共創パートナー様向け
最適化セミナー

パートナー事例と最適化アプリのサンプルコードの紹介

FIXSTARS Amplify

目次

1. 最適化クラウドサービス「Fixstars Amplify」
2. 共創開発の事例
3. サンプルコードの紹介

講師紹介

株式会社 Fixstars Amplify

小林 隆之

商品企画部　シニアディレクター

大学院修士過程修了後、味の素に入社し、生産技術開発、新事業開発、グループのDX戦略担当等を経験。『最適化技術』を世界に広げるため、株式会社Fixstars Amplifyへ転職。エンタープライズ事業の責任者として、業界を問わず様々な業務の最適化を実現するプロジェクトに参画している。

株式会社 Fixstars Amplify

末次 健太郎

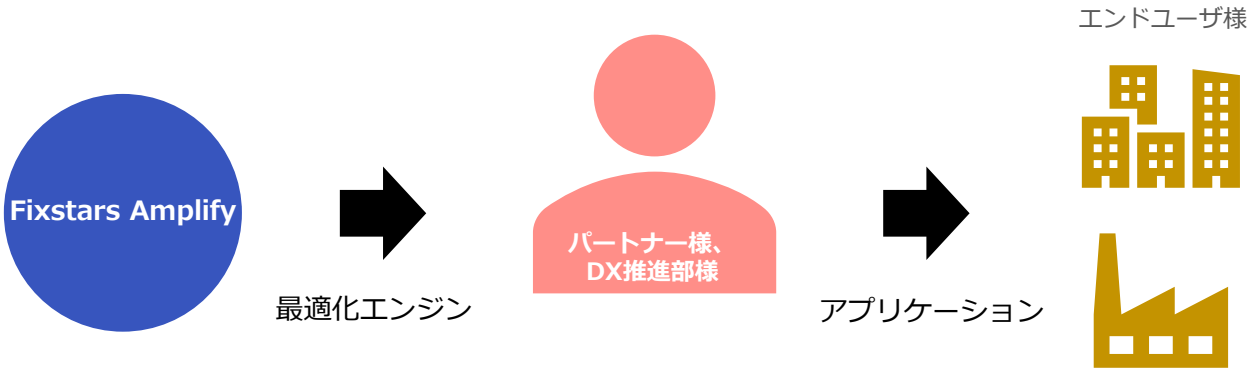
リードエンジニア

学生時代には素粒子物理学を専攻し高エネルギー加速器実験に取り組む。その後、計測システムの設計開発でプログラミングとプロジェクト管理の経験を積む。多様な知識をキャリアに活かしたいと考えて、株式会社Fixstars Amplifyに転職。さまざまな業種において、最適化技術を実業務に直結したシステムをエンドユーザー様およびパートナー様と共創している。

本セミナーの趣旨

本セミナーでは、Fixstars Amplifyの最適化技術を使いアプリケーションにするヒントを提供します。お客様に最適化アプリの提供を考えているパートナー様や、社内ユーザーにアプリケーションを提供するDX推進部の方にも参考になるセミナーとなっています。

Fixstars Amplifyの定式化やBBOについて学びたい方は、ぜひ研究・開発者向けセミナーにもご参加ください。



最適化クラウドサービス
「Fixstars Amplify」

フィックスターズグループの基本情報

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

※グループ会社一覧：https://www.fixstars.com/ja/company/group

Fixstars Solutions, Inc.
連結子会社
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
連結子会社
量子コンピューティングのクラウド事業を運営

(株)Sider
連結子会社
開発支援ツールCloneTrackerを提供

(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

Fixstars Amplify の歩み

フィックスターズの持つ高い技術力で
最新の最適化計算ができる環境を提供してきました。

2025年
Fixstars Amplify AE v1リリース
登録組織数　1000超
累計実行回数　1億回突破

2024年
産総研次世代スパコンABCI-Qへの採用
光量子コンピュータのクラウド化研究
登録組織数　700超

2023年
新製品 Fixstars Amplify Scheduling Engine リリース
累計実行回数3,000万回突破

2022年
Fixstars Amplify がGurobi、IBM-Quantumをサポート
累計実行回数1,000万回突破

2021年
量子アニーリングクラウドサービス「Fixstars Amplify」提供開始
Fixstars Amplifyを設立
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

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」

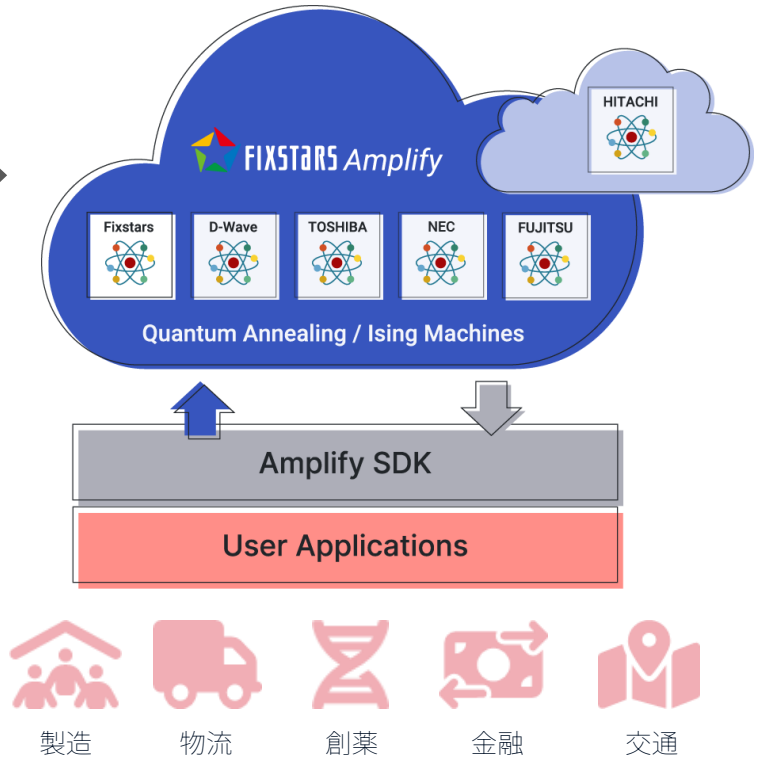
2018年
NEDOのプロジェクトに採択
「イジングマシン共通ソフトウェア基盤の研究開発」

2017年
日本で初めて
D-Wave Systems社と提携

Fixstars Amplify

最適化クラウドサービス

量子技術など高性能コンピュータによる
高精度・高効率で使いやすい最適化計算環境を
クラウドで提供しています



登録企業・大学等
1100
以上

累計実行回数
1.1億回
以上

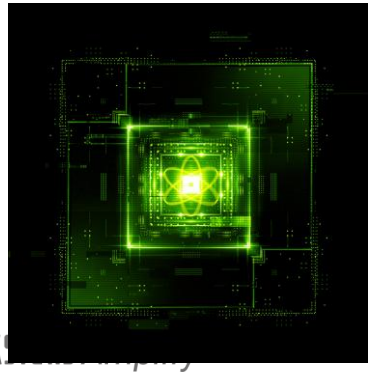
FIXSTARS Amplify

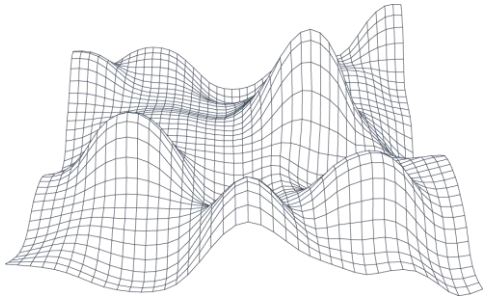
Amplify AEの強み

高性能

ハードウエアの性能を最大限引き出す

Amplify AEは、シミュレーテッドアニーリングを基盤とした独自のアルゴリズムを実装しています。当社の特許技術を用いた並列化手法により、GPUの計算能力を極限まで活用し、探索プロセスを劇的に高速化します。
従来は膨大な計算時間を要した大規模で複雑な問題に対しても、実用的な時間内で精度の高い解を導き出します。さらに、Amplify AE は求解パラメータの調整が不要で、専門知識がない方でも簡単に高い性能を引き出せます。





直感的

現実世界の複雑な制約に柔軟に対応

Amplify AEは、最大4次までの多項式で表現される目的関数や制約条件を扱うことが可能です。これにより、現実の複雑な問題の定式化を容易にし、より高度で精度の高い最適化計算を実現します。
さらに、直感的に使える「Fixstars Amplify SDK」と組み合わせることで、多様な最適化問題をシンプルなコードで記述でき、Amplify AEの強力な計算能力を最大限に引き出すことが可能です。

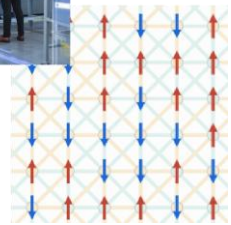
実用向け

研究から実用まで、選べる求解モード

用途に応じた2つのモードを提供しています。

01. **制約モード**
実社会の複雑な問題をそのまま入力できるモードです。複雑な制約条件はソルバーが内部で自動的に処理します。

02. **PUBO・QUBOモード**
量子アニーリングやイジングモデルの研究用途に最適なモードです。高次の項も扱える汎用イジングマシンとして、高精度に動作します。



研究・開発・実務の幅広い現場で利用が拡大しています

量子技術を使ったアニーリングエンジンやスケジュール最適化エンジン、お客様専用のエンジンなど高性能な最適化エンジンで、お客様の課題解決を実現してきました。

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　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

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

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

FIXSTARS Amplify

パートナー事例の紹介

日本テレビ放送網株式会社

地上波テレビの広告割り当てエンジンを提供

日本テレビ放送網株式会社様
放送法による基幹放送事業及び一般放送事業、メディア事業、その他放送に関連する事業を行う

背景

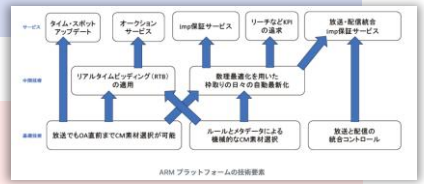
インターネット広告との差を埋める新技術が必要

- デジタル技術の進化により、地上波広告の高価値を最大化するための広告枠選定が課題に。
- 契約時に決定された広告露出量をリアルタイムで最適化する技術が求められている。

最適化

数理モデルとアルゴリズムで最適化を実現

- 地上波広告の仕様をモデル化し、PoCでアルゴリズムを改良。
- 計算時間やコストを考慮した実現可能なプラン（専用最適化エンジン）を提案。



成果

インプレッション保証＆KPI達成を自動化

- 広告主の目標達成を支援する広告枠割り当てが可能に。
- 2024年度末からARMプラットフォームで運用開始予定、さらに名古屋地区の中京テレビ放送も参画。

FIXSTARS Amplify

物流センターの従業員最適配置エンジン

住友商事株式会社様

Fixstars Amplify Annealing Engine → 住友商事様 → ベルメゾンロジスコ様

背景

毎日、リーダーたちが集まって次の日のシフトを2時間掛けて作成

従業員スキルや業務負荷を考慮したシフト作成に2時間を要し、リーダーに大きな負担がかかっていた

最適化

生産性向上と従業員の負荷軽減を両立する最適化エンジンを導入

目的：全ジョブの作業効率を最大化

制約条件：スキルマッチ、対人相性の考慮など

成果

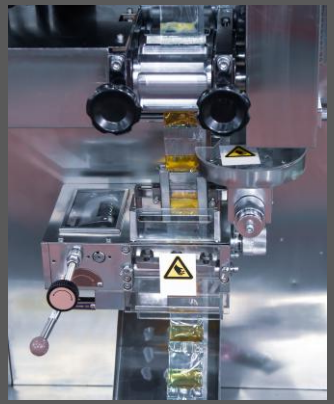
生産性の高い計画の立案を実現

- 生産性の高いシフト計画を迅速に自動生成
- 欠勤や急な変更にも柔軟対応
- 配置決定の負担を軽減し、リーダーの業務を効率化

FIXSTARS Amplify

充填（包装）ラインの生産計画最適化エンジン

システム開発企業様



Fixstars Amplify Annealing Engine → システム開発企業様 → 食品会社様

背景

生産計画立案の属人化により、担当者が休みを取れない！

ベテラン社員が液体調味料を充填する生産計画を立案している。
生産日は休むことができず、残業も多い。そんな働き方を見ているため、後任が育たない。

		1号機	2号機	3号機	・・・	10号機
9/14	製品	A	K	S		P
	数量	30,000	5,000	21,000		5,000
9/15	製品	A	J	L		B
	数量	32,000	16,000	30,000		32,000
9/16	製品	A				
	数量	29,000				
9/17	製品					

最適化

納期を守り、最も設備稼働率の高くする最適化エンジンを導入

制約条件：製造可能日、納期、設備の取り合いなど

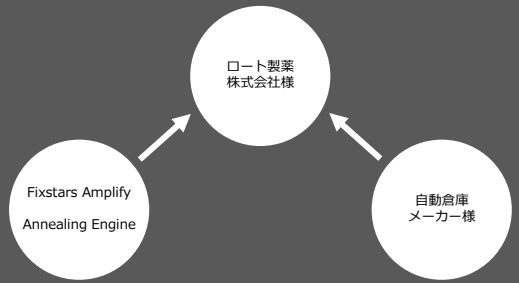
成果

- ベテランが納得する計画を自動生成し、属人化を解消。
- 製造制約をすべて満たし、人より効率的な計画を実現。
- 業務負荷が軽減され、後任への引継ぎもスムーズに進行。

FIXSTARS Amplify

自動倉庫でのサイバーフィジカルシステムの実現

ロート製薬様



背景

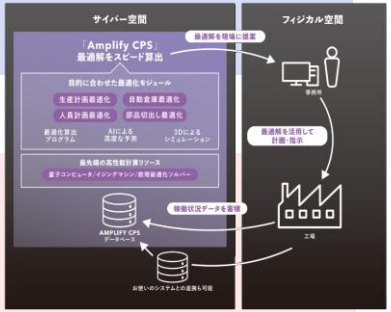
サイバーフィジカルシステムの導入で生産性向上を加速

- 仮想空間で蓄積・分析したデータを活用し、最適化されたプロセス実現
- トラックの渋滞削減を目指し、ピッキング時間の短縮に着手

最適化

自動倉庫の夜間棚替えを最適化する新しい仕組み

- 目的：ピッキング時間を最短化する棚替え計画を自動生成
- 手法：AIと最適化技術を活用し、効率的な棚配置を実現



成果

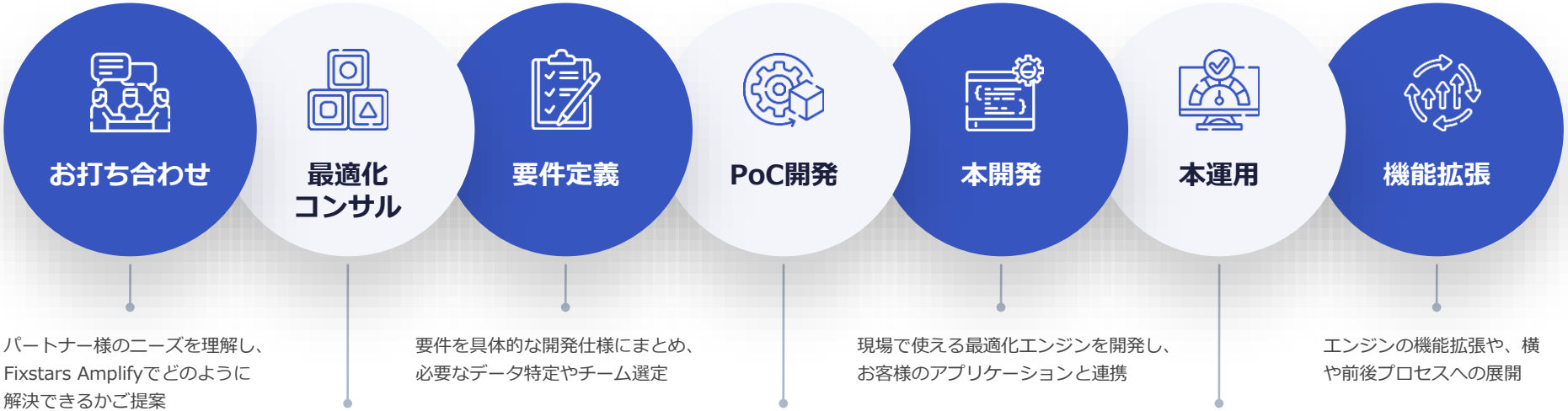
ピッキング作業時間を短縮し、効率を最大化

- 常に最適な棚替えを自動実行し、人的介入を削減！
- ピッキング時間を**10%**短縮し、物流効率を大幅改善

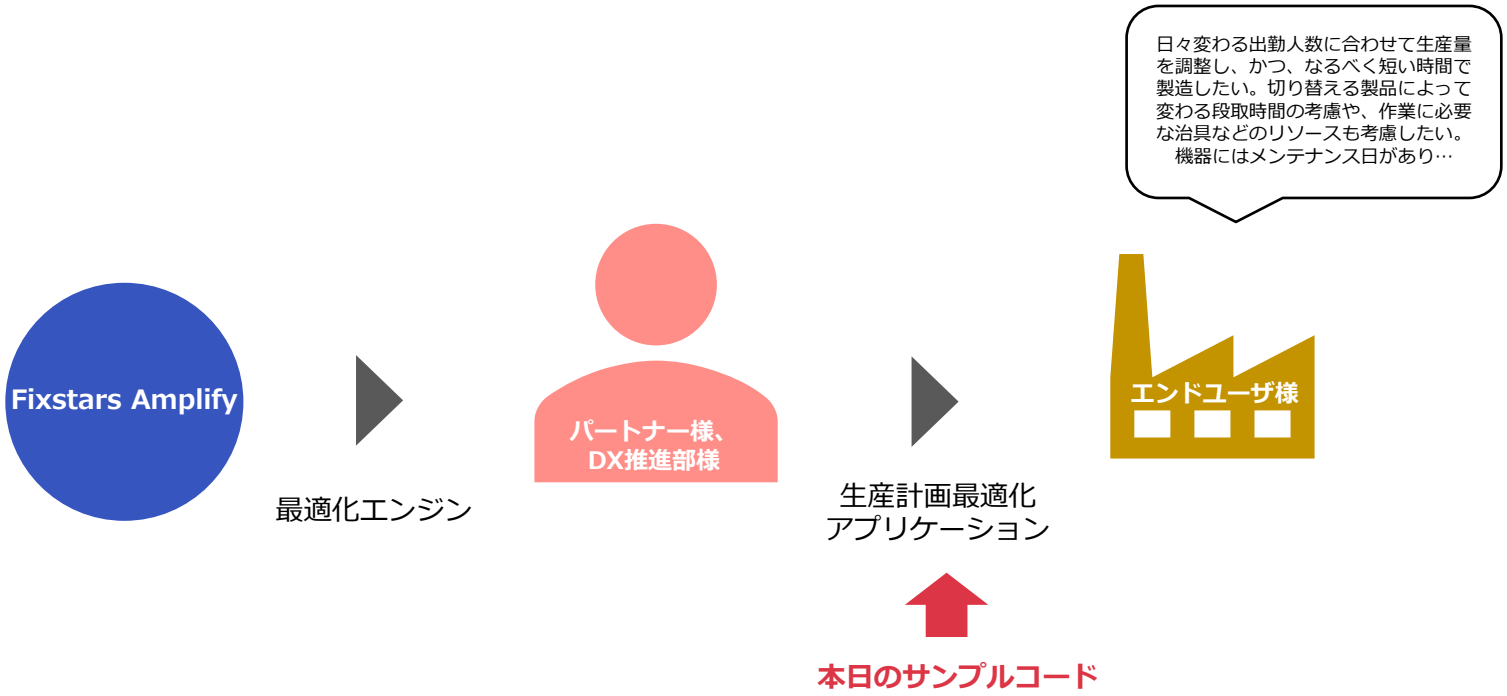
FIXSTARS Amplify

取組みステップ

パートナー様の環境や取組み内容に合わせて柔軟に対応します



サンプルコードのご紹介



Fixstars Amplify クイックスタート

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 = AmplifyAEClient()
client.token = "Amplify AE のアクセストークン"
client.parameters.time_limit_ms = 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 の無料トークンの取得（1）

- ご自身の PC のブラウザ上で Python の簡単なプログラムを実行して簡易的に動作確認できます。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/

Fixstars Amplify の無料トークンの取得（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/1xdrIVKEK0ndQYMMIhjGXFb3yIw4Ci05M

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)
```

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

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

制約条件のプログラミング

```python
from amplify import *

# Variables
g = VariableGenerator()
q = g.array("Binary", 4)

# Constraints
c1 = equal_to(q[0] + q[1] + q[2], 2)    # q[0] + q[1] + q[2] == 2
c2 = less_equal(q[0] + q[2] + q[3], 1)  # q[0] + q[2] + q[3] <= 1

# Model and solve
model = Model(c1 + c2)
result = solve(model, client)

# Result
values = result.best.values
objective = result.best.objective

>>> f"result: {q} = {q.evaluate(values)} (f = {objective})"
result: [q_0, q_1 ,q_2, q_3] = [0., 1., 1., 0.] (f = 0.0)
```

制約条件の生成関数

- `equal_to` / `one_hot`
- `less_equal`
- `greater_equal`
- `clamp`

制約条件の和
→ 制約条件リスト

多項式と制約条件の和
→ `amplify.Model` クラス

Amplify による制約条件の定式化

- Amplify SDK が制約条件(ペナルティ関数)の生成・管理をサポート
- 多項式fと定数vに対する制約条件オブジェクトの生成関数
  - `equal_to(f, v)`: $f = v$
  - `one_hot(f)`: $f = 1$
  - `less_equal(f, v)`: $f \leq v$
  - `greater_equal(f, v)`: $f \geq v$
  - `clamp(f, v1, v2)`: $v_1 \leq f \leq v_2$
- ペナルティ関数を与えて生成する場合
  - Constraint() クラスを構築する
    - 詳細はドキュメントを参照してください
    - https://amplify.fixstars.com/ja/docs/amplify/v1/penalty.html#specify-penalty

デモアプリの紹介

Amplify AE を活用した生産計画最適化の実装紹介

本実装紹介のゴール

数理モデルの構築、制約の実装、多目的最適化の技術習得

紹介する内容

製造現場の複雑な制約（納期、人員、設備）を Fixstars Amplify AE で解決するアプリケーションの実装例を紹介します。

多目的最適化の実現
- メイクスパン（総工期）の最小化による生産効率の向上
- 人員の平準化（人数充足率の最大化）による現場負荷の安定化

現場実務に即した高度な制約の実装
- 段取り替え時間の考慮
- 共有リソース（金型や特定の設備、技能や資格など）の競合回避の記述方法
- 既存計画を維持するためのジョブの固定

ポイント

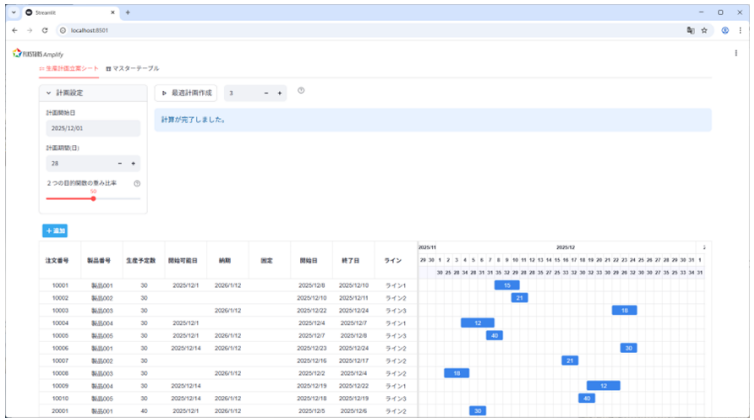
- Fixstars Amplify AEを使うことで、膨大な組み合わせの中から現実的な計算時間で優れた計画を見つけ出します。
- 「目的関数」と「ハード制約」を適切に使い分けて実行可能かつ高品質な計画を両立できます。
- 多くの実案件に基づいた、現場担当者が使いやすいUI構成のベストプラクティスを紹介します。

アプリ仕様

生産計画立案シート

- ✓ ボタン一つで最適化計算して、結果のガントチャートを自動生成します。
- ✓ ジョブ情報を表形式とガントチャートで連動表示します。

本アプリの製造モデル
・製品ごとに(最大)日産数が決められている。
・連続製造、つまり、生産予定数を満たすまでそのラインで作り続ける。

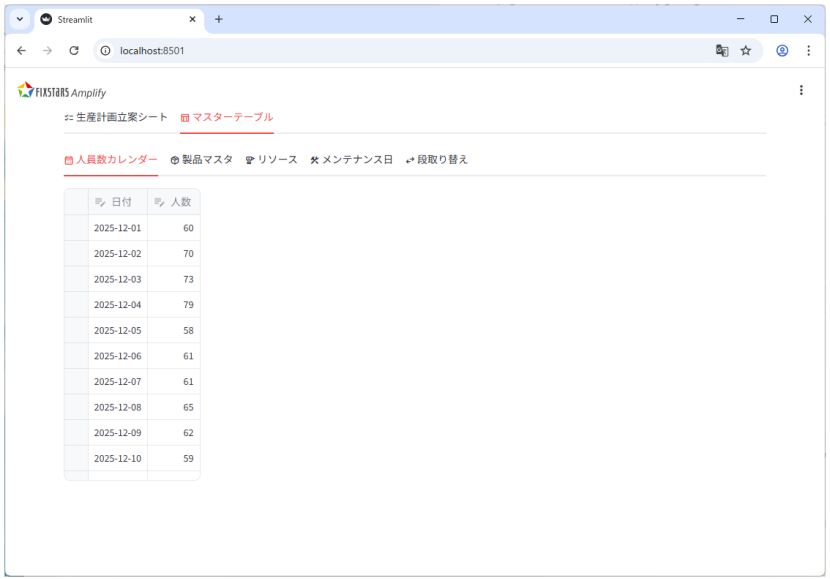


ジョブテーブル
- 生産する製品の製品番号
- その生産予定数
- 開始可能日/納期
- 固定フラグ
- 最適化計算で割り当てた開始日/終了日/ライン

計算の条件
- 計画開始日/期間
- 2つの目的関数の重み比率
- 計算時間

マスターテーブル

カレンダー、製品マスタ、共有リソース、メンテナンス日、段取り替えをタブで一括管理します。



- 人員数カレンダー
- 製品マスタ
- リソース
- メンテナンス日
- 段取り替え

[マスターテーブル] カレンダー

人員数カレンダー

- その日に作業可能な人数
- 定義されていない日は非稼働日
- その非稼働日をまたいで製造可能

メンテナンス日

- そのラインを使用できない日
- その日をまたいで製造可能

[マスターテーブル] 製品情報


人員数カレンダー　製品マスタ　リソース　メンテナンス日

製品番号	ライン	最大日産数	必要人数	必要リソース{個数}
製品001	ライン1	10	15	治具
製品001	ライン2	20	30	金型{2}
製品002	ライン2	15	21	金型{2},治具
製品003	ライン1	12	18	
製品003	ライン2	12	18	ホイスト,玉掛技能者
製品003	ライン3	12	18	金型
製品004	ライン1	8	12	
製品005	ライン3	25	40	ホイスト,フォークリフト


人員数カレンダー　製品マスタ　リソース　メンテナンス日

リソース名	数量
治具	5
金型	3
ホイスト	4
玉掛技能者	3
フォークリフト運転技能者	2

製品マスタ

- その製品をそのラインで製造する場合の情報
  - ✓ 最大日産数
  - ✓ 必要人数
  - ✓ 必要リソース: カンマ区切り {}内の数字は必要数量

リソース

- 共有して仕様するリソースの名前とその数量

[マスターテーブル] 段取り替え

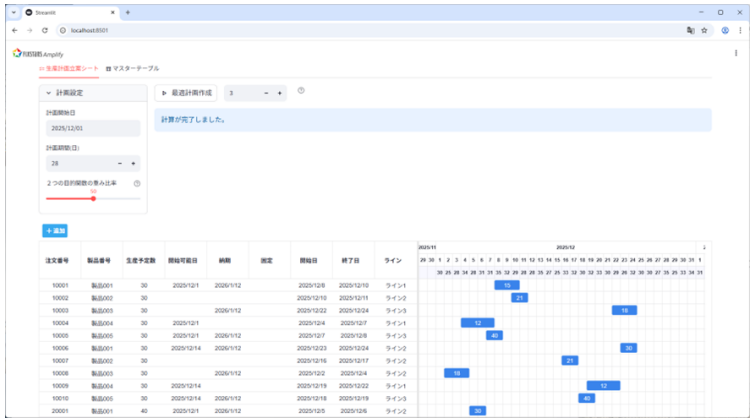
「切替元」と「切替先」の製品番号と、その「切替日数」を1次元テーブルに定義します。
右側のプレビューで2次元の表として確認可能です。

コード解説

生産計画立案シート

- ✓ ボタン一つで最適化計算して、結果のガントチャートを自動生成します。
- ✓ ジョブ情報を表形式とガントチャートで連動表示します。

本アプリの製造モデル
・製品ごとに(最大)日産数が決められている。
・連続製造、つまり、生産予定数を満たすまでそのラインで作り続ける。



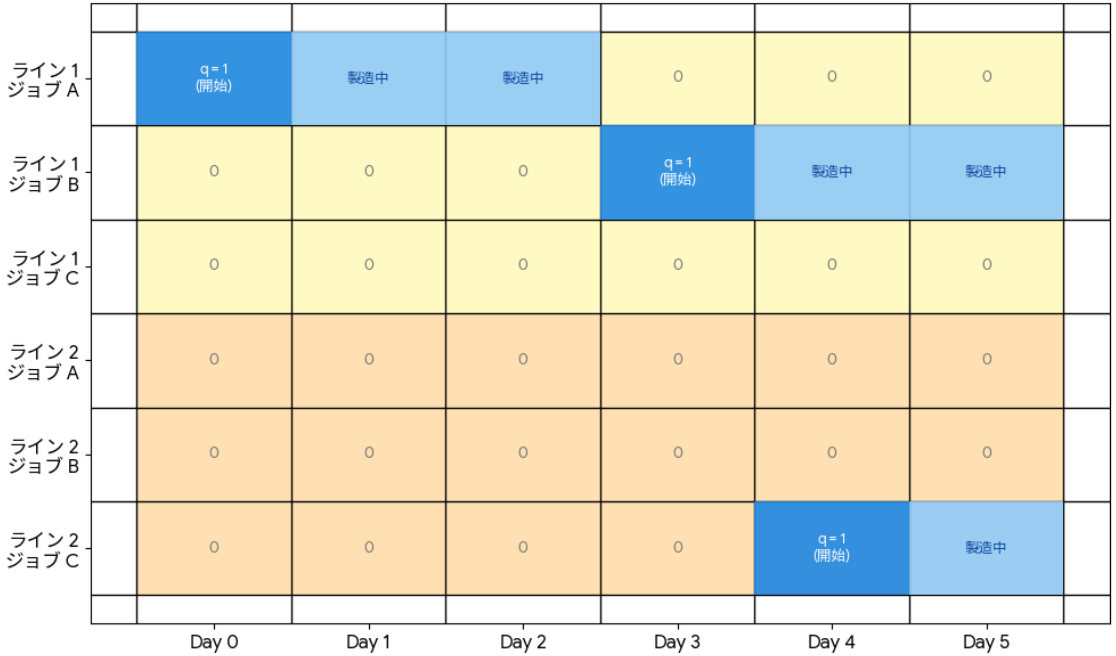
ジョブテーブル
- 生産する製品の製品番号
- その生産予定数
- 開始可能日/納期
- 固定フラグ
- 最適化計算で割り当てた開始日/終了日/ライン

計算の条件
- 計画開始日/期間
- 2つの目的関数の重み比率
- 計算時間

決定変数の設計

どのジョブがどのラインでいつから製造開始するか？

⇒ 膨大なマス目の中から、制約を満たしつつ最も効率的な場所に 1 を書き込む「パズル」



決定変数の設計

コード抜粋 (sched.py):

```
#バイナリ変数の3次元配列を生成
gen = VariableGenerator()
q = gen.array("Binary", shape=(num_lines, num_jobs, num_days))
```

決定変数 q[m, j, d]

ライン m でジョブ j が日付 d に「開始する」かどうかをバイナリ(0/1)変数の3次元配列で表現します。

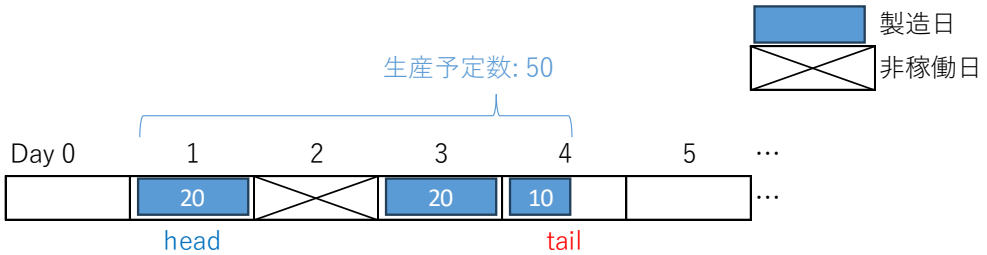
「ジョブは(全ラインと全期間で)1回だけ処理される」制約を `one_hot(q[:, j, :])` のように実装します。

コード抜粋 (intable_extends.py):

```
#製造に必要な日数を計算（非稼働日をスキップ）
for d in range(head, intbl.num_days):
 if is_disabled_slot(intbl, line, jid, d):
  skipped.append(d)
  continue
 prod += prod_per_day
 if quantity <= prod: # 予定数に達したら終了
  return TailSlot(d, tuple(skipped), production)
```

連続製造と日産数

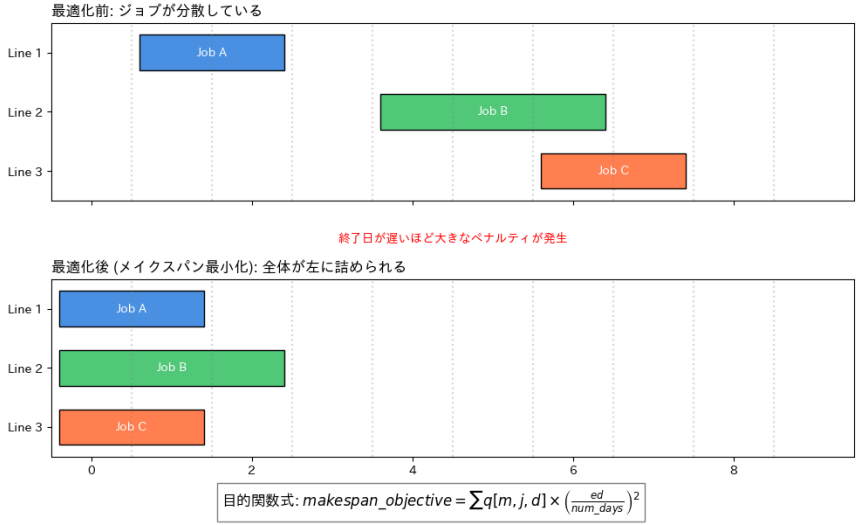
各製品の「最大日産数」に基づき、生産予定数を満たすまで作り続けます。



目的関数

① メイクスパン最小化

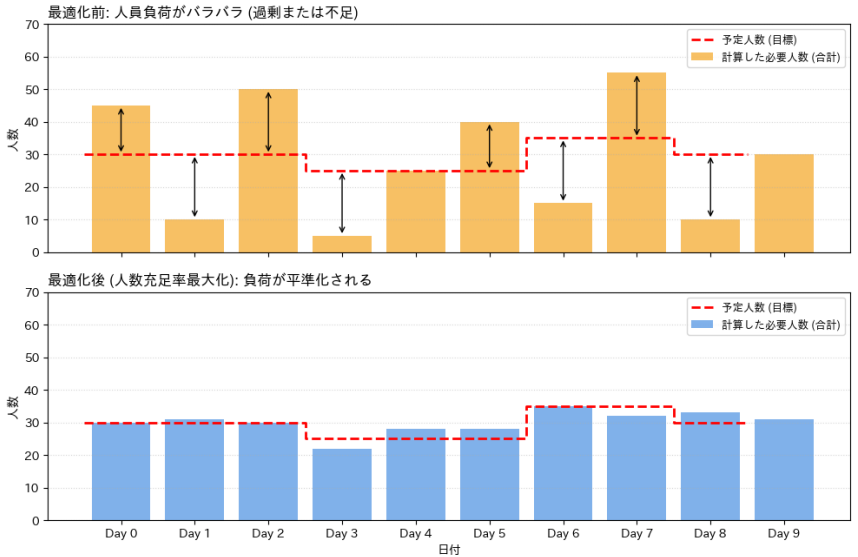
全体の製造終了をできるだけ早くします。



目的関数式: $makespan\_objective = \sum q[m, j, d] \times \left(\frac{ed}{num\_days}\right)^2$

② 人数充足率の最大化

日ごとの人数を、予定人数に可能な限り近づけます。



目的関数式: $headcount\_objective = \sum \left(\frac{hc\_d}{required} - 1\right)^2$

多目的最適化

① メイクスパン最小化
全体の製造終了をできるだけ早くします。

```
#各ジョブの終了日(ed)が遅いほど目的関数値を大きくする
makespan_objective = Poly()
for m, j, d のループ:
  ed = tail_slot(intbl, m, j, d).tail
  makespan_objective += q[m, j, d] * (ed / num_days) ** 2
```

② 人数充足率の最大化
日ごとの人数を、予定人数に可能な限り近づけます。

```
#(当日の必要人数 / 予定人数 - 1)^2 を最小化
headcount_objective = Poly()
for d in range(num_days):
  required = calendar_headcount(intbl.calendar, _date(d))
  hc_d = Poly() #当日の全ラインの合計人数
  for m, line in enumerate(lines):
    for j, job in enumerate(jobs):
      hc_d += product_of(…) ["必要人数"] * polys[m][j][d]
  headcount_objective += (hc_d/required - 1) ** 2
```

```
w1, w2 = (intbl.weight / 100, (100 - intbl.weight) / 100)
objective = w1 * makespan_objective + w2 * headcount_objective
model = objective + constraints
```

係数で2つの目的関数の寄与の重みを与える。
実運用では、対象にしている問題や入力データを考慮して適切な重みを調整する。

ジョブの固定

よくあるユースケース:
最適化計算して、その計画で製造する。しばらく(数日ほど)して、新たなオーダーが発生した。
製造中のジョブはそのままで、製造が始まっていないジョブと新たなオーダーのジョブで再度計算したい。
⇒ 製造中のジョブを「固定」し、新たなオーダーのジョブを追加し、ふたたび最適化計算する。

注文番号	製品番号	生産予定数	開始可能日	納期	固定	開始日	終了日	ライン
10001	製品001	30	2025/12/1	2026/1/12		2025/12/8	2025/12/10	ライン1
10002	製品002	30			✓	2025/12/10	2025/12/11	ライン2
10003	製品003	30		2026/1/12		2025/12/22	2025/12/24	ライン3
10004	製品004	30	2025/12/1			2025/12/4	2025/12/7	ライン1
10005	製品005	30	2025/12/1	2026/1/12	✓	2025/12/7	2025/12/8	ライン3
10006	製品001	30	2025/12/14	2026/1/12		2025/12/23	2025/12/24	ライン2
10007	製品002	30				2025/12/16	2025/12/17	ライン2
10008	製品003	30		2026/1/12		2025/12/2	2025/12/4	ライン2

ジョブ固定の実装案

	固定したいスロットに1を立てる	固定したいスロットではない他の範囲を0で埋める
✔ Pros.	・1を立てるので必ずそこに固定される。	・ある許容範囲で固定することが実現可能(例: 12月2日～3日を許容) ・制約条件を満たさなかった時の結果を理解しやすい。
✕ Cons.	・例えば同時製造不可のジョブを同時に固定した場合に、制約条件を破った同時に1の解が得られてしまう。	・0で埋める範囲を決める条件の見落とし(つまりバグ)に注意が必要

コード抜粋 (sched.py):

```
for j, job in enumerate(jobs):
  if fixed := job_fixed_range(job):
    m = lines.index(fixed.line)
    if is_fixed_in_range(intbl, fixed):
      d = (fixed.start - intbl.start_date).days
      q[m, j, d] = 1
    else:
      warn.append(f"{_job_label(j)} の固定範囲が開始日
＋日数の範囲外です。")
```

左記コードは固定したい位置に1を立てる例
制約条件を作る前、つまり、決定変数qを生成した直後の「プリセット制約」で、固定の1や0を立ててください。

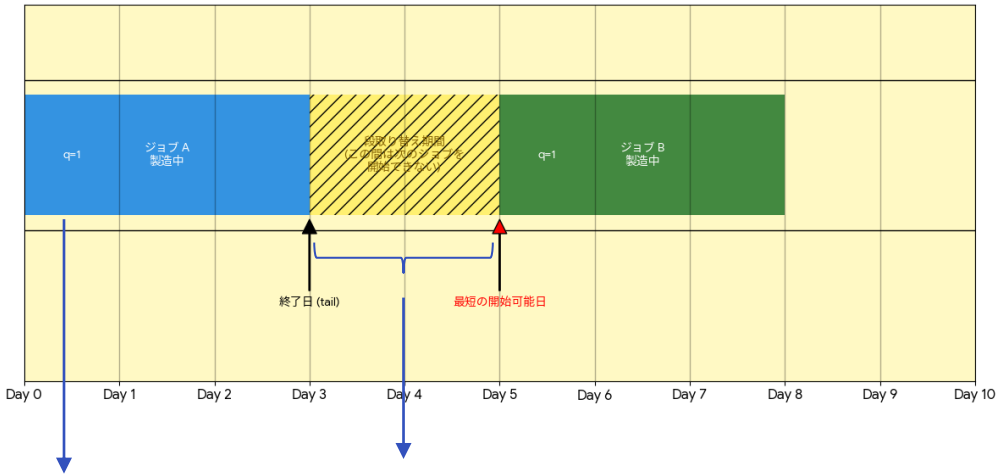
【再掲】

[マスターテーブル] 段取り替え

「切替元」と「切替先」の製品番号と、その「切替日数」を1次元テーブルに定義します。
右側のプレビューで2次元の表として確認可能です。

段取り替えの実装例

ジョブjの終了後、切替時間(co)の間に他のジョブojが開始されるのを禁止する。



q[m, j, d] ✖ q[m, oj, (ed+1):(ed+1)+co].sum() = 0

コード抜粋 (sched.py):

```
for m, line in enumerate(lines):
  for j, job in enumerate(jobs):
    for d in range(num_days):
      ed = tail_slot(intbl, line, j, d).tail
      for oj, ojob in enumerate(jobs):
        co = changeover_time(
          intbl.changeover, job["item_id"],
ojob["item_id"]
        )
        # q[m, j, d]が1なら、 切替時間内のq[m, oj, ...]
の合計は0でなければならない
        poly_d = q[m, oj, (ed + 1) : (ed + 1) +
co].sum()
        constraints += equal_to(q[m, j, d] * poly_d,
0)
```

[マスターテーブル] 製品情報

人員数カレンダー　製品マスタ　リソース　メンテナンス日

製品番号	ライン	最大日産数	必要人数	必要リソース{個数}
製品001	ライン1	10	15	治具
製品001	ライン2	20	30	金型{2}
製品002	ライン2	15	21	金型{2},治具
製品003	ライン1	12	18	
製品003	ライン2	12	18	ホイスト,玉掛技能者
製品003	ライン3	12	18	金型
製品004	ライン1	8	12	
製品005	ライン3	25	40	ホイスト,フォークリフト

人員数カレンダー　製品マスタ　リソース　メンテナンス日

リソース名	数量
治具	5
金型	3
ホイスト	4
玉掛技能者	3
フォークリフト運転技能者	2

製品マスタ

- その製品をそのラインで製造する場合の情報
  - ✓ 最大日産数
  - ✓ 必要人数
  - ✓ 必要リソース: カンマ区切り {}内の数字は必要数量

リソース

- 共有して仕様するリソースの名前とその数量

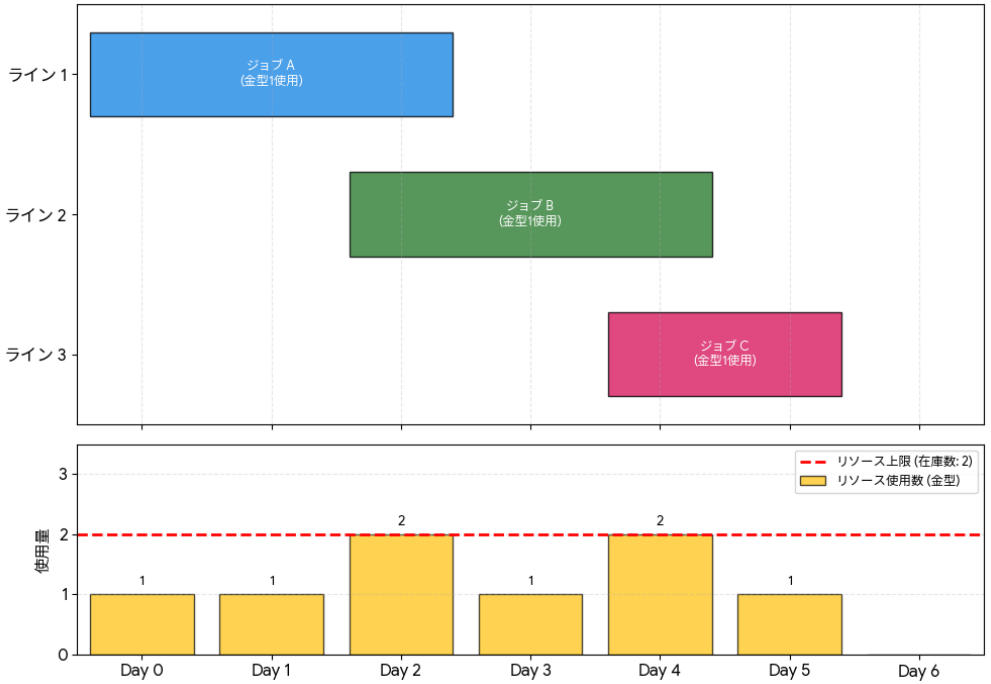
共有リソース

金型や技能者など、複数ラインで共有するリソースの数に制限を設けます。

実装方法

各ジョブが「その日に稼働しているか」を示す変数polysを導入します。

この変数は人数充足率最大化の目的関数の式でも使いました。



共有リソースの実装例

コード抜粋 (sched.py):

```
for m, line in enumerate(lines):
  for j, job in enumerate(jobs):
    for d in range(num_days):
      ed = tail_slot(intbl, line, j, d)
      for s in range(d, ed.tail + 1):
        if s in ed.skipped:
          continue
        polys[m][j][s] += d[m, j, d]
```

polys[m][j][s]
あるジョブjがラインm,日付dに製造開始したとき、それが日付sで製造しているか(1か0)を示す変数。
つまり、m, j, sにおいて製造しているかどうかを判定できる。

```
for res, job_ress in common_resource_jobs(intbl,
lines).items():
  res_count = resource_count(intbl.resources, res)
  for d in range(num_days):
    res_d = Poly()
    for j, m, cnt in job_ress:
      res_d += cnt * polys[m][j][d]
    constraints += less_equal(res_d, res_count)
```

cnt * polys[m][j][d]
あるジョブjがラインm,日付dで必要とするリソースの個数。
cntは製品マスタテーブルで定義されている。

less_equal(res_d, res_count)
そのリソースを使用している全てのジョブを足して(res_d)、それがリソースの数量(res_count)以下であるべき。

Infeasible(解なし)な結果の救済

制約を満たすことができないinfeasible(解なし)な結果になることがあります。
Infeasibleな結果の場合もできる限り結果画面に表示することで、ユーザーが問題の箇所を特定するためのヒントになり、ユーザービリティーが向上します。

Infeasibleで得られる結果の例	表示案
qがすべて0 1回も処理されない、つまり、製造開始の日付が得られない。	A. 開始日/終了日をブランクにする。 B. 開始日/終了日を既定値で表示し赤色でハイライト
qに2個以上の1が立つ 見かけには複数回処理されてしまう。	A. 上記の「qがすべて0」と同じ B. 最初の解を採用して開始日/終了日を表示し赤色でハイライト

コード抜粋 (sched.py):

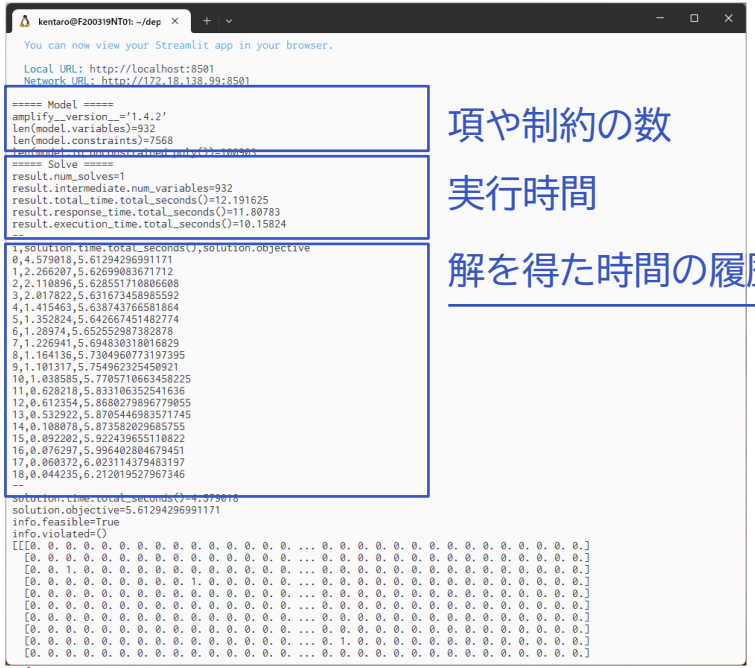
```
for alc in alloc:
  line, j, d = lines[int(alc[0])], int(alc[1]), int(alc[2])
  ts = tail_slot(intbl, line, j, d)
  if j in gantt:
    warn.append(f'ジョブID"{j}"の製造回数が異常')
    continue
  gantt[j] = GanttDict(line=line,  start=d, …)
```

最初の解を採用する例
警告でその旨を伝えるとさらにユーザービリティーが向上します。

エネルギープロットと解の収束の確認

計算が進むにつれて目的関数(エネルギー)が低下する様子を確認します。
解の改善が止まって収束した時間を特定し、計算時間(client.parameters.time_limit_ms)の目安にします。

コンソールログ

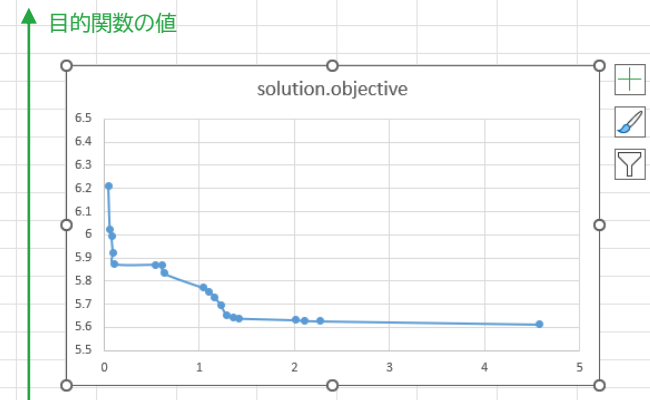


項や制約の数

実行時間

解を得た時間の履歴

i	solution.time.total_seconds()	solution.objective
0	4.579018	5.612943
1	2.266207	5.626991
2	2.110896	5.628552
3	2.017822	5.631673
4	1.415463	5.638744
5	1.352824	5.642667
6	1.28974	5.652553
7	1.226941	5.69483
8	1.164136	5.730496
9	1.101317	5.754962
10	1.038585	5.770571
11	0.628218	5.833106
12	0.612354	5.868028
13	0.532922	5.870545
14	0.108078	5.873582
15	0.092202	5.92244
16	0.076297	5.996403
17	0.060372	6.023114
18	0.044235	6.21202



演習問題

1. それぞれのジョブは「1回だけ処理される」を「たかだか1回処理される」で実装してください。
2. 「毎日の作業人数が、予定人数を超えない」制約を実装してください。

ヒント
1. one_hot(…)の代わりにless_equal(…, 1)を使います。
2. 人数充足率最大化の目的関数を参考に、当日の合計人数hc_dを使って less_equal(hc_d, required) を制約に追加します。

まとめ

アプリ紹介のまとめ

製造現場の複雑な制約(納期、人員、設備)を Fixstars Amplify AE で解決するアプリケーションの実装例を紹介しました。

- 数理モデルの構築
- 多目的最適化の実装
- 制約の実装

ベテランの経験に頼っていた計画業務を自動化・高度化できます!

研究開発分野で多くご利用いただいているブラックボックス最適化についても
デモ・チュートリアルや簡易アプリをご用意しています。

デモ・チュートリアル

FMQA導入

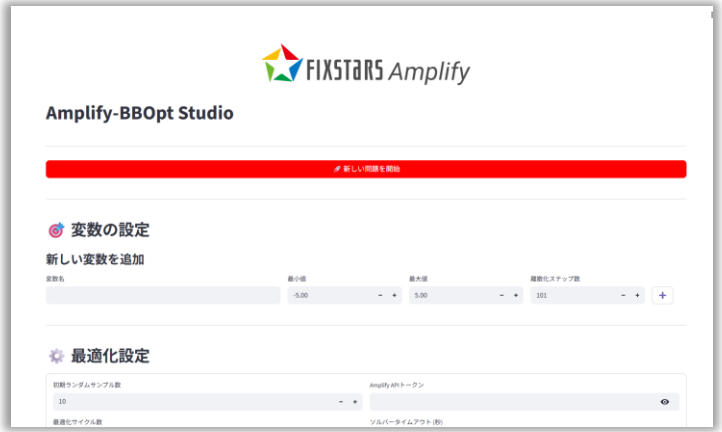
材料最適化

化学プラント
運転条件最適化

翼形状最適化

信号制御最適化

簡易アプリ



「お客様から最適化に関する要望が出ているけれど、社内に取り組み実績がない」、「自社プロダクトに最適化技術を入れて差別化したい」という方など、ぜひ一度お話をさせてください！

Q&A

- 最適化アプリの一般的な導入期間は？
- 最適化のプロジェクトを進める時に陥りがちな失敗例は？
- 最適化エンジンを扱うのは難しいのではないか？