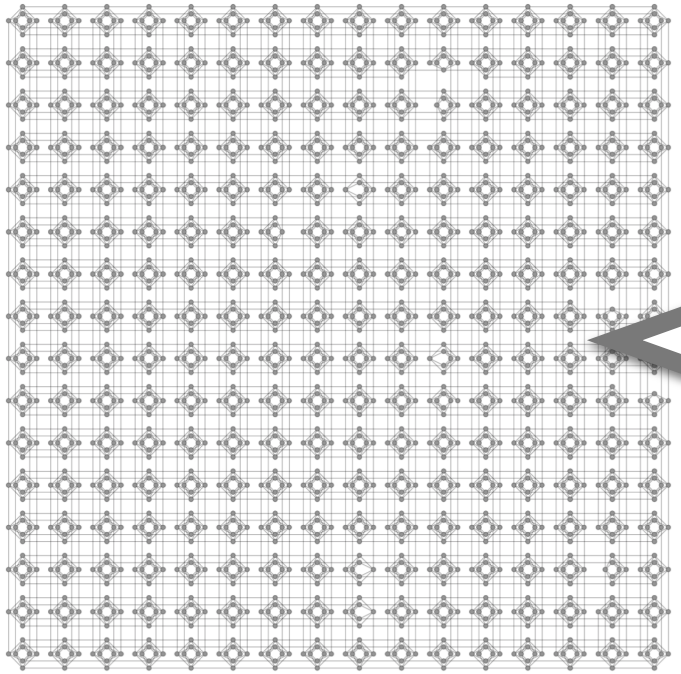
S. Rendle, IEEE International Conference on Data Mining pp. 995–1000 (2010).



Factorization machineをQUBOに書き換えると，
QUBOの基底状態は，予測特性が最も良い材料の説明変数

イジングマシンによる選択

D-Wave 2000Qを用いて選択を実行
（あらゆるイジングマシンが適用可能）



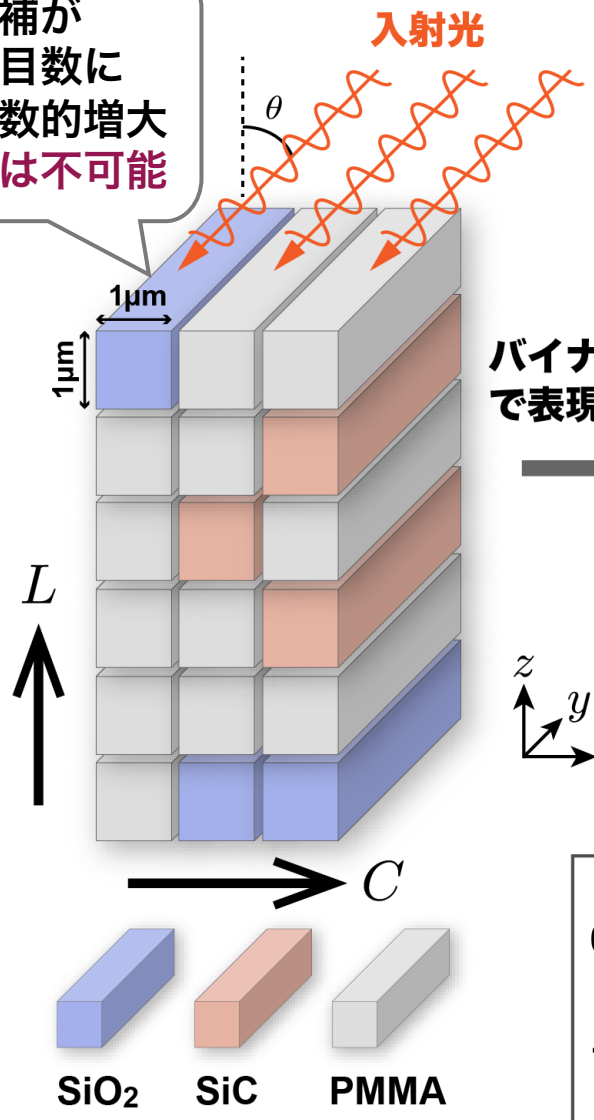
num_reads=50で
最小エネルギー状態を
次の候補に採用

2038キュービットのキメラグラフ
(63ビットの全結合グラフの計算が可能)

すでに選択された材料が選択された場合　→　ランダムに選択

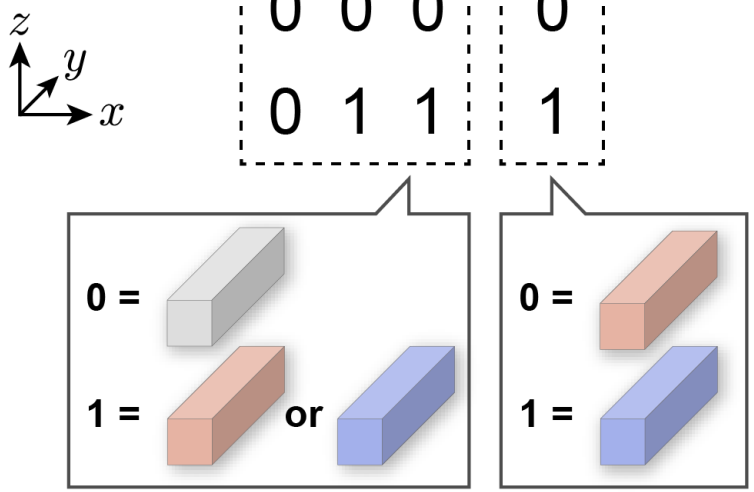
放射冷却用メタマテリアル開発

構造候補が
さいの目数に
指数関数的増大
全計算は不可能



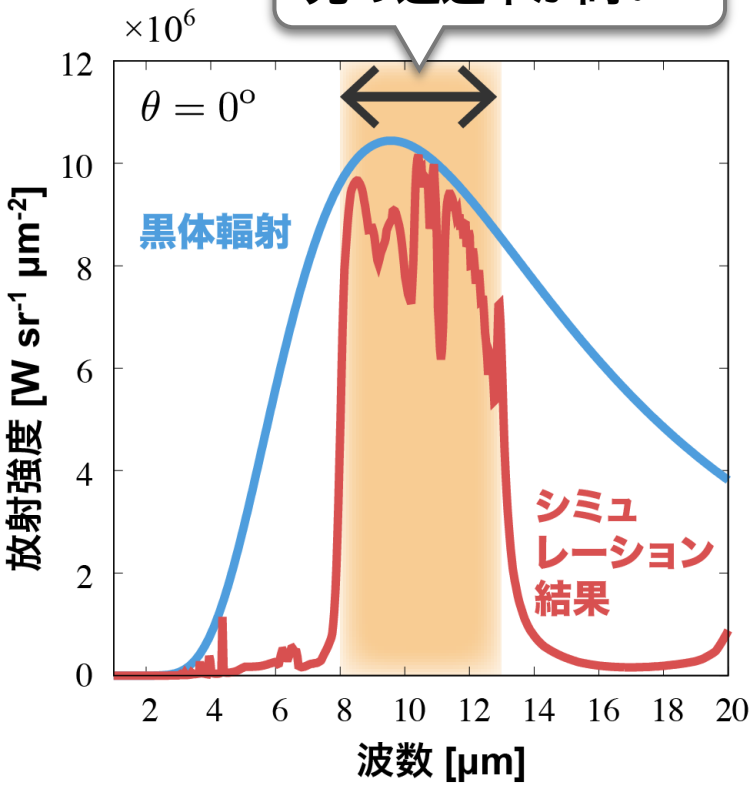
バイナリ変数で表現

Si 材料 or PMMA を表す変数			SiC or SiO2 を表す変数
1	0	0	1
0	0	1	0
0	1	0	0
0	0	1	0
0	0	0	0
0	1	1	1



特性シミュレータ

大気の窓
大気の影響が少なく
光の透過率が高い



スコアを評価

FOM=0.686

電磁波が宇宙空間に放出され温度が低下

RCWAとFOM

厳密結合波理論(RCWA)により，
分光放射率を計算

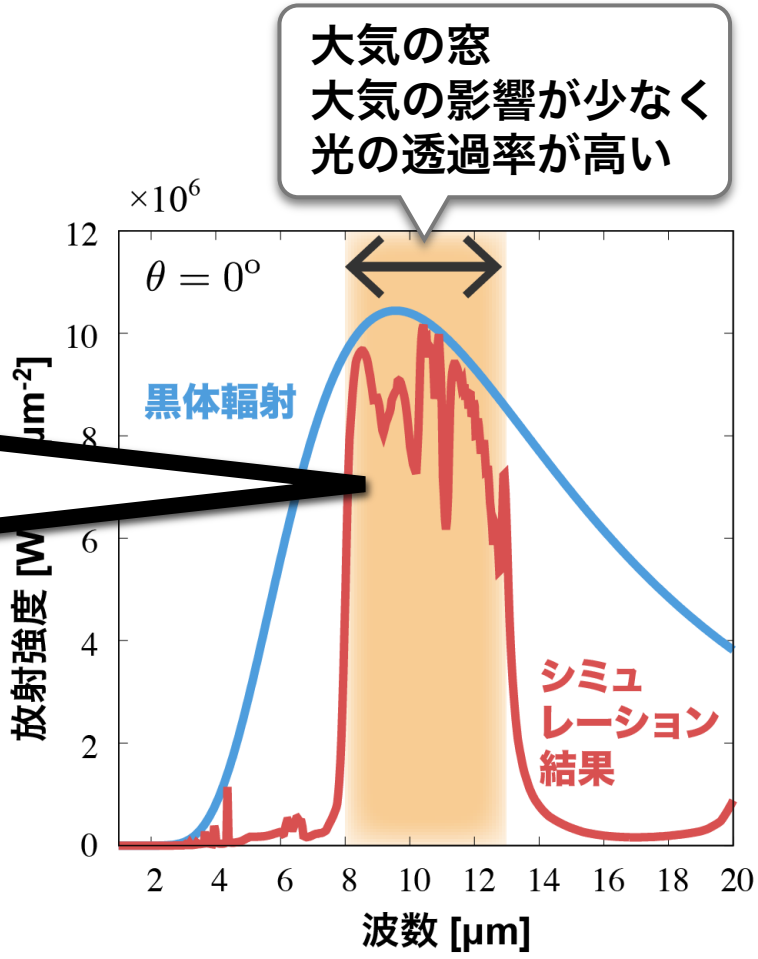
Figure-Of-Merit(FOM)の定義

$$\mathrm{FOM} = \frac{\int_{\lambda_\mathrm{i}}^{\lambda_\mathrm{f}} \epsilon_\lambda E_{b\lambda} d\lambda}{\int_{\lambda_\mathrm{i}}^{\lambda_\mathrm{f}} E_{b\lambda} d\lambda}$$ ：8~13μmに入っている部分

$$- \frac{\int_{\lambda_\mathrm{min}}^{\lambda_\mathrm{i}} \epsilon_\lambda E_{b\lambda} d\lambda}{\int_{\lambda_\mathrm{min}}^{\lambda_\mathrm{i}} E_{b\lambda} d\lambda}$$ ：8μm以下の部分

$$- \frac{\int_{\lambda_\mathrm{f}}^{\lambda_\mathrm{max}} \epsilon_\lambda E_{b\lambda} d\lambda}{\int_{\lambda_\mathrm{f}}^{\lambda_\mathrm{max}} E_{b\lambda} d\lambda}$$ ：13μm以上の部分

大きいと良い材料
(FMを学習する際はマイナスをつける)



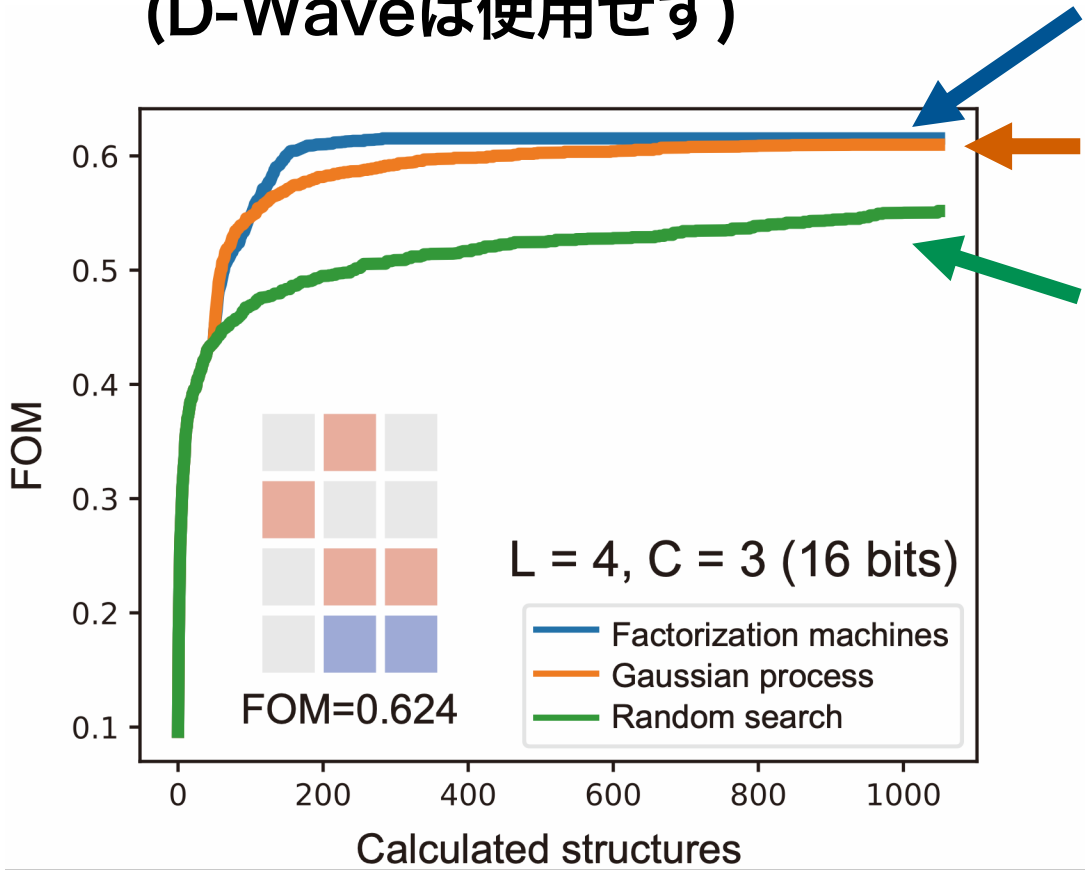
スコアを評価

FOM=0.686

FMQA有用性の実証

- 16ビットで表せる構造
  (2^{16}=65536)

全探索による選択
(D-Waveは使用せず)



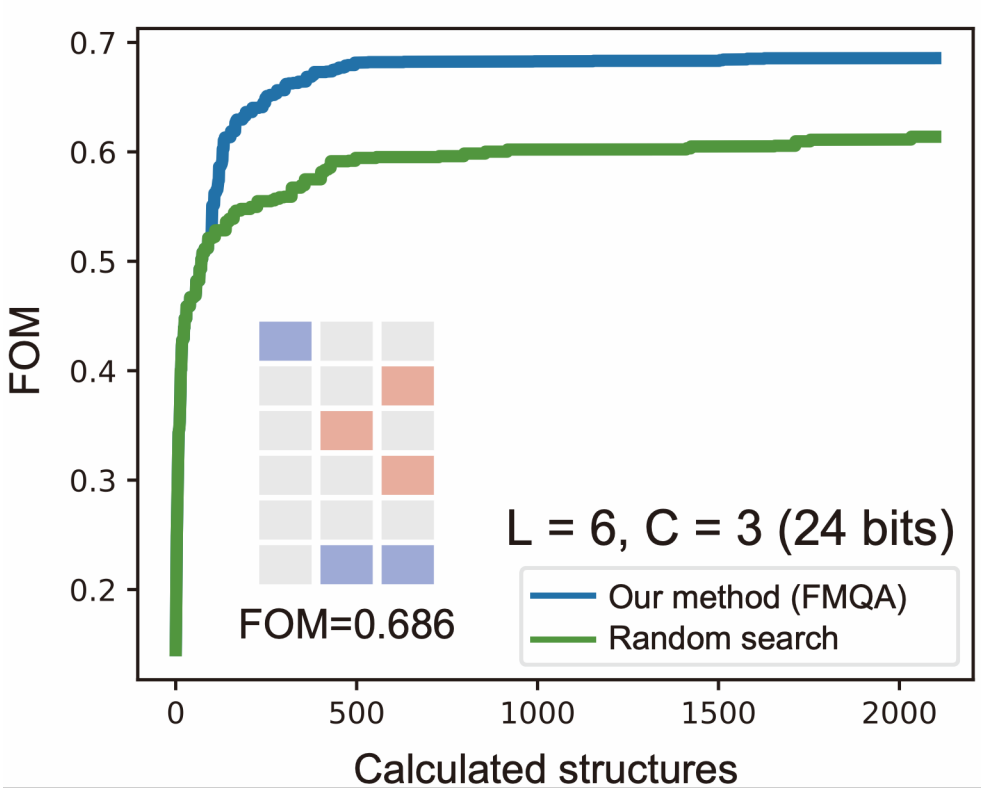
予測モデルとしてFMを利用

予測モデルとしてGPを利用
(ベイズ最適化)

ランダムに探索

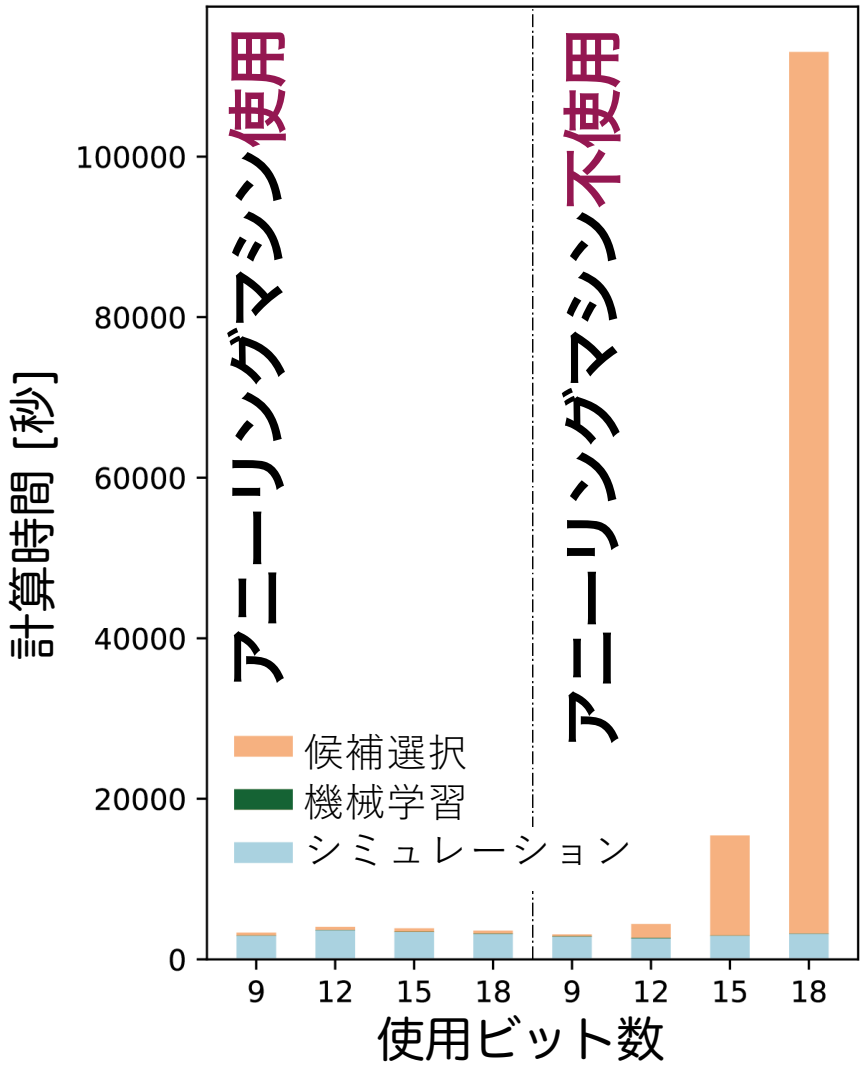
- 24ビットで表せる構造
  (2^{24}=16777216)

D-Waveを用いた選択
(候補数が多くなった)

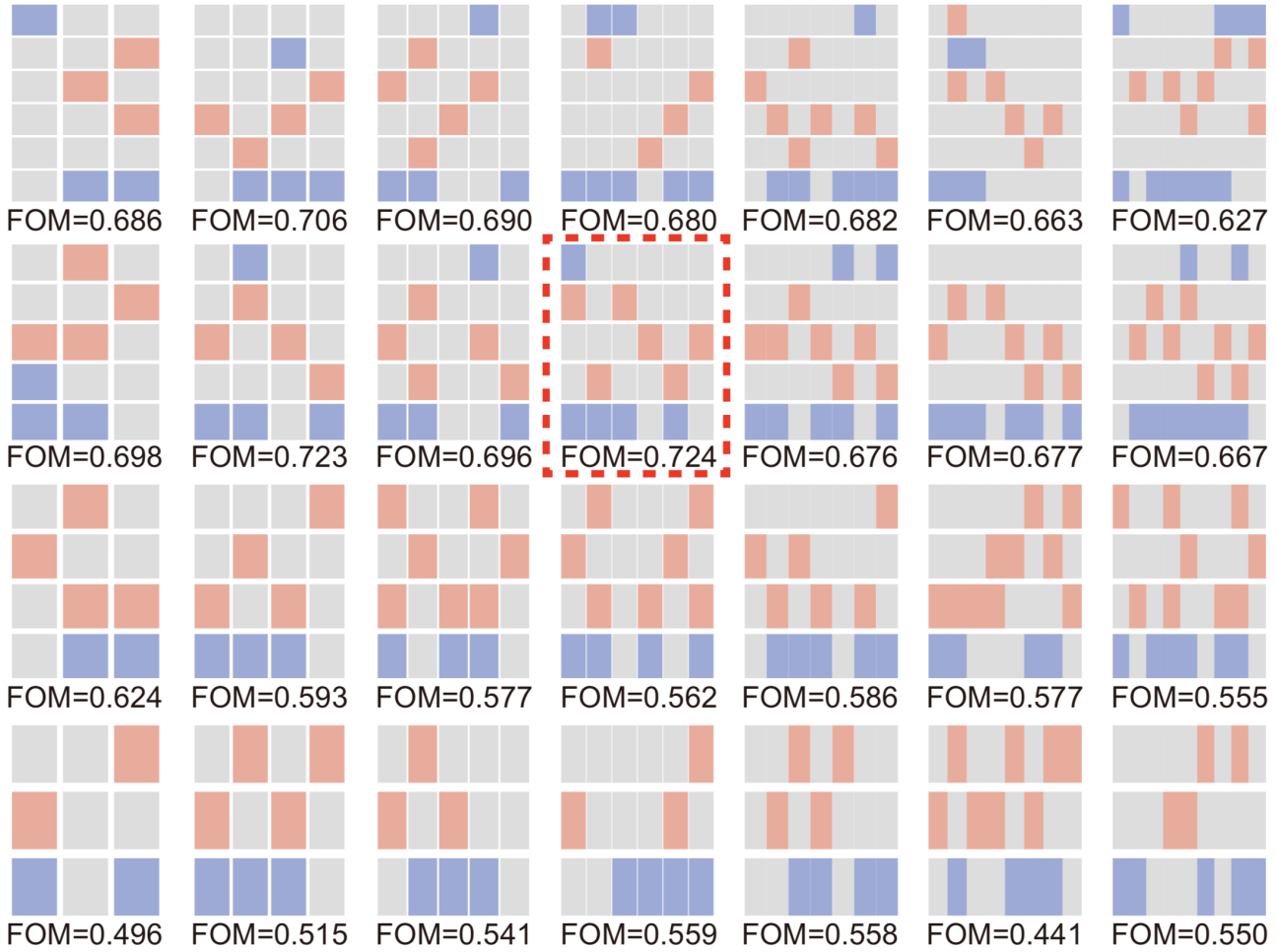


計算時間と探索結果

計算時間比較



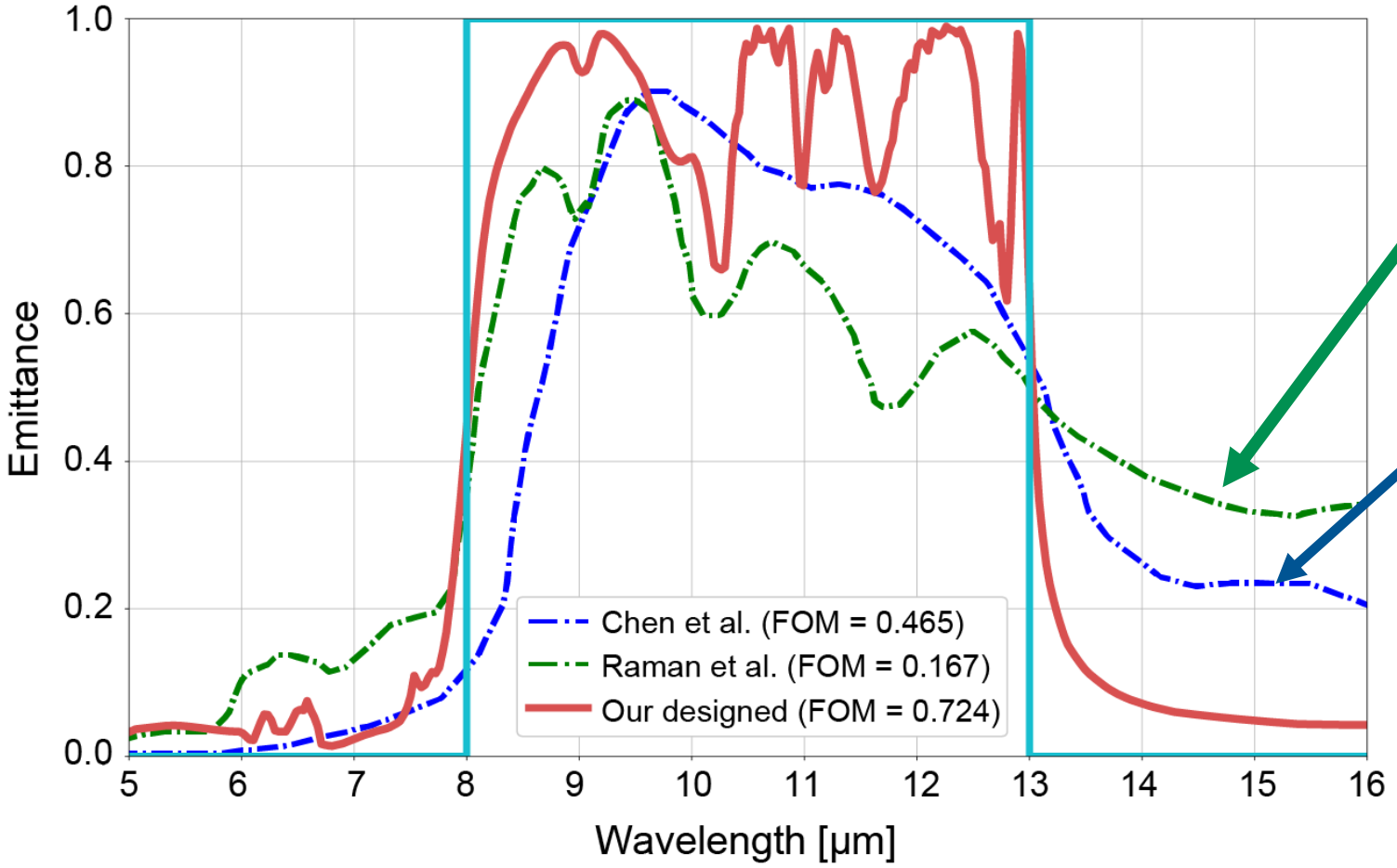
FMQAが見つけた最適構造



計算時間の短縮により網羅計算が実行可能に

これまでの材料との比較

アニーリングマシンを用いてデザインした
メタマテリアルは既存の材料より良いFOM値を示す.

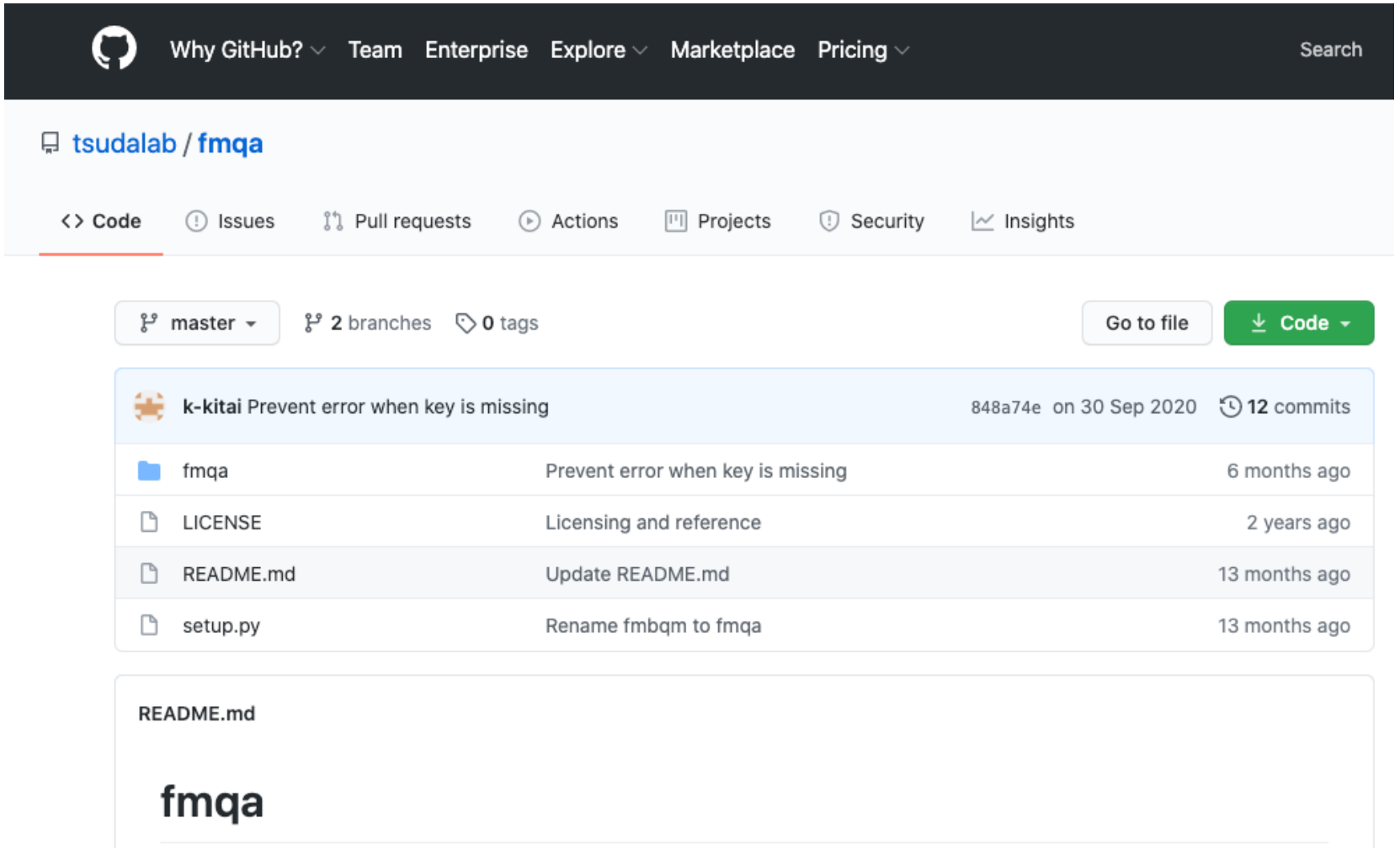


SiO_2, HfO_2の積層

Si_3N_4, Si, Alの積層

さいの目構造により,
高いFOM値を実現

FMQA package

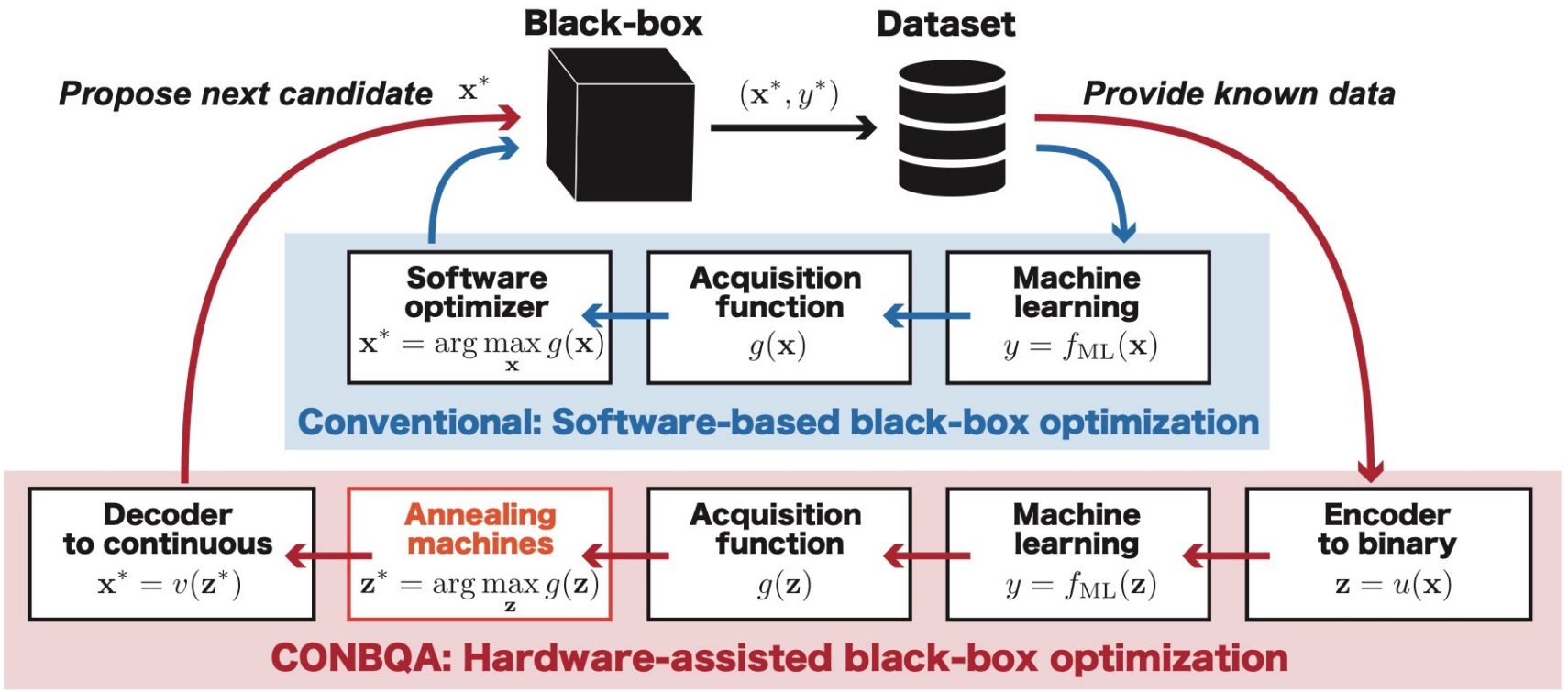


シミュレーテッドアニーリングも利用できる.

https://github.com/tsudalab/fmqa

連続値も扱いたくなってくる…

CONtinuous Black-box optimization with Qubo model solved by Annealer (CONBQA)

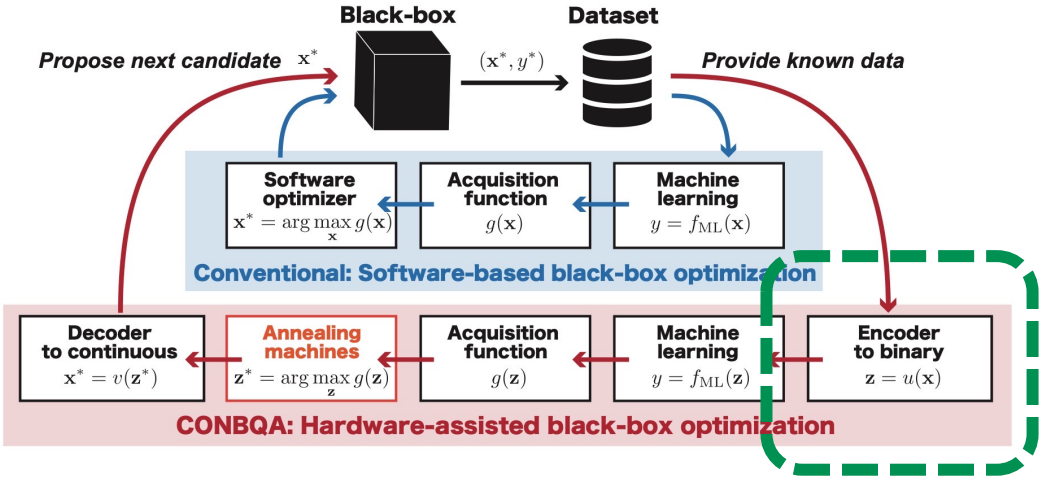


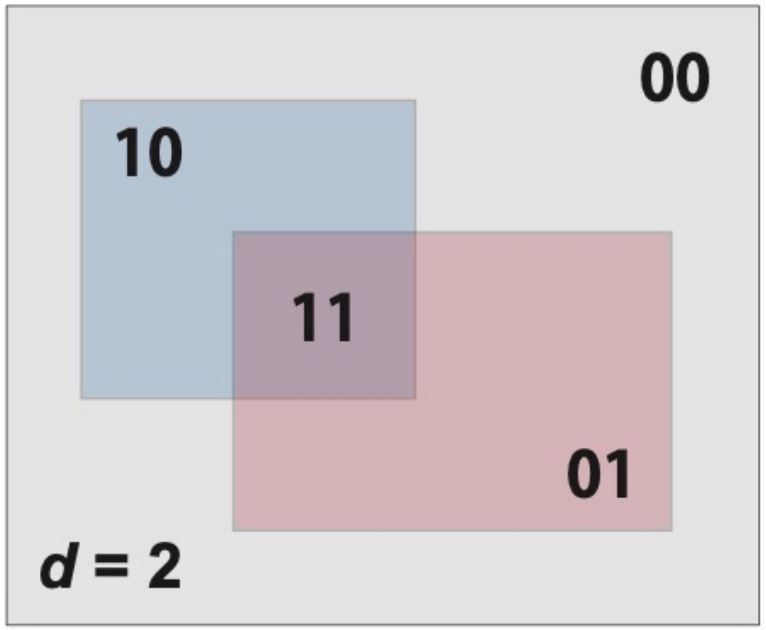
S. Izawa, K. Kitai, S. Tanaka, R. Tamura, and K. Tsuda, 4, 023062 (2022).

連続値をバイナリに変換

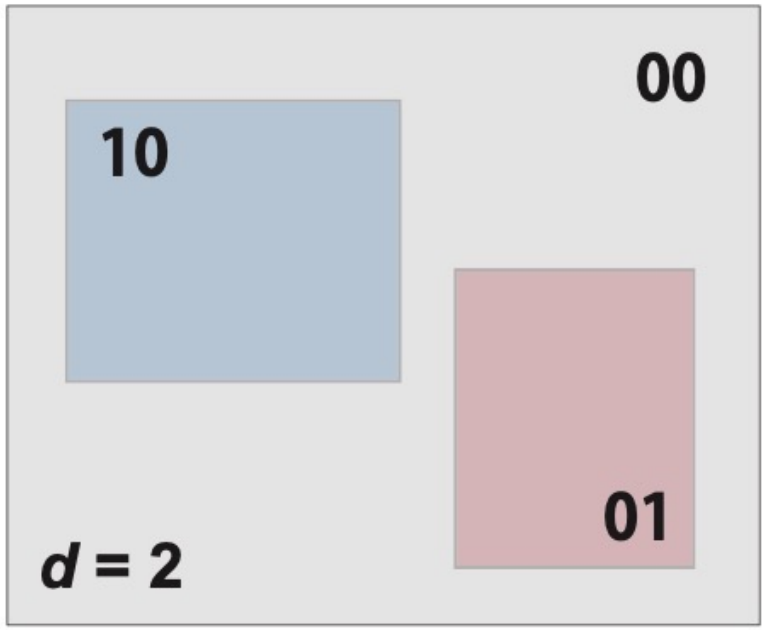
Random subspace codingを用いて変換

例：２次元連続変数を２ビットで表す





Surjective



Non-surjective

Non-surjectiveな場合

11となる領域は存在しない．
11が選ばれるとdecoding
できない．

Decodingできる解を
選ぶことが重要

Hyperrectangle(長方形)の数を増やすことで，resolutionが向上

回帰＆候補選択

学習データを集め，非負値の線形モデル

で回帰モデルを作成

$$y = \sum_{k=1}^{m} w_k z_k, \quad w_k \geq 0$$

バイナリ

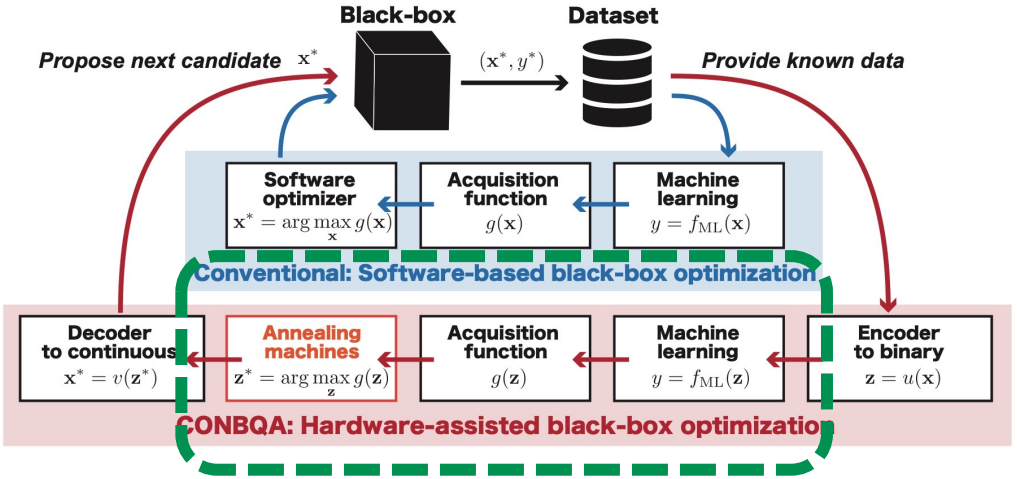
イジングマシンで，yが最大となるzを求める.

注：Decodingできる解を選ばないといけない.

Acquisition function

$$\min_{z \in \{0,1\}^m} -A \sum_{i=1}^{m} w_i^* z_i + B \sum_{\{(i,j)|R_i \cap R_j = \emptyset, i<j\}} z_i z_j$$

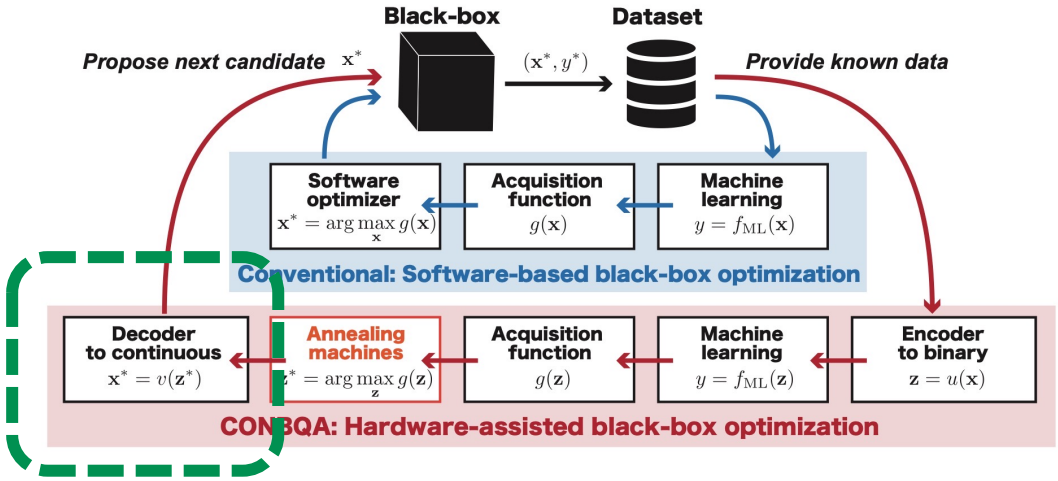
重なりがないhyperrectangle同士のビットが同時に１にならない制約

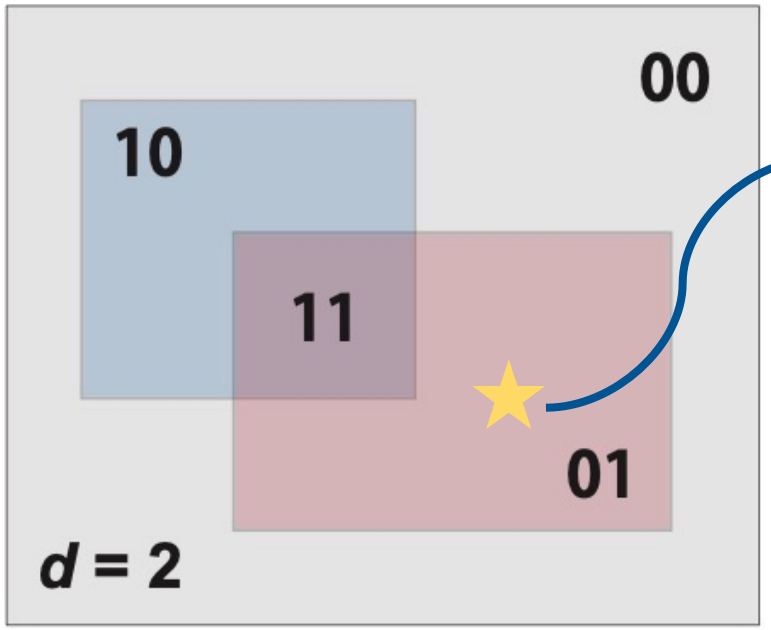


バイナリを連続値に変換

選ばれた領域の中心を利用する.

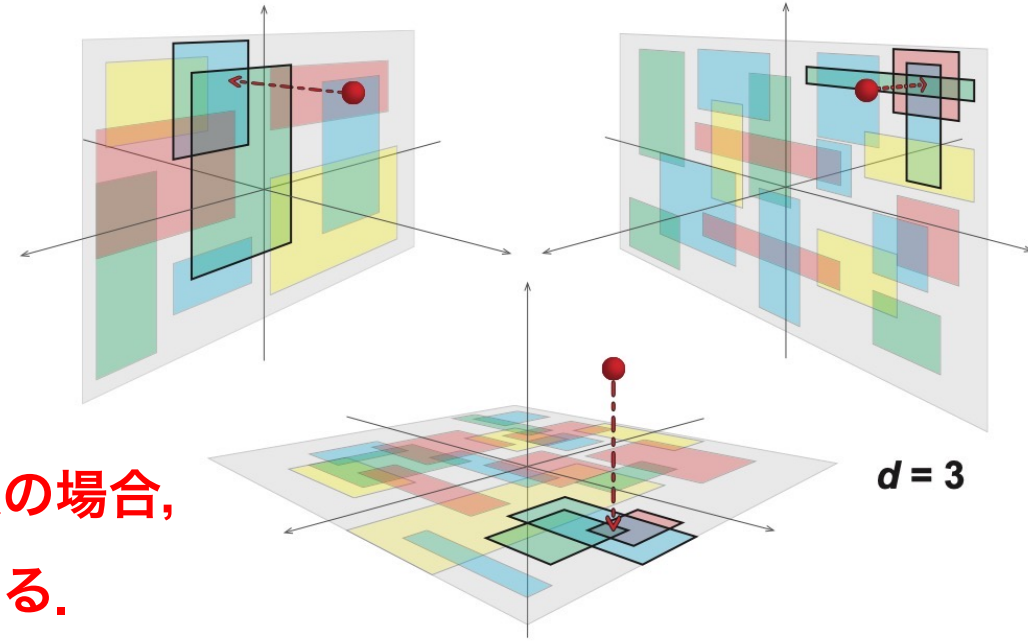
※ここは色々な方法がある.





実験・シミュレーションにより

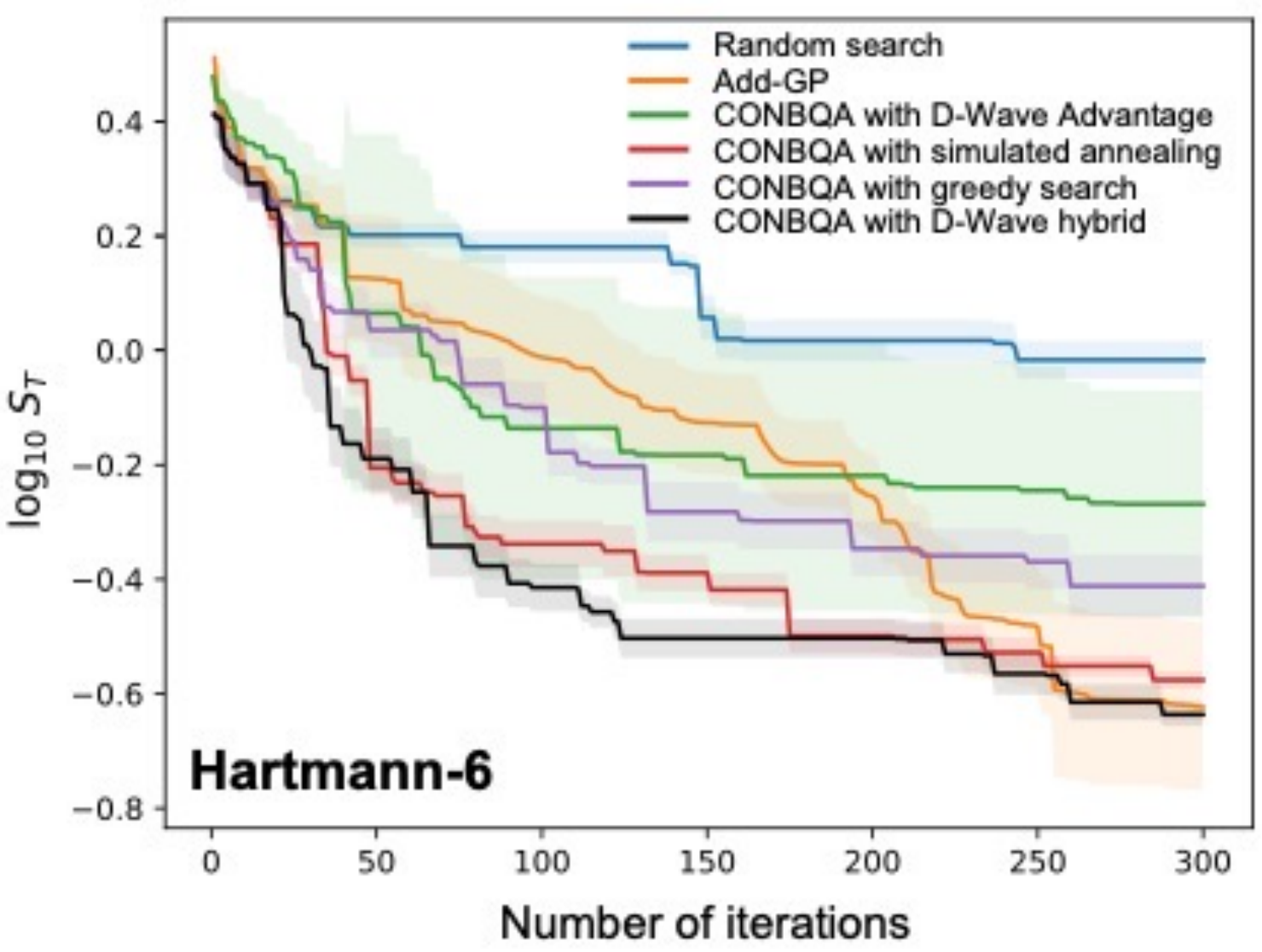
この点の対象yを評価する.



３次元連続変数の場合,

立方体になる.

テスト関数を利用したデモンストレーション

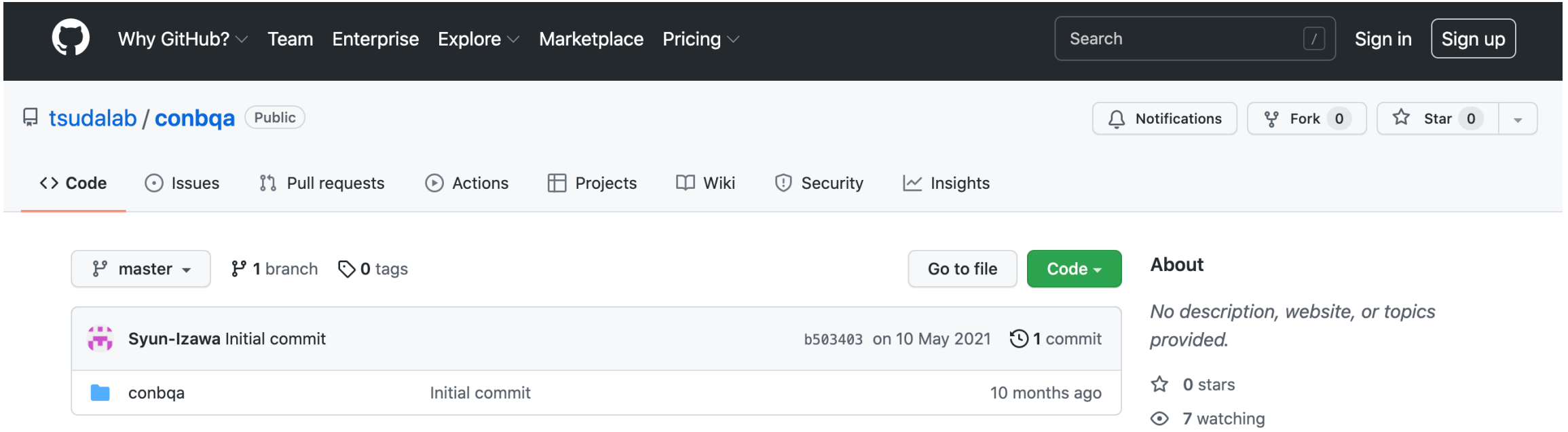


６次元連続変数問題

QUBO solver	Empty	Admissible	Decodable
D-Wave Advantage	44.4 %	54.4 %	1.2 %
Simulated annealing	0 %	12.1 %	87.9 %
Greedy search	0 %	3.6 %	96.4 %
D-Wave hybrid	0 %	53.5 %	46.5 %

CONBQAは良い
最適化性能を示す

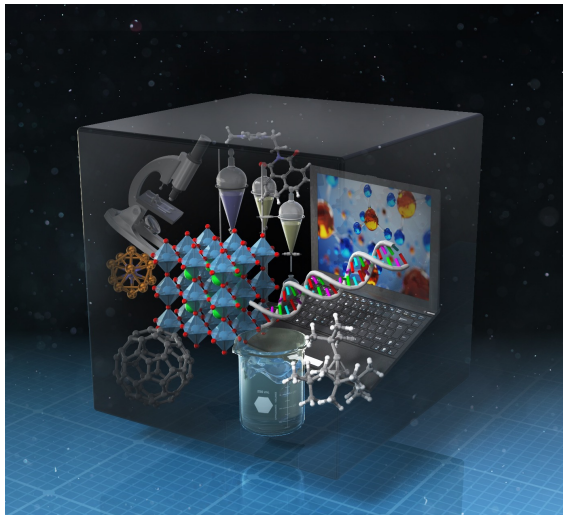
CONBQA package

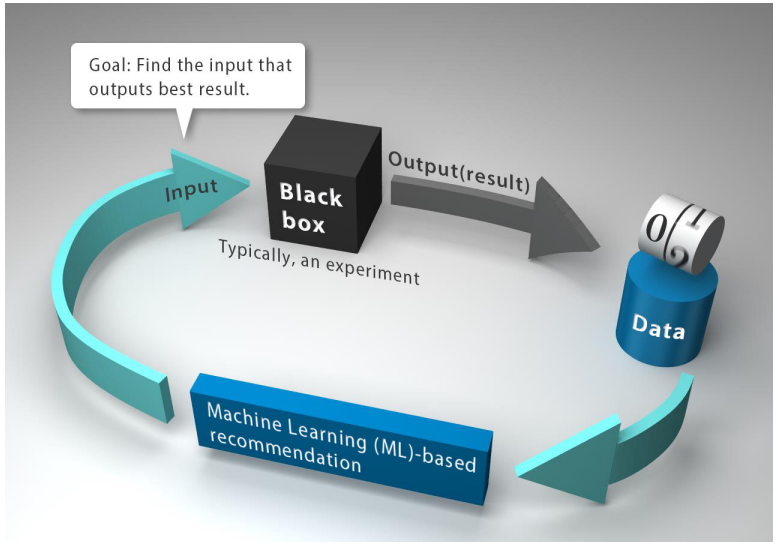


https://github.com/tsudalab/conbqa

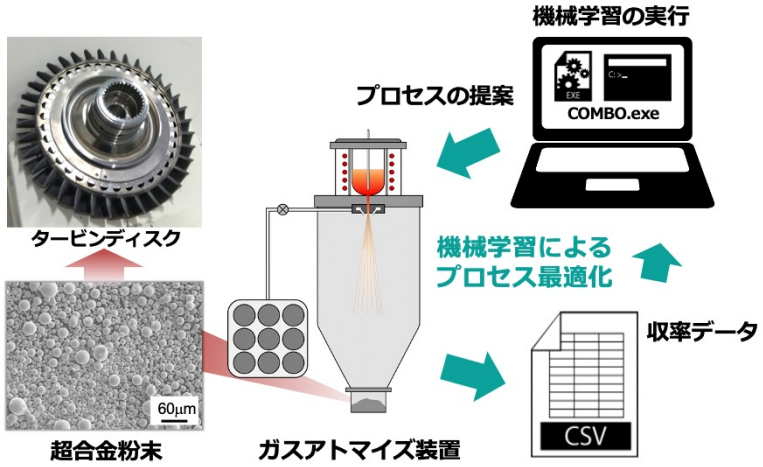
まとめ

ブラックボックス最適化を利用することで材料最適化が高速に実行可能

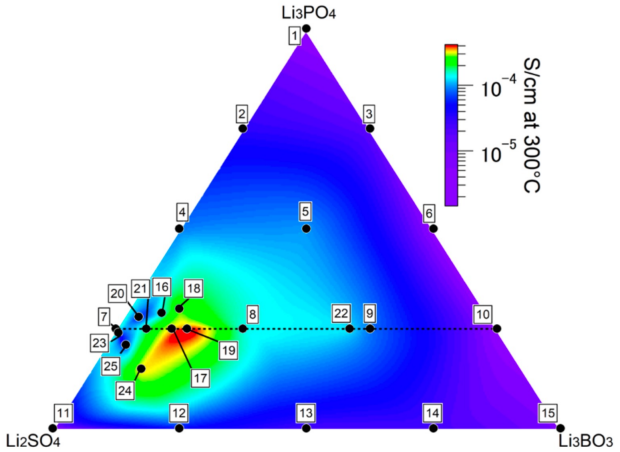




プロセス最適化



材料組成最適化



構造最適化