FIXSTARS Amplify

# 量子コンピュータ時代の最適化セミナー

～ 最新事例と技術解説 ～

# 本日の予定

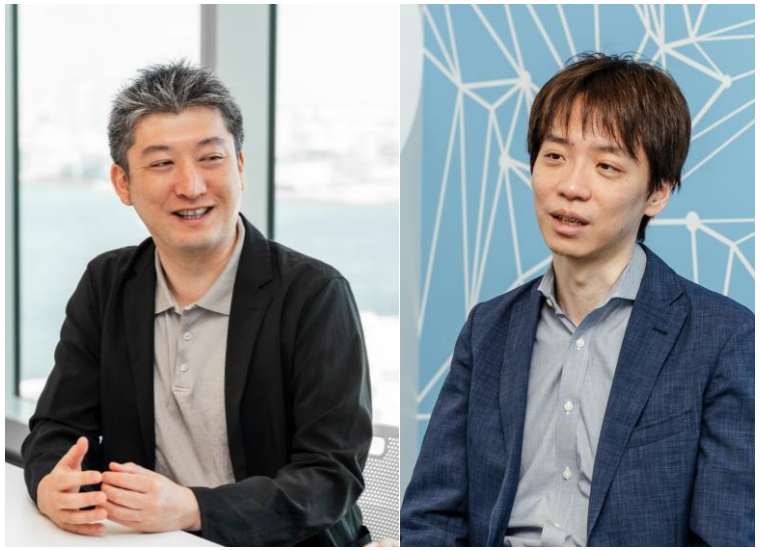
- Fixstars Amplify の紹介
- Amplify SDK 及び AE における処理フロー
- 定式化実装におけるヒント
  - タイムアウトの決定方法
  - 制約重み・目的関数スケーリング手法
  - 求解性能向上のための工夫

質問は随時、ZoomのQ&Aへお願いします

# (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の提供

*Fixstars Amplify*



- 2021年に設立
  - 代表取締役社長CEO：平岡 卓爾
  - 取締役CTO：松田 佳希（博士）

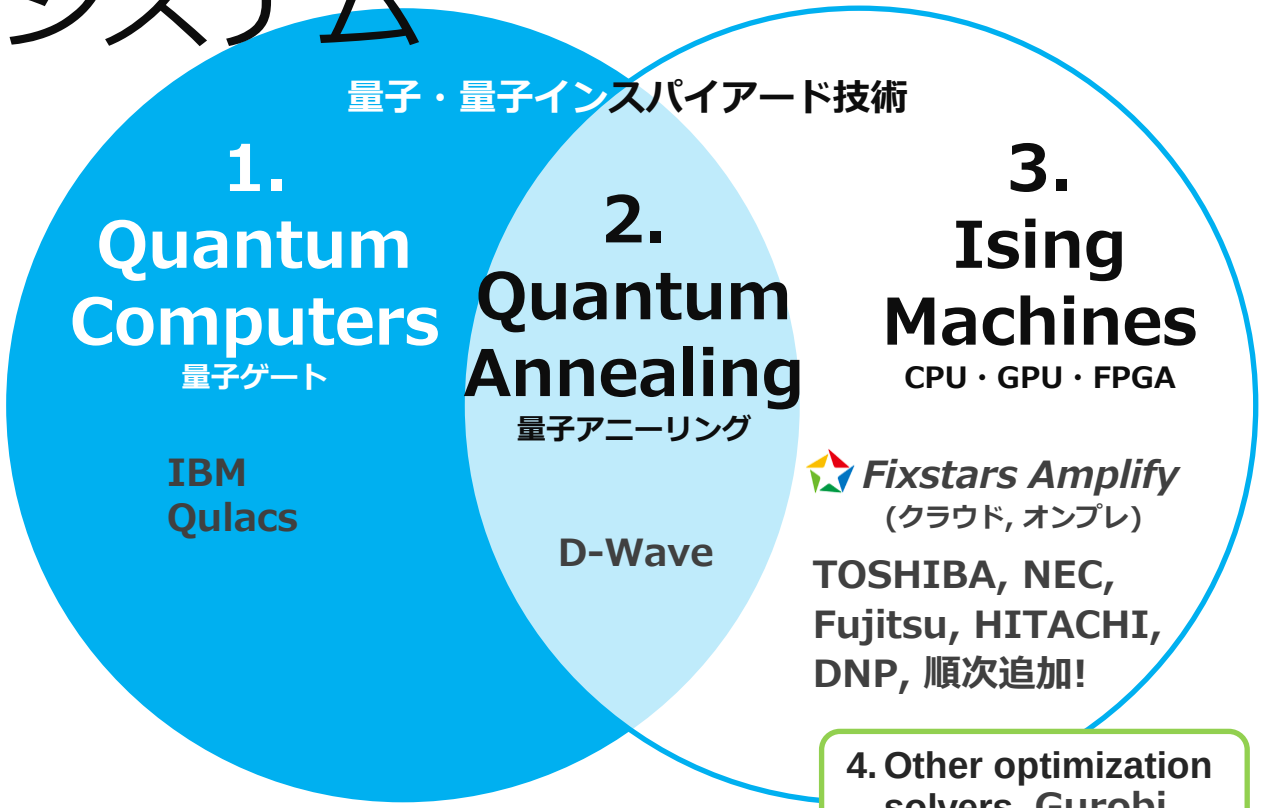
- 親会社 (株) フィックスターズ
  - 東証プライム市場上場。
  - ソフトウェア高速化プロフェッショナル集団。

# *Fixstars Amplify* のエコシステム

- **開発環境 Amplify SDK**
  - **簡単・最速**な組合せ最適化アプリ実装
- **実行環境 Amplify AE**
  - GPU上で実行されるイジングマシン
  - 大規模求解可能な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




FIXSTARS *Amplify* SDK

**種々の組合せ最適化問題**
**(利用ソルバーに関わらずQUBOを超越)**

# 富士通 Digital Annealer の標準マシン化

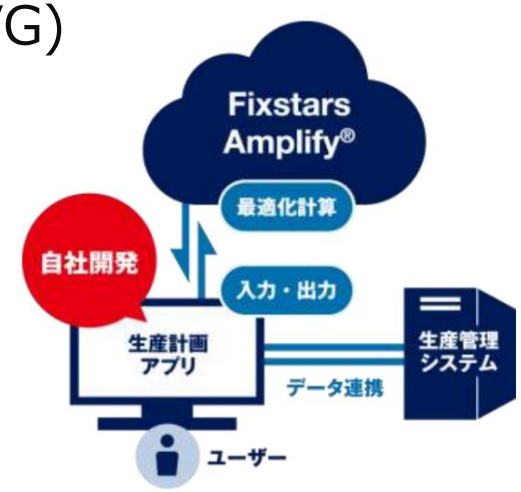
- Fixstars Amplify 対応マシン
  - Fixstars Amplify Annealing Engine (AE)
  - D-Wave The Leap Quantum Cloud Service
  - 東芝 SQBM+
  - NEC Vector Annealing サービス
  - 富士通 Digital Annealer ← **new!**
  - 日立製作所 CMOS アニーリングマシン
  - 大日本印刷 DAS
  - Gurobi Optimizer
  - IBM Quantum
  - Qulacs

**標準マシン**
Fixstars Amplify 社がトークンを発行することが可能なマシン

FIXSTARS Amplify

# ユースケースと活用例（実稼働を含む）

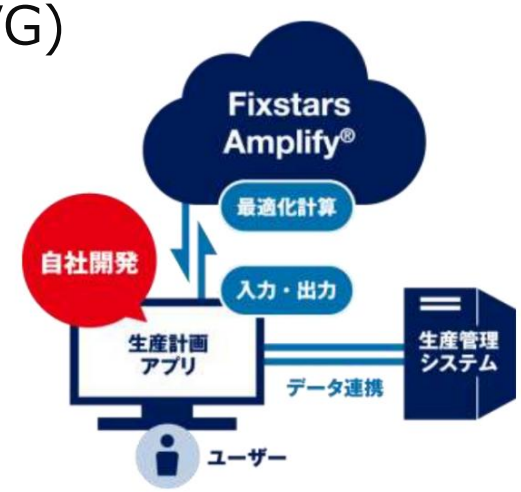
- **生産計画**
  - 多品種少量生産、保全計画、設備投資、在庫
- **従業員割り当て**
  - 食品、輸送、製造
- **エネルギーマネジメント**
  - エネルギーミックス、機器の運転制御
- **経路**
  - 配送、無人搬送車 (AVG)
- **メディア**
  - 最適広告配信
- **研究開発、設計**
  - 材料設計
  - 物理シミュレーション
  - ブラックボックス最適化



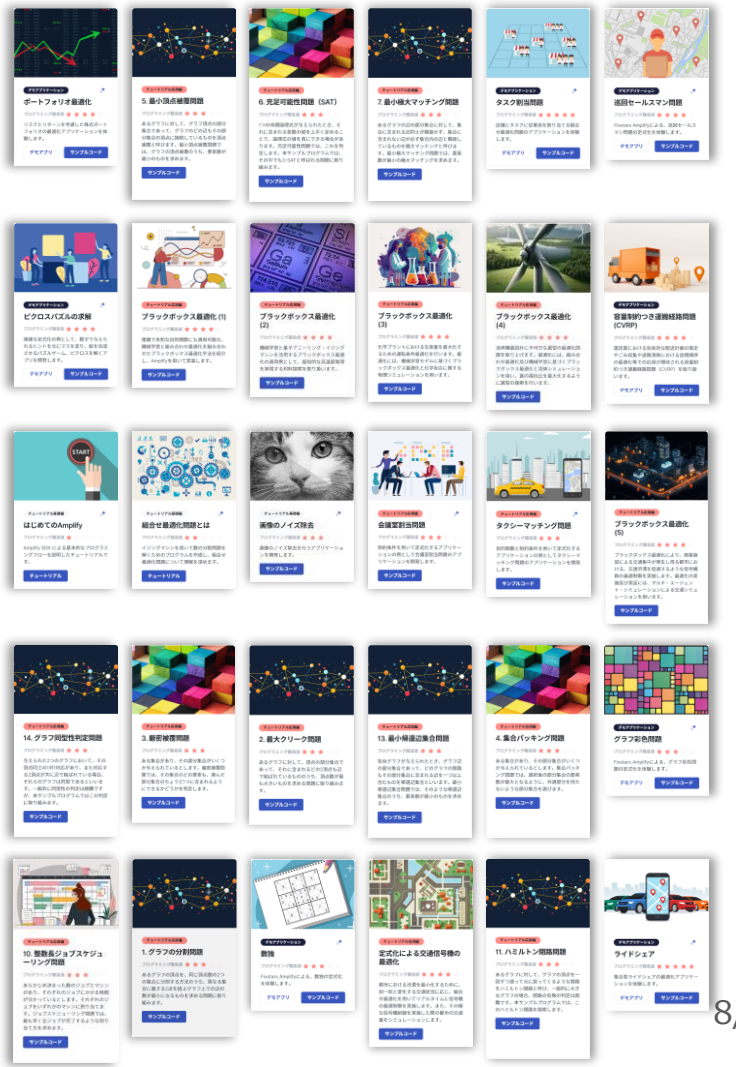

Amplify インタビュー　検索

# ユースケースと活用例（実稼働を含む）

- **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
- **エネルギーマネジメント**
  - エネルギーミックス、機器の運転制御
- **経路**
  - 配送、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
  - ブラックボックス最適化




Amplify デモ Search



# ユースケースと活用例（実稼働を含む）

- **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**
  - 食品、
- **エネルギー**
  - エネルキ
- **経路**
  -
- **メ**
  -
- **研究**
  - 材料設計
  - 物理シミュレーション
  - ブラックボックス最適化

**800**を超える企業、研究所、大学

**6,000万**を超える実行回数 (Amplify AE)


# Amplify SDK による実装

- **制約条件**を満たしながら**目的関数**が最小となる**決定変数**の値を探索
  - **決定変数** バイナリ変数 `q[0]`, `q[1]`
  - **目的関数** `q[0] - 2 * q[0] * q[1]`
  - **制約条件** one_hot ➔ q の要素1つが1
    - その他の制約ヘルパー関数
      - equal_to
      - less_equal
      - greater_equal
      - clamp
      - domain_wall

```python
# 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))  # [1. 0.]
print(objective.evaluate(result.best.values))  # 0.0
```

FIXSTARS Amplify

# 制約条件のペナルティ化

- 制約条件のペナルティ化
  - 制約違反の場合には内部的に「1」がペナルティとして目的関数に加算されるように定式化
  - ペナルティ込み目的関数を最小化するよう探索
  - 結果として、全ペナルティゼロ（制約充足）な解が得られることが期待される
- 制約の重み
  - 大きな値を伴う目的関数の場合、制約を違反し、ペナルティ (=1) を加算されたとしても、目的関数を改善する方がよい、となる
    - ➔ 実行可能解が得られにくい
    - ➔ 適切な **制約重み・目的関数スケーリング** が必要

```python
# 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))  # [1. 0.]
print(objective.evaluate(result.best.values))  # 0.0
```

# Amplify AE 求解フロー

- 探索の実行
  - **タイムアウト** の中で、アルゴリズム実行単位の「探索」を繰り返し、より小さい目的関数値を実現する解を取得
- 解の返却
  - 探索回数分の解を SDK に返却
  - 最初の「探索」で真の最適解が得られた場合、SDK に返却される解は1つ
  - 最低1つの「解」が得られるまで求解を継続
- 制約充足の判定
  - AE は制約をペナルティ値のみを考慮。
  - 最終的な制約充足は SDK によって判定。

```python
# 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))  # [1. 0.]
print(objective.evaluate(result.best.values))  # 0.0
```

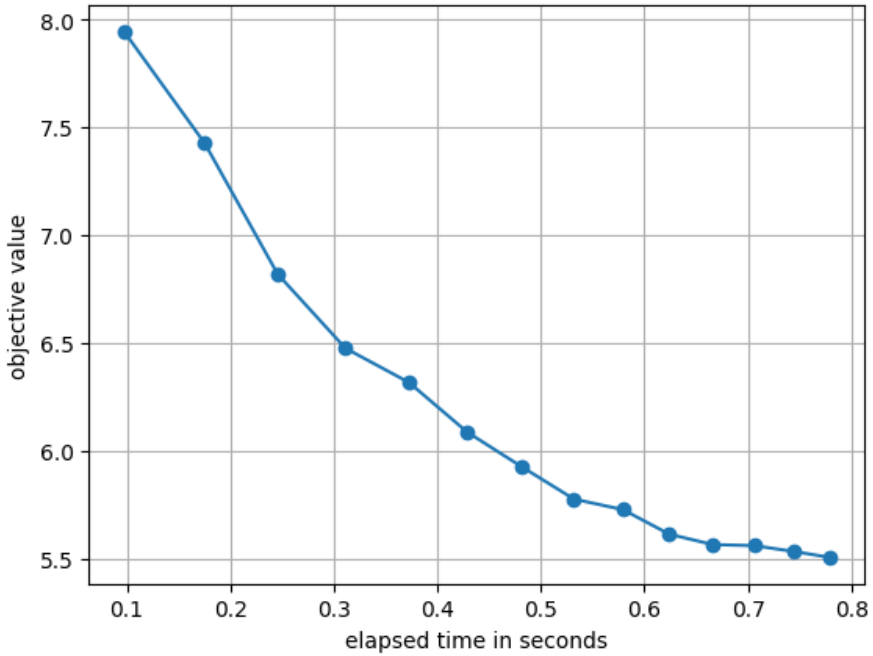
# タイムアウト設定に関するヒント

- 適切なタイムアウト設定は？
  - 短すぎると、探索が十分行われず、良い解が得られない
  - 長すぎると、探索解が十分収束した後も探索を繰り返すため、非効率
- 必要タイムアウトの一般的な傾向
  - 決定変数の数：大　→　タイムアウト：大
  - 制約条件：多　→　タイムアウト：大

# タイムアウト設定に関するヒント

https://amplify.fixstars.com/ja/docs/amplify/v1/timing.html#id4

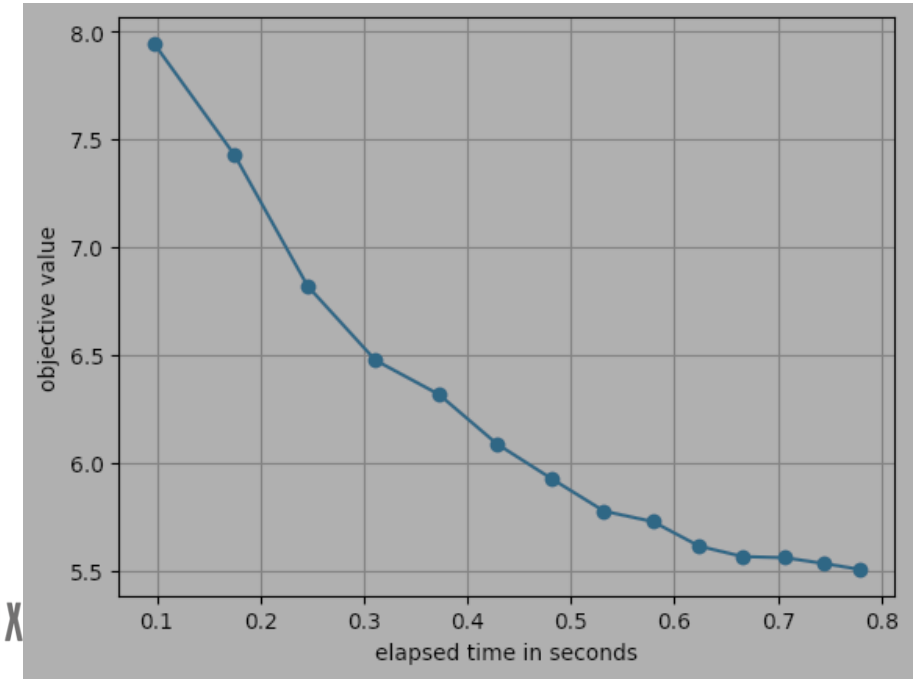
- 解の時間情報の取得
  - AE はタイムアウト内に得られた「探索」回数分の「解」全てを SDK に返却。
  - SDK は返却された「解」から、制約を満たす解（実行可能解）のみを `result` に格納
  - 実行可能解及びタイムスタンプをプロット




```python
import matplotlib.pyplot as plt

# 定式化
・・・

# 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)
client.parameters.outputs.num_outputs = 0  # 全ての探索解を返却

# ソルバーの実行
result = solve(model, client)

# それぞれの解の時刻と目的関数の値を取得
times = [solution.time.total_seconds() for solution in result]
objective_values = [solution.objective for solution in result]

# プロット
plt.plot(times, objective_values, "-o")
plt.xlabel("elapsed time in seconds")
plt.ylabel("objective value")
plt.grid(True)
```

# タイムアウト設定に関するヒント

- 解の時間情報の取得
  - AE はタイムアウト内に得られた「探索」回数分の「解」全てを SDK に返却。
  - SDK は返却された「解」から、制約を満たす解（実行可能解）のみを result に格納
  - 実行可能解及びタイムスタンプをプロット




```python
import matplotlib.pyplot as plt

# 定式化
・・・

# 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)
client.parameters.outputs.num_outputs = 0  # 全ての探索解を返却

# ソルバーの実行
result = solve(model, client)

# 探索された解の個数(アルゴリズム実行単位である探索の回数)を表示
print(result.client_result.execution_parameters.num_iterations)
# > 20

# SDK によってフィルターされた実行可能解の個数を表示
# ＝履歴プロットの点の数
print(len(result))
# > 11
```

# 制約重みに関するヒント

- 大きな値を伴う目的関数の場合、制約を違反し、ペナルティ (=1) を加算されたとしても、目的関数を改善する方がよい、となる
  - ➔ 実行可能解が得られにくい
  - ➔ 適切な **制約重み・目的関数のスケーリング** が必要
- 基本的には、制約重みとして、目的関数値の大きさと同じか、1オーダー大きいくらいの値を設定する。
  - 0.01, 0.1, 10, 100, 1000 のようなオーダー単位の調整でOK

# 単目的最適化：制約重みに関するヒント

- 制約重み weight の例
  - 目的関数値が取りそうな値とする
    - max(d)
    - sum(d)
    - …
- 重みの自動調整
  - Amplify AE の場合、制約重みはある程度自動調整するため、SDKで記述する重みは目安程度で良い
- 目的関数が複数の場合は？

```python
# 巡回セールスマン問題の定式化例

gen = VariableGenerator()
q = gen.array("Binary", N + 1, N)
q[-1, :] = q[0, :]

objective = 0
for k in range(N):
    for i in range(N):
        for j in range(N):
            objective += d[i, j] * q[k, i] * q[k + 1, j]

constraint1 = one_hot(q[:-1], axis=1)
constraint2 = less_equal(q[:-1], axis=0)

model = objective + weight * (constraint1 + constraint2)
```

$$\sum_{i,j,k}^{N} d_{i,j}\, q_{k,i} q_{k+1,j}$$

# 多目的最適化：制約重みに関するヒント

- 多目的最適化
  - 目的関数が複数存在
  - 「ソフト制約」も目的関数の1つ
  - 複数の目的関数が異なるレンジを取る
- 多目的最適化における制約重み
  - それぞれの目的関数をスケーリング
  - スケーリング係数 (s_1, s_2) は、(obj_1, obj_2) の定式化に基づいて見積もり

```
# 多目的最適化問題の例
obj_1 = ...
obj_2 = ...

const_1 = one_hot(...)
const_2 = less_equal(...)


# obj_1 と obj_2 のレンジが大きく異なる場合、不適
model = obj_1 + obj_2 + weight * (const_1 + const_2)

# obj_1 と obj_2 のレンジが大きく異なる場合でも、対応可
# ここで、s_1 と s_2 はそれぞれ obj_1 と obj_2 の代表値
model = obj_1 / s_1 + obj_2 / s_2 + const_1 + const_2


# スケーリング後に各目的関数の重みを微調整しても良い
model = 10 * obj_1 / s_1 + obj_2 / s_2 + const_1 +...
```

# 多目的最適化：発展的なスケーリング係数決定

- スケーリング係数の自動決定 (AE)

1. 制約問題として求解
   - ➔ 短いタイムアウトで多くの解を取得
   - ➔ 制約を満たすランダム解とみなせる
2. 解を個々の目的関数に代入、代入結果を目的関数のスケーリング係数とする
3. 目的関数を除算しスケーリングを実施
   - ➔ 目的関数の取りうる値が 1 程度になることが期待される
4. スケーリング後の目的関数と制約条件を考慮し、最適化問題として求解

```python
obj_1, obj_2 = ...
const_1, const_2 = one_hot(...), less_equal (...)

# 制約問題として求解
model = const_1 + const_2

# 同じ目的関数値の解を複数取得するオプション
#（制約問題では、目的関数は常に0であるため）
client.parameters.outputs.duplicate = True

result = solve(model, client)

# s_1 と s_2 はそれぞれ obj_1 と obj_2 の代表値
s_1 = max([obj_1.evaluate(sol.values) for sol in result])
s_2 = max([obj_2.evaluate(sol.values) for sol in result])

# obj_1 と obj_2 のレンジが大きく異なる場合でも、対応可
model = obj_1 / s_1 + obj_2 / s_2 + const_1 + const_2

# 実際の最適化問題として求解
result = solve(model, client)
```

# 多目的最適化：発展的なスケーリング係数決定

- スケーリング係数の自動決定 (A

1. 制約問題として求解
   - ➔ 短いタイムアウトで多くの解を
   - ➔ 制約を満たすランダム解とみな
2. 解を個々の目的関数に代入、代入
   を目的関数のスケーリング係数と
3. 目的関数を除算しスケーリングを
   - ➔ 目的関数の取りうる値が 1 程
     なることが期待される
4. スケーリング後の目的関数と制約
   を考慮し、最適化問題として求解

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

```
one_hot(...), less_equal (...)


const_2

を複数取得するオプション
関数は常に0であるため)
outputs.duplicate = True

el, client)

ぞれ obj_1 と obj_2 の代表値
valuate(sol.values) for sol in result])
valuate(sol.values) for sol in result])

レンジが大きく異なる場合でも、対応可
l + obj_2 / s_2 + const_1 + const_2

して求解
result = solve(model, client)
```

# 求解性能を向上させる様々な工夫

- 不等式制約におけるペナルティ生成法を指定
  - 緩和法
    - https://amplify.fixstars.com/ja/docs/amplify/v1/penalty.html#ineq-penalty
    - 正確なペナルティではないが、多くの場合において不等式制約を含む問題の求解性能が上がる可能性が有る。
    - 生成されるペナルティは正確な定式化ではないが、制約条件充足はSDKにより正しく判定されるため、得られた解は制約を正しく満たす。

```
constraint = less_equal(q, 2, penalty_formulation="Relaxation")
constraint = greater_equal(q, 2, penalty_formulation="Relaxation")
constraint = clamp(q, (1, 2), penalty_formulation="Relaxation")
```

# 求解性能を向上させる様々な工夫

- ソルバーの直列実行 (num_solves)
  - https://amplify.fixstars.com/ja/docs/amplify/v1/serial.html
  - ソルバーによっては、長時間のタイムアウトを指定して 1 回実行するよりも短時間のタイムアウトで何回か繰り返し実行するほうがより良い解を見つける可能性

```
result = solve(model, client, num_solves=3)
```

# 求解性能を向上させる様々な工夫

- GPU並列実行 (num_gpus)
  (Amplify AEでマルチGPUオプション設定の場合のみ)
  - アニーリングにて使用するGPU数を増やすことで、より高速・高精度な求解が期待

```
client = FixstarsClient()
client.token = "Amplify AE のアクセストークン"
client.parameters.timeout = timedelta(seconds=1)
client.parameters.num_gpus = 4
```

# 求解性能を向上させる様々な工夫

- 求解が失敗するたびに、
  - 制約重みを2倍にする（右例→）
  - タイムアウトを増やす
  - num_solvesを増やす
  - など
- 1回の最適化で複数回の求解を行う必要がある場合に有効
  - スケジュール最適化
  - ブラックボックス最適化
  - など

```
max_retries = 3
for _ in range(max_retries):
    result = solve(model, client)

    if len(result) > 0:
        break

    model.constraints *= 2
```

# まとめ

- Amplify SDK 及び AE における処理フロー
- 定式化実装におけるヒント
  - タイムアウトの決定方法
  - 制約重み・目的関数スケーリング手法
    - 単目的
    - 多目的
  - 求解性能向上のための工夫
    - 不等式制約のペナルティ生成法
    - 直列実行
    - マルチGPU
    - リトライ機能

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

# 数理最適化・組合せ最適化とは

- 目的を表す何らかの関数の最小値を求め、更にその最小値を与える入力要素（決定変数）の値を決定する数学的な手法

  - 目的となる対象を数式で表現できる
  - 変更できる要素がある
  - 求めたい成果がある

**お遍路巡り経路最適化問題**

- 目的となる対象を数式で表現できる → 対象：総移動距離
- 変更できる要素がある → 要素：霊場の訪問順序
- 求めたい成果がある → 成果：総移動距離の最小化

数式で表現できるものが「数理最適化」になるため、 日常的に使う「最適化」よりも範囲は狭くなります。使用できる範囲がかなり限定されるように感じますが、想像以上に様々な計画業務を数式に表現できることが分かっています。

# 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)<br>**30万円** (最大5名) | **20万円** (1名)<br>**60万円** (最大5名) | **30万円** (1名)<br>**90万円** (最大5名) | **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) | | |
| **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<br>(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<br>(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 |
| 月間実行回数上限<br>(実行回数追加オプションあり) | 制限の可能性あり | 無制限 | | | - (制限をかける可能性あり) | | |
| 東芝 SQBM+オプション | 無料 | 30万円 (1名)、90万円 (最大5名) | | | - | | |
| NEC VAオプション | 無料 | 30万円 (1名)、90万円 (最大5名) | | | - | | |
| 富士通 DAオプション | 無料 | 50万円 (1名)、150万円 (最大5名) | | | - | | |
| D-Wave の利用 | 無料プログラム (3分/月) | | | | - | | |
| **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

FIXSTARS Amplify

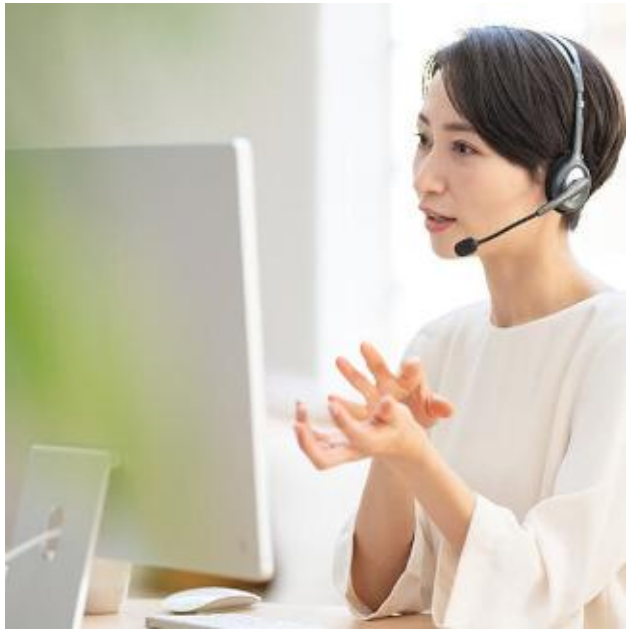
# Plusオプション（プレミアムプラン/Sプレミアムプランで追加可能なオプション）

## 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

### コード最適化レビュー

- 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