

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
- 会社および量子コンピューティングクラウド「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

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

質問は随時ZoomのチャットかQ&Aでお願いします

会社紹介

フィックスターズグループの基本情報

コンピュータの性能を最大限に引き出す、**ソフトウェア高速化**のエキスパート集団

会社名	株式会社フィックスターズ
本社所在地	東京都港区芝浦3-1-1 msb Tamachi 田町ステーションタワーN 28階
設立	2002年8月
上場区分	東証プライム（証券コード：3687）
代表取締役社長	三木 聡

資本金	5億5,446万円
社員数（連結）	292名（2023年9月現在）
主なお客様	キオクシア株式会社 ルネサスエレクトロニクス株式会社 トヨタグループ（トヨタ自動車株式会社・豊田通商株式会社・株式会社デンソー） みずほ証券株式会社 キヤノン株式会社

グループ会社

Fixstars Solutions, Inc.

完全子会社
米国での営業及び開発を担当

(株) Fixstars Autonomous Technologies

株式会社ネクスティ エレクトロニクスとのJV
自動運転向けソフトウェアを開発

(株) Fixstars Amplify

完全子会社
量子コンピューティングのクラウド事業を運営

2021/10/1 設立

(株) Sider

完全子会社
開発支援SaaS「Sider」を運営

(株) Smart Opinion

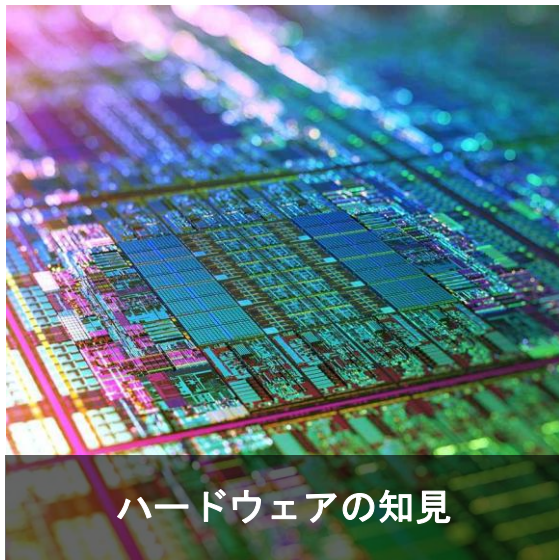
連結子会社
乳がんAI画像診断支援事業を運営

オスカーテクノロジー (株)

連結子会社
ソフトウェア自動並列化サービスを提供

フィックスターズの強み

コンピュータの性能を最大限に引き出す、**ソフトウェア高速化**のエキスパート集団



目的の製品に最適なハードウェアを見抜き、その性能をフル活用するソフトウェアを開発します。



ハードウェアの特徴と製品要求仕様に合わせて、アルゴリズムを改良して高速化を実現します。



開発したい製品に使える技術を見抜き、実際に動作する実装までトータルにサポートします。

フィックスターズの量子・最適化技術への取り組み

次世代技術を先取りし
今ある課題の解決を目指す

2018年

NEDOのプロジェクトに採択
「イジングマシン共通ソフトウェア
基盤の研究開発」

2017年

日本で初めて
D-Wave Systems社と提携

2019年

SIPの研究開発に参画
「光・量子を活用したSociety 5.0実現化技術：光電子情報処理」

2021年

2月：量子アニーリングクラウドサービス「Fixstars Amplify」提供開始
10月：子会社Fixstars Amplifyを設立
11月：Q-STAR 量子技術による新産業創出協議会に特別会員として加入

2022年

5月：Fixstars Amplify がGurobi、IBM-Quantumをサポート
7月：累計実行回数1,000万回突破

2023年

9月：新製品 **Fixstars Amplify Scheduling Engine** リリース
11月：Toshiba SQBM+を標準マシンに追加
12月：累計実行回数3,000万回突破

量子コンピューティング事業

多様なハードウェアでのソフトウェア高速化サービスに加え、
量子コンピュータ活用支援とシステム開発を提供しています。

お客様の課題



量子コンピューティングが課題の解決に役立つ
か確信が持てない



量子コンピューティングの検討をどう進めたら
良いかわからない



作りたいアプリケーションがあるが、開発が
難しい

ご支援内容

セミナー・トレーニング

量子コンピュータの研究動向や活用事例、実際の利用方法等

クラウド実行環境のご提供

クラウド経由での量子コンピュータ利用サービスを提供

コンサルティング

セットアップ支援、処理の分割や変換等のコンサル

ソフトウェア高速化・開発支援サービス

量子コンピュータを組み合わせることでシステムの高速化を実現

様々な領域での利用拡大中（実稼働含む）

700 を超える企業、研究所、大学

5,000万 を超える実行回数 (Amplify AE)



Fixstars Amplify のご紹介

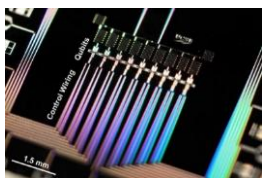
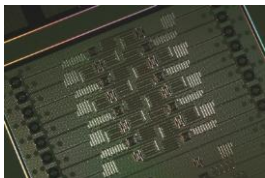
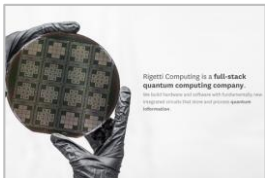
量子技術とFixstars Amplifyの対応領域

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。



1
量子コンピュータ

IBM/Google/Rigetti/IonQ

2
量子
アニーリング

D-Wave/NEC

3
イジングマシン

富士通/日立/東芝/Fixstars

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。



Amplify AE

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。

Copyright© Fixstars Group

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)

QUBO目的関数 (0-1整数二次計画問題)

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

f : 目的関数

q : 決定変数

Q : 係数

量子アニーリング・イジングマシン

Q uadratic	二次形
U nconstrained	制約条件なし
B inary	0-1整数 (二値)
O ptimization	計画 (最適化)



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



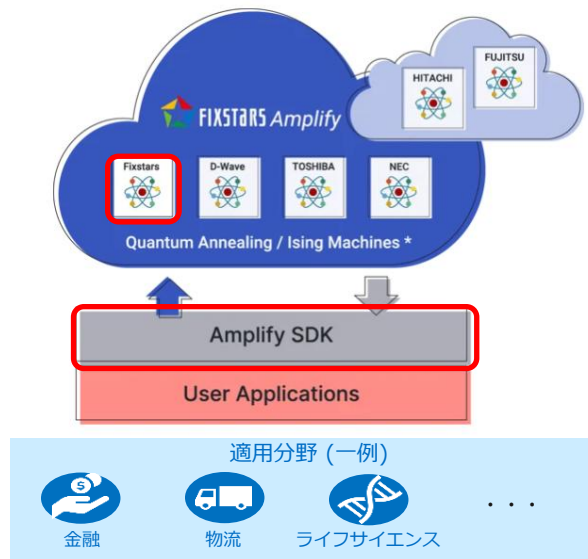
クラウドサービス : **Fixstars Amplify**

クラウドサービス: 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独自開発のGPUアニーリングマシンなど、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

量子アニーリング・イジングマシン  標準マシン フィックスターズ Amplify Annealing Engine	量子アニーリング・イジングマシン  標準マシン D-Wave Systems 2000Q / Advantage	量子アニーリング・イジングマシン  標準マシン 東芝 デジタルソリューションズ SQBM+	量子アニーリング・イジングマシン 日本電気株式会社 (NEC) 標準マシン NEC Vector Annealing
量子アニーリング・イジングマシン  富士通 デジタルアニーラ	量子アニーリング・イジングマシン  HITACHI 日立製作所 CMOSアニーリングマシン	数値最適化ソルバー  Gurobi Gurobi Optimizer	ゲート式量子コンピュータ  IBM IBM Quantum
量子回路シミュレータ  Qulacs Qulacs	様々なソルバーも 順次追加予定！		

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

最速エネルギー管理システム

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た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・インジマンを用いた最適化を行います。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 (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、組み合わせた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、真の最適化を最大化するように実装の探索を行います。

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 (4)

プログラミング難易度 ★★★★★
流体機器設計に不可欠な異型の最適化問題を取り上げます。最適化には、組み合わせた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、真の最適化を最大化するように実装の探索を行います。

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 ★★★★★
配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巡回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、商業施設における交通量が増加する都市における、交通渋滞を低減するような信号機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、第一歩に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くつ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



チュートリアル応用編

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



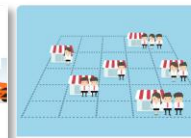
デモアプリケーション

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開発します。

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開発します。

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開発します。

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 ならアニーリングのプログラミングが圧倒的に短縮されます

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。

開発環境インストール

```
$ pip install amplify
```

最適化コード例

```
1 from amplify import VariableGenerator, FixstarsClient, solve
2
3 # 入力モデルの構築
4 q = VariableGenerator().array("Binary", 2)
5 f = 1 - q[0] * q[1]
6
7 # 実行マシンの設定
8 client = FixstarsClient()
9 client.parameters.timeout = 1000
10
11 # アニーリングの実行
12 result = solve(f, client)
13
14 # 結果の解釈
15 solution = q.evaluate(result.best.values)
16
17 print(f"result: {q} = {solution}")
18 # result: [q_0, q_1] = [1, 1.]
```

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:
Wikipedia

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！

実行環境：Fixstars Amplify Annealing Engine (AE)

NVIDIA GPU V100/A100/H100 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

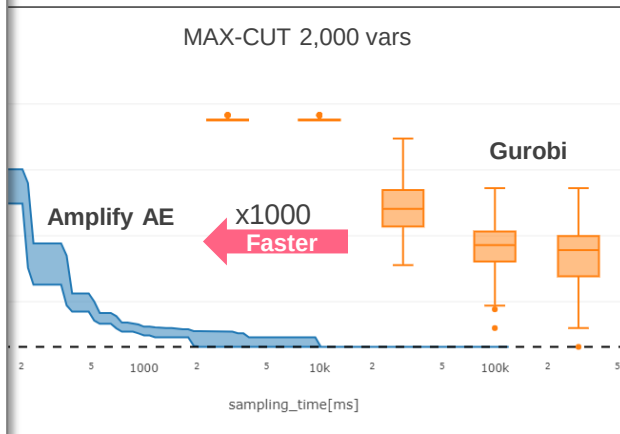
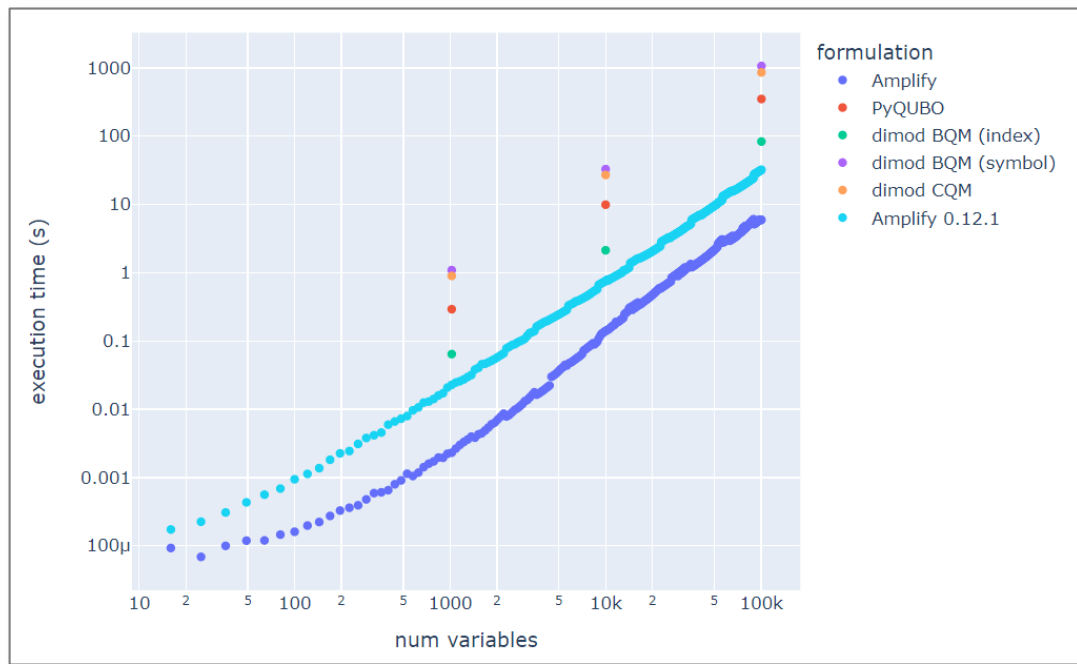
商用マシンでは最大規模・最高速レベル

- 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+) ^(*) / 1,000 (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

Fixstars Amplify SDK/AE パフォーマンス

Fixstars Amplify は最速レベルの定式化・求解速度を達成しています ←「高速に」



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_X8RufFbc4wzRlyxj8M5FIgorkcKIE?usp=sharing#scrollTo=joih5kayrIJt
Step2	https://colab.research.google.com/drive/1ssgrvSQ7cZrLhGzmaLZUJRYu2JJJeSOiv?usp=sharing#scrollTo=Ca8VcqcfFsXSi
Step3	https://colab.research.google.com/drive/1SIYCT_eS5abwzl_rldM-Vf9g2s-KihRp?usp=sharing#scrollTo=WKVyZz2wtV9L

質問は随時、Zoomの チャット か Q&A でお願ひします。
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

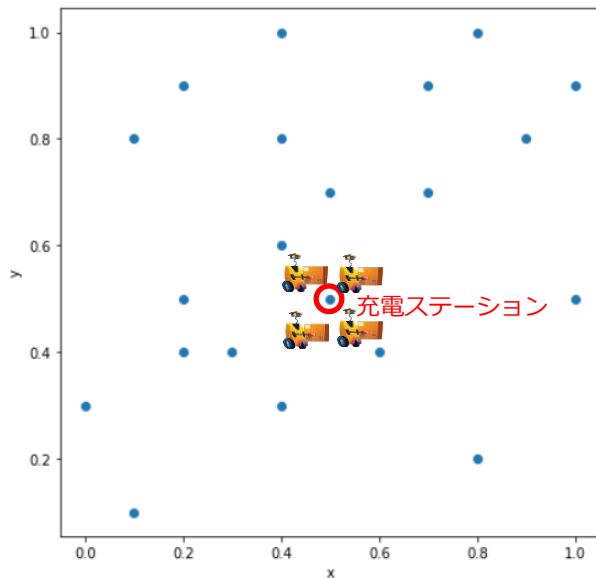
一言に「経路最適化」と言っても、搬送ロボット (AGV) が運用されている環境や要件などは様々で、それぞれに対する問題設定や定式化は異なります

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

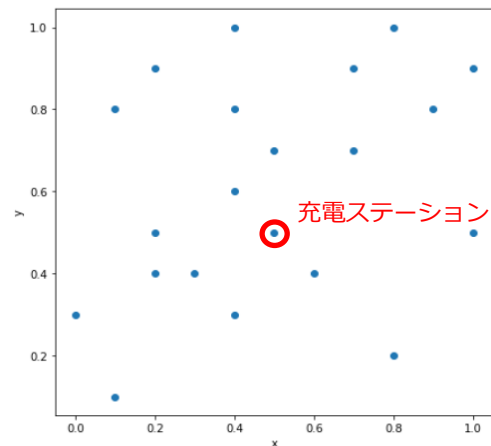
制約②: 各訪問先を回るのは1回のみ

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

```
# 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 (j)	0	0	0.54	0.22	...	0.64
	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	$q_{0,1}(0 \text{ or } 1)$	$q_{0,2}(0 \text{ or } 1)$	...	$q_{0,20}(0 \text{ or } 1)$
1	$q_{1,0}(0 \text{ or } 1)$	$q_{1,1}(0 \text{ or } 1)$	$q_{1,2}(0 \text{ or } 1)$	...	$q_{1,20}(0 \text{ or } 1)$
2	$q_{2,0}(0 \text{ or } 1)$	$q_{2,1}(0 \text{ or } 1)$	$q_{2,2}(0 \text{ or } 1)$	...	$q_{2,20}(0 \text{ or } 1)$
...	...	...	...	...	...
20	$q_{20,0}(0 \text{ or } 1)$	$q_{20,1}(0 \text{ or } 1)$	$q_{20,2}(0 \text{ or } 1)$	...	$q_{20,20}(0 \text{ or } 1)$

イジングマシン
で最適な(0,1)の
組合せを探す



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

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)
```

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

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

$$\sum_{i=0}^{20} q_{n,i} = 1 \quad n \in \{0,1,\dots,20\}$$

制約2: 各訪問先を回るのは1回のみ

→ one hot 制約

$$\sum_{n=0}^{20} q_{n,i} = 1 \quad i \in \{0,1,\dots,20\}$$

順番(n)	訪問先 (i)					
	0	1	2	...	20	
0	1	0	0	...	0	→ one_hot
1	0	0	0	...	1	→ one_hot
2	0	1	0	...	0	→ one_hot
⋮	⋮	⋮	⋮	⋮	⋮	
20	0	0	1	...	0	→ one_hot

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

$$\sum_{i=0}^{20} q_{n,i} = 1 \quad n \in \{0,1,\dots,20\}$$

制約2: 各訪問先を回るのは1回のみ

→ one hot 制約

$$\sum_{n=0}^{20} q_{n,i} = 1 \quad i \in \{0,1,\dots,20\}$$

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

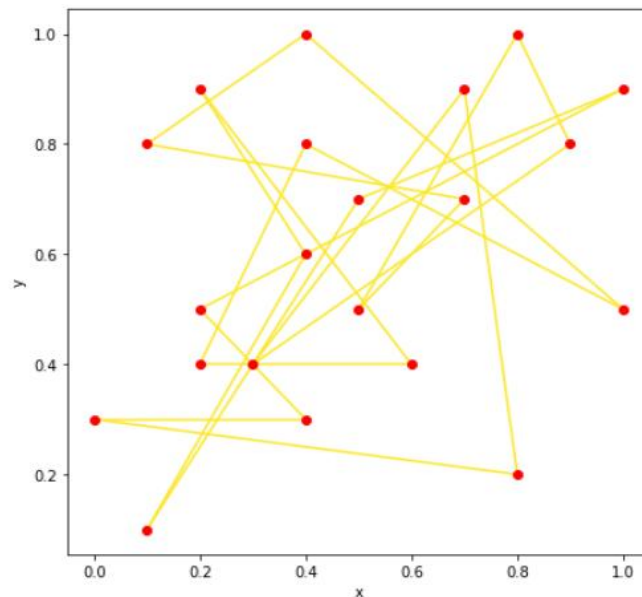
制約①: 各順番で回れる訪問先 (充電ステーション+各搬送先) は1か所

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

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

決定変数 × 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 (j)	0	0	0.54	0.22	...	0.64
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)
```

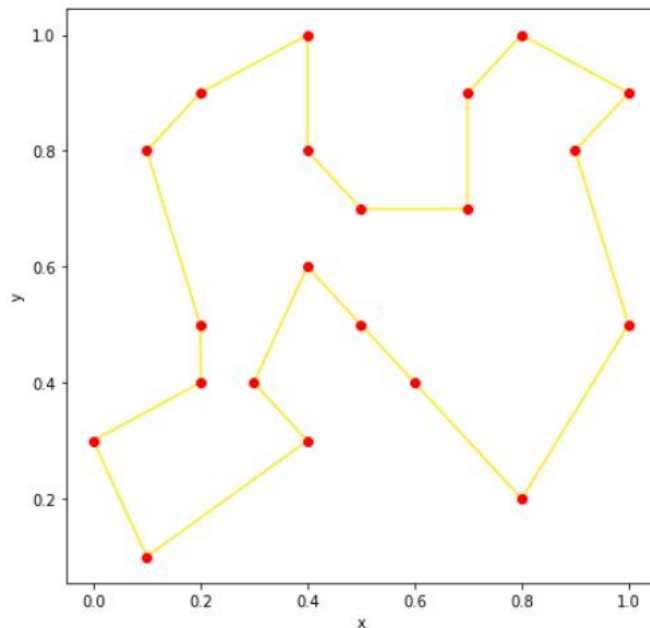
イジングマシンは、制約を満たす中で、このdistance (目的) を最小にする組合せを探します

```
# 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 ($n_vehicle = 4 \rightarrow n_visits = 24$)

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)

訪問先 (i)	0	1	2	...	20
順番(n)	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

```
# 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, remainder = divmod(number, divisor)
    ... return [quotient + 1] * remainder + [quotient] * (divisor - remainder)

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 ($n_vehicle = 4 \rightarrow n_visits = 24$)

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	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
#####

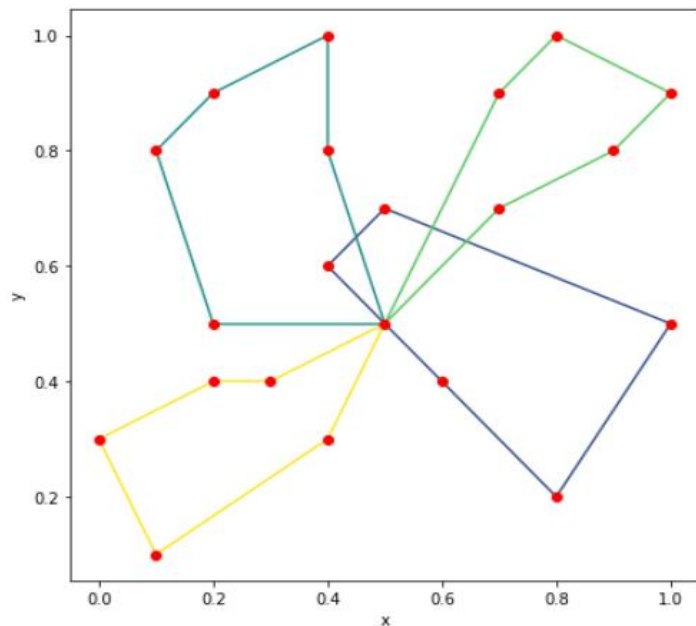
from amplify import one_hot

# 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

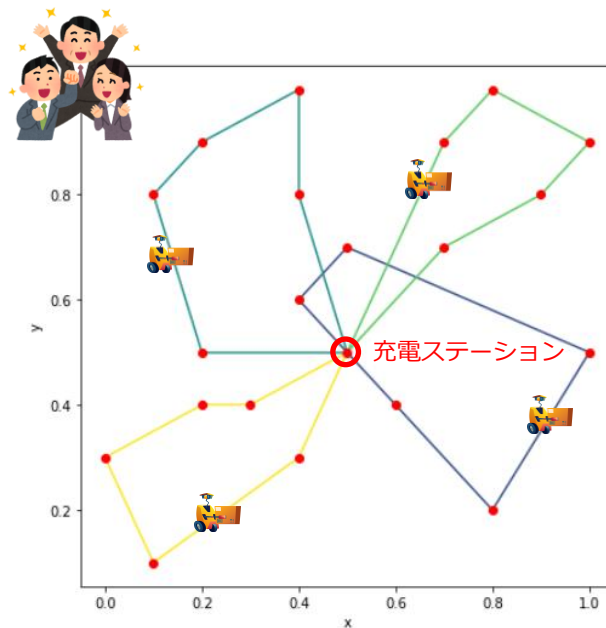
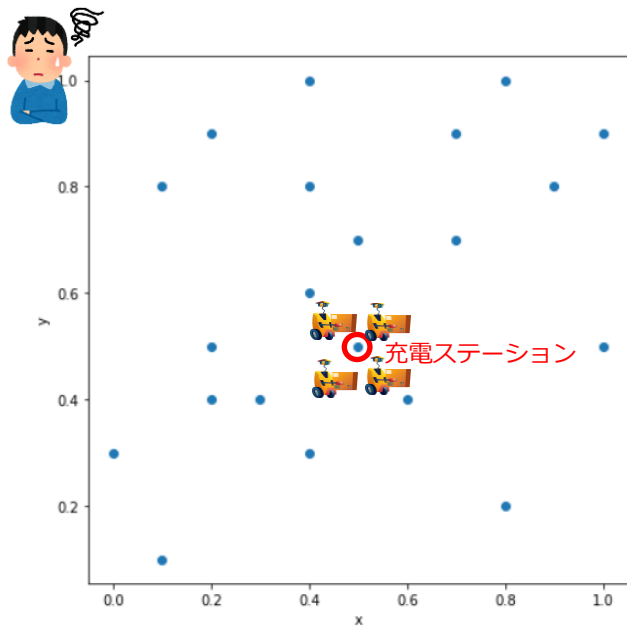
AGVを4台に増やします ($n_vehicle = 4 \rightarrow n_visits = 24$)



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 of 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



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

制約条件のみで
解く問題



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

目的関数 + 制約条件で
解く問題



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

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。

サンプルコード

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>

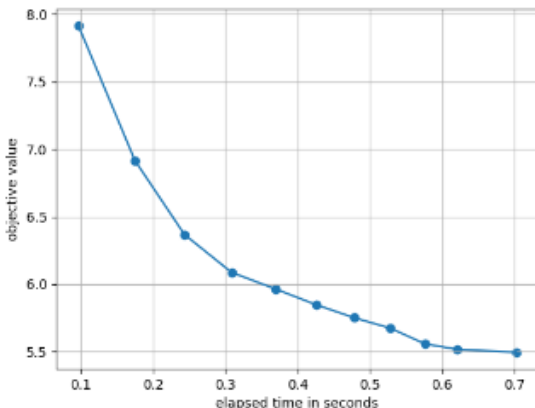


Tips: 適切なタイムアウト値の設定について

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)
```


Wrap Up

最適配置自動化サービス（物流梱包業務のDX）

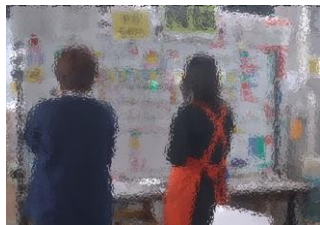
<https://www.fixstars.com/ja/services/cases/amplify-bellemaison>

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



生産計画最適化（電気機器製造メーカー A社様）

複数の製品事業部から様々なプリント基板の注文を受け、生産を行う部門

課題

生産する基板に応じて製造装置の部品や材料を交換する「段取り時間」が必要。段取り時間を考慮した効率的な生産スケジュールを作成したい
従来は、専任者が、一日数回・毎回数十分かけて経験に基づいてスケジュールを作成。更なる生産性向上やノウハウ継承のため、生産スケジュール作成の自動化に着手



効果

生産スケジュール作成の時間・コストの大幅な削減！
(一日あたり数時間 → 数分)

段取りのための製造装置の停止回数の削減！
(10%以上削減)

最適化未経験のご担当者様1人がプログラム試作開始から約1~2カ月間取り組んでこの効果を実現
現在は試作段階で、実運用に向けてモデルを改良中！

次期フェーズでは、Amplifyの活用領域の拡大を検討中！



Fixstars Amplify を活用したアカデミアの研究事例

	大学	研究室	論文タイトル
量子アニーリング/イジングマシンに関する研究	早稲田大学	戸川研究室	イジングマシンによる制約付きグラフ彩色問題の彩色数最小化手法 (リンク)
	東京大学	Prof. Codognet	Modeling the Costas Array Problem in QUBO for Quantum Annealing (リンク)
	名古屋大学	片桐研究室	Amplifyを用いたCMOSアニーリングマシンの特性の分析 (リンク)
	東北大学	小松研究室	組み合わせクラスタリングによるアニーリングマシンの評価 (リンク)
応用研究	慶應大学	村松研究室 (材料工学)	Phase-fieldモデルの量子アニーリングシミュレータ (リンク)
	東京大学	長谷川研究室 (量子ゲート)	ISAAQ:イジングマシンを活用した量子コンパイラ (リンク)
	山梨大学	鈴木研究室 (情報工学)	量子アニーリングによる疎行列直接解法向けフィルイン削減オーダリング (リンク)
	東京大学	津田研究室 (MI)	Designing metamaterials with quantum annealing and factorization machines (リンク)
	京都大学	野田研究室 (電子工学)	量子アニーリングを活用したフォトニック結晶レーザーの構造最適化 (リンク)
	東京大学	津田研究室 (MI)	Chemical Design with GPU-based Ising Machines (リンク)

量子アニーリング・イジングマシンを活用したブラックボックス最適化 (FMQA)

Fixstars Amplify ユーザー様インタビュー

Amplify インタビュー 検索

amplify.fixstars.com/ja/customers/interview

・ 業務・研究開発利用



顧客事例 2024年5月
日本テレビ放送網
数値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
食品製造業A社
従業員が納得する最適な人員シ
フトを実現



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



ユーザーインタビュー 2023年11月
株式会社タアフ
多品種少量生産の工程にマッチ
した生産計画アプリを開発



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

・ 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
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・イメージングマシンの活用

Fixstars Amplify ユーザー様インタビュー

Amplify インタビュー 検索

amplify.fixstars.com/ja/customers/interview



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

お話を伺った方々



非住宅建築用ZEB設計サポートツールの開発



アニーリングマシンを用いた学校給食推薦システムの開発



個別指導塾向けのコマ組の自動化アプリ「塾コマ」の開発



アニーリングマシンによる新規配送最適化手法の開発



アニーリングマシンを用いた合金触媒の探索・解析支援ツールの開発

組合せ最適化問題の概要

数あるレシピの中から、複数の条件を満たしながら、より良い組み合わせを選択する組合せ最適化問題。児童、生徒の学年に合わせた目標栄養価やカロリー、設定した費用などの条件を満たすような1か月分の学校給食の献立を作成します。

決定変数

レシピ数 (i) × 日数 (r) の2次元の決定変数 (レシピiを日にちrに選択するとき1、しないとき0)

決定変数のイメージ

		日数(r)					
		4月1日	4月2日	4月3日	4月4日	4月5日	...
レシピ (i)	ごはん	q (0 or 1)	q (0 or 1)	q (0 or 1)	q (0 or 1)	q (0 or 1)	
	ばん	q (0 or 1)	q (0 or 1)	q (0 or 1)	q (0 or 1)	q (0 or 1)	
	パスタ	q (0 or 1)	q (0 or 1)	q (0 or 1)	q (0 or 1)	q (0 or 1)	
	焼き魚	q (0 or 1)	q (0 or 1)	q (0 or 1)	q (0 or 1)	q (0 or 1)	
	とんかつ	q (0 or 1)	q (0 or 1)	q (0 or 1)	q (0 or 1)	q (0 or 1)	
	・						

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)

月額利用料

計算環境

利用GPU
(マルチGPUオプションあり)

1ジョブの実行時間
(実行時間延長オプションあり)

月間実行回数上限
(実行回数追加オプションあり)

SQBM+の利用

D-Waveの利用

サポート

Plus オプション

ベーシック

スタンダード

プレミアム

Sプレミアム

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)

スモール

NVIDIA V100

ミディアム

NVIDIA V100

ラージ

NVIDIA A100

スーパーラージ

NVIDIA H100

10秒

1分

10分

15分

制限の可能性あり

無制限

無料 (1ヶ月無料
プログラム)

SQBM+オプション: 30万円 (1名)、90万円 (最大5名)

無料プログラム (3分/月)

ベーシック

スタンダード

プレミアム

プレミアム

-

月額50万/人

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

20万円 (1アプリ)

40万円 (1アプリ)

60万円 (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)

ミディアム

NVIDIA V100

1分

ラージ

NVIDIA A100

10分

スーパーラージ

NVIDIA H100

15分

- (制限をかける可能性あり)

-

-

スタンダード

スタンダード

スタンダード

-

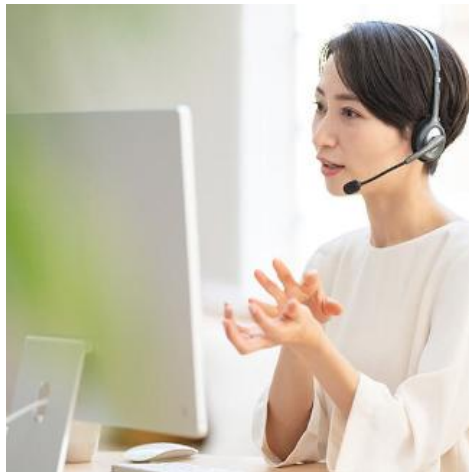
クラウド利用料: Plusオプション

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



開発支援サービス

ユーザーの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

評価サービス

ユーザー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。本サービスでは、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、ユーザーにプログラムやデータを送付してもらい、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
- 評価サービスにかかる期間については個別相談となります

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ますのち、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。

第1回
3時間

...
1週間

第2回
3時間

課題
2週間

第3回
1.5時間

...
2週間

第4回
1.5時間

今後のセミナーのご案内

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

今後も無料セミナーを開催します！第二部からの参加も可能です

2024/12/12 (木) 「経路最適化」

- ・ 会社および量子コンピューティングクラウド「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のご紹介
- ・ Wrap Up & QA

2025/1/30 (木) 「ブラックボックス最適化を活用した材料開発」

- ・ 会社および量子コンピューティングクラウド「Fixstars Amplify」のご紹介
- ・ Fixstars Amplify を用いたワークショップ
 - ・ ブラックボックス最適化
- ・ 事例や今後の進め方のご紹介
- ・ Wrap Up & QA

2025/2/19 (水) 12:00-13:00 and 14:00-15:00

「定式化ベースの最適化の技術解説」

- ・ 会社および量子コンピューティングクラウド「Fixstars Amplify」のご紹介
- ・ 定式化ベースの最適化の事例紹介
- ・ 躰きポイントとTips
- ・ Wrap Up & QA

ご質問・ご不明点がございましたら、お問い合わせフォームでご連絡下さい

<https://amplify.fixstars.com/ja/contact>

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**

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