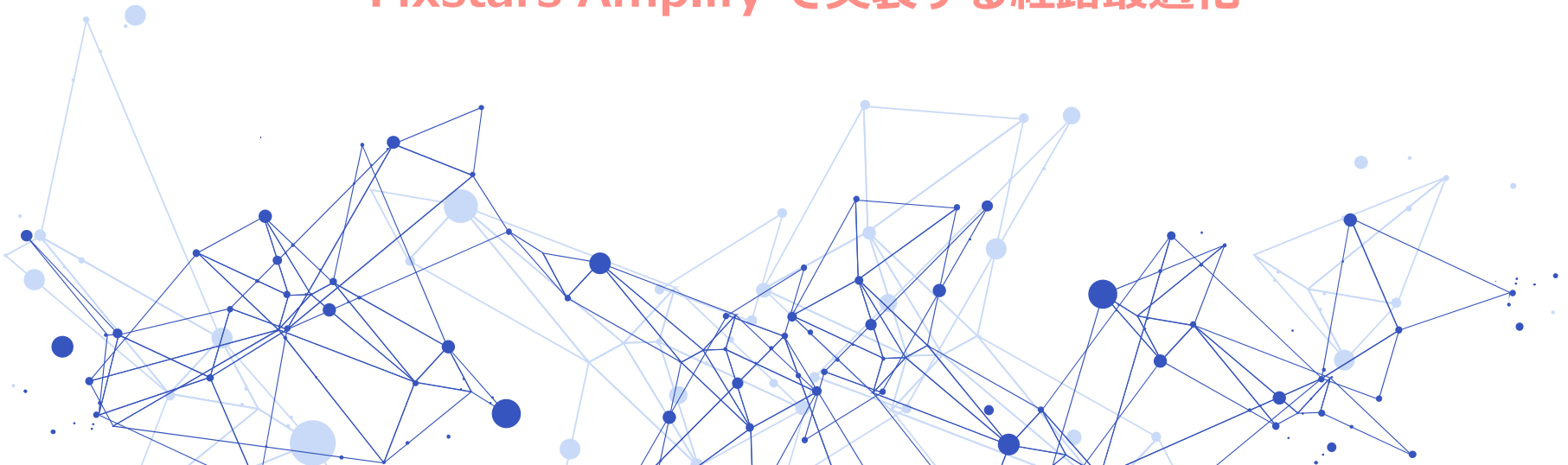
FIXSTARS Amplify

量子コンピュータ時代のプログラミングセミナー

～ Fixstars Amplify で実装する経路最適化 ～



本日のAgenda

- 本セミナーのゴール
- 会社および「Fixstars Amplify」のご紹介
- Fixstars Amplify を用いたワークショップ
  - 経路最適化
- 事例や今後の進め方等のご紹介
- Wrap Up & QA

質問は随時ZoomのチャットかQ&Aでお願いします

本セミナーのゴール

- 身の回りには組合せ最適化問題がたくさんあることを知る
- 組合せ最適化問題を解くための専用マシン（量子アニーリング・イジングマシン）があることを知り、解くためための流れを理解する（決定変数、目的関数、制約条件など）
- ワークショップを通して、実際にイジングマシンを動かしてみることで、実問題への適用の足掛かりを得る

会社紹介

(株) Fixstars Amplify の紹介

- **組合せ最適化**のための量子コンピューティングクラウドプラットフォーム「Fixstars Amplify」の提供

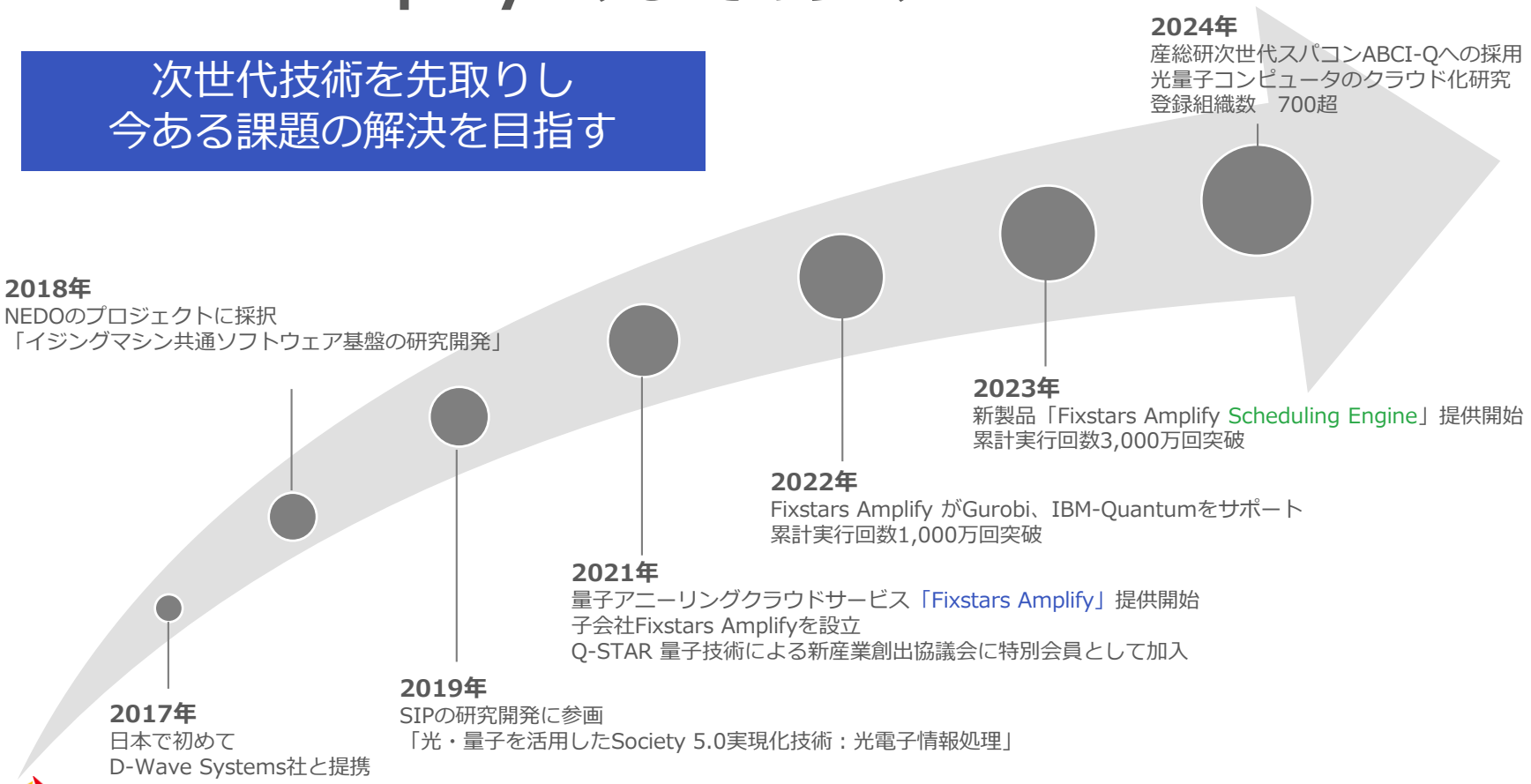
- 2021年に設立（株式会社フィックスターズからスピンアウト）
  - 代表取締役社長CEO：平岡 卓爾
  - 取締役CTO：松田 佳希（博士）

- 親会社 (株) フィックスターズ
  - 東証プライム市場上場
  - 約300名の従業員のうち、約9割がソフトウェアエンジニア
  - ソフトウェア高速化プロフェッショナル集団

FIXSTARS Amplify

Fixstars Amplify: 今までの歩み

次世代技術を先取りし
今ある課題の解決を目指す



Fixstars Amplify: 製品ポートフォリオ

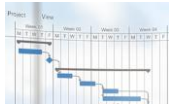
Fixstars Amplify
(量子コンピューティングクラウド, 2021/2～)

- **<u>汎用的な問題</u>**に対応する SDK と実行環境 (AE)
- 2次の QUBO 問題が得意
- シフト最適化、経路最適化、設計変数最適化向けに自分で**<u>自由に定式化</u>**した目的関数や制約条件の中で一番良い組み合わせを高速・高精度に探索
- 量子アニーリング・イジングマシンを活用してブラックボックス最適化問題を解くことも可能

本日のセミナーはこちら

Fixstars Amplify Scheduling Engine
(スケジューリング最適化クラウド, 2023/9～)

- **<u>スケジューリング問題に特化</u>**した SDK と実行環境 (Scheduling Engine)
- 最適化の目的を Makespan の最小化に絞り、**<u>定式化不要</u>**で、複雑な制約条件を通常のプログラミングの延長で実装可能
- 生産計画・人員シフト計画・配送計画など幅広いスケジューリングに適用可能



様々な領域での利用拡大中（実稼働含む）

900 を超える企業、研究所、大学

7,000万 を超える実行回数 (Amplify AE)

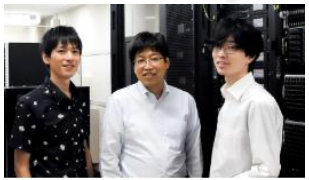
Fixstars Amplify のご紹介

組合せ最適化問題

- 組合せ最適化問題とは、膨大な選択肢の中から、制約条件を満たし、ある評価値（目的関数値）が最もよくなる（最小 or 最大）解を求めること。量子アニーリング・イジングマシンは組合せ最適化問題を解くための専用マシン
- 組合せ最適化問題は、生産計画最適化や、シフト最適化、構造物の設計最適化、材料開発（MI）、エネルギーマネージメント最適化など、様々な領域に存在し、研究・開発や社会実装が積極的に進められている

Fixstars Amplify の活用事例

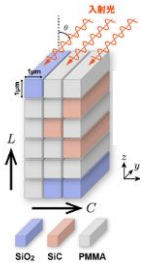
生産計画最適化 / 材料開発（MI） / 経路最適化 / 人員シフト最適化 / 機械学習の特徴量抽出 / 実験のパラメータ最適化 / 設計最適化 / エネルギーマネジメント最適化 / 電力最適化



ユーザーインタビュー　2023年8月

東京大学

ブラックボックス最適化手法（FMQA）の開発に成功



ユーザーインタビュー　2023年6月

住友商事株式会社

現場で愛されて育つ、量子コンピュータ技術活用

ユーザーインタビュー　2024年10月

マツダ株式会社

ブラックボックス最適化を活用した車両設計最適化

FIXSTARS Amplify

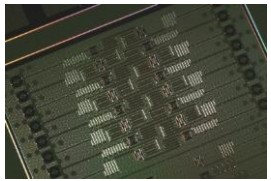
組合せ最適化問題、量子技術、Fixstars Amplify の関係

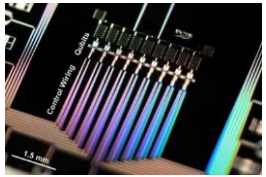
1. 量子コンピュータ

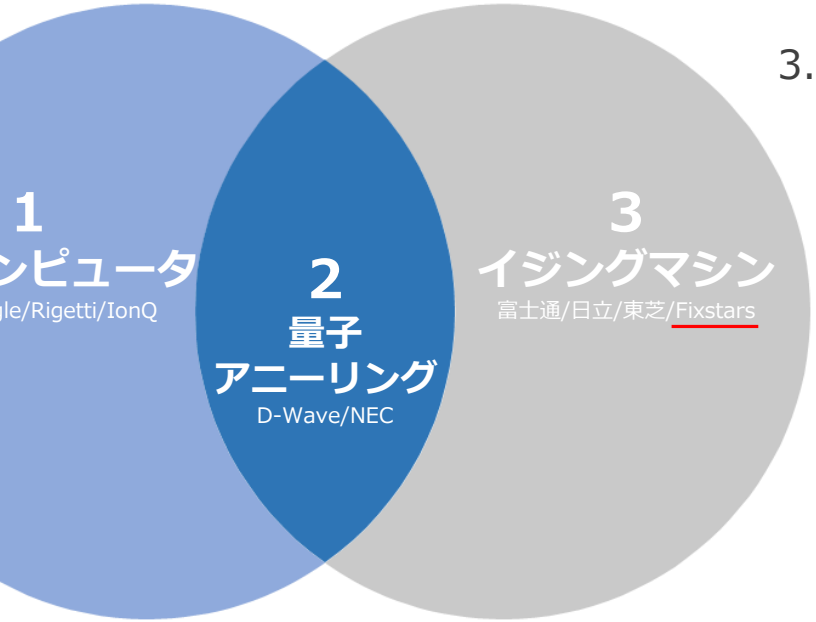
量子ゲート方式

古典汎用コンピュータの上位互換。量子力学の重ね合わせ状態を制御する量子ゲートを操作し、特定の問題を汎用的かつ高速に処理する。

QAOAにより組合せ最適化問題(**QUBO**)を取り扱うことが可能。







3. イジングマシン

二値二次多項式模型

二次の多変数多項式で表される目的関数の組合せ最適化問題（**QUBO**）を扱う専用マシン。

変数は0,1または±1。統計物理学におけるイジング模型（磁性体の性質を表す模型）に由来。様々な実装により実現されている。

2. 量子アニーリング方式

量子焼きなまし法

イジングマシンの一種であり、量子焼きなまし法の原理に基づいて動作する。量子イジング模型を物理的に搭載したプロセッサで実現する。自然計算により低エネルギー状態が出力される。組合せ最適化問題 (**QUBO**) を扱う専用マシン。

Amplify AE

FIXSTARS Amplify

組合せ最適化問題 (QUBO)

数理最適化問題

- 連続最適化問題
  - 決定変数が連続値（実数など）
- 決定変数が離散値 (整数など)
  - 整数計画問題 (決定変数が整数)
  - 0-1整数計画問題 (決定変数が二値)

量子アニーリング・イジングマシン

Quadratic	二次形
Unconstrained	制約条件なし
Binary	0-1整数 (二値)
Optimization	計画（最適化）

QUBO目的関数 (0-1整数二次計画問題)

$$f(\boldsymbol{q}) = \sum_{i<j} Q_{ij} q_i q_j + \sum_i Q_{ii} q_i$$

$f(\boldsymbol{q})$を最小化するような $\boldsymbol{q}$ を求める

クラウドサービス：**Fixstars Amplify**

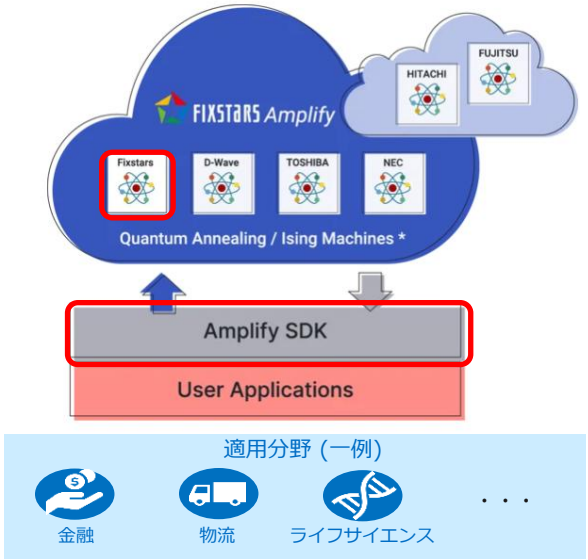
f: 目的関数　q: 決定変数　Q: 係数

量子コンピューティングクラウド: Fixstars Amplify

- 量子コンピューティングを想定したシステム開発・運用のクラウドプラットフォーム
- 量子コンピュータやGPUベースのイジングマシンなど、組合せ最適化問題の専用マシンを効率的に実行できる

https://amplify.fixstars.com/ja/

サービス概要



簡単

- SDKをインストールするだけですぐに使える (pip install amplify)
- ハードウェアの専門知識不要でアプリケーションが開発できる

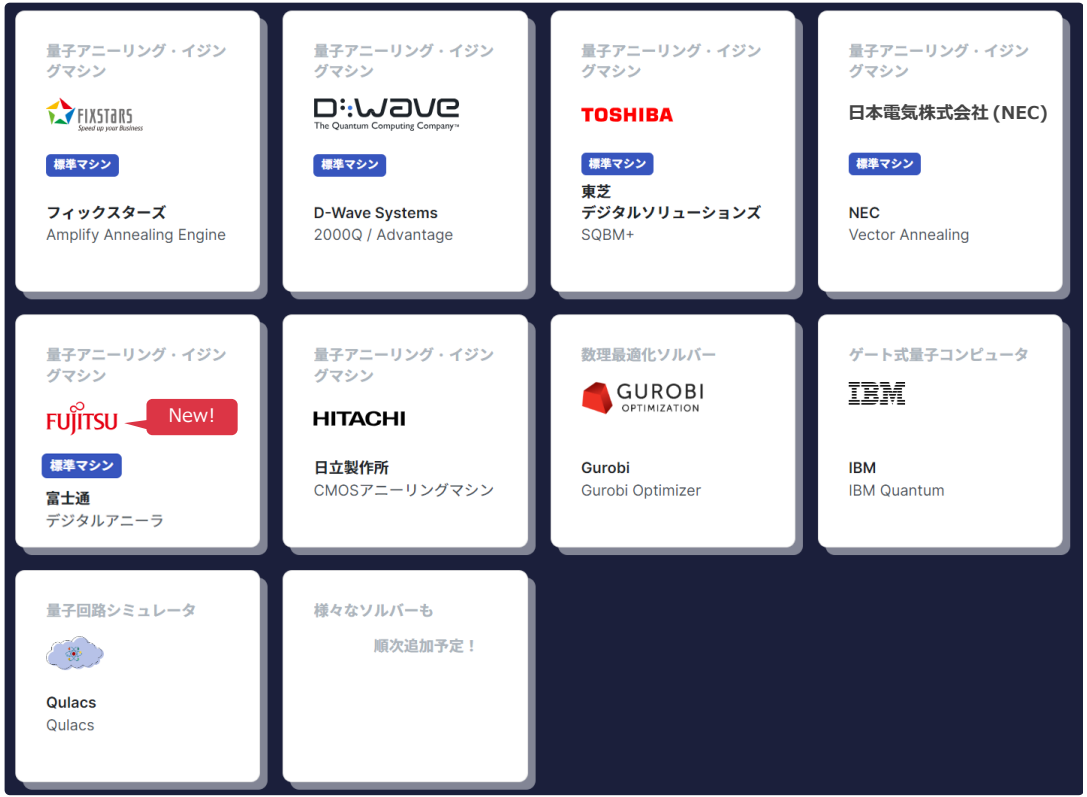
ポータブル

- すべての量子アニーリング/イジングマシンに対応
- Fixstars の26万ビット級のアニーリングマシン実行環境も利用可能

始めやすい

- 評価・検証用途には開発環境と実行環境が無償で利用可能
- 多くのチュートリアル、サンプルコードを整備・拡充

Fixstars Amplify の対応マシンの一例



標準マシン は、

- ベンダ各社と個別マシン利用契約なし、
- 評価・検証用ベーシックプランなら無料、

で利用可能！←「いつでも」、「誰でも」

今後も幅広い対応マシンの追加が続々と行われる予定です！←「あらゆる」

オンラインデモ & チュートリアル

Amplify デモ　検索

https://amplify.fixstars.com/ja/demo

New!

チュートリアル応用編

最適エネルギーマネジメント

プログラミング難易度 ★★★★☆

本サンプルプログラムでは、ホーム・エネルギー・マネジメント・システム (HEMS) を想定し、種々のエネルギーコストや供給量に基づき、2日分の最適エネルギーミックスの探索を行います。

サンプルコード

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

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

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

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

デモアプリケーション

容量制約つき運搬経路問題 (CVRP)

プログラミング難易度 ★★★★☆

運送業における効率的な配送計画の策定やごみ収集や道路清掃における訪問順序の最適化等での応用が期待される容量制約つき運搬経路問題（CVRP）を取り扱います。

デモアプリ　サンプルコード

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

チュートリアル応用編

定式化による交通信号機の最適化

プログラミング難易度 ★★★★☆

都市における渋滞を最小化するために、刻一刻と変化する交通状況に応じ、組合せ最適化を用いてリアルタイムに信号機の最適制御を実施します。また、その様な信号機制御を実施した際の都市の交通量をシミュレーションします。

サンプルコード

チュートリアル応用編

10. 整数長ジョブスケジューリング問題

プログラミング難易度 ★★★☆☆

あらかじめ決まった数のジョブとマシンがあり、それぞれのジョブにかかる時間が分かっているとします。それぞれのジョブをいずれかのマシンに割り当てます。ジョブスケジューリング問題では、最も早く全ジョブが完了するような割り当て方を求めます。

サンプルコード

チュートリアル基礎編

画像のノイズ除去

プログラミング難易度 ★★★☆☆

画像のノイズ除去を行うアプリケーションを開発します。

サンプルコード

チュートリアル応用編

会議室割当問題

プログラミング難易度 ★★★☆☆

制約条件を用いて定式化するアプリケーションの例として会議室割当問題のアプリケーションを開発します。

サンプルコード

チュートリアル応用編

タクシーマッチング問題

プログラミング難易度 ★★★☆☆

目的関数と制約条件を用いて定式化するアプリケーションの例としてタクシーマッチング問題のアプリケーションを開発します。

サンプルコード

デモアプリケーション

グラフ彩色問題

プログラミング難易度 ★★★☆☆

Fixstars Amplifyによる、グラフ彩色問題の定式化を体験します。

デモアプリ　サンプルコード

デモアプリケーション

巡回セールスマン問題

プログラミング難易度 ★★★★☆

Fixstars Amplifyによる、巡回セールスマン問題の定式化を体験します。

デモアプリ　サンプルコード

デモアプリケーション

数独

プログラミング難易度 ★★★☆☆

Fixstars Amplifyによる、数独の定式化を体験します。

デモアプリ　サンプルコード

デモアプリケーション

ライドシェア

プログラミング難易度 ★★★★★

集合型ライドシェアの最適化アプリケーションを体験します。

デモアプリ　サンプルコード

デモアプリケーション

タスク割当問題

プログラミング難易度 ★★★★★

店舗とタスクに従業員を割り当てる組合せ最適化問題のアプリケーションを体験します。

デモアプリ　サンプルコード

デモアプリケーション

ポートフォリオ最適化

プログラミング難易度 ★★★★☆

リスクとリターンを考慮した株式ポートフォリオの最適化アプリケーションを体験します。

デモアプリ　サンプルコード

Fixstars Amplify の内容と特徴

- 開発環境：Amplify SDK
- 実行環境：Amplify Annealing Engine (AE)

開発環境：Fixstars Amplify SDK

Fixstars Amplify SDK を使うと組合せ最適化問題の実装が圧倒的に直感的で簡単です

通常のプログラミング

1. **課題を定式化**
   マシンのSDKやAPI仕様に合わせて物理モデルをデータ化
2. **論理モデルへ変換**
   目的関数をマシンの動作モデルで再定義
3. **物理モデルへ変換**
   マシン仕様や制約を考慮した物理モデルに再変換
4. **マシンにデータを入力**
   マシンのSDKやAPI仕様に合わせて物理モデルをデータ化
5. **マシンを実行**
   特定マシンのみで実行可能

Fixstars Amplifyを用いたプログラミング

1. **課題を定式化**
   定式化された数式をプログラムコードで表現
2. **マシンを実行**
   複数マシンの中から選択可能

SDKが提供するAPIが、自動で各マシンに合った形式へ多段変換をして入力。実行結果は逆変換をして、ユーザーにとって結果の解釈が容易な形式で返却されます。

```
# 決定変数の発行
q = VariableGenerator().array("Binary", 2)

# 目的関数と制約条件の定義
objective = q[0] - 2 * q[0] * q[1]
constraint = one_hot(q)

# モデルの構築
model = Model(objective, constraint)

# ソルバーの指定
client = FixstarsClient()
client.token = "Amplify AE のアクセストークン"
client.parameters.timeout = timedelta(seconds=1)

# 最適化の実行
result = solve(model, client)

# 結果の表示
print(q.evaluate(result.best.values))
print(objective.evaluate(result.best.values))
```

FIXSTARS Amplify

Fixstars Amplify SDK によるシンプルプログラミング

数独を解くサンプルアプリ

SDKなし
最適化しても
200行以上

SDKあり
30行程度

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

出典:
Wikipdia

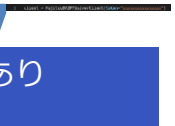
富士通・デジタルアニーラの設定用コード

SDKなし
59行

SDKあり
1行

日立CMOSアニーリングマシンの設定用コード

SDKなし
183行



SDKあり
1行

SDKが、各マシンに対して最適な形式に実装式を多段変換！

FIXSTARS Amplify

実行環境：Fixstars Amplify Annealing Engine (AE)

NVIDIA GPU V100/A100 で動作

- 独自の並列化シミュレーテッド
  アニーリングアルゴリズム

WEB経由で計算機能を提供

- 社会実装・PoC・検証が加速
- Amplify SDK の実装を直ぐに実行可能

商用マシンでは最大規模・最高速レベル

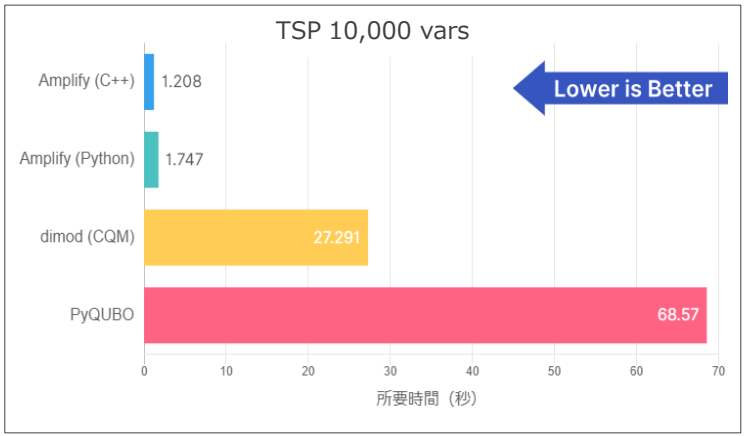
- 120,000 ビット（全結合）
- 260,000 ビット超（疎結合）

	標準マシン Fixstars Amplify AE	標準マシン D-Wave 2000Q/Advantage	標準マシン 東芝 SQBM+	日立 CMOS Annealing	富士通 Digital Annealer
装置型式	GPU	量子回路	GPU	デジタル回路	デジタル回路
最大ビット数	262,144以上	2,048 (16x16x8)/ 5,760 (16x15x24)	100,000 (SQBM+)/ 10,000 (SBM PoC 版)	61,952 (352x176)	8,192 (DA2)/ 100,000 (DA3)
係数パラメータ	デジタル (32/64bit)	アナログ (5bit程度)	デジタル (32bit)	デジタル (3bit)	デジタル (16/64 bit)
結合グラフ	全結合	キメラグラフ/ ペガサスグラフ	全結合	キンググラフ	全結合
全結合換算ビット数	131,072	64/124	31,000程度 (SQBM+)(*1)/ 1,000 (SBM PoC 版)	176	8,192 (DA2)/ 100,000 (DA3)
APIエンドポイント	Fixstars Amplify	D-Wave Leap	Fixstars Amplify / AWS	Annealing Cloud Web	DA Cloud

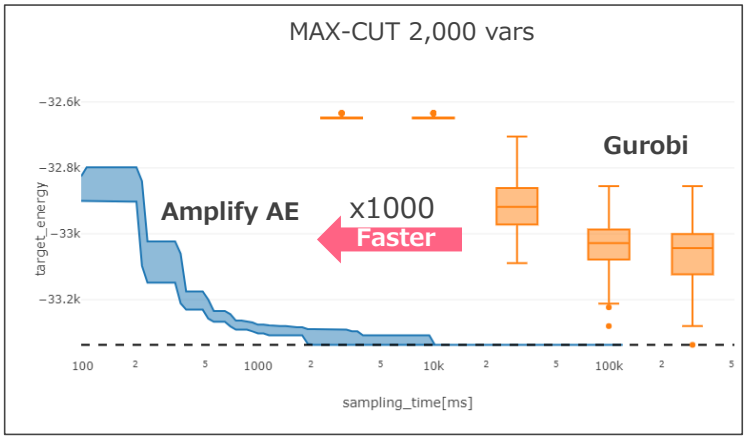
※評価・検証用途では無償提供

Fixstars Amplify SDK/AE パフォーマンス

Fixstars Amplify は、SDK も AE も最速レベルの定式化・求解速度を達成しています



SDK 定式化処理速度



AE 求解性能・速度

ワークショップ
～事前準備～

ワークショップの事前準備（1）

- ご自身の PC のブラウザ上で Python のプログラミングを行います。Google Colaboratory を使うので、事前に Google Colaboratory にログインできることをご確認ください（Google アカウントが必要です）。

Google Colab 検索

https://colab.research.google.com/

- Fixstars Amplify の無料トークンの取得有無をご確認ください。まだの方は、こちら からユーザー登録をして無料トークンを取得してください（1分で完了します）。

Fixstars Amplify 検索

https://amplify.fixstars.com/

ワークショップの事前準備（2）

取得された Fixstars Amplify AE の無料トークンを用いてトークンチェック用のサンプルコードが動くか、以下のステップでご確認をお願いします。

1. 以下の URL にアクセスしてください。サンプルコードは閲覧のみ可能な状態なので、「ファイル」→「ドライブにコピーを保存」して、ご自身の Google ドライブにコピーを作成してください。
   https://colab.research.google.com/drive/1bg2Ql3McJck_Sto8uvxtmPUMWtRFhf7a

2. コピーしたファイルの1番目のセルにご自身の無料トークンを入力してください（***印の部分を書き換えてください）。ご自身の無料トークンは、「アクセストークン」ページの「Fixstars Amplify AE」のセクションでご確認いただけます。トークンを入力後、再生ボタンまたは Shift +Enter で1番目のセルを実行して下さい。

```
token = ”**********”  # ご自身のトークンを入力
```

3. 1番目のセルの実行が完了したら、2番目のセルも再生ボタンまたは Shift + Enter で実行してください。実行後、以下の結果が出力されればOKです。

```
result: [q_0, q_1] = [1. 1.] (f = 0.0)
```

ワークショップの事前準備（3）

- ワークショップで使うサンプルコードを以下のURLより取得して下さい
- それぞれのサンプルコードにご自身のトークンを入力いただく必要があります。それぞれのサンプルコードを「ドライブにコピー」の上、トークンを入力し実行して下さい

➢ サンプルコード

Step1	https://colab.research.google.com/drive/1nu_X8RufFbc4wzRlyxj8M5FlGorkcKIE?usp=sharing#scrollTo=joih5kayrIJt
Step2	https://colab.research.google.com/drive/1ssgrvSQ7cZrLhGzmaLZUJRYu2JJeSOiv?usp=sharing#scrollTo=Ca8VcqcFsXSi
Step3	https://colab.research.google.com/drive/1SlYCT_eS5abwzl_rldM-Vf9g2s-KihRp?usp=sharing#scrollTo=WKVyZz2wtV9L

質問は随時、Zoomの チャット か Q&A でお願いします。
対応可能なメンバーが対応致します。

ワークショップ
～ 経路問題 ～

搬送経路最適化

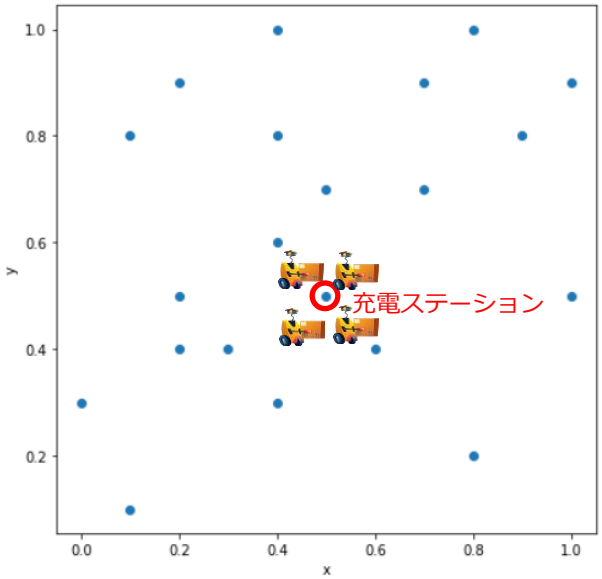
一言に「経路最適化」と言っても、搬送ロボット (AGV) が運用されている環境や要件などは様々で、それぞれに対する問題設定や定式化は異なります

状況や要件の例

- AGVが通過できる経路が定められている
- AGVごとに稼働時間に制限がある
- AGVごとに積載量に制約がある
- 配達地ごとに要求される到着時間がある
- AVGの衝突回避の制御が必要
- 運ぶ材料や部品によって各配達地で作業時間が異なる

搬送経路最適化

【問題】工場内の4台のAGVを用いて、AGVの充電ステーションから20か所の搬送先に部品を届けるときの各搬送ロボットの走行距離の合計が最短になるような経路を求めます。尚、搬送ロボットをできるだけ平均的に利用したいため各AGVの搬送先は5か所とします



搬送経路最適化

【問題】工場内の4台のAGVを用いて、AGVの充電ステーションから20か所の搬送先に部品を届けるときの各搬送ロボットの走行距離の合計が最短になるような経路を求めます。尚、搬送ロボットをできるだけ平均的に利用したいため各AGVの搬送先は5か所とします

全てを一度にやるのは難しいので、3つのステップに分けてプログラムを作成します

Step1

AGVを1台とし、制約を守るだけのプログラムを作ります
制約①: 各順番で回れる訪問先 (充電ステーション+各搬送先) は1か所
制約②: 各訪問先を回るのは1回のみ

解の候補多数あり

Step2

Step1に「総距離の最小化」という目的を追加し、複数の解の候補から目的を実現する解を求めるプログラムを作ります

Step3

AGVを4台に増やします。具体的には、Step2に対して以下の2つの変更を加えます
変更①: 決定変数の一部を0/1で固定
変更②: 制約条件の対象範囲を微調整

Step1

AGVを1台とし、制約を守るだけのプログラムを作ります
制約①: 各順番で回れる訪問先 (充電ステーション+各搬送先) は1か所
制約②: 各訪問先を回るのは1回のみ

Step1のサンプルコードのレビュー

(本ワークショップでは、最適化のコードにフォーカスするため、
可視化のコードの詳細は割愛します)

Step1

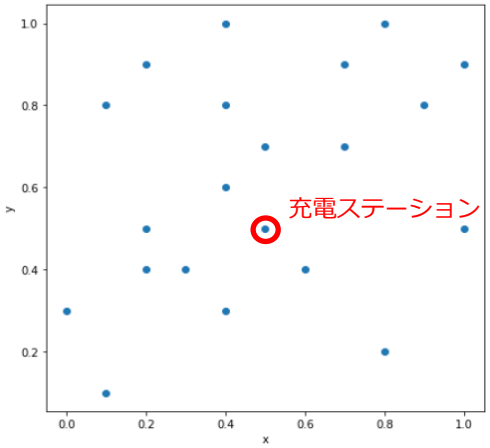
AGVを1台とし、制約を守るだけのプログラムを作ります
制約①: 各順番で回れる訪問先 (充電ステーション+各搬送先) は1か所
制約②: 各訪問先を回るのは1回のみ

距離行列の作成

各訪問先の座標

```
[0.5 0.5]
[0.  0.3]
[0.3 0.4]
[0.2 0.5]
[0.4 0.8]
[0.5 0.7]
[0.2 0.4]
[0.2 0.9]
[0.7 0.7]
[0.6 0.4]
[0.8 1. ]
[0.1 0.1]
[0.1 0.8]
[0.4 0.6]
[0.8 0.2]
[0.7 0.9]
[0.4 0.3]
[0.4 1. ]
[0.9 0.8]
[1.  0.5]
[1.  0.9]
```

充電ステーション



実装

```python
# 距離行列の作成
all_diffs = np.expand_dims(locations, axis=1) - np.expand_dims(locations, axis=0)
distances = np.sqrt(np.sum(all_diffs**2, axis=-1))
print(distances)
```

各訪問先間の距離行列

訪問先 (0) は充電ステーション

(km)		訪問先 (i)				
		0	1	2	・・・	20
訪問先	0	0	0.54	0.22	・・・	0.64
(j)	1	0.54	0	0.32	・・・	1.17
	2	0.22	0.32	0	・・・	0.86
	・ ・ ・	・ ・ ・	・ ・ ・		・・・	・ ・ ・
	20	0.64	1.17	0.86	・・・	0

Step1

AGVを1台とし、制約を守るだけのプログラムを作ります
制約①: 各順番で回れる訪問先 (充電ステーション+各搬送先) は1か所
制約②: 各訪問先を回るのは1回のみ

決定変数を準備

順番(n)	訪問先 (i) 0	1	2	・・・	20
0	1	q_0,1(0 or 1)	q_0,2(0 or 1)	・・・	q_0, 20(0 or 1)
1	q_1,0(0 or 1)	q_1,1(0 or 1)	q_1,2(0 or 1)	・・・	q_1,20(0 or 1)
2	q_2,0(0 or 1)	q_2,1(0 or 1)	q_2,2(0 or 1)		q_2,20(0 or 1)
・	・	・			・
・	・	・		・・・	・
・	・	・			・
20	q_20,0(0 or 1)	q_20,1(0 or 1)	q_20,2(0 or 1)	・・・	q_20,20(0 or 1)

BinaryPoly型 (21×21) = 441 [qbit]
1は訪問する、0はしない事を示す

イジングマシンで最適な(0,1)の組合せを探す

得られる解のイメージ

順番(n)	訪問先 (i) 0	1	2	・・・	20
0	1	0	0	・・・	0
1	0	0	0	・・・	1
2	0	1	0		0
・	・	・			・
・	・	・		・・・	・
・	・	・			・
20	0	0	1	・・・	0

充電ステーションから出発を示す

充電ステーションの次に行くのは訪問先 (20) であることを示す

実装

```
# 決定変数の作成
from amplify import VariableGenerator

gen = VariableGenerator()  # 変数のジェネレータを宣言
q = gen.array("Binary", shape=(n_visits, n_locations))  # 決定変数を作成
q[0, 0] = 1  # 出発点（充電ステーション）を表す決定変数に1を代入
print(q)
```

Step1

AGVを1台とし、制約を守るだけのプログラムを作ります
制約①: 各順番で回れる訪問先 (充電ステーション+各搬送先) は1か所
制約②: 各訪問先を回るのは1回のみ

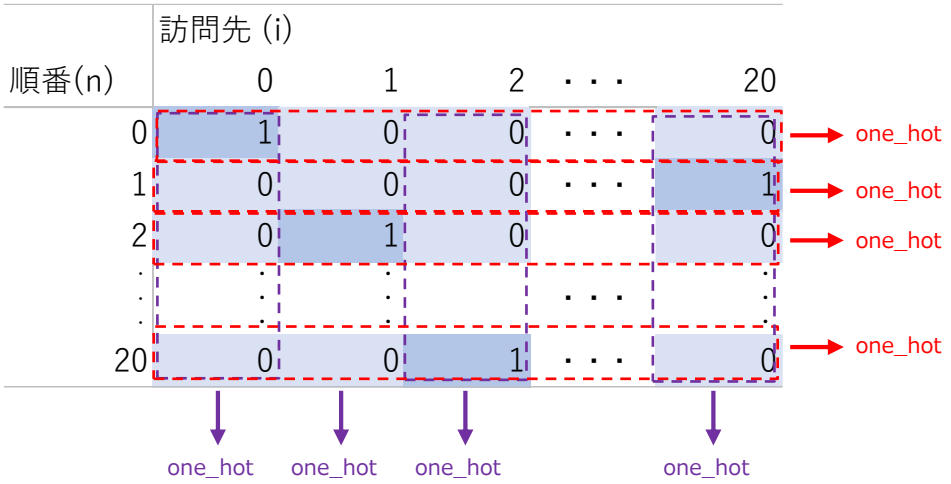
制約の定式化

制約1:各順番で回れる訪問先は１か所
→ one hot 制約

$$\sum_{i=0}^{20} q_{n,i} = 1 \qquad n \in \{0,1,\cdots,20\}$$

制約2: 各訪問先を回るのは1回のみ
→ one hot 制約

$$\sum_{n=0}^{20} q_{n,i} = 1 \qquad i \in \{0,1,\cdots,20\}$$



Step1

AGVを1台とし、制約を守るだけのプログラムを作ります
制約①: 各順番で回れる訪問先 (充電ステーション+各搬送先) は1か所
制約②: 各訪問先を回るのは1回のみ

制約の定式化

制約1:各順番で回れる訪問先は1つ
→ one hot 制約

$$\sum_{i=0}^{20} q_{n,i} = 1 \qquad n \in \{0,1,\cdots,20\}$$

制約2: 各訪問先を回るのは1回のみ
→ one hot 制約

$$\sum_{n=0}^{20} q_{n,i} = 1 \qquad i \in \{0,1,\cdots,20\}$$

実装

```
##########
# 制約
##########

from amplify import one_hot

# 制約1 各順番で回れる訪問先は1つ
row_constraints = one_hot(q, axis=1)

# 制約2 各訪問先を回るのは1回のみ
col_constraints = one_hot(q, axis=0)

# 制約
constraints = row_constraints + col_constraints
```

Step1

AGVを1台とし、制約を守るだけのプログラムを作ります
制約①: 各順番で回れる訪問先 (充電ステーション+各搬送先) は1か所
制約②: 各訪問先を回るのは1回のみ

求解

- 先ほど作ったconstraints (制約)をmodelに格納して、マシンに投げます
- 制約条件だけを与えた場合、制約条件を満たす解を探してきてくれます

```
##########
# 求解
##########

from amplify import FixstarsClient, solve

# 実行マシンクライアントの設定
client = FixstarsClient()
client.token = token
client.parameters.timeout = 1 * 1000  # タイムアウト1秒 (1000m秒)

# モデル化
model = constraints

# アニーリングマシンの実行
result = solve(model, client)
```

Amplify AE

無料版は1ジョブ10秒まで設定可。有料版では、1分 (スタンダード)、10分（プレミアム）、15分（Sプレミアム）まで設定可能

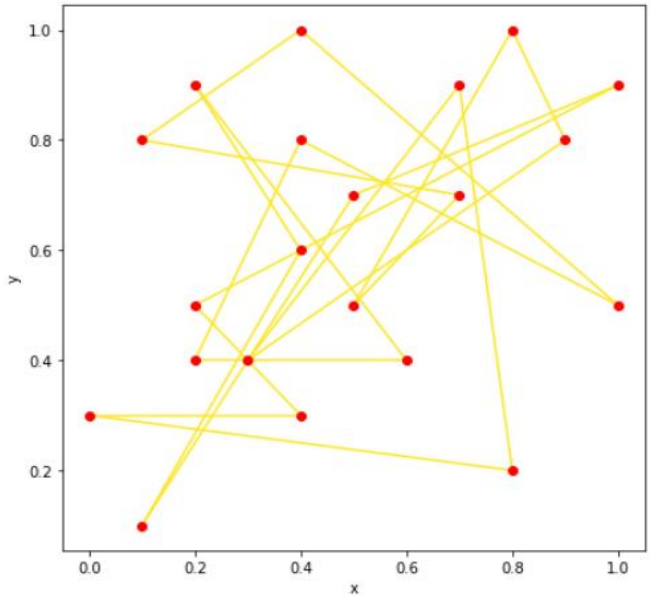
Step1

AGVを1台とし、制約を守るだけのプログラムを作ります
制約①: 各順番で回れる訪問先 (充電ステーション+各搬送先) は1か所
制約②: 各訪問先を回るのは1回のみ

結果の取得

```
#################
# 結果の取得
#################

q_solutions = q.evaluate(result.best.values)
print(q_solutions)
```

可視化



各順番で回れる訪問先は1つ、各訪問先を回るのは1回のみ、という二つの制約を満たす経路を求めることができました。解の候補がたくさんある状況ですので、次のStepでは、目的を追加して解を絞ります

Step2

Step1に「総距離の最小化」という目的を追加し、複数の解の候補から目的を実現する解を求めるプログラムを作ります

Step2のサンプルコードのレビュー

Step2

Step1に「総距離の最小化」という目的を追加し、複数の解の候補から目的を実現する解を求めるプログラムを作ります

目的関数の定式化

総距離の最小化

$$distance = \sum_{n=0}^{20}\sum_{i=0}^{20}\sum_{j=0}^{20} d_{i,j} \cdot q_{n,i} \cdot q_{n+1,j}$$

決定変数 x 決定変数 (2次)

計算のイメージ

n=1のときi=20が1、n=2のときj=1が1なので、$d_{20,1}$である1.17がdistanceに加算される

$distance =$

各訪問先間の距離行列 ($d_{i,j}$)

(km)		訪問先 (i)				
		0	1	2	・・・	20
訪問先	0	0	0.54	0.22	・・・	0.64
(j)	1	0.54	0	0.32	・・・	1.17
	2	0.22	0.32	0	・・・	0.86
	⋮	⋮	⋮		・・・	⋮
	20	0.64	1.17	0.86	・・・	0

×

決定変数 ($q_{n,i}$, $q_{n+1,j}$)

	訪問先 (i)				
順番(n)	0	1	2	・・・	20
0	1	0	0	・・・	0
1	0	0	0	・・・	1
2	0	1	0		0
⋮	⋮	⋮		・・・	⋮
20	0	0	1	・・・	0

Step2

Step1に「総距離の最小化」という目的を追加し、複数の解の候補から目的を実現する解を求めるプログラムを作ります

実装

：追加 or 変更部分

```
##########
# 目的関数
##########

distance = 0
for n in range(n_visits):
    for i in range(n_locations):
        for j in range(n_locations):
            distance += distances[i, j] * q[n, i] * q[(n + 1) % n_visits, j]
```

剰余を使って、最後の訪問先の次は0番目に戻るようにしています

```
##########
# 制約
##########

# 制約の重み
constraint_weight = 1

# 制約
constraints = constraint_weight * (row_constraints + col_constraints)
```

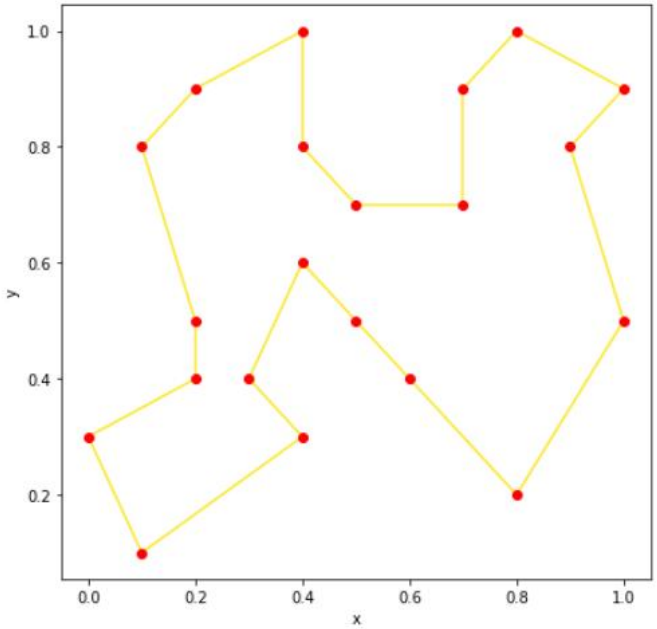
イジングマシンは、制約を満たす中で、このdistance (目的) を最小にする組合せを探します

```
# モデル化
model = distance + constraints

# アニーリングマシンの実行
result = solve(model, client)
```

【補足】制約条件には、目的関数の値を考慮して適切な値の重みを設定する必要があります。制約条件の取り扱いに関する詳細は、こちらのドキュメントも合わせてご参照下さい
https://amplify.fixstars.com/ja/docs/amplify/v1/constraint.html#id8

Step2

Step1に「総距離の最小化」という目的を追加し、複数の解の候補から目的を実現する解を求めるプログラムを作ります



2つの制約を満たした上で、総距離を最小化する経路が求められるようになりました。次のStepでは、AGVを4台に増やします。

Step3

AGVを4台に増やします。具体的には、Step2に対して以下の2つの変更を加えます
変更①: n_vehicle = 4にして決定変数の一部を0/1で固定
変更②: 制約条件の対象範囲を微調整

Step3のサンプルコードのレビュー

Step3　AGVを4台に増やします (n_vehicle = 4 → n_visits = 24)

変更①: 決定変数の一部を0/1で固定します。これにより、各AGVの搬送先を5か所に固定しています

決定変数

充電ステーションには、0番目、6番目、
12番目、18番目に訪問します (計4回訪問)

順番(n) \ 訪問先 (i)	0	1	2	・・・	20
0	1	0	0	・・・	0
1	0	q_1,1	q_1,2	・・・	q_1,20
2	0	q_2,1	q_2,2	・・・	q_2,20
3	0	q_3,1	q_3,2	・・・	q_3,20
4	0	q_4,1	q_4,2	・・・	q_4,20
5	0	q_5,1	q_5,2	・・・	q_5,20
6	1	0	0	0	0
7	0	q_7,1	q_7,2	・・・	q_7,20
8	0	q_8,1	q_8,2	・・・	q_8,20
9	0	q_9,1	q_9,2	・・・	q_9,20
10	0	q_10,1	q_10,2	・・・	q_10,20
11	0	q_11,1	q_11,2	・・・	q_11,20
12	1	0	0	0	0
13	0	q_13,1	q_13,2	・・・	q_13,20
⋮	⋮	⋮	⋮		⋮
23	0	q_23,1	q_23,2	・・・	q_23,20

これらの中で
合計が最短と
なる経路を探
します

実装

：追加 or 変更部分

```python
# 決定変数の作成
from amplify import VariableGenerator

gen = VariableGenerator()  # 変数のジェネレータを宣言
q = gen.array("Binary", shape=(n_visits, n_locations))  # 決定変数を作成

if len(locations) // n_vehicle <= 1:
    raise RuntimeError(f"n_vehicle is too large: {n_vehicle}")


def split_number(number, divisor):
    quotient, reminder = divmod(number, divisor)
    return [quotient + 1] * reminder + [quotient] * (divisor - reminder)


i = 0
q[:, 0] = 0
for j in split_number(n_visits, n_vehicle):
    q[i] = 0
    q[i, 0] = 1
    i += j
```

Step3　AGVを4台に増やします (n_vehicle = 4 → n_visits = 24)

変更②: 制約条件の対象範囲を微調整します。訪問先 (0) である充電ステーションを、one_hot制約の対象から外します

決定変数

one_hot制約対象外

順番(n) \ 訪問先 (i)	0	1	2	・・・	20
0	1	0	0	・・・	0
1	0	q_1,1	q_1,2	・・・	q_1,20
2	0	q_2,1	q_2,2	・・・	q_2,20
3	0	q_3,1	q_3,2	・・・	q_3,20
4	0	q_4,1	q_4,2	・・・	q_4,20
5	0	q_5,1	q_5,2	・・・	q_5,20
6	1	0	0	0	0
7	0	q_7,1	q_7,2	・・・	q_7,20
8	0	q_8,1	q_8,2	・・・	q_8,20
9	0	q_9,1	q_9,2	・・・	q_9,20
10	0	q_10,1	q_10,2	・・・	q_10,20
11	0	q_11,1	q_11,2	・・・	q_11,20
12	1	0	0	0	0
13	0	q_13,1	q_13,2	・・・	q_13,20
	.	.	.		.
	.	.	.		.
	.	.	.		.
23	0	q_23,1	q_23,2	・・・	q_23,20

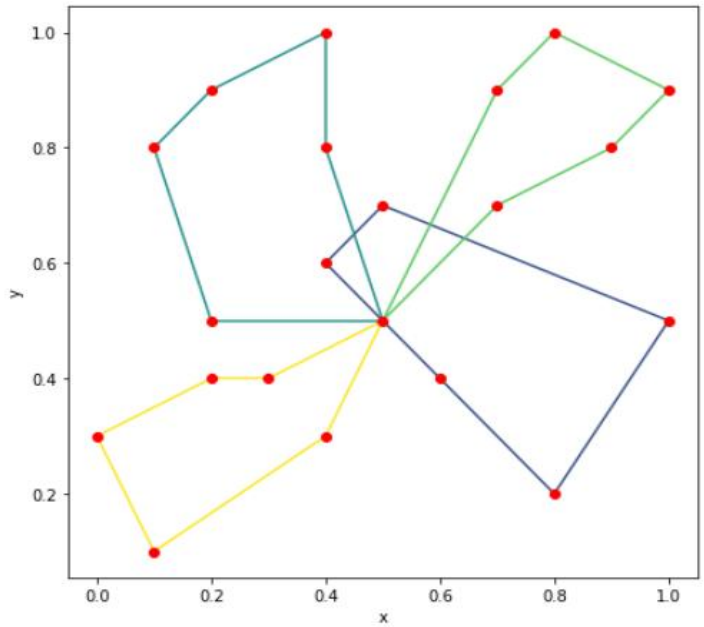
実装

：追加 or 変更部分

```
##########
# 制約
##########

from amplify import one_hot

# 制約1 各順番で回れる訪問先は1つ
row_constraints = one_hot(q, axis=1)

# 制約2 各訪問先を回るのは1回のみ
col_constraints = one_hot(q[:, 1:], axis=0)
```

Step3

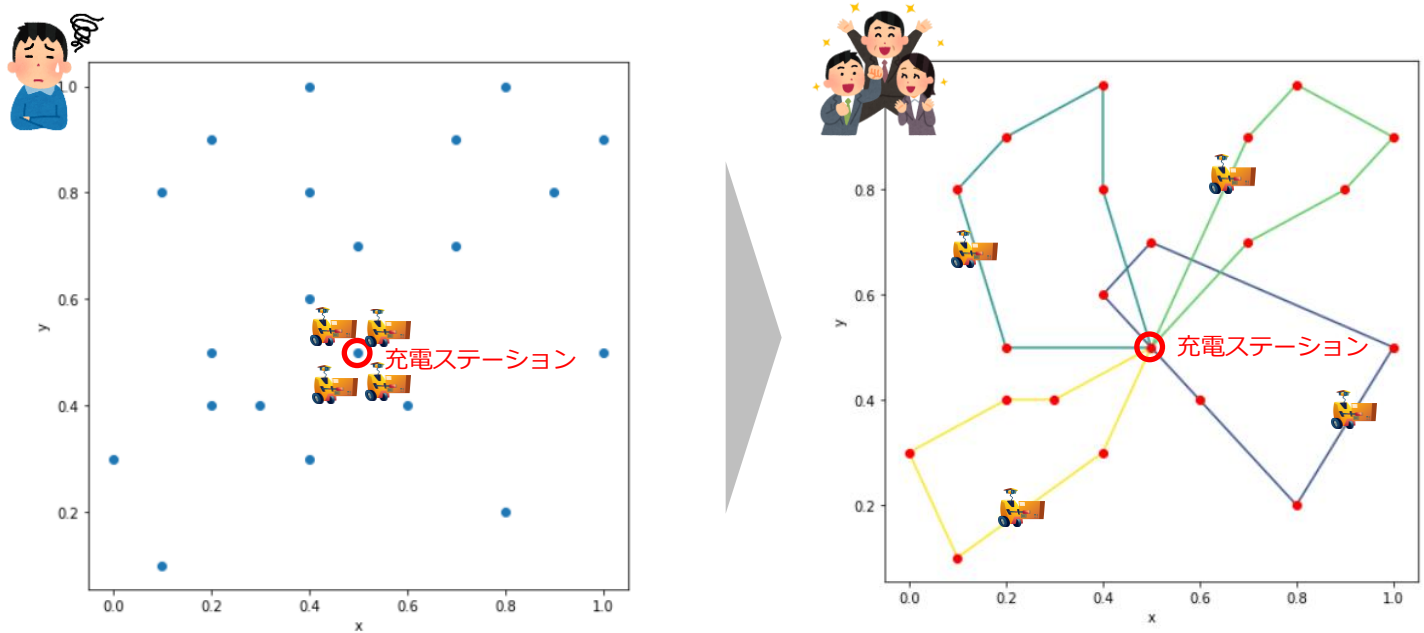
AGVを4台に増やします (n_vehicle = 4 → n_visits = 24)



4台のAGVそれぞれが5か所ずつ搬送し、総走行距離が最短になるような経路を求める事ができました！

ワークショップ：おさらい

AGV1台の制約のみから始めて、目的を追加し、最終的には、4台のAGVそれぞれが5か所ずつ搬送し総走行距離が最短になるような経路を求めることができました！



今後について

ぜひ、デモ・チュートリアルの様々な問題に挑戦してください！

目的関数のみで解く問題

制約条件のみで解く問題

目的関数＋制約条件で解く問題

複数台の車両のそれぞれに積載容量の制約がある中で、どの車両がどの荷物をどういうルートで運ぶかを解く問題です。不等式制約 (less_equal) を使って解いています！

困った時はドキュメンテーションを！

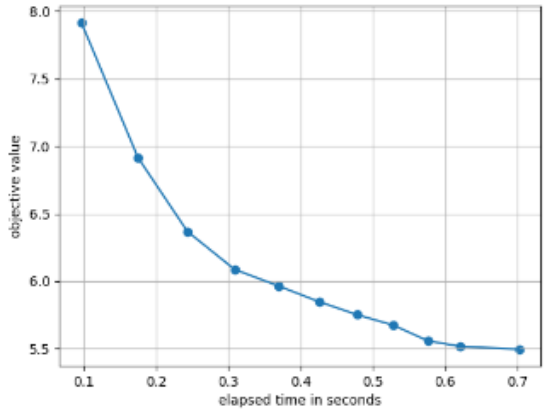
https://amplify.fixstars.com/docs/amplify/v1/index.html

Tips: 適切なタイムアウト値の設定について

実行後にヒストリー情報や num_iterations などを確認しながら適切なタイムアウト値を調整します

ヒストリー情報 (ドキュメントはこちら)

- Amplify AE は1回の実行の中で探索を繰り返し、時間の許す限り最良解を徐々に改善していきます。
- 必要な設定変更を行うことにより、1回の実行の中で、最良解を更新した時間とその解の値を確認することができます。解の収束の様子などから実行時間の過不足の判断に用いることができます。



num_iterations (ドキュメントはこちら)

- 実行後に num_iterations を確認することにより、どの程度の「探索」が行われたか確認することができます。「探索」とは、Amplify AE が実装しているアルゴリズムの実行単位であり、この値が大きいほど広く探索が行えたことを意味します。この値が一桁など小さい場合には探索が十分でない可能性があります。
- Amplify AE は初めに1回だけ探索を行い、タイムアウトまで余裕がある場合には、時間の許す限り解を徐々に改善していくという動作になっています。設定したタイムアウト値が問題規模に対して小さすぎる場合には、最初の探索が指定したタイムアウト時間内に終わらないこともあります。その場合、初めの探索だけで実行が終わり、num_iterations は 1 となります。

```
print(result.client_result.execution_parameters.num_iterations)
```

事例・価格・今後の進め方のご紹介

シフト最適化の事例: 物流倉庫の最適配置

https://www.fixstars.com/ja/services/cases/amplify-bellemaison

業務内容：
梱包業務担当者（１チーム３名）を
コンベア前のブースに割り当て

従来の方法：
前日夕方に、翌日の予測出荷目標数と出勤予定に基づいて、３人程度のリーダーが相談し**数時間をかけて**決定

課題：
公平になるよう、様々な配慮を行う必要があり、割り当て担当者に心理的負担がかかっていた

成果：　→ 2022年10月より実稼働開始！
アニーリング技術を活用して自動化・デジタル化
- 作業時間 → 15分程度に
- 心理負担 → ほぼゼロに

手動配置
一部事前配置
自動配置結果の微調整

自動配置（アニーリング）
各種条件を満たす形で
未配置のメンバーを一括割り当て

割当業務の時間削減
担当者の心理的負担低減
配置情報のデジタル化
Smileboard Connectと連携

国家プロジェクトSIP「光・量子を活用したSociety 5.0実現化技術」の一環として、住友商事、SCSK、ベルメゾンロジスコと、2019年より共同研究

FIXSTARS Amplify

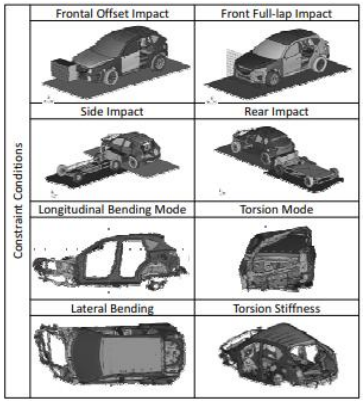
※Smileboard Connect: 住友商事様開発の物流業務可視化サービス

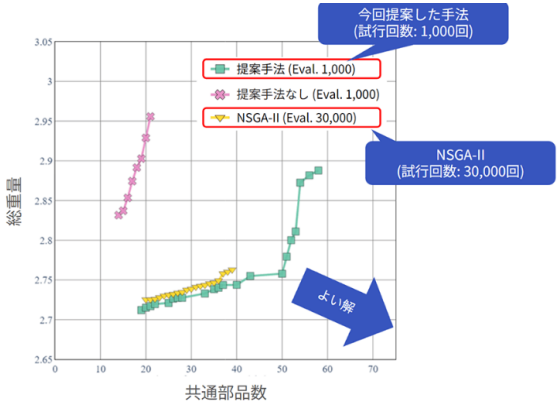
車体構造の設計最適化（マツダ株式会社）

https://amplify.fixstars.com/ja/customers/interview/vehicle-co-optimization

- 複数車種の車体内部の各部位に異なる厚さの鋼板を割り当てる問題（設計変数が計222個の大規模な問題）。衝突性能などの品質特性の制約条件を満たした上で、部品の軽量化と3車種間の共通部品数の最大化の実現する多目的最適化問題
- 2017年にマツダがベンチマーク問題として一般公開し*1、進化学会でのコンペ*2や英国オックスフォード大学や米国Meta社の研究者によるベイズ最適化に基づく解法*3など様々な手法が試されてきており、1～3万回程度の試行を繰り返せばある程度よい解が見つけられることは確認されていた
- 2023年より量子アニーリング・イジングマシを活用したブラックボックス最適化（FMQA）に着目し、検証を開始

- 量子アニーリング・イジングマシを活用した FMQA により、で従来の代表的な手法の1/30の1,000回の試行で同等以上の解を見つけることに成功！
- 今後は、QUBO 式近似の計算コストを削減しつつ、最適化性能も向上させるような手法の検討を予定





*1 応答曲面法を用いた複数車種の同時最適化ベンチマーク問題の提案
*2 進化計算コンペティション2017開催報告
*3 Multi-objective Bayesian optimization over high-dimensional search spaces

Fixstars Amplify ユーザーインタビュー

Amplify インタビュー 検索
amplify.fixstars.com/ja/customers/interview

- 業務・研究開発利用

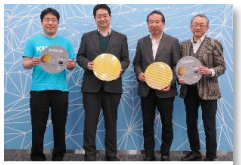
ユーザーインタビュー 2024年10月
マツダ株式会社
ブラックボックス最適化を活用した車両設計最適化

開発事例 2024年5月
日本テレビ放送網
数理最適化技術で地上波テレビの広告取引を最適化

ユーザーインタビュー 2023年6月
住友商事株式会社
現場で愛されて育つ、量子コンピュータ技術活用

ユーザーインタビュー 2023年11月
株式会社アパールデータ
生産計画の属人化解消と設備投資計画への活用を目指して

パートナー事例 2022年10月
通販向け物流倉庫の人員最適配置自動作成サービス
住友商事株式会社と、物流倉庫での実運用を開始



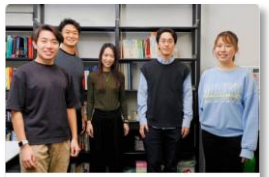
ユーザーインタビュー 2024年6月
会津大学・東京工業大学・キオクシア株式会社
半導体製造におけるマスク最適化に量子アニーリング・イジングマシンを活用

- 学術利用

FMQAの生みの親

ユーザーインタビュー 2023年8月
東京大学
ブラックボックス最適化手法（FMQA）の開発に成功

有識者インタビュー 2023年2月
慶應義塾大学
量子コンピュータの活用を促進する「量子バイリンガル」というキャリアデザイン



ユーザーインタビュー 2022年11月
慶應義塾大学
Fixstars Amplifyを使って、次世代技術を活用した高速な解析手法の開発に成功

ユーザーインタビュー 2023年4月
早稲田大学
情報工学領域で進む量子コンピュータ・イジングマシンの活用

Fixstars Amplify ユーザーインタビュー

Amplify インタビュー　検索

amplify.fixstars.com/ja/customers/interview

MITOU TARGET

特集インタビュー　2024年6月

2023年度IPA未踏ターゲット事業

Fixstars Amplifyを活用してプロジェクトに取り組まれた5組の方々へのインタビュー

Fixstars Amplify: クラウド利用料

	個人単位のプラン ～ 主に研究者・開発者向け ～				組織単位のビジネスプラン ～ 社内システムの利用向け ～		
(金額は税抜)	ベーシック	スタンダード	プレミアム	Sプレミアム	ビジネス スタンダード	ビジネス プレミアム	ビジネス Sプレミアム
月額利用料	**無料**	**10万円** (1名) **30万円** (最大5名)	**20万円** (1名) **60万円** (最大5名)	**30万円** (1名) **90万円** (最大5名)	**20万円** (1アプリ) (同一組織内であれば同一アプリのユーザー数は無制限)	**40万円** (1アプリ)	**60万円** (1アプリ)
計算環境	**スモール**	**ミディアム**	**ラージ**	**スーパーラージ**	**ミディアム**	**ラージ**	**スーパーラージ**
利用GPU (マルチGPUオプションあり)	NVIDIA V100	NVIDIA V100	NVIDIA A100	NVIDIA H100	NVIDIA V100	NVIDIA A100	NVIDIA H100
1ジョブの実行時間 (実行時間延長オプションあり)	10秒	1分	10分	15分	1分	10分	15分
月間累計実行回数 (実行回数追加オプションあり)	制限の可能性あり	無制限			- (上限を設定する可能性あり)		
月間累計実行時間	制限の可能性あり	無制限			- (上限を設定する可能性あり)		
東芝 SQBM+ オプション	10秒	30万円 (1名)、90万円 (最大5名)			-		
富士通 DA オプション	10秒	50万円 (1名)、150万円 (最大5名)			-		
D-Wave オプション	3分/月	ご相談可			-		
サポート	**ベーシック**	**スタンダード**	**プレミアム**	**プレミアム**	**スタンダード**	**スタンダード**	**スタンダード**
Plus オプション	-	-	月額50万/人		-		

次ページ

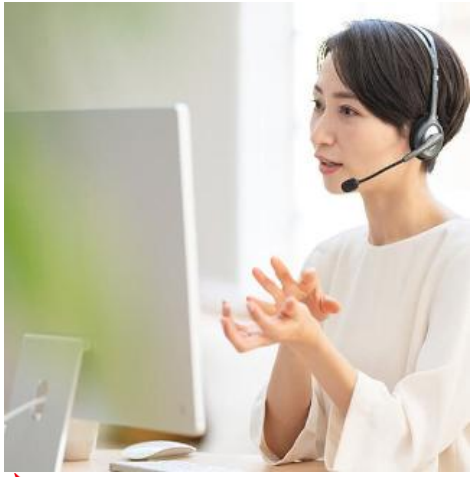
Plusオプション（プレミアムプラン/Sプレミアムプランで追加可能なオプション）

Plusオプション

料金：月額50万円（税込55万円）/ユーザー

- 問い合わせ回数は無制限
- ご質問には翌営業日までに回答（目安）
- 定式化・実装等のご相談
- 特別対応窓口や定例会の設置
- 特別技術支援※

※特別技術支援の内容に応じて期間等は個別にご相談



特別技術支援の例

開発支援

- ユーザー様の実装にお困りの部分に関して、弊社エンジニアがサンプルコードを作って提供します
- 開発支援にかかる期間については個別相談となります

コードレビュー

- 弊社エンジニアがユーザー様が実装したコードを確認し、よりよい実装などがあればサンプルコードを作って提供します

評価支援

- ユーザー様にご提供いただく問題設定で、弊社のエンジニアが様々な計算環境で実験・評価して結果をレポートします
- 複雑な問題になると限られた計算環境では十分な精度の解が得られない可能性があります。本支援では、異なる GPU (V100/A100/H100) や、GPU 数 (1機～4機)、実行時間 (～1時間) で実験・評価し、最適な計算環境の評価・検討のご支援をします
- 問題設定については、ユーザー様にプログラムやデータを送付してもらう、もしくは、問題の概要をテンプレートで回答いただく形になります
- 評価支援にかかる期間については個別相談となります

FIXSTARS Amplify

今後について

ぜひ、デモ・チュートリアルにあるサンプルコードにも挑戦してください！

一般的な組合せ最適化問題

目的関数のみで定式化

制約条件のみで定式化

目的関数 + 制約条件

ブラックボックス最適化問題

概要

材料探索

翼型最適化

信号機制御

困った時はドキュメンテーションを！

https://amplify.fixstars.com/docs/amplify/v1/index.html

セミナー・トレーニングのご紹介

https://amplify.fixstars.com/ja/news/seminar

お客様の実際の課題解決をご支援するために、無料セミナーや有償トレーニングを提供しています。

無料セミナー・ワークショップ

ビジネス向け、エンジニア向けに分けて
開催しています！

ビジネス向け

製造業向け量子コンピュータ時代のDXセミナー

見える化、予測・分析、その先の最適化へ

組合せ最適化問題や量子アニーリング・イジングマシンの概要をご紹介したのち、製造業における組合せ最適化を活用したDX推進の一例として、生産計画最適化や生産ラインのシフト最適化などの事例とデモをご紹介いたします。「Fixstars Amplify」を通じて量子アニーリング・イジングマシンを活用することで、どのようなビジネス上の効果が期待できるのかを感じていただきたいと思います。

エンジニア向け

製造業向け量子コンピュータ時代のDXセミナー

最適化の中身を覗いてみよう

製造業における組合せ最適化を活用したDX推進の一例として、生産計画最適化、勤務シフト最適化などの事例を用いて、問題設定の考え方、目的関数や制約条件の定式化、実装のポイントなどを実際のコードを見ながら解説します。また、サンプルコードを用いて、ご自身の環境で実際に量子アニーリング・イジングマシンを動かす体験をしていただきます。

企業向けプライベートトレーニング

お客様が抱える実際の課題やデータを使った
カスタムメイドのトレーニングです！

全4回のレクチャーとお客様に実施いただく「課題」を含む約1.5か月のコースです。コースの前半では、量子アニーリング・イジングマシン専用の開発／実行環境であるFixstars Amplifyを用いてPython言語による組合せ最適化アプリケーション開発方法を学びます。後半では、お客様が抱える実際の課題やデータを使ったトレーニングを実施します。量子アニーリング・イジングマシンを使って実課題の解決に取り組んでみたい方に最適なコースです。

展示会に出展します

今後のセミナーのご案内

https://amplify.fixstars.com/ja/news/seminar

今後も研究・開発者向けの無料セミナーを定期的に開催します！

2025/5/14（水）
14:00-16:00

「AGVの搬送経路最適化」

- 会社および「Fixstars Amplify」のご紹介
- Fixstars Amplify を用いたワークショップ
  ・AGVの搬送経路最適化
- 事例や今後の進め方のご紹介
- Wrap Up & QA

2025/6/19 (木)
11:00-12:00 & 14:00-15:00

「BBO 技術解説」

- 会社および「Fixstars Amplify」のご紹介
- ブラックボックス最適化のご紹介
- FMQA の最新事例および技術解説
- Wrap Up & QA

※ 2回とも同じ内容です

2025/7/24（木）（仮）
14:00-16:00

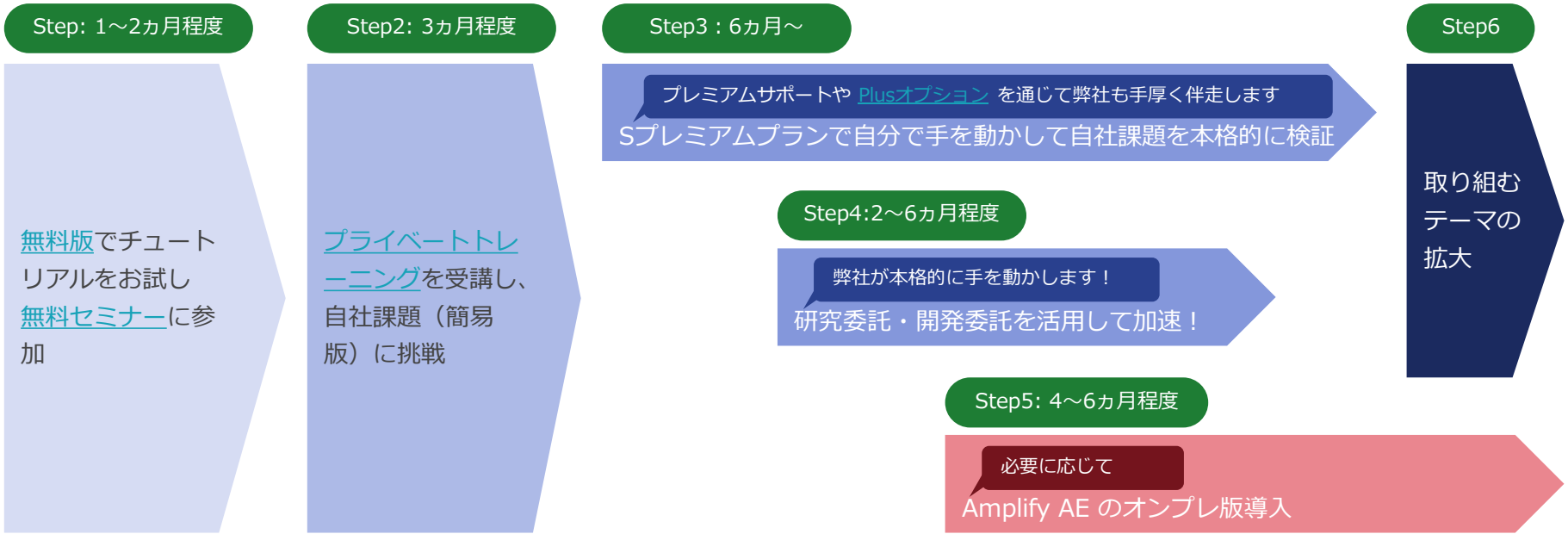
「BBO 材料開発」

- 会社および「Fixstars Amplify」のご紹介
- Fixstars Amplify を用いたワークショップ
  ・BBOを用いた材料開発
- 事例や今後の進め方のご紹介
- Wrap Up & QA

ご質問・ご不明点がありましたら、お問い合わせフォームでご連絡下さい
https://amplify.fixstars.com/ja/contact

研究・開発者向けおすすめの進め方

二次・非線形を上手に使いこなせるように、**弊社と一緒に**取り組みを進めていきましょう！



本セミナーのゴール

- 身の回りには組合せ最適化問題がたくさんあることを知る
- 組合せ最適化問題を解くための専用マシン（量子アニーリング・イジングマシン）があることを知り、解くためための統一的なフレームワークを理解する（決定変数、目的関数、制約条件など）
- ワークショップを通して、実際にイジングマシンを動かしてみることで、実問題への適用の足掛かりを得る

FIXSTARS Amplify

ご参加いただきありがとうございました

アンケートへのご回答もお願いいたします