NTT DATA Mathematical Systems Inc.

Numerical Optimizer V24 ご紹介

株式会社ＮＴＴデータ数理システム

Trusted Global Innovator
NTT DATA Group

はじめに

Numerical Optimizer は、

次期バージョン（V24：2022 年 3 月リリース予定）より

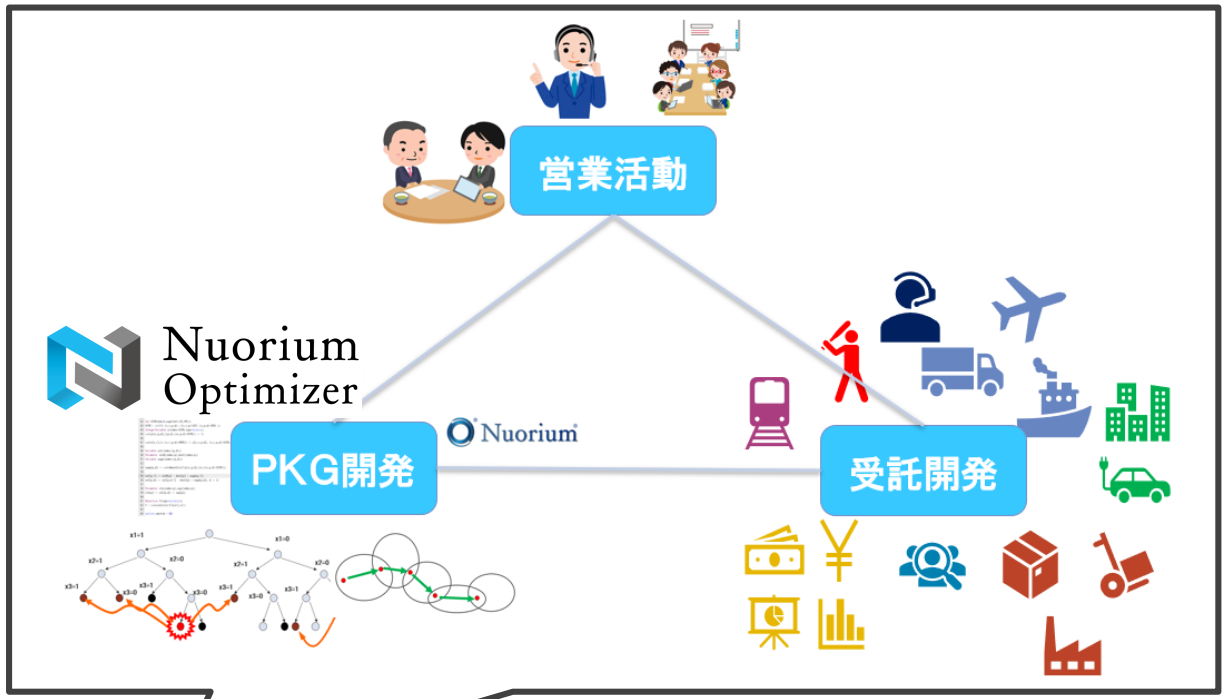
Nuorium Optimizer

に改名となります。（読み方：ニューオリウム オプティマイザー）

- Nuorium は、「数理最適化をおこなう場所」という意味を持ち、既にエディタとして皆様にご愛用いただいております。
- Nuorium Optimizer は、皆様が数理最適化をおこなう上で欠かせない存在として今後もお役に立ちたいという想いを込めて名付けました。

数理最適化とは

NTTデータ数理システムの数理最適化ビジネスの内容



数理最適化とは

大量にある候補の中から最もよいものを選び出す

変数

生産量，製造順序，シフト，
配送ルート，在庫量… etc.

目的関数

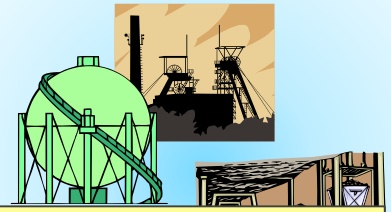
コスト最小化，
売上最大化，
作業平準化… etc.

制約条件

需要や必要人数の充足，
納期遵守，スキルの考慮… etc.

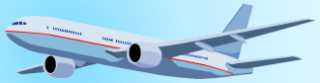
数理最適化の適用分野

エネルギー



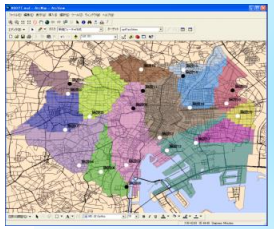
・燃料タンク繰り
・発電起動停止計画
・エネルギー配分

運輸


・配船計画
・配車計画
・貨物積み付け

サービス・流通

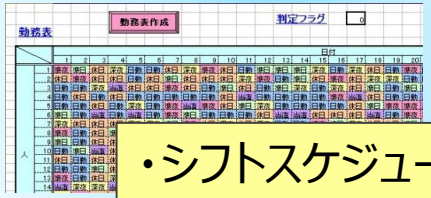



・広告配信計画
・出店計画

製造

・生産スケジューリング
・在庫配置
・資材切り出し

人員配置(業界共通)



・シフトスケジューリング
- コールセンター
- 病院
- 店舗

金融

・ポートフォリオ最適化
・イールドカーブ推定

数理最適化導入の意義や目的

1. **数理最適化** により **現状よりも良い計画を立案**
   - ➢コスト削減
   - ➢売上や効果の増大

2. **数理最適化** による計画作成の **自動化**
   - ➢得られた解は入力された「全ての制約式を満たしている」という安心感
   - ➢人手では工数のかかる困難な作業を自動化し、工数を削減

3. 属人化した計画作成業務を**数理最適化**により自動化（**属人化解消**）
   - ➢熟練者のノウハウ（ルールなど）を抽出して**数理最適化**問題に落とし込む
   - ➢熟練者でない方でも計画を立案することが可能に
     - ・単一現場だけでなく複数現場の自動化・標準化を視野に入れることも可能

課題解決に対する数理最適化を用いたアプローチ

【様々な課題】

数理モデルに対し数理最適化ソルバを用いて最適な答を導出

数理モデル

変数：
配送量・経路・シフト etc.
目的関数：
コスト最小・売上最大 etc.
制約条件：
予算遵守・容量遵守
納期遵守 etc.

数理最適化ソルバ

Nuorium Optimizer

解きたい問題・課題を数式・数理モデルで表現

Nuorium Optimizer

数理最適化パッケージ Nuorium Optimizer

Nuorium Optimizer

モデリング言語

PySIMPLE

SIMPLE

RSIMPLE

V24 NEW!
R言語を
インターフェース

C++ API

計算ライブラリ

単体法/内点法(LP,NLP,SDP)
有効制約法/逐次二次計画法
分枝限定法/wcsp/wls/rcpsp

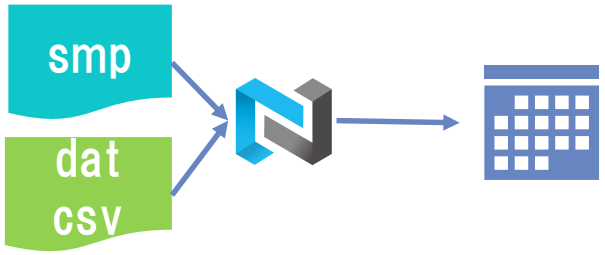
GUI

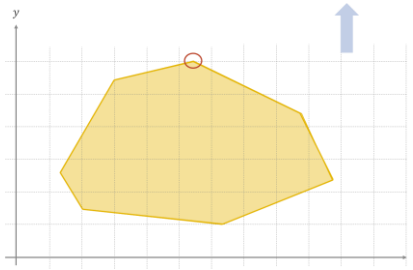
Nuorium

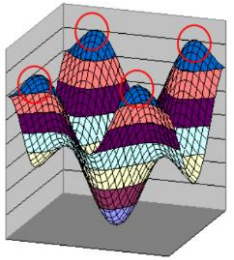
Nuorium Optimizer を
使ったモデル開発の
サポート機能が豊富な
専用エディタ

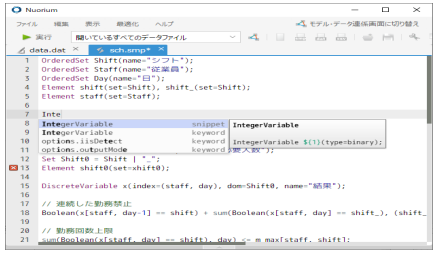
Excel アドイン

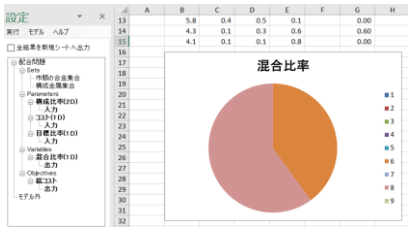
Excel のデータを
Nuorium Optimizer に
与えるための機能。
使い慣れた Excel で
データ加工ができます











定式化

変数

x （整数）
y （整数）

目的関数

$500x+400y$ → 最小化

制約条件

$60x+10y \geq 400$
$10x+50y \geq 300$
$0 \leq x \leq 7$
$0 \leq y \leq 7$

SIMPLE

C++ ベースの言語

```
IntegerVariable x;
IntegerVariable y;

Objective f(type=minimize);
f=500*x+400*y;

60*x+10*y>=400;
10*x+50*y>=300;
0<=x<=7;
0<=y<=7;
```

PySIMPLE

Python ベースの言語

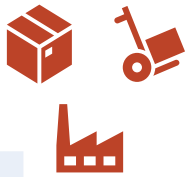
```
x = IntegerVariable(lb=0, ub=7)
y = IntegerVariable(lb=0, ub=7)

problem = Problem(type=min)
problem += 500*x+400*y

problem += 60*x+10*y>=400
problem += 10*x+50*y>=300
```

■ 扱える問題の種類（解法）

- 線形計画問題(simplex/higher)
- 混合整数線形計画問題(simplex)
- 整数計画問題(simplex/wcsp/wls), 重み付き制約充足問題(wcsp/wls)
- 非線形計画問題(tipm), 凸二次計画問題(asqp/tipm)
- 半正定値計画問題(lsdp), 非線形半正定値計画問題(trsdp)

■ 問題の具体例

- 輸送計画
- 配送計画
- 船舶スケジューリング



- 生産スケジューリング
- カッティングストック
- 積み付け最適化

- ポートフォリオ最適化
- イールドカーブ推定

- 需要を考慮した
  シフトスケジューリング

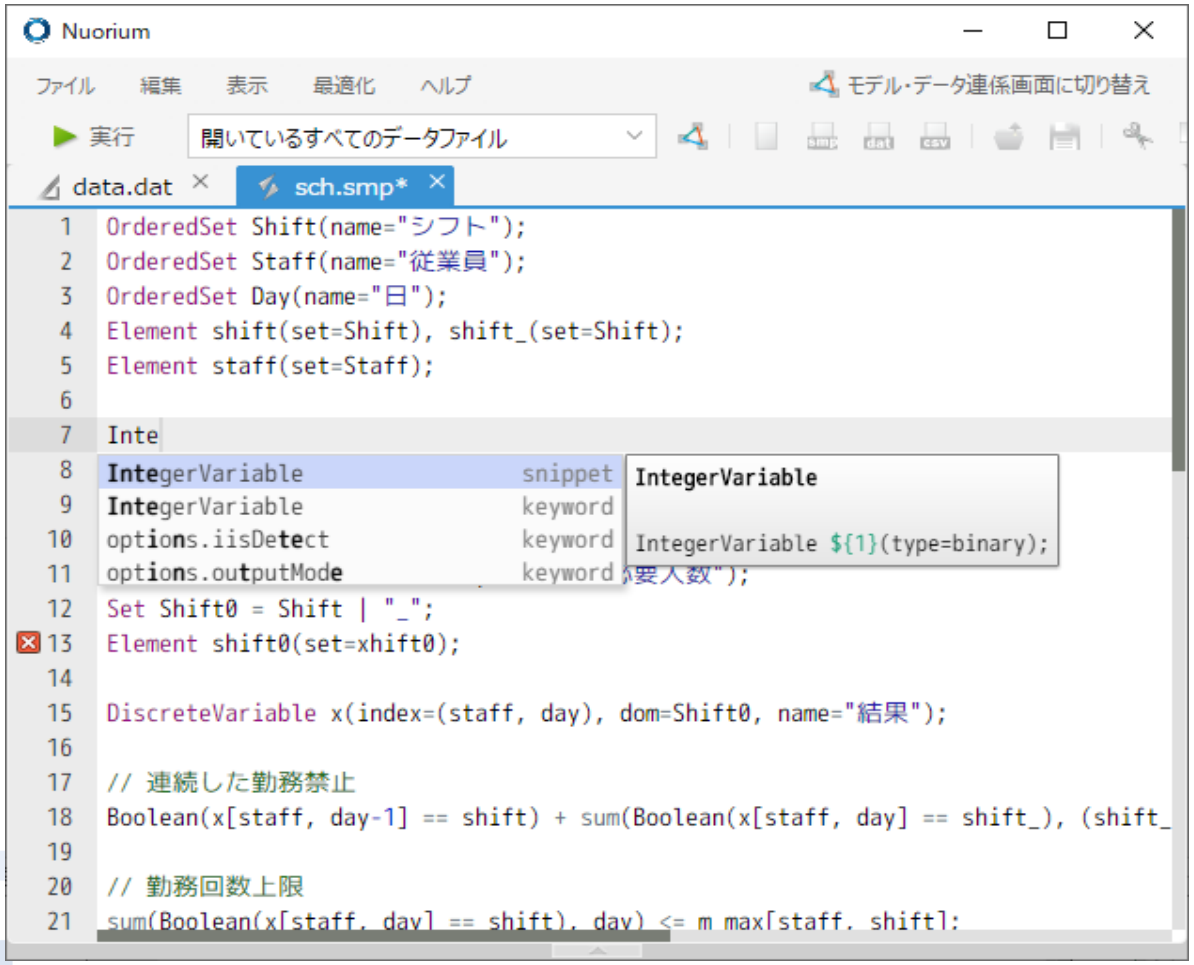
- エネルギーマネジメント

- CSナンバー算出

- 効率的な保線計画

- マッチング
- レコメンデーション

■ モデリング環境 Nuorium



■ Excel アドイン

- Excel からデータを与えて数理最適化計算
- 結果も Excel で確認

■ 弊社 HP にて、Nuorium Optimizer の導入事例をご紹介しております。
https://msi.co.jp/nuopt/interview/index.html

■ 主な導入事例

	東京ガス様 エネルギー設備の最適運用・最適設計プログラム「オプトパス®」	カネカ様 「食用油脂配送 配車計画システム」	日本将棋連盟様 「関東奨励会における対戦表自動作成システム」
問題の種類	• エネルギー設備運転計画問題	• 複雑な配送計画	• 対戦表
導入前	• 現場の勘で計画	• 配送履歴を元に手作業で計画	• 感覚的な制約を考慮して手作業で作成
導入後	• コスト削減（平均 5% 程度のエネルギーコスト削減）	• 20 分で計画作成 • 制約条件の考慮漏れが無い	• 感覚的な制約も考慮 • 計画作成時間の短縮（ほぼ 0）
	現状よりも良い計画を立案	自動化	属人化解消

豊富なアルゴリズム

線形計画問題から整数計画問題，非線形計画問題まで幅広く対応

強固な開発環境と丁寧なドキュメント

GUI 環境やモデリング言語．27 個の例題集やオンラインマニュアル

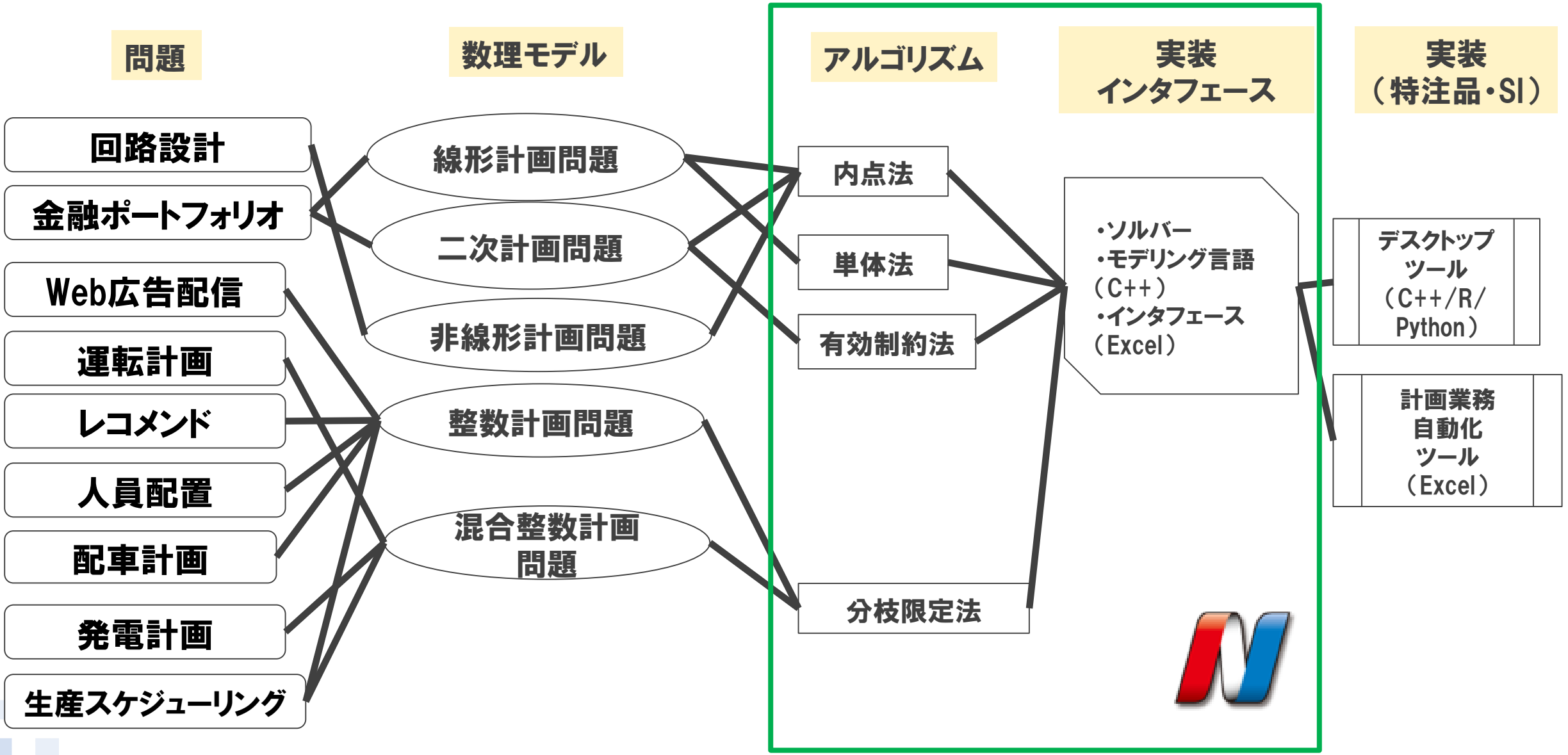
迅速なサポートと充実したバージョンアップ

技術者による 1 営業日以内の回答．性能向上や新機能を含む毎年のバージョンアップ

お客様の課題に対する受託開発・コンサルティング業務もお受け致します

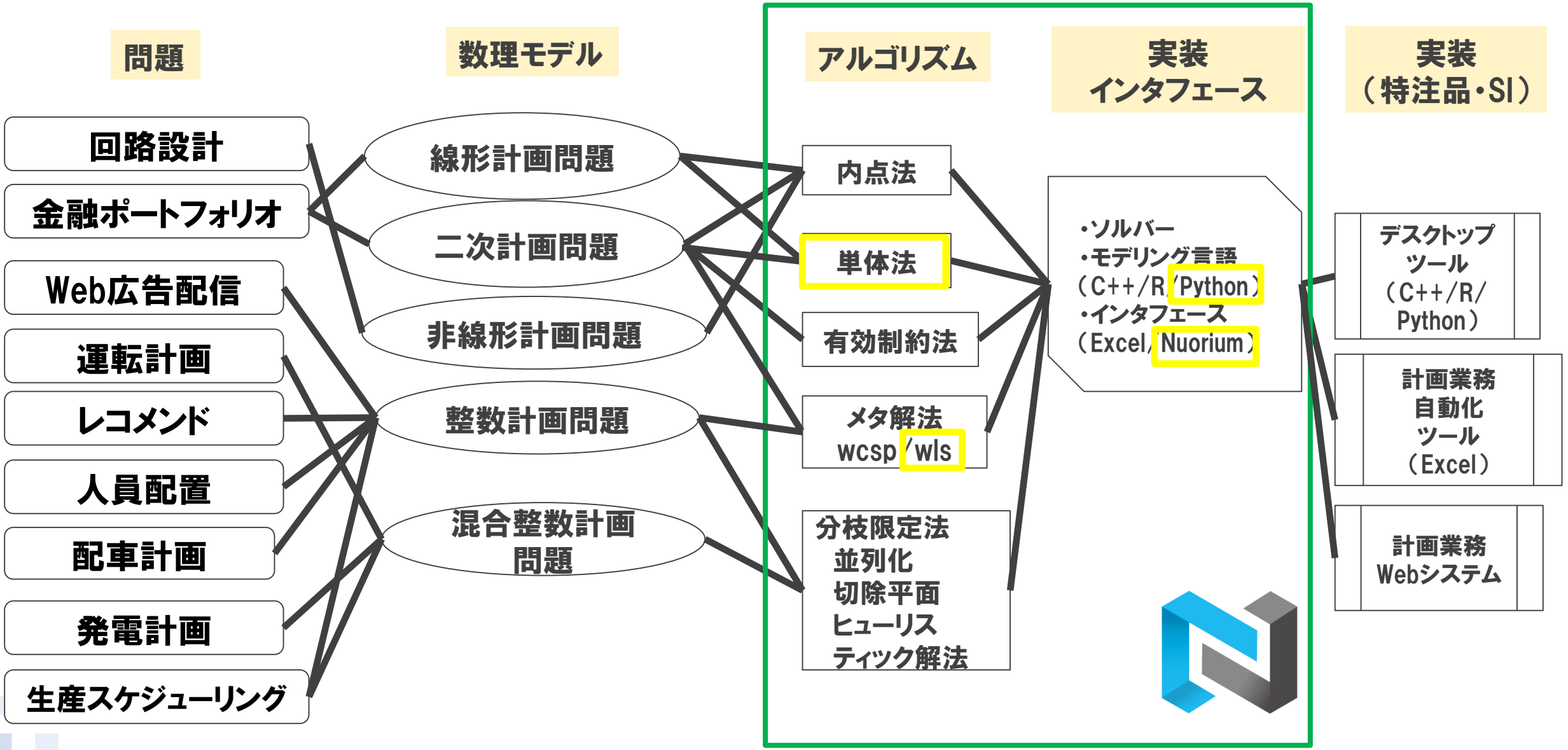
Nuorium Optimizer
～ 昔と今 ～

数理最適化アプリケーションと Nuorium Optimizer　2000年頃



数理最適化アプリケーションと Nuorium Optimizer　現在

Nuorium Optimizer V24 新機能のご紹介



Nuorium Optimizer V24
2022 年 3 月末リリース予定

1 PySIMPLE

- IIS オブジェクトの取得
- PySIMPLE クラスのシリアライズ

2 新アルゴリズム hsimplex

- スパースな LP/QP に対する単体法
- 当社従来解法よりも 20 % 高速化

3 wls QCQP

- 二次制約に対応
- 目標値とのブレを抑える問題に対応

4 Nuorium と定式化技法集

- Nuorium にターミナル機能を追加
- ユーザー様に向けた技術資料を公開

Nuorium Optimizer V24 新機能
PySIMPLE

PySIMPLE : IIS オブジェクト

- IIS オブジェクトを取得可能に
  - IIS(Irreducible Infeasible Set) は互いに矛盾する式の最小の組

```
from pysimple import *

x = Variable(lb=0)
y = Variable(lb=0, ub=1)
z = Variable(lb=0)
p = Problem()
p += y + z >= 0
p += x + y >= 5
p += x + z <= 3
p += x + y + z
p.solve(silent=True)
assert p.status == NuoptStatus.INFEASIBLE
print(p.result.iis)  # IIS_info オブジェクト
```

これらを同時に満たす
x, y, z は存在しない

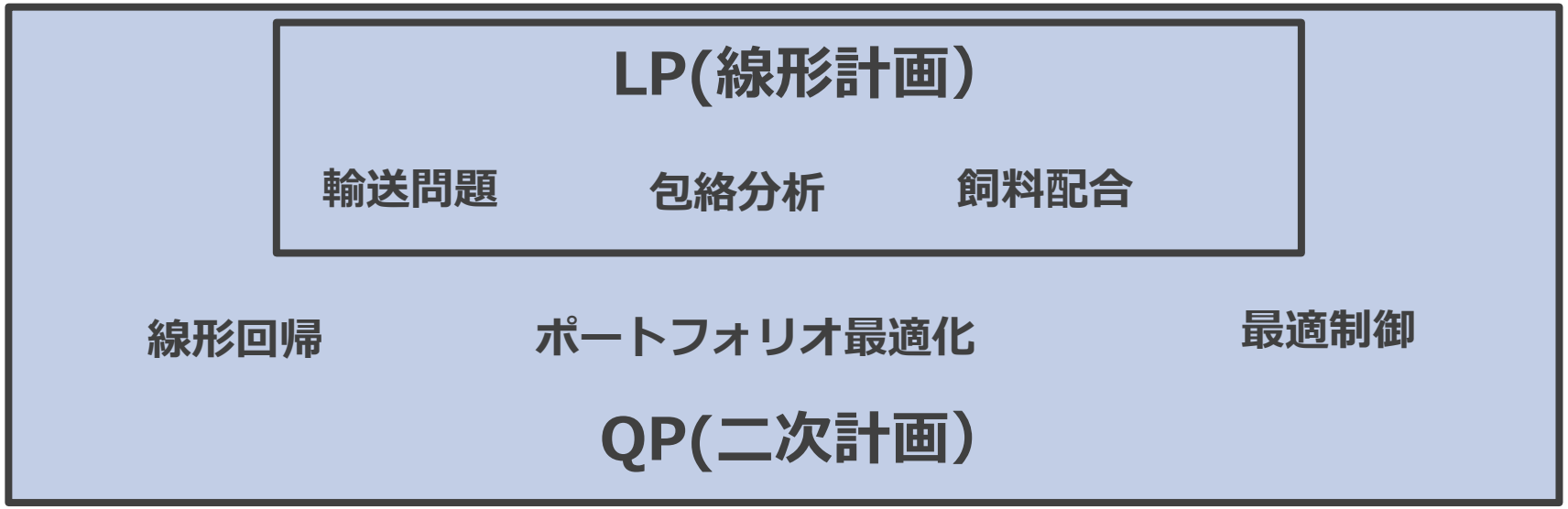
PySIMPLE：シリアライズ

- PySIMPLE オブジェクトのシリアライズ(直列化)が可能に
  - PySIMPLE オブジェクトであれば何でもシリアライズ可能
  - 使い方は標準モジュールの pickle と同じ

```
from pysimple import *

i = Element(value=[1, 2, 3])
x = Variable(index=i)
with open('dump.pkl', 'wb') as f:
    Serialize.dump(x, f)

with open('dump.pkl', 'rb') as g:
    x_ = Serialize.load(g)
print(x_)
```

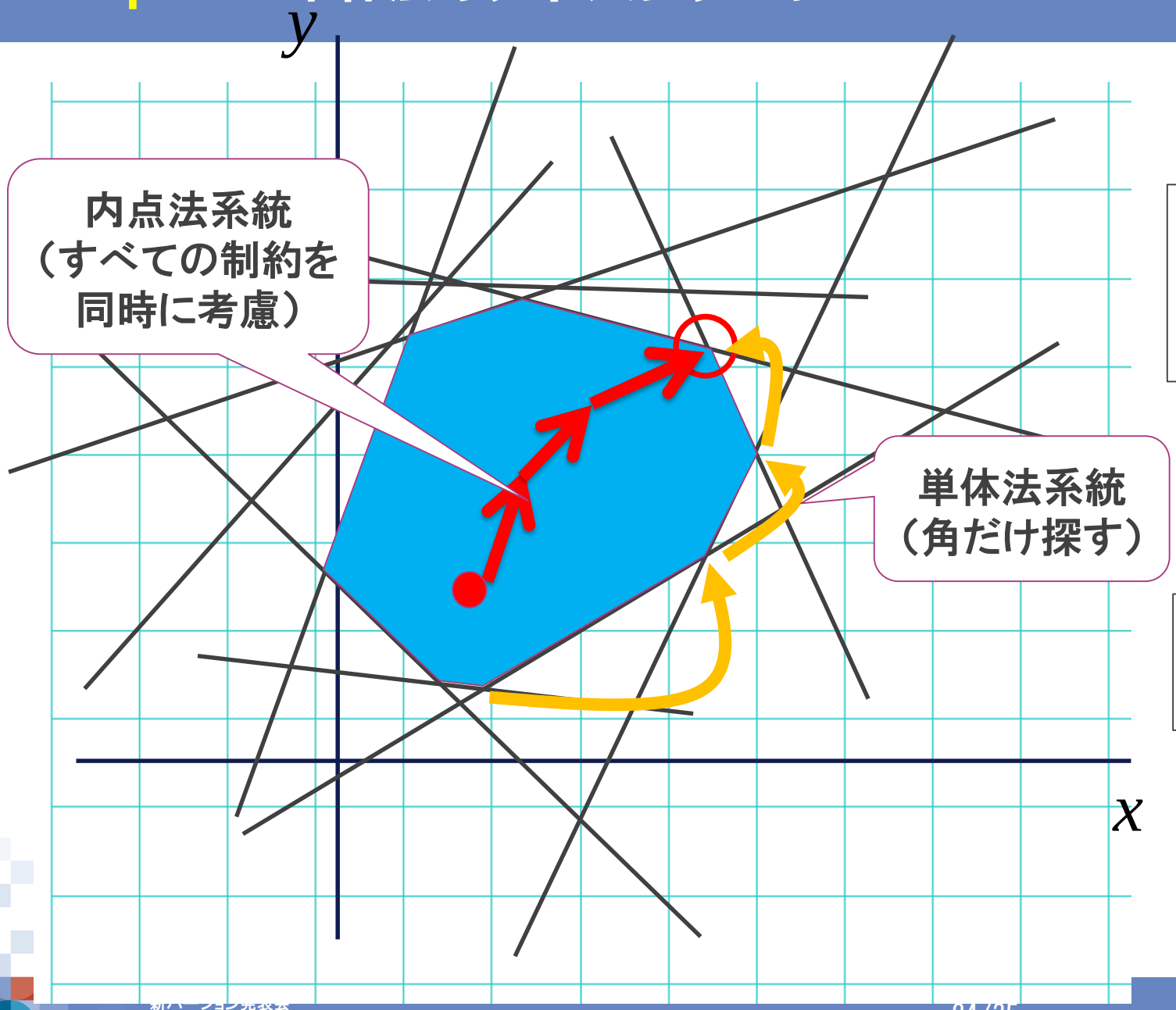
Nuorium Optimizer V24 新機能
hsimplex

hsimplex：LP（線形計画）と　QP （二次計画）



	単体法系統	内点法系統
LP	単体法 (simplex/dual_simplex/hsimplex)	高次内点法
QP	有効制約法 (asqp) 単体法 (hsimplex)	内点法

hsimplex：単体法のアドバンテージ



解に 0 が多い場合は
高速になる傾向！
（ポートフォリオ最適化など）

高速なリスタート
（単体法＋分枝限定法）

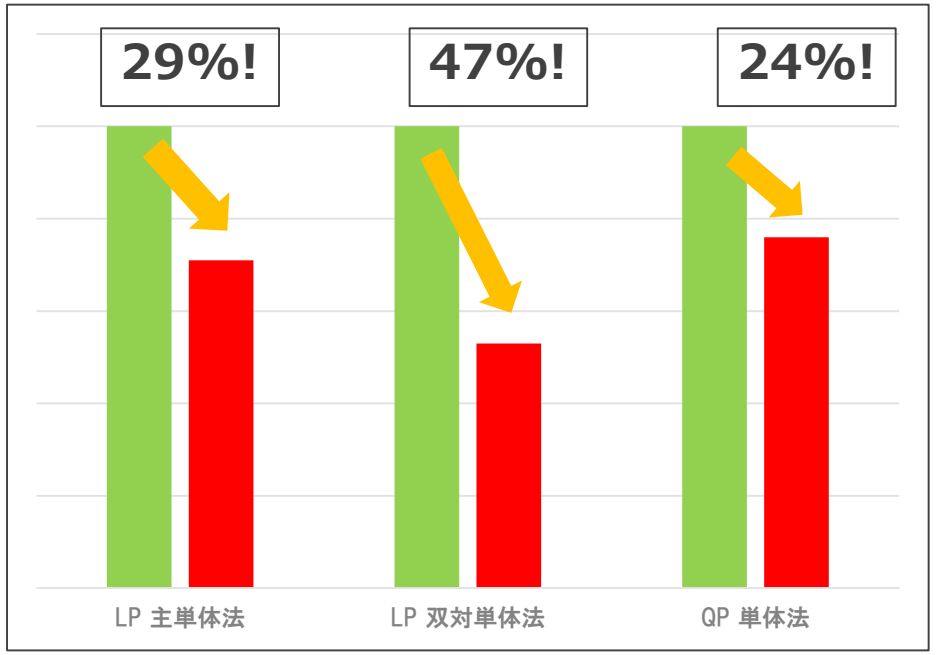
hsimplex:LP 及び QP に対するスパース単体法

- HSIMPLEX : スパース性をより活用した LP 及び QP に対する単体法

  ✓ スパースな問題(一つの制約式に少数の変数しか現れない問題)に対して高速！

  実装にあたり、共同研究をしているエジンバラ大学の Julian Hall 博士にご助言を頂きました。
  ここに感謝申し上げます。

- 当社従来の単体法と比較して
  - ✓ スパースな LP 主単体法 → 29% 高速化
  - ✓ スパースな LP 双対単体法 → 47% 高速化
  - ✓ スパースな QP 主単体法 → 24% 高速化



- ポートフォリオ最適化問題や輸送問題、起動停止問題の緩和などに是非お役立てください！

Nuorium Optimizer V24 新機能
wls QCQP

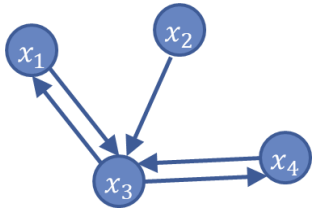
wls QCQP：二次の制約式に対応！

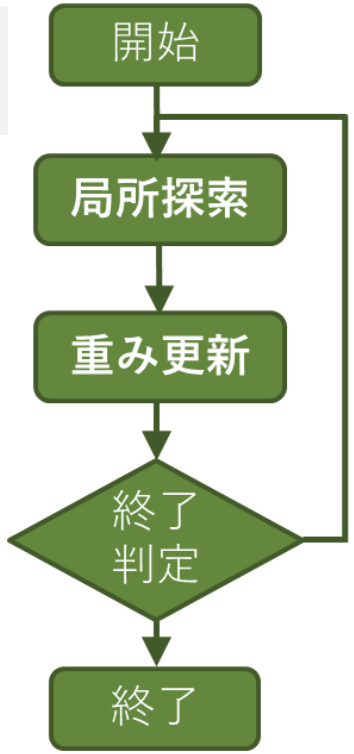
WLS （Weighting Local Search，重み付き局所探索法）

- 新たな局所探索法として V23 にてリリース

参考： S. Umetani: Exploiting variable associations to configure efficient local search algorithms in large-scale binary integer programs. *Eur. J. Oper. Res.*, Vol. 263, pp. 72－81, 2017.

- 整数計画問題に対する近似解法（二次の目的関数，ソフト制約も対応）
- 重み付き局所探索により，高速に近似解を探索
  - ✓ 隣接リストを用い，有用な近傍操作を厳選
- 0-1 係数の制約式に対し，特に高い性能を発揮
  - ✓ 問題構造の例： 割り当てる，選択する，…





V24 にて，さらに幅広い最適化問題を扱えるように

wls QCQP：二次の制約式に対応！

二次制約の最適化が可能に (IQCQP: 整数二次制約二次計画問題)

- V23: 二次の目的関数 & 線形の制約式（ソフト制約可）
- V24: 二次の目的関数 & 二次の制約式（ソフト制約可）

目的	$x - 2y + 3z + x^2 + yz - z^2$ → 最小化
条件	$x + z = 2$, $3x + 2y - z \leq 0$（重み 1 のソフト制約）,
変数	x, z は 0 or 1, $-1 \leq y \leq 3$, y は整数

V23: 二次の目的関数 に対応

- **使用事例**
  - **目標とのブレの最小化**
  - **平準化**
  - **二次割当問題**
  - **巡回経路の長さ最小化**　　**等々**

wls QCQP：二次の制約式に対応！

二次制約の最適化が可能に (IQCQP: 整数二次制約二次計画問題)

- V23: 二次の目的関数 & 線形の制約式（ソフト制約可）
- V24: 二次の目的関数 & 二次の制約式（ソフト制約可）

目的	$x - 2y + 3z + x^2 + yz - z^2$ → 最小化
条件	$x + z = 2$,
	$3x + 2y - z \leq 0$（重み 1 のソフト制約）,
	$x + y^2 + xz - z^2 \leq 5$
変数	x, z は 0 or 1,
	$-1 \leq y \leq 3$, y は整数

V23: 二次の目的関数 に対応

V24: 二次の制約式 に対応

- 使用事例
  - 目標とのブレを一定値以下に
  - 程よく平準化

等々

wls QCQP：二次制約の具体例

■ 総労働時間のバラつきを抑える（シフト割当）

	月 9:00	月 10:00	月 11:00	…	日 16:00	日 17:00
	○	○	○	…	—	—
	—	—	○	…	○	○
⋮	⋮	⋮	⋮	…	⋮	⋮

目的　…

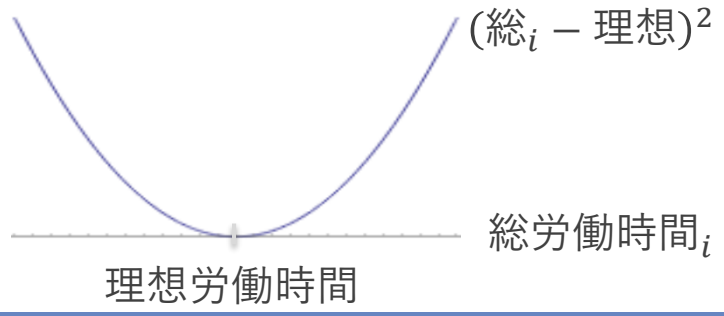
条件　…

$\sum_i$(総労働時間$_i$ − 理想労働時間)$^2 \leq V^2$

変数　…

総労働時間$_i$：従業員 i の労働時間の合計

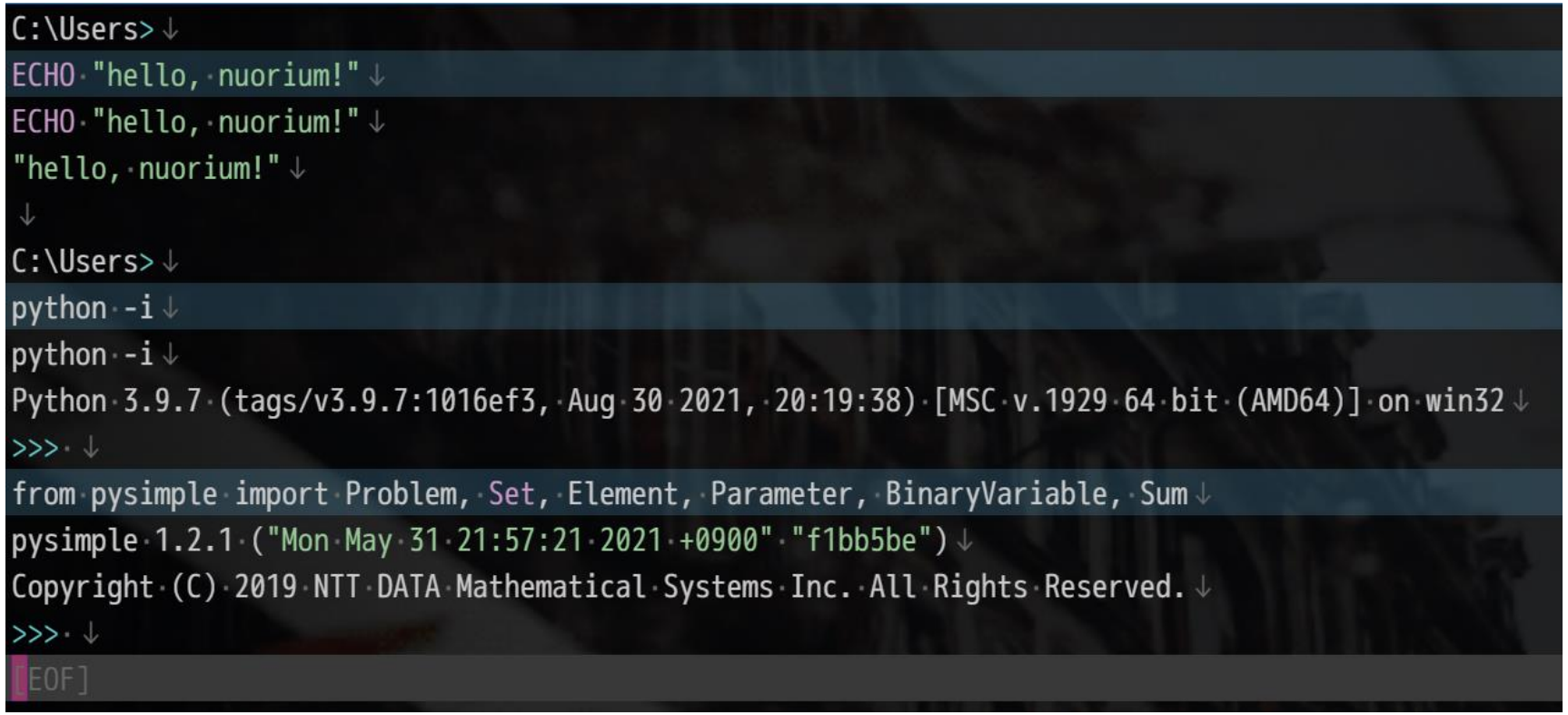
労働時間を程よく平準化



Nuorium Optimizer V24 新機能
Nuorium, 定式化技法集

Nuorium：ターミナル機能

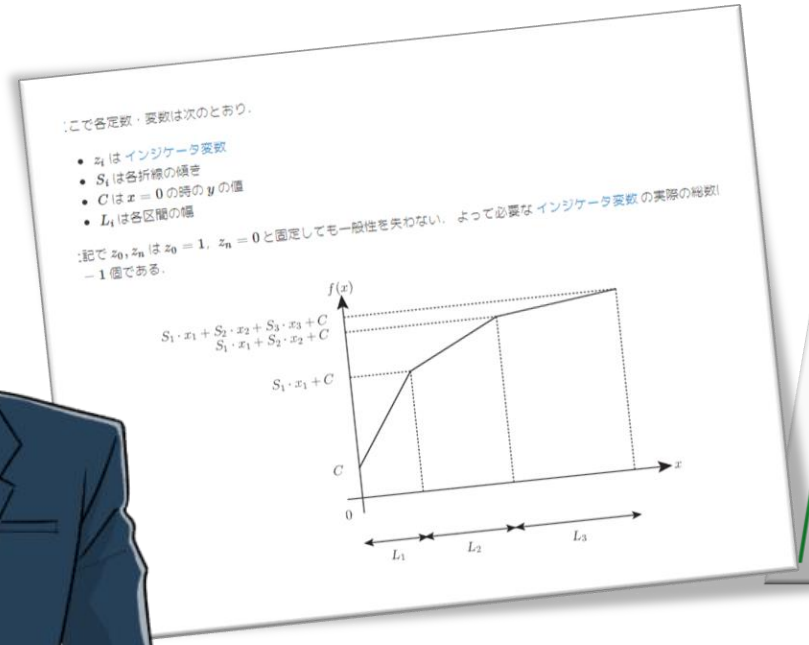
Nuorium にターミナル機能を実装しました。
コマンドプロンプトを呼び出して、様々なタスクの実行や、
PySIMPLE をインタラクティブに使用することもできます！

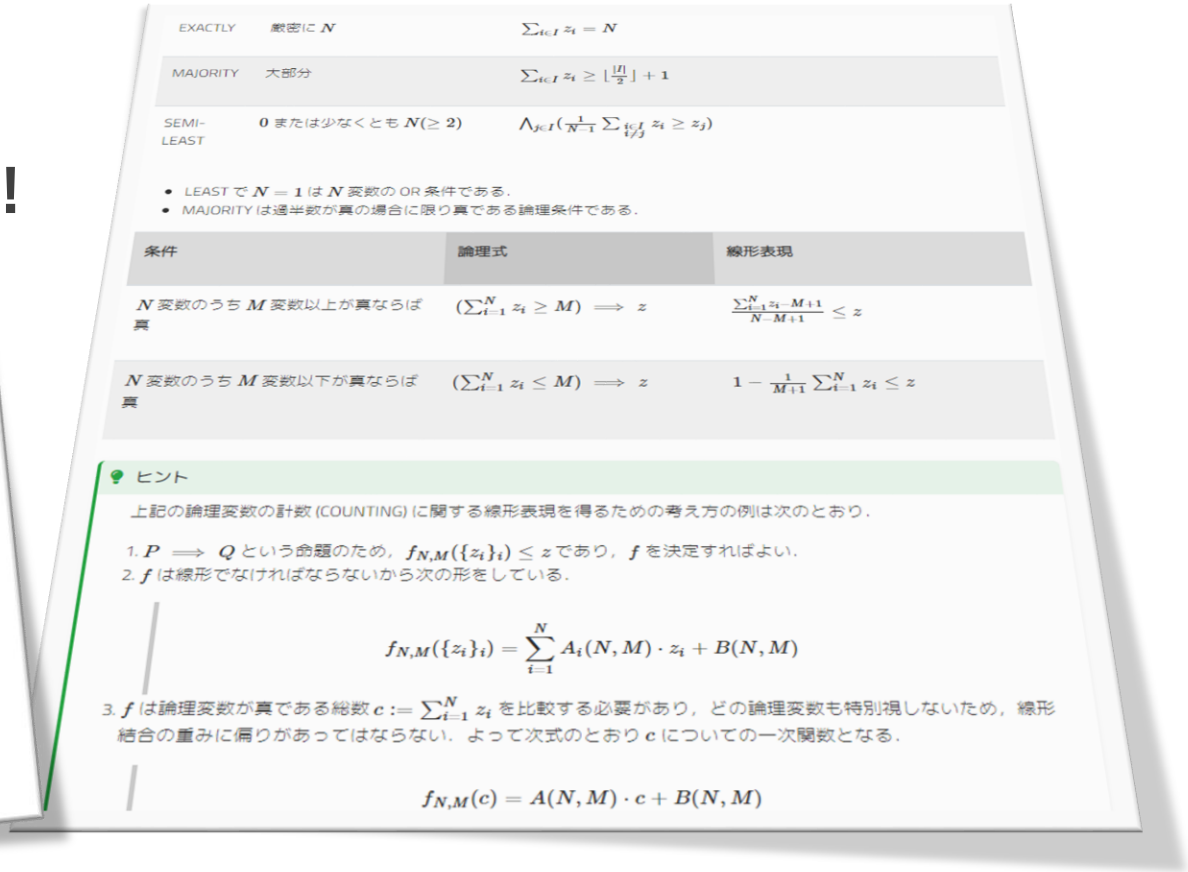


定式化技法集：モデリングのためのテクニカルドキュメント

開発版をご購入されたユーザー様がモデリングを独自に行う際に、
役立つ様々なテクニカルドキュメントを公開します。

実際に役立つテクニックばかりで、
実務家としてモデリングに取り組む際には必携です！





EXACTLY	厳密に N	$\sum_{i\in I} z_i = N$
MAJORITY	大部分	$\sum_{i\in I} z_i \geq \lfloor \frac{\lvert I\rvert}{2} \rfloor + 1$
SEMI-LEAST	0 または少なくとも $N(\geq 2)$	$\bigwedge_{j\in I}(\frac{1}{N-1}\sum_{i\in I, i\neq j} z_i \geq z_j)$

- LEAST で $N=1$ は N 変数の OR 条件である．
- MAJORITY は過半数が真の場合に限り真である論理条件である．

条件	論理式	線形表現
N 変数のうち M 変数以上が真ならば真	$(\sum_{i=1}^{N} z_i \geq M) \Longrightarrow z$	$\frac{\sum_{i=1}^{N} z_i - M + 1}{N-M+1} \leq z$
N 変数のうち M 変数以下が真ならば真	$(\sum_{i=1}^{N} z_i \leq M) \Longrightarrow z$	$1 - \frac{1}{M+1}\sum_{i=1}^{N} z_i \leq z$

ヒント

上記の論理変数の計数 (COUNTING) に関する線形表現を得るための考え方の例は次のとおり．

1. $P \Longrightarrow Q$ という命題のため，$f_{N,M}(\{z_i\}_i) \leq z$ であり，f を決定すればよい．
2. f は線形でなければならないから次の形をしている．

$$f_{N,M}(\{z_i\}_i) = \sum_{i=1}^{N} A_i(N,M)\cdot z_i + B(N,M)$$

3. f は論理変数が真である総数 $c := \sum_{i=1}^{N} z_i$ を比較する必要があり，どの論理変数も特別視しないため，線形結合の重みに偏りがあってはならない．よって次式のとおり c についての一次関数となる．

$$f_{N,M}(c) = A(N,M)\cdot c + B(N,M)$$

定式化技法集：テクニカルドキュメントの目次（一部）

目次

質疑応答

NTT DATA

NTT DATA Mathematical Systems Inc.