

量子コンピュータ時代の最適化セミナー

～ブラックボックス最適化の最新事例の紹介と技術解説～



本日の予定

- Fixstars Amplify の紹介
- ブラックボックス最適化とは
- QA-BBOの概要と活用例
- 技術解説
 - QA-BBOの躰きポイント
 - 多目的最適化、発展的な技術
 - Kernel-QA
 - Amplify-BBOpt
- まとめ

質問は随時、ZoomのQ&Aへお願いします

本セミナーのゴール

- QA活用のブラックボックス最適化手法（QA-BBO）に関する
 - 基本的な流れ
 - 躰きポイント
 - 高度なテクニック

について知っていただく

質問は随時、ZoomのQ&Aへお願いします

会社概要

- 株式会社 Fixstars Amplify

- 設立 2021年10月
- オフィス 東京都港区芝浦3-1-1 田町ステーションタワーN 28階
- 事業 最適化のための量子コンピューティング プラットフォーム事業



- 株主

- 株式会社 フィックスターズ(東証プライム市場)100%
 - ✓ ソリューション(受託)事業
 - ✓ ソフトウェア高速化プロフェッショナル集団
 - ✓ 日本で初めて D-Wave Systems 社と提携 (2017年)



JSTムーンショット事業

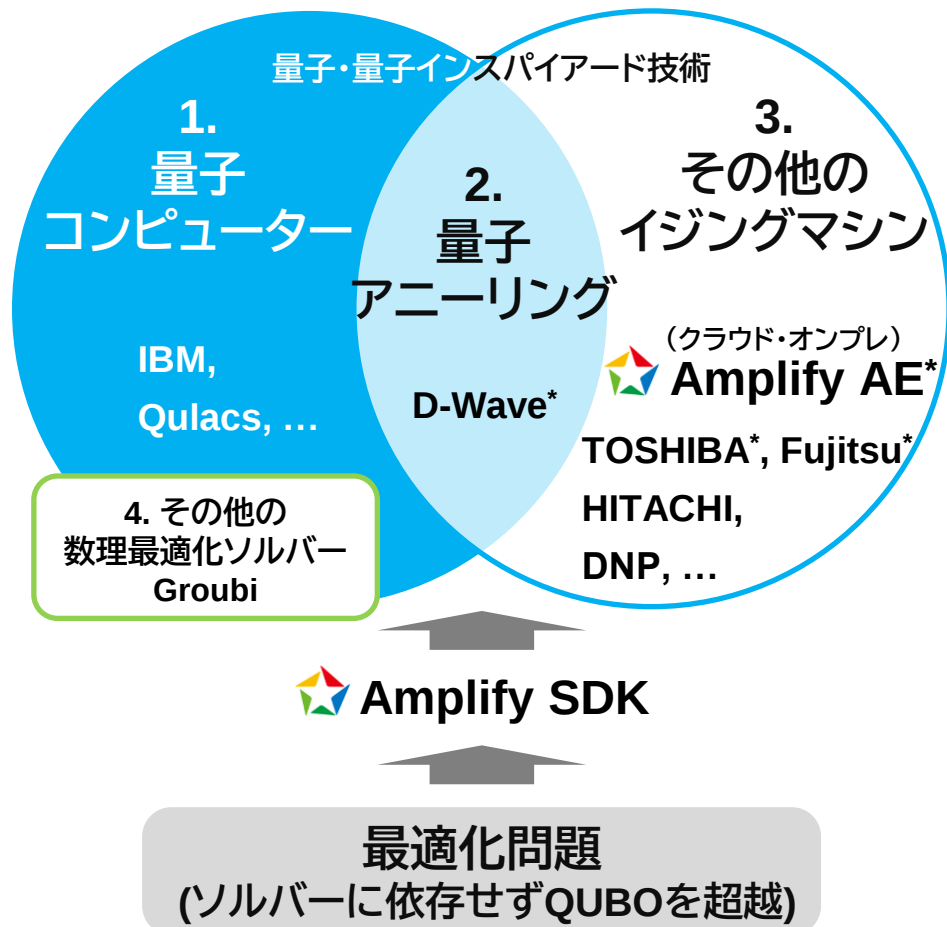
Fixstars Amplify の役割

- **開発環境 Amplify SDK** 簡単・最速な組合せ最適化アプリ実装 (Python)
- **実行環境 Amplify AE** GPUクラウド上で実行されるイジングマシン (リファレンスマシン)

全結合問題: 131,072 ビット

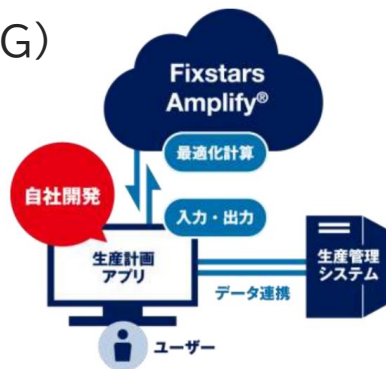
疎結合問題: 262,144 ビット

➔ 短期～長期を見据えた組合せ最適化の社会実装の実現 (無料で利用スタート可能)



活用領域とユースケース（PoC→実稼働）

- 生産計画
 - 多品種少量生産、保全計画、設備投資、在庫
- 従業員割り当て
 - 食品、輸送、製造
- エネマネ
 - エネルギーミックス、装置の運転制御
- 経路
 - 配送、船舶、無人搬送車（AVG）
- メディア
 - 最適広告配信
- 研究開発、設計
 - 材料設計
 - 物理シミュレーション



活用領域とユースケース（PoC→実稼働）

Amplify インタビュー

検索

- 生産計画
 - 多品種小量生産、保全計画、設備投資、在庫

- 従業員割

- 食品

- エネマネ

- エネルギーミックス、装置の運転制御

- 経路

-

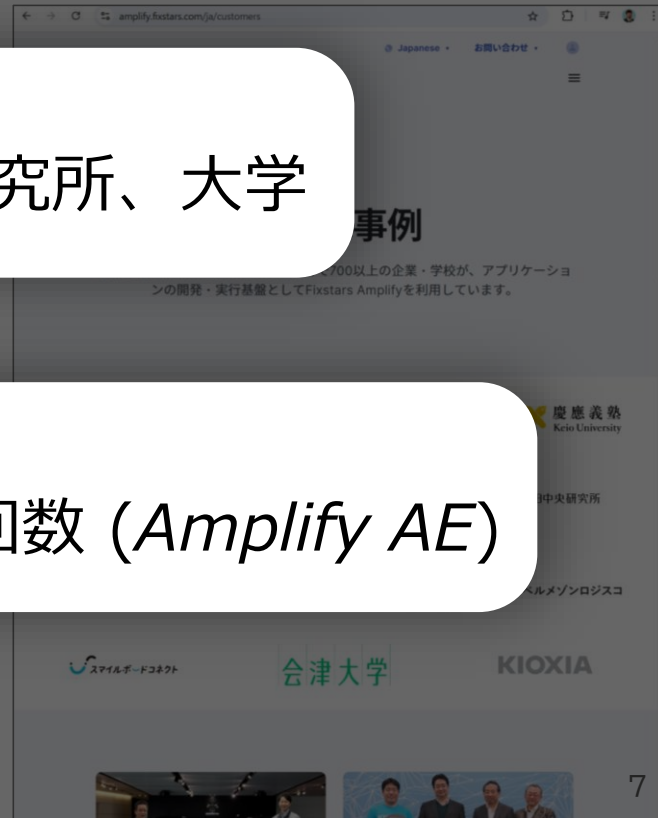
- 研究開発、設計

- 材料設計

- 物理シミュレーション

900を超える企業、研究所、大学

8,000万を超える実行回数 (Amplify AE)



QA活用のブラックボックス 最適化と活用事例

通常の数理最適化とブラックボックス最適化 (BBO)

通常の数理最適化

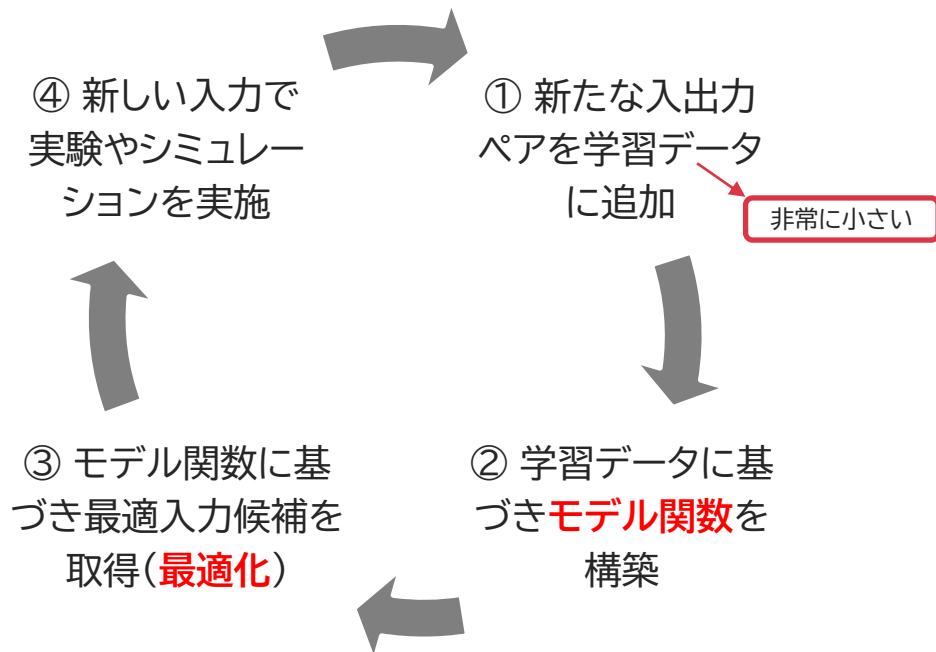
- 目的関数を定式化(例:2次2値形式)
 - 生産計画(メイクスパン最小)
$$f = \sum \sum \sum (P_i q_{t,m,p} + \sum S_{p,r} q_{t,m,p} q_{t,m,r}) \dots$$
 - 経路最適化(経路距離最小)
$$f = \sum \sum \sum d_{i,j} q_{n,i} q_{n+1,j} \dots$$
- 最適化の実施
 - イジングマシンにより定式化された目的関数を最小化

ブラックボックス最適化 (BBO)

- 直接の定式化が困難な目的関数
 - 低損失な流体デバイス形状？
 - 高性能な材料/構造トポロジー？
- 最適化の実施
 - 実験やシミュレーションの試行錯誤により、定式化不可な目的関数を最小化

BBOのフローとQA-BBO

● 逐次最適化：最適化サイクルの実施



● QA-BBO[†]

FMQA [Kitai, et al., Phys. Rev. Res. \(2020\)](#) (東大津田先生)

- **モデル関数** → FM ※
- **最適化** → QA[†]

Kernel-QA [Minamoto & Sakamoto, arXiv:2501.04225](#)

- **モデル関数** → Kernel model ※
- **最適化** → QA[†]



高次元の最適化問題に強い！

※ 2次多項式形式のモデル。バイナリ化により2次以上の関数形も表現可

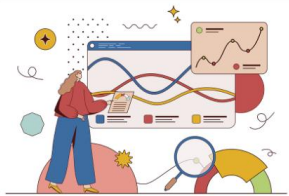
† QA: Quadratic-optimization Annealing

BOCSもQAベースだが、高次元向けとは必ずしも言えないので、取り上げない

QA-BBO: 活用例 (Amplify サンプルプログラム)

Amplify デモ

検索



チュートリアル応用編

ブラックボックス最適化 (1)

プログラミング難易度 ★★★★★

複雑で未知な目的関数にも適用可能な、機械学習と組み合わせ最適化を組み合わせたブラックボックス最適化手法を紹介し、Amplifyを用いて実装します。

サンプルコード

FMQA導入



チュートリアル応用編

ブラックボックス最適化 (2)

プログラミング難易度 ★★★★★

機械学習と量子アニーリング・イジングマシンを活用するブラックボックス最適化の適用例として、疑似的な高温超電導を実現する材料探索を取り扱います。

サンプルコード

材料最適化

FMQA
×
物理モデル



チュートリアル応用編

ブラックボックス最適化 (3)

プログラミング難易度 ★★★★★

化学プラントにおける生産量を最大化するための運転条件最適化を行います。最適化には、機械学習モデルに基づくブラックボックス最適化と化学反応に関する物理シミュレーションを用います。

サンプルコード

化学プラント
運転条件最適化

FMQA
×
化学シミュレーション



チュートリアル応用編

ブラックボックス最適化 (4)

プログラミング難易度 ★★★★★

流体機器設計に不可欠な翼型の最適化問題を取り上げます。最適化には、組み合わせ最適化及び機械学習に基づくブラックボックス最適化と流体シミュレーションを用い、翼の揚抗比を最大化するように翼型の探索を行います。

サンプルコード

翼形状最適化

FMQA
×
流体シミュレーション



チュートリアル応用編

ブラックボックス最適化 (5)

プログラミング難易度 ★★★★★

ブラックボックス最適化により、商業施設による交通集中が発生し得る都市における、交通渋滞を低減するような信号機群の最適制御を実施します。最適化の実施及び実証には、マルチ・エージェント・シミュレーションによる交通シミュレーションを用います。

サンプルコード

信号制御最適化

FMQA
×
マルチ・エージェント・シミュレーション

QA-BBO: 活用例 (Amplify ユーザー)

- 活用領域

- 化学、創薬、食品、自動車、電機、通信、重工、エネルギー、ヘルスケア・・・

非線形現象の逆
問題

機械学習：
コスト↓精度↑

設計開発におけ
る部品選定

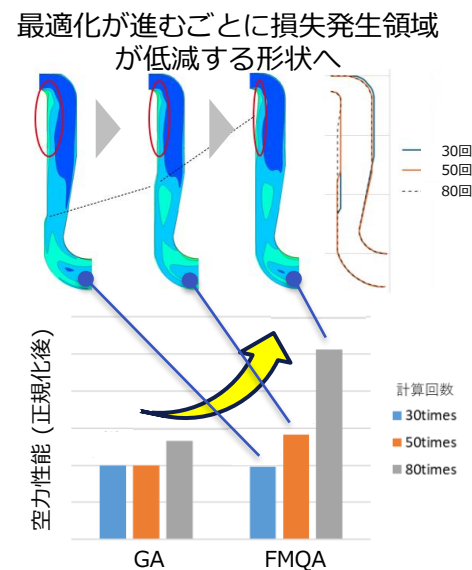
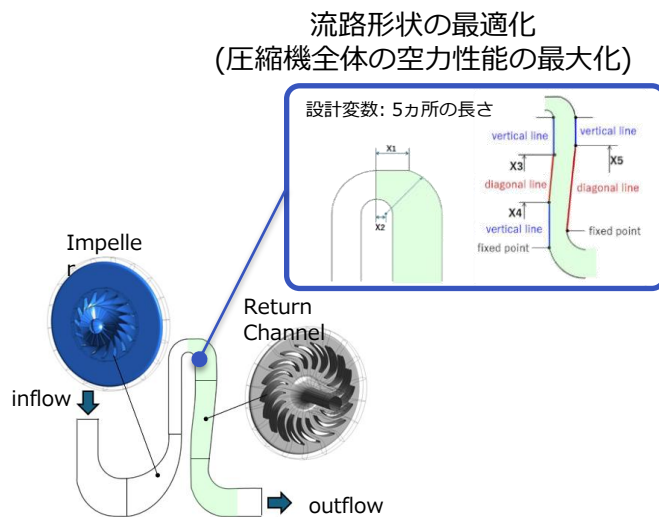
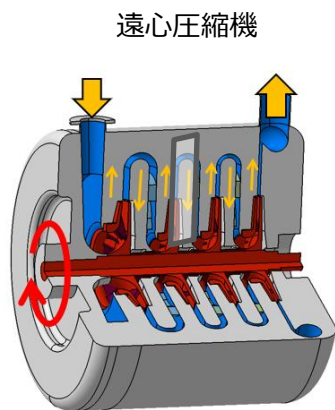
材料配合最適化

多目的最適化

物理モデルの
簡略化

事例: ターボ機械の形状最適化(川崎重工工業様)

- 従来より遺伝的アルゴリズム (GA) により形状最適化を行うことが多かった。規模が大きくなると求解までに時間がかかり、開発期間が長期化するという課題があった
- FMQA により、従来手法と比べ、**同じ計算回数でもより優れた解が得られる**ことを確認。



事例：車両設計最適化(マツダ様)

Amplify マツダ

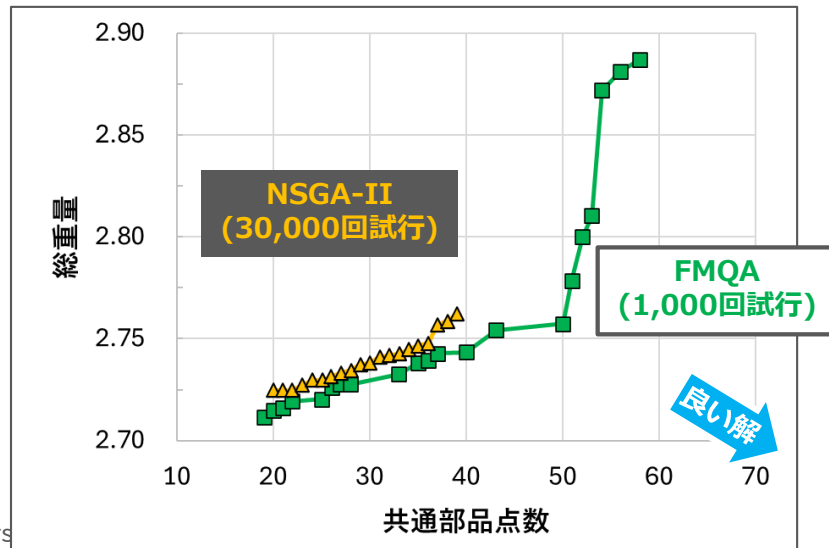
検索

- 車体設計の複数車種同時最適化問題。**実数変数200**以上の大規模問題。
- **車体の軽量化**と**共通部品数の最大化**の実現(衝突性能・製造、構造などの制約条件を満たした上で)
- 国内外研究グループ*1,2 により様々な手法が試されてきており、**1~3万回程度の試行**により、ある程度良い解が見つけれられることは確認されていた

- ▼
- FMQA により、**1,000回程度の試行**で、従来手法と同等以上の解を見つけることに成功！

*1 [進化計算コンペティション2017開催報告](#)

*2 [Multi-objective Bayesian optimization over high-dimensional search spaces](#)

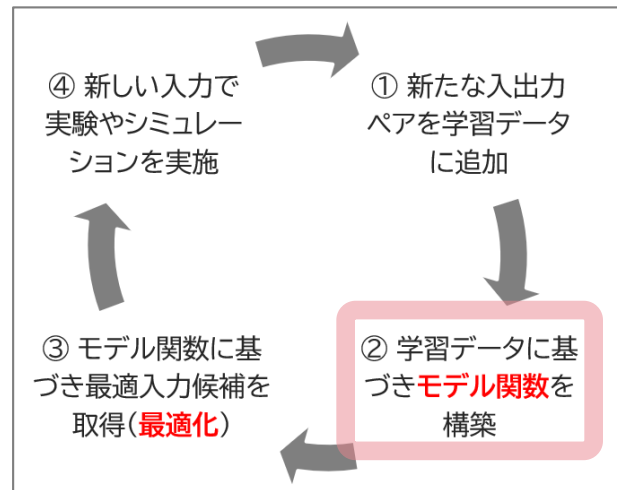


QA-BBO

- ・ 躰きポイント

QA-BBO 躓きポイント

- モデル関数の性能が鍵
(ベイズ最適化のように探索を考慮しないため)
 - 性能→✕精度
 - 性能→✓ブラックボックス目的関数との(順位)相関
- モデル関数構築の基礎となる学習データのスケールリング
 - ダイナミックレンジを補正する



精度の良いモデルよりも、最適化しやすいモデル構築が重要

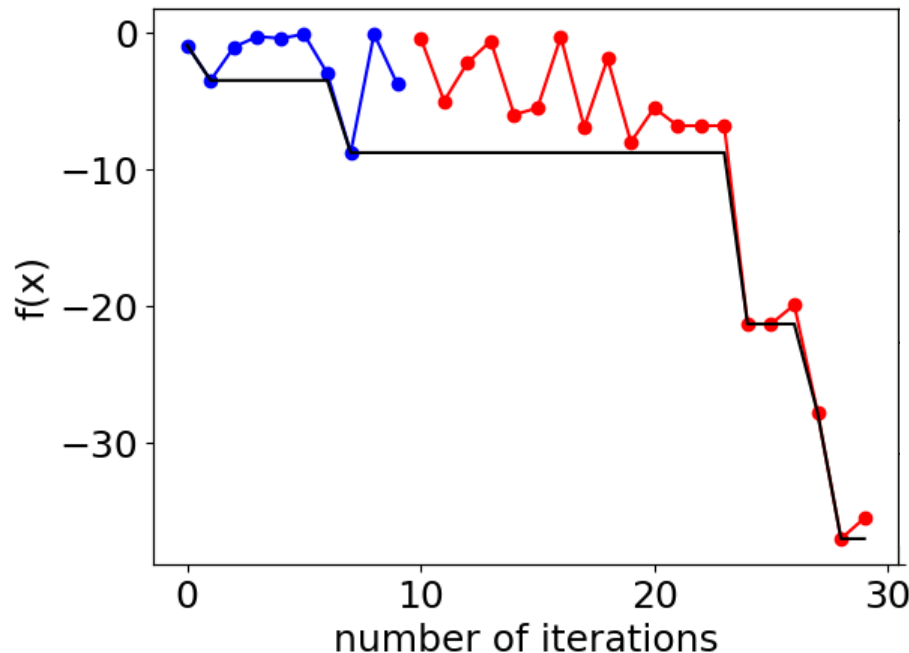
相関: 真値とモデル予測値との相関係数

```
def train(  
    x: np.ndarray,  
    y: np.ndarray,  
    model: TorchFM,  
) -> None:  
    """FM モデルの学習を行う"""
```

```
# イテレーション数  
epochs = 2000
```

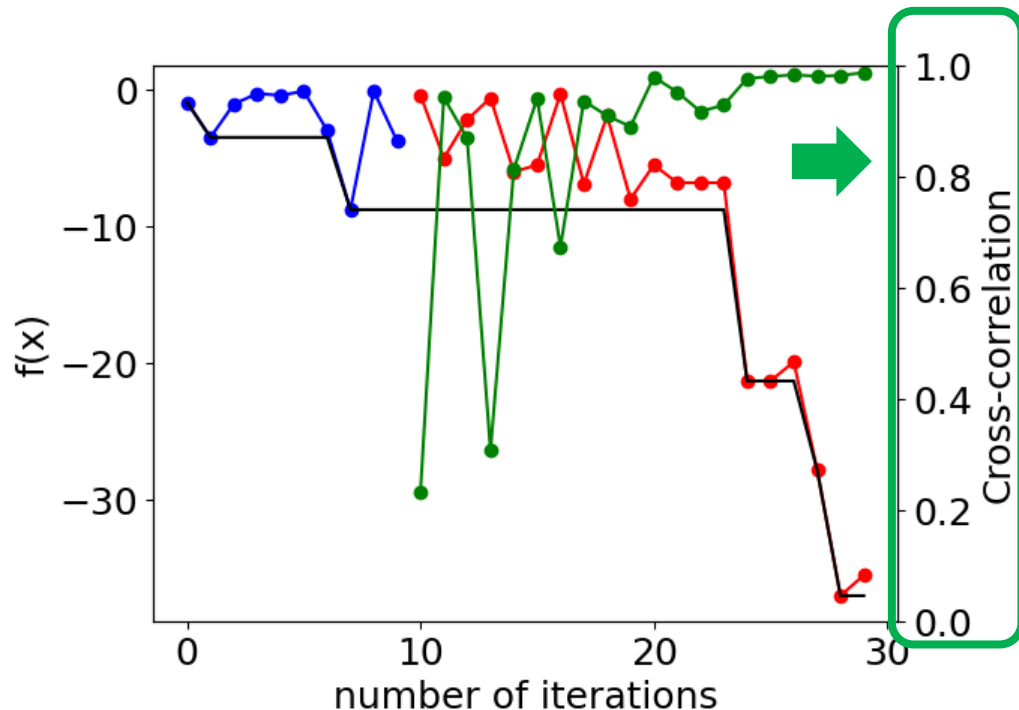
```
# . . .  
# 中略  
# . . .
```

```
# モデルを学習済みパラメータで更新  
model.load_state_dict(best_state)
```



相関: 真値とモデル予測値との相関係数

```
def train(  
    x: np.ndarray,  
    y: np.ndarray,  
    model: TorchFM,  
    ) -> None:  
    """FM モデルの学習を行う"""  
  
    # イテレーション数  
    epochs = 2000  
  
    # . . .  
    # 中略  
    # . . .  
  
    # モデルを学習済みパラメータで更新  
    model.load_state_dict(best_state)  
  
    # 真値とモデル予測値の間の相関係数を表示  
    y_pred = model(x_tensor)  
    yy = torch.stack([y_tensor, y_pred])  
    print(f"corrcoef: {torch.corrcoef(yy)[0,  
1]})")
```



例: FMQAの場合

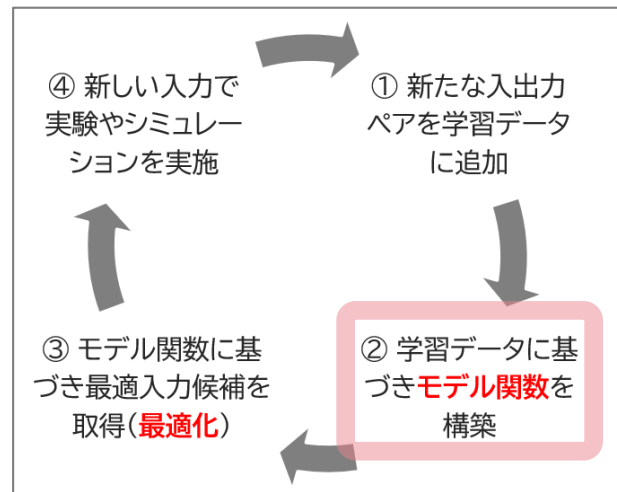
K. Kitai, J. Guo, S. Ju, S. Tanaka, K. Tsuda, J. Shiomi, & R. Tamura
"Designing metamaterials with quantum annealing and factorization machines"
Phys. Rev. (2020).

- Factorization machine (FM)

$$g(x|\mathbf{w}, \mathbf{v}) = w_0 + \sum_{i=1}^d w_i x_i + \frac{1}{2} \sum_{f=1}^k \left(\left(\sum_{i=1}^d v_{if} x_i \right)^2 - \sum_{i=1}^d v_{if}^2 x_i^2 \right)$$

- ハイパーパラメータ k

- $k = d$ の時、 $g(x)$ は全ての相互作用項を考慮。
- BBOでは、比較的小さな k を利用(過学習を避ける)
 - ✓ 経験上、 d や $f(x)$ の複雑さに関わらず、 $k = 10$ で良い
- 学習曲線を(時々)確認→相関係数とのセットで学習の妥当性を把握



QA-BBO

- ・ 発展的な技術

多目的BBOにおける発展的な技術

近藤 俊樹, 小平 剛央, 源 勇気, FMQA を用いた複数車種の車体構造同時設計最適化, 56 巻 2 号 p. 229-236 (2025).

事例: 車両設計最適化(マツダ様)

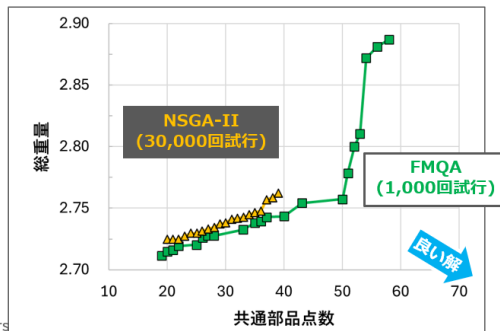
Amplify マツダ 検索

- 車体設計の複数車種同時最適化問題。実数変数200以上の大規模問題。
- 車体の軽量化と共通部品数の最大化の実現(衝突性能・製造制約などの制約を満たした上で)
- 国内外研究グループ^{*1,2} により様々な手法が試されてきており、1~3万回程度の試行により、ある程度良い解が見つけれられることは確認されていた

- ▼
- FMQA により、1,000回程度の試行で、従来手法と同等以上の解を見つけることに成功！

^{*1} 進化計算コンペティション2017開催報告

^{*2} [Multi-objective Bayesian optimization over high-dimensional search spaces](#)



車両設計最適化(詳細)

- <https://ladse.eng.isas.jaxa.jp/benchmark/>
- 目的関数: 2個(多目的最適化問題)
 - 3車種合計重量
 - 共通部品点数
- 制約条件: 53個
 - 衝突性能、固有値、車体剛性、構造制約、製造制約・・・
 - 内、**ブラックボックスな制約条件**: 42個
- 実数決定変数: 222個(バイナリ変数換算: 1905個)
- ブラックボックス関数の評価1,000回

Benchmark Problem Based on Real-World Car Structure Design Optimization Mazda Benchmark Problem

August 1, 2018

Design problems in real world often have many constraints. Design optimization methods that can efficiently obtain an optimal design or Pareto-optimal designs for such problems are needed. Evolutionary algorithms can efficiently obtain optimal design(s) for such problems and have been studied using benchmark problems such as CDTLZ problems and real-world-like problems. However, Tanabe et al showed that those benchmark problems have inappropriate features [1]. For example, most of infeasible designs violate only one constraints.

For this reason, we decided to provide simultaneous optimization problem of multiple car models as a benchmark problem under a joint research by Mazda Motor Corporation, Japan Aerospace Exploration Agency, and Tokyo University of Science. This problem was solved on "K computer" [2].

Here, total weight of three types of cars is minimized while number of common thickness parts among the three types of cars is maximized. Design parameters are thickness of 222 structural parts. This problem has 54 constraint functions such as collision safety performances. This benchmark problem can be used as a single-objective design optimization benchmark problem (weight minimization problem) or multiobjective design optimization benchmark problem (weight minimization and number of common parts maximization).

In the optimization performed on K computer, structural simulation software LS-DYNA was used to evaluate the constraints on collision safety [3]. In the benchmark problem, constraints on collision safety are modeled by response surface approximation. Please refer to reference [4] for details. Sample results are presented in [5].

This benchmark problem is released as C++ source code. Thus, it should work on any operating system. You can compile it on your own environment to use it. Executable file that is checked on windows 10 machine (64 bit version, Visual Studio 2015/2017) is also included.

We expect that this benchmark problem contributes to research and development of a robust and efficient design optimization algorithm for real-world problems.

References

- [1] Ryoji Tanabe and Akira Oyama, "A Note on Constrained Multi-Objective Optimization Benchmark Problems," 2017 IEEE Conference on Evolutionary Computation, 2017.
- [2] What is K?
- [3] Akira Oyama, Takehisa Kohira, Hiromasa Kemmotsu, Tomoaki Tatsukawa, and Takeshi Watanabe, "Simultaneous structure design optimization of multiple car models using the K computer," 2017 IEEE Symposium Series on Computational Intelligence, 2017.
- [4] Takehisa Kohira, Hiromasa Kemmotsu, Akira Oyama, and Tomoaki Tatsukawa, "Proposal of Benchmark Problem Based on Real-World Car Structure Design Optimization," The Genetic and Evolutionary Computation Conference (GECCO) 2018, 2018.
- [5] Hiroaki Fukumoto and Akira Oyama, "Benchmarking Multiobjective Evolutionary Algorithms and Constraint Handling Techniques on a Real-World Car Structure Design Optimization Benchmark Problem," The Genetic and Evolutionary Computation Conference (GECCO) 2018, 2018.

How to download the benchmark problem

Download the benchmark problem from [here](https://ladse.eng.isas.jaxa.jp/benchmark/). We appreciate it if you provide your name and affiliation to benchmark@flab.isas.jaxa.jp after you download. The provided information may be used for update notices and statistical analysis.

When you publish results

We appreciate it if you refer to the reference [4] in your publications.
We also appreciate it if you send the publication to benchmark@flab.isas.jaxa.jp

ブラックボックスな制約条件とは？

- 必ず満たさなければならない品質特性

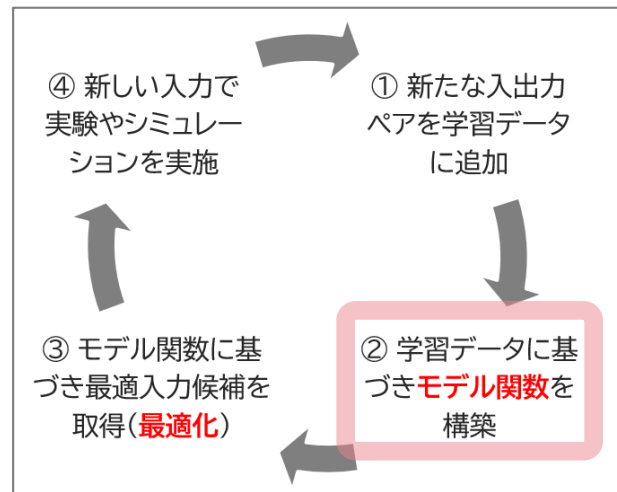
- 衝突性能、固有値、車体剛性、構造制約、製造制約・・・

- 各部品の鋼板厚さの多項式で充足を判定できない！

- これら個々の品質特性 j に関するシミュレーション：

$$g_j(x) \begin{cases} \geq 0 \cdots \text{制約 } j \text{ を満足} \\ < 0 \cdots \text{制約 } j \text{ の違反} \end{cases}$$

- これらブラックボックスな制約のパナルティを荷重和した出力に対するモデル関数を構築



ブラックボックスな制約条件とは？

- 最適化の対象となる最終的な目的関数

- 目的関数(2つ)
- ブラックボックスな制約ペナルティに対するモデル関数
- 定式化可能な制約条件ペナルティ

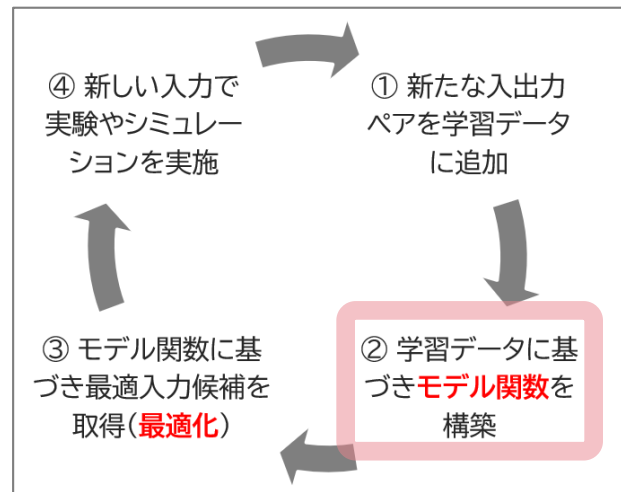
上記、多項式の線形結合モデルを構築

→ 最適化(アニーリング)

- 発展的な工夫ポイント

- 各項の重みの動的決定法
- ブラックボックス関数の並列評価

✓ 並列評価により、1000サイクル→333サイクルに！



QA-BBOの今後の展開:

- Kernel-QA
- Amplify-BBOpt

QA活用BBOのアップデート

背景

- FMQA は高次問題において高性能。しかし、
 - FM 学習が計算コスト大
 - FM 学習パラメータの調整が困難な場合有
- QA-BBO ユーザー: **より簡単でシンプルなモデル関数を使いたい！**



Kernel-QA

(Minamoto & Sakamoto, arXiv:2501.04225)

Kernel-QA:期待されること

- より高速なモデル構築
 - モデル関数は**解析的**に構築
 - ⇔ FMIは機械学習で構築
- より最適なモデル関数
 - **全体最適なモデル係数**を保証する数学的手法
 - ⇔ 機械学習で用いられる確率的勾配降下法は局所最適に陥る可能性
- 高速な最適化
 - FMQAと同様に**QAを活用**可能 ☺
 - ⇔ 通常のバイズ最適化手法

デメリット:実装が大変 ☹

Amplify-BBOpt:QA-BBOをより簡単に！ (FMQA, Kernel-QA)

- *Fixstars Amplify* の機能をフル活用:
 - Amplify SDK
 - ✓ 種々の変数やソルバーの利用
 - ✓ ソフト・ハード制約条件の考慮
 - Amplify AE
 - ✓ 高速・大規模な最適化
 - ✓ 数1,000 のバイナリ決定変数 (より大規模でもBBO可能)
 - ✓ 数100 の実数・離散決定変数 (より大規模でもBBO可能)

Amplify-BBOpt: プログラム例 overview

←250行を超える実装
が
30 行の実装に→



```
from amplify bbopt import BinaryVariableList, blackbox, DatasetGenerator, FMQAOptimizer, plot_history
import amplify
from datetime import timedelta

num_materials = 100 # Number of decision variables

my_func = make_blackbox_func(num_materials)

# Define black-box function and decision variables in the Amplify-BBOpt manner
@blackbox
def blackbox_func(x: list[bool] = BinaryVariableList(length=num_materials)):
    return my_func(np.array(x))

# Set solver client
client = amplify.FixstarsClient()
client.parameters.timeout = timedelta(milliseconds=2000) # 2 seconds
client.token = token

# Construct initial training data (10 samples)
data = DatasetGenerator(objective=blackbox_func).generate(num_samples=10)

# Instantiate the optimizer
optimizer = FMQAOptimizer(data=data, objective=blackbox_func, client=client)

# Perform optimization cycle for [num_cycles] times
optimizer.optimize(num_cycles=100)

# Plot optimization history
plot_history(optimizer.fetch_history())
```

まとめ

● QA活用ブラックボックス最適化に関する

- 最近の事例
- 躰きポイント
- 最新の技術
- 今後の展開

などをご紹介

● 今後について

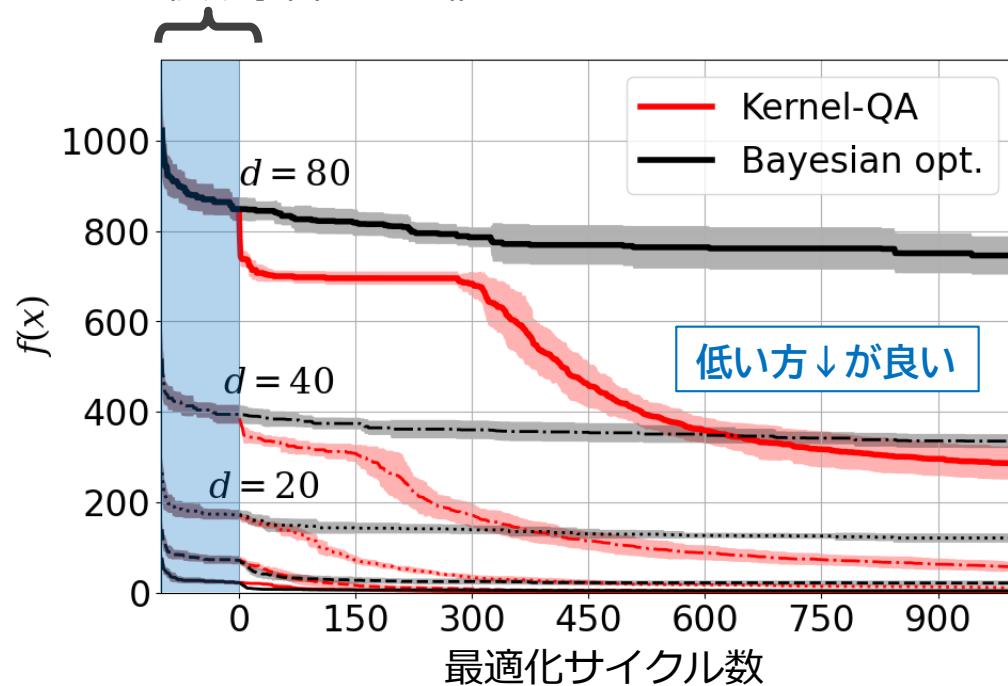
- デモ・チュートリアルを試す
 - <https://amplify.fixstars.com/ja/demo#blackbox>
 - 無料で利用スタート可能！
- 打ち合わせにて、より詳細な
 - Kernel-QA
 - Amplify-BBOptに関する情報を取得する
- 公開セミナーやプライベートトレーニングを受講する
- ご質問などはお気軽にお問合せください

ご参加ありがとうございます

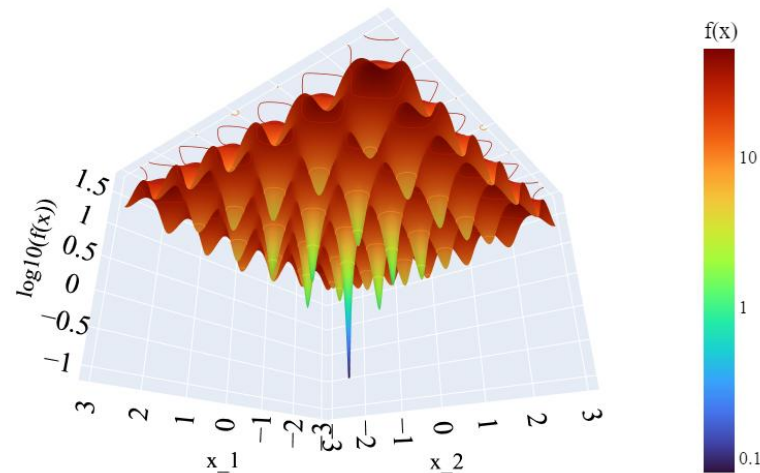
アンケートのご回答、よろしくお願いいたします。

Kernel-QA: パフォーマンス (実数変数)

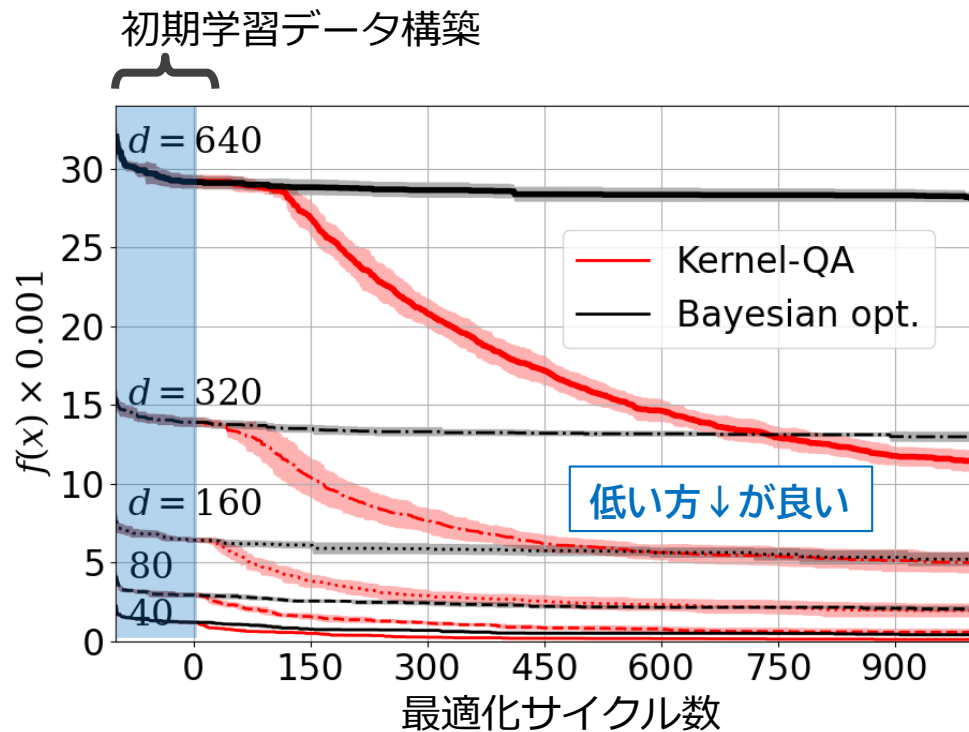
初期学習データ構築



Rastrigin 関数 (2D 可視化)
真の最適解 $x = [0, 0, \dots, 0]$, $f(x) = 0$



Kernel-QA: パフォーマンス (バイナリ変数)



Rastrigin 関数 (2D 可視化)
真の最適解 $x = [0, 0, \dots, 0]$, $f(x) = 0$

