# 数理計画法モデリングとLocalSolver

宮崎　知明

1960 年代に，数理計画法システムは商用化され，現在に至っている．筆者は 1974 年に富士通株式会社に入社し，以来，数理計画法システムを中心とした OR 分野の研究開発，顧客支援に携わってきた．2016 年のオペレーション・リサーチ (Vol.61 No.1 2016) [1] では，「数理計画法システム進化の歴史と今後の方向」という題で数理計画法システムの進化について述べた．数理計画法システムで扱う最適化問題は，実数変数を定義し，線形不等式として表現した制約条件と目的関数によるモデル（LP モデル）を SIMPLEX 解法（LP 計算）で解くことが標準であった．また，整数変数を定義し，実数変数と整数変数が混合したモデル（MIP モデル）を解く混合整数計画法（MIP 計算）は，LP 問題として解いた解で，実数値をとる整数変数を，実数値を挟んだ上限制約（実数値の下限値）と下限制約（実数値の上限値）に分割した二つの子問題を作成し，それぞれで LP 計算を行いながら，最適化をしていく分枝限定法 (Branch & Bound) で解くことが標準であった．MIP モデルと LP モデルは同一のフォーマットでデータ化できた．数理計画法のモデリングは，解法に依存した多次元の線形不等式（LP モデル）を対象としたものとなっていた．このため，多次元問題，非線形問題等への適用はあまり進んでいないのが現状である．最近になって，色々な解法を装備した最適化システムが開発されつつあり，数理計画問題のモデリングの範囲も広がりつつある．本稿では，数理計画法システム用に特化した数理計画法モデリングについて，進化の歴史を振り返るとともに，次世代の数理計画法システムとして LP，MIP，NLP などの非線形最適化を実現した All-In-One ソルバーとしての LocalSolver の概要を紹介する[1]．

キーワード：数理計画法システム，SIMPLEX 法，B&B（分枝限定）法，MPS 形式データ，LP 形式データ

## 1. 1970 年代から 1980 年代

1970 年代は，計算センターの利用から，ユーザ各社がホストコンピュータを導入した時代であり，日本では，富士通，日本電気，日立などが，海外では，IBM，UNIVAC，CDC などが高性能のホストコンピュータを毎年のように市場に投入していた．現在のように PC もなく，GUI 端末もまだなかった時代であり，データは 80 桁のカードに一行一行パンチしたカードデータか磁気テープで，計算機に読ませる時代であった．磁気ディスクも最大で 200 MB であった．

また，オペレーティングシステム (OS) が各社ごとに異なるため，ホストベンダ各社は独自にソフトウェアを開発する必要があった．その当時の共通言語は技術計算用では FORTRAN，事務計算用では COBOL であった．

数理計画法システムはホストコンピュータの CPU を 90%以上占有したため，1000 式の LP モデルの実行に数時間かかったり，0-1 整数変数が 1,000 個を超える MIP モデルだと物理的に解けないことが発生していた．

### 1.1 数理計画法システムのモデル

その当時主流の数理計画法システムは，IBM が開発した MPSX：Mathematical Programming System eXtended，B&M 社が開発した UNIVAC，CDC の FMPS：Functional Mathematical Programming System であった．富士通は，FMPS をもとに MPS/X を開発し，日立は，IBM MPSX をもとに日立の MPSX を開発した．FMPS は FORTRAN で開発されていたが，IBM に対抗するため，富士通の MPS/X はアセンブラで開発を行った．また，富士通では，非線形最適化を行う独自の OSIV NLPS の製品化も行った．

当時の数理計画法システムは，線形制約で表現される LP 計算と MIP 計算だけが標準であった．当時，非線形計画問題は，多次元表現が必要なため，解法とデータの定義がばらばらとなることから，標準化はされず，FORTRAN などで解法ごとに異なるシステムであった．

当時，数理計画法システムの標準データとなった MPS 形式データについて述べる．

MPS 形式データで扱える問題は，LP：線形計画問題，MIP：混合整数計画問題，SEP：Separable Programming 問題（折線近似問題）であった．

当時は，個人画面（TSS 端末）もなく，80 桁の文字形式データであった．このため，数理計画問題を定義す

みやざき　ともあき
MSI 株式会社　数理モデリング研究所
〒 261–7102 千葉市美浜区中瀬 2–6 WBG マリブウエスト 2 階

[1] 本稿は文献 [1] をもとに加筆している．

るため，シンプレクスタブローを作成してデータを作成した．下記にシンプレクスタブローのイメージを示す．

説明のために数理計画法の問題例を示す．

**ナップサック問題**

**― ナップサックに入れられる，x1 から x8 の品がある．**

**― ナップサックには，総重量 102 kg まで入る**

**― 102 kg の重量制限で，価値の高いものを入れたい．**

数学表現は以下：

目的関数（最大化）：

```
1*x1+10*x2+15*x3+40*x4+60*x5+90*x6
    +100*x7+15*x8
```

重量制約（102kg 以下）：

```
1*x1+10*x2+60*x3+30*x4+40*x5+90*x6
    +20*x7+2*x8
```

シンプレクスタブロー表現は以下：

| | COLUMINS | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ROWS | s1 | s2 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 | | RHS |
| Object | 1 | | 1 | 10 | 15 | 40 | 60 | 90 | 100 | 15 | max | |
| Weight | | 1 | 10 | 60 | 30 | 40 | 30 | 20 | 20 | 2 | =< | 102 |

**図 1**　シンプレクスタブローの例

### 1.2　MPS 形式データ

当時の数理計画法システムの入力データは，IBM が提唱した MPS 形式データと呼ばれる業界標準のデータ形式だけであった．MPS 形式データは当時の計算機環境に適した構造をもち，現在でも世界標準として使うことができる．

MPS 形式の大きな特徴を以下に示す：

― 1 レコード 80 バイト（カードデータを意識）

― 行名，列名等の変数名は，アルファニューメリック（英数文字）で 8 文字以内

― 係数値は，12 桁の文字列

― データ定義は，行セクション，列セクション，RHS セクション等の順番で LP 問題を定義する

― マトリクスの係数値を列単位で定義する

当時はシンプレクスタブローを頼りに MPS 形式データを作成した．

MPS 形式データの表現例を以下に示す．

```
NAME                TOYMODEL
     ROWS
      N  OBJECT
      L  WEIGHT
    COLUMNS
        X_1       OBJECT       1.0
        X_1       WEIGHT       10.0
        X_2       OBJECT       10.0
        X_2       WEIGHT       60.0
          :
        X_8       OBJECT       15.0
        X_8       WEIGHT       8.0
    RHS
        RHS       WEIGHT       102.0
ENDATA
```

当時の数理計画法システムは，長時間かけて解いた最適解を有効活用するため，データの入力から最適化，解の保存・回復などを指示する独自の実行プログラムをもち，簡易言語プログラムで処理を定義し，独自のコンパイラで実行形式プログラムを作成して実行する形式であった．

MPS/X の簡易言語プログラムの例を以下に示す．

```
MCALL INITIALIZE
ADATA  = "TOYMODEL"   :入力データデックの名前
APBNAME = "TOYMPFEL"  :問題ファイル名
CALL INPUT        :MPS形式データの入力
AOBJ = "OBJECT"   :目的関数名
ARHS = "RHS"      :RHS名
AMINMAX = "MAX"   :最大化指定
CALL SETUP        :Matrixファイル等の準備
CALL SCALE        :スケーリング
CALL CRASH        :有効基底作成
CALL OPTIMIZE     :シンプレクス法による最適化
CALL SOLUTION     :解出力
STOP
END
```

**MPS/X** の簡易言語のなかには MPS/X-MG (Matrix Generator) と MPS/X-RG (Report Generator) の機能があり，シンプレクスタブローをイメージした MPS データを作成支援する機能や帳票出力機能があった．

計算機性能の進歩とともに，問題が大規模化し，固変数名の定義文字数（8 文字），数値の定義フィールド（12 桁）など固定のデータ形式では十分に解くことができなくなりつつあった．

## 2.　1990 年代から 2010 年代

1980 年代後半からは，コンピュータ環境が大きく変わってきた時代である．IT 技術の進歩は目覚しく，1980 年代では想像もしなかった環境の変化である．スーパーコンピュータの出現，超並列コンピュータの出現，CPU の超高速化，大量のメモリの使用である．

1988 年と 2003 年とで比較すると，アルゴリズムで 2,360 倍，ハードウェアで 800 倍早くなったため，トータルで 190 万倍早くなった実測がある．現在では，さらにコンピュータの性能が，飛躍的に向上するとともに，最適化計算機能の理論的，技術的な進化により 20 年間で，約 1,000 万倍の性能向上が実現でき，1000 式の LP 問題が数時間かかっていたのが，1 秒以下で最適化結果を得ることができるようになっている．

また，個人の計算機の使い方も大きく変わった．TSS 端末から PC 端末に変わりカード入力がなくなり，PC から直接プログラム，データを定義，開発することができるようになり，カードの時の入力データ制約の必要がなくなった．

大きく変化したのは，最適化アルゴリズムが進化するだけでなく，数理計画問題が UNIX，PC で簡単に解けるようになったことである．PC の一人一台の普及，インターネットの普及，ホストコンピュータの端末としての画面の利用により GUI が大きく進化したことで数理計画法システムも変わった．また，ソフトウェア環境が Windows と UNIX として標準化が進み，専門のアプリケーションベンダが台頭したことにより，数理計画法システムも大きく変わっている．主な商用ソフトは，XPRESS [2–4]，CPLEX [5]，Gurobi Optimizer [6] などである．以下にこの時代の進化を述べる．

### 2.1 LP 形式データ

IT 技術の発展に伴い，個人の PC 環境が進化するとともに，式形式のままで，LP 問題を定義したいという要求に対し，LP 形式がデファクトスタンダード化された．既存の数理計画法システムは MPS 形式データだけでなく LP 形式データを扱うようになった．LP 形式では，変数名字数制約，数値桁制約がなく自由に定義可能となった．このことは，精度トラブルの減少，変数定義の拡張が可能となり，問題定義の幅が大きく広がっていった．ただ，LP 形式データは，IBM の CPLEX がほぼ標準であるが，数理計画法ソフト毎に微妙な違いが残っており注意が必要となる．ここでは，Xpress の LP 形式データをベースに紹介する．

LP 形式データと MPS 形式データの大きな違いは，MPS 形式データが列変数ごとのカラムワイズ (column-wise) であるのに対し，LP 形式データは行ごとのロウワイズ (row-wise) であることである．

LP 形式データは以下から構成される．

— 目的関数
— 制約式
— 変数等の属性宣言

以下に LP 形式データの例を示す．詳細は，各ソフトウェアのマニュアルを参照されたい．

```
  Maximize
  Object: 1x_1 + 10x_2 + 15x_3 + 40x_4
          +60x_5 + 90x_6 + 100x_7 + 15x_8
  Subject To
  Weight: 1x_1 + 10x_2 + 60x_3 + 30x_4
          + 40x_5 + 90x_6 + 20x_7 + 2x_8
          <= 102;
  Binary
    x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8
End
```

LP 形式データは，MPS 形式データと同じように係数値をもつ完全な式にする必要がある．

大規模なモデルとなると，完全な式形式データを作成するためには，プログラムが必要となる．そこで，最近では，式データを生成するモデリングと最適化の実行制御を指示するモデリングシステムが商用化されている．代表的なソフトは AMPL である．また，各ソフトウェアベンダでも，モデリングシステムがある．

Xpress-Mosel はモデル作成用のプログラミング言語であり，配列記述や外部データ入力を駆使することで，式本体とデータを分離してモデルの構築が可能である．以下に mosel 言語の記述例を示す．

```
model "Chess"
 declarations
  small: mpvar ! Number of small chess
         sets
  large: mpvar ! Number of large chess
         sets
 end-declarations
 Profit:= 5*small + 20*large ! Objective
 Lathe:= 3*small + 2*large <= 160
 Boxwood:= small + 3*large <= 200
end-model
```

### 2.2 IT の進化による数理計画法システムの発展

ホストコンピュータのダウンサイジングが進み，技術計算分野でも UNIX が主流となりつつあった．PC，UNIX の画面は，GUI が優れており，このグラフィカル性を利用することを考えた．数理計画問題のモデリング（定式化）は，変数を定義し，目的関数と制約条件を，変数を使い一次結合式で定義することで，多次元の連立不等式を考えることである．そのため，MPS 形式

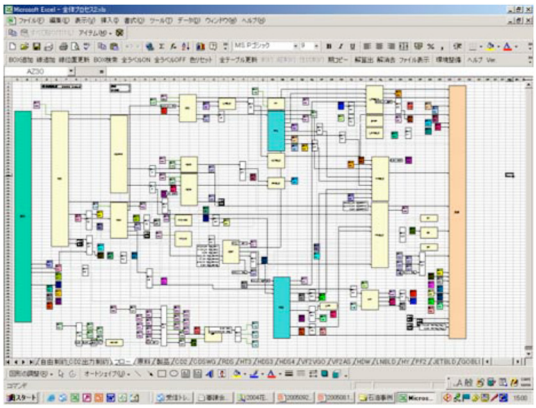

**図 2**　Excel ベースのモデリングのイメージ

のカードデータで 1000 式を超えるモデルを新規に開発する場合には，最低でも 6 ヶ月以上の期間が必要であった．数理計画問題は，物の流れと物と物との関係（推移）が基本である．装置系ではプロセスフローが，ロジスティクス系では，ネットワークフローが典型的な例である．物（原料，製品）がどう変化していくかを表現するのに，図で表現することが重要であった．式を考慮せず，Excel ベースでフローでモデルを定義できるよう，AMPS/VI：AMPS/Visual Intrface（UNIX 版），FRISolver（PC 版）として商品化を行った．

基本的な考え方は，「箱―線―箱」の「線」で物（変数）を定義し，「箱」で変数と変数の一次結合（線形関係）を表形式で定義することで，変数間の線形関係式を定義していき，最終的に，連立方程式を生成する．図 2 は製造プロセスのイメージであり，原料が処理され，最終的に製品に到達する流れを示している．この例で，作成された LP 問題は，1000 式以上の規模をもつ．

AMPS/VI，FRISolver は，従来の方法では，完成まで 6 ヶ月以上かかっていたモデルの構築を 1 ヶ月未満で完成させることを実現した．ビジュアルにモデルを表現し，最適化結果を図と表に反映させることで，LP 問題を局所化して検証していくことを可能とした．

### 2.3　最近のモデリング環境（GUI の進化と非線形最適化）

C 言語などの開発環境が出てきて，プログラムをリアルタイムで作成，実行することができるようになった．数理計画法システムでも開発環境を備えるシステムができた．当初の数理計画法システムは線形計画問題を入力して，最適解を出力するための専用の制御言語プログラムを実行させるだけであった．現在主流となった数理計画法システムは問題の構築と実行制御を同時に行うことができる開発環境をもつようになっている．

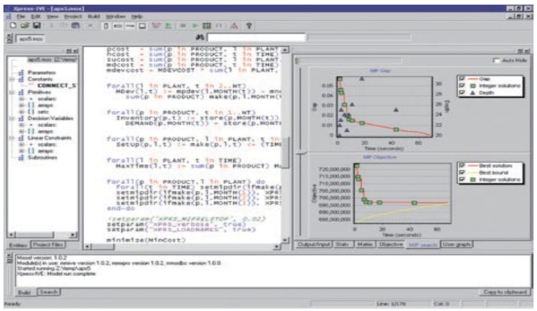

**図 3**　Xpress-IVE のイメージ

Xpress-IVE を例に最近のモデリング環境を紹介する．Xpress-IVE は Windows 環境で，Xpress-Mosel のための完全なビジュアル開発環境を提供する．

Xpress-IVE は，Mosel で書かれたモデルを編集，コンパイル，および実行を行う環境を用意している．また，Optimizer の性能分析，モデルのマトリクス表示，ソリューション表示をすることができる．さらに，解の表示，レポートの作成，解のグラフ表示も可能である [4]．

図 3 に Xpress-IVE のイメージを示す．

現在の各社数理計画法システムは，開発環境を持つ C++，C#，Java，Python 用の API を装備していることが多い．最近は，Python からの利用が多くなっている．たとえば，AMPLPY や PySIMPLE など．

また，各種最適化解法を装備し，線形問題から脱却した非線形を含む数理最適化問題を解くことをねらったシステムとして，LocalSolver，FICO Xpress，Gurobi Optimizer，NTT データ数理システム Numerical Optimizer，SIMPLE [7, 8] などが開発され，同一手順，定式化で非線形最適化機能を備えて，非線形最適化問題をも適用できるようになってきている．

## 3.　LocalSolver でみる数理計画法の最新動向

ビッグデータとして，社内データだけでなく，社外データをも自由に活用する環境が成立しつつあり，さまざまな情報の取得により，データの精度を高め，さらには，プランニング，スケジューリングによる情報活用が重要になっている．

最近は，大規模組合せ最適化の要求が高まっており，解法も進化している．制約論理プログラミング（CP：Constraint Programming），ニューロ，GA など，従来の数理計画法では解くことができなかった問題対応型の解法も実用化されている．本節では，**All-In-One Solver** を実現した次世代の数理計画法システム：**LocalSolver** を紹介する．[9–14]

従来の数理計画法システムとの違いは，以下にある．

— **自然なモデリングの実現**
- 線形不等式に固執する必要なく数式ベースで定式化が可能
- 意思決定変数を定義し，目的関数と制約条件式を意思決定変数で定義するだけでよい
- Python，Java，C#，C++ での自然な形式 (LSP) でモデリングが可能（それぞれ用の API を完備）
- モデル定義と一体化した最適化の実行（従来の各種最適化制御言語は不要）

— **高速な最適化計算**
- 実行可能な初期解を求め，実行可能性を崩さずず高速な解探索による解探索を実現（MIP 問題でも最初から整数解を改善していく）
- 高速なイタレーション計算により，大規模な問題でも 1 秒間に数十万回の解評価が可能
- 組合せ最適化に適した SetBound を導入した意思決定変数の定義による実用的な最適化問題の定義（List，Set）

— **LP，MIP，NLP を同一システムで最適化実現**
- 実行可能解が凸領域であれば，同一形式のモデリング定義によりモデルを事前分析し実行可能性と上界値を最初に計算し，最適化を実行

### 3.1 LocalSolver の概要

LocalSolver はフランスのコングロマリットの一つであるブイグループの研究所メンバー 4 人とマルセーユ大学の準教授 2 人により 2007 年から研究開発をした成果として，2012 年からブイグループの研究部門の子会社である Innovation24 社が商用化を開始しており，昨年（2019 年），LocalSolver 社と改名している．当初は，メタヒューリスティクスによる問題対応型の解法研究開発者の集まりであり，既存の数理計画法システムでは解けなかったスケジューリングおよび最適化問題を一つのシステムで解けるようにすることが目的であった．ビッグデータ時代を迎え，自動化したいスケジューリングや最適化計算問題のモデリング範囲，制約条件が複雑になっており，大規模な問題を汎用的に扱える実用解法の要求が高まっている．

100 万以上の 0-1 整数変数をもつような大規模な数理最適化問題は，既存の混合数理計画法システムや制約論理システムでは，解探索で組合せ爆発が起こり，実用時間内に解くことはできなかった．LocalSolver はこれらの問題を実用的に解くことを目的としている．

現実世界の多くの問題は大規模組合せ最適化問題となる．特に，下記のようなスケジューリング問題に関して，汎用的に解くことができるようすることを，LocalSolver は実現した．

- 車両の優先順位付け（組立）問題
- 裁断計画問題（フィルムなど）
- SCM 問題（製造－輸送－在庫－販売など）
- 最短路問題（カーナビのルート検索など）
- ネットワーク問題（交通網，通信網，電気，ガスなどの設計）
- 配送計画問題（宅配便，店舗への商品配送，ゴミ収集など）
- 施設配置問題（工場，店舗，公共施設などの配置など）
- 人員スケジューリング問題（看護士等の勤務表，時間割の作成など）
- 機械スケジューリング問題（工場の運転計画，装置稼働計画など）

LocalSolver の最適化部分は以下の意思決定問題を解くことができるようになっている．

**2012 年：0-1 変数 (bool)，ヒューリスティック**
**2013,2014：実数変数，整数変数**
**2015：List 集合，set 集合**

これは非線形最適化も実現したいという最近の要求を反映したものである．

解法としては，ローカルサーチ，メタヒューリスティックス，LP，MIP，MINLP，NLP を解くことができ，All-In-One ソルバとして実用に供する性能をもつ．

大規模最適化問題や非線形最適化問題を定式化するのに，従来のデータ形式では，開発に多大の工数を必要としていた．LocalSolver のモデル記述言語（LSP 言語）は，コンピュータの能力を最大限に利用した最新の関数型言語として，どんな最適化問題でも簡単に定式化できるようデータとロジックを分離したインタープリタ型言語となっている．本稿では，最新の LocalSolver の概要と LSP 言語によるモデリングの考え方を紹介する．

### 3.2 LocalSolver のモデリング言語 (LSP)

LSP は，第 4 世代言語である最新の関数型言語であり，作業変数の型宣言等は不要であり，コンパクトに問題と実行ステップを定義することが可能である．

LSP 言語モデルは，以下の要素から構成される．

— **意思決定変数**：bool() 他

— **副生変数**：任意の変数であり，プログラミングをわかりやすくすることができる．変数の定義には，<- を使用する．

— 制約：constraint（予約語）で，制約条件を定義する．Constraint の条件が実行可能性の判定で使用される．
— 目的関数：minimize（予約語）または maximize（予約語）で目的関数を定義する．

目的関数は複数定義可能であるが，定義された順番に最適化を行う．目的関数を複数定義して解く多目的計画法としても利用可能である．

LSP 言語は，モデリングおよびモデルのチューニングを行うフェーズで試行錯誤を行うのに最適な環境を提供する．

LSP 言語は，第 4 世代の関数型プログラミング言語であり，型推論を備えている．Java や C 言語と異なり，コンパイラが自動的にデータの種類（型など）を推定するため，データの型などを指定する必要がない．その結果，プログラムの記述は Ruby など軽量言語のように簡潔である．

LSP 言語の特徴は以下：

— 迅速に開発できる（開発生産性が良い．従来に比べて，1/5 から 1/2 の開発量）
— バグを抑えやすい（コンパイラが型などを自動的に定義しチェックする）
— アプリケーションの性能を向上させやすい
— 簡潔かつシンプルなモデリング言語（できるだけ省略できるよう設計）
※大規模問題でも制約条件およびデータがそろっていれば，1 日でモデリングと実行が可能である．
— 作成（修正）↔実行が同時にできる（一つはエディタ，もう一つは DOS コマンドプロンプトの二つのウインドウを操作しながら開発が可能である．
— 目的計画法のように目的を段階的に設定することができるため，モデルの開発及び解の検証を段階的に行うことができる．

LSP 言語プログラムの例を以下に示す．

```
function model(){
  // 0-1 decisions
  x[0..7] <- bool()；
  weights = {10,60,30,40,30,20,20,2};
  values = {1,10,15,40,60,90,100,15};
  // weight constraint
  knapsackWeight <- 10*x[0]+ 60*x[1]+
    30*x[2]+ 40*x[3]+ 30*x[4]+ 20*x[5]+
    20*x[6]+ 2*x[7];
  constraint knapsackWeight <= 102;
  // maximize value
  knapsackValue <- 1*x[0]+ 10*x[1]+
    15*x[2]+ 40*x[3]+ 60*x[4]+ 90*x[5]+
    100*x[6]+ 15*x[7];
  maximize knapsackValue;
  // secondary objective: minimize product
     of minimum and maximum values
  knapsackMinValue <- min[i in 0..7](x[i]
   ? values[i] : 1000);
  knapsackMaxValue <- max[i in 0..7](x[i]
   ? values[i] : 0);
  knapsackProduct <- knapsackMinValue *
   knapsackMaxValue;
  minimize knapsackProduct;
}
```

従来の LocalSolver は，0-1 意思決定変数 (bool) で，制約条件，目的関数を定義するのが基本であり，bool 変数の数がたとえ 1,000 万変数を超えても実用的な意味で解を求めることができた．

最新の LocalSolver は，上下限をもつ実数意思決定変数 (float)，上下限をもつ整数意思決定変数 (int) および list 関数で指定するセット化された変数を意思決定変数として定義できる．意思決定変数で定義された制約条件，目的関数を計算しながら，超高速に試行することで，最適化を目指している．

また，LocalSolver は，意思決定変数の入れ替えで，目的関数，制約条件の値を差分計算し，目的関数の単調性を保証することで，最適化を行っている．このため，目的関数，制約条件を線形式にする必要はなく，自由に非線形関数として表現可能である．

また，非線形関数を定義できない場合には，区分線形関数として定義することも可能である．

以下に list 関数を使った TSP の例を示す．

TSP の例

```
function model(){
  // cities[i] is the ith city in the tour
  cities <- list(nbCities);
  // All cities must be visited
  constraint count(cities) == nbCities;
  // Minimize the total distance
  obj <- sum(1..nbCities-1,i =>
  distanceWeight[cities[i-1]][cities[i]])
    + distanceWeight[cities[nbCities-1]]
    -[cities[0]];
  minimize obj;
}
```

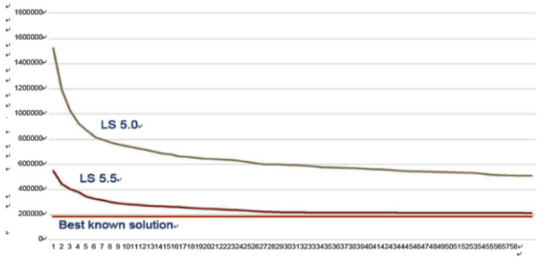


**図 4**　TSP 問題の改善例

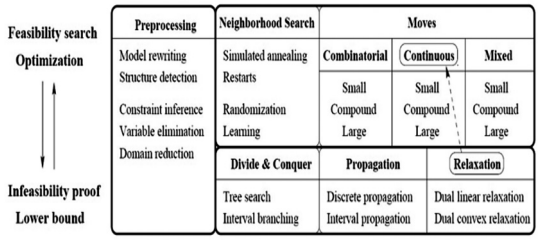


**図 5**　LocalSolver が取り入れている手法

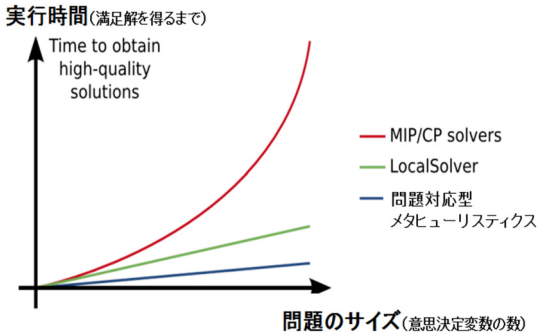


**図 6**　LocalSolver の計算時間の特徴

List 関数でセット化した意思決定変数の効果を図 4 に示す．200 箇所以上を廻る TSP の実例である．

List 関数の導入により，大規模な TSP，VRP 問題を実用的に解くことが可能となった．

LSP 言語は，短いステップでモデリングを行うことができ，モデルのチューニングを行うフェーズではリアルタイムで LSP 言語を変更しながら，試行錯誤を行うことができる．

### 3.3　LocalSolver の解法

LocalSolver は技術の集大成である．

— **最新の問題解析による上界の計算（凸最適化問題）**
— 非凸，非平滑な解空間でも最適化計算が可能とする各種最適化手法を内蔵
— 解を改良していく仕組み（**超高速な探索**）
— モデルの縮小と再定式化をする事前処理

LocalSolver は，近傍探索だけでなく，前処理，制約伝播など，既存の数理計画法システムが改善，開発してきたノウハウを取り入れている．図 5 に LocalSolver が取り入れている手法を示す．

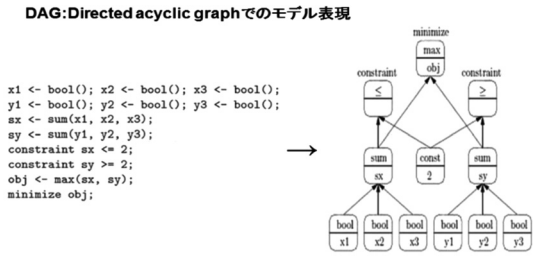


**図 7**　LocalSolver の変数選択のイメージ

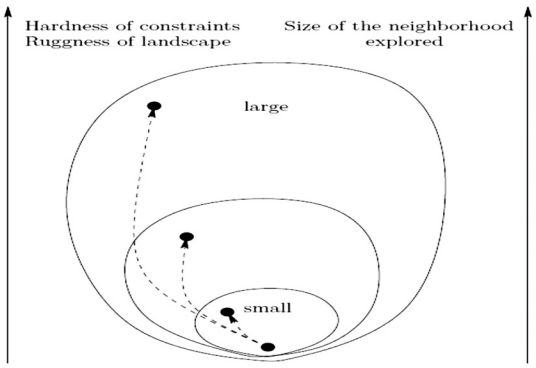


**図 8**　LocalSolver のイタレーションのイメージ

LocalSolver の解法は，図 6 に示すように，問題の規模が大きくなっても，計算時間が線形で比例することにある．LocalSolver のイタレーションは，意思決定変数を一つずつ動かしていくことで行われる．意思決定変数を変更したことによる，目的関数，制約条件を差分計算することで 1 イタレーションの計算量となる．このため，従来の数理計画法システムと違い，子問題の情報の保存や，掃き出し計算のような割り算が必要ない．

また，LocalSolver のイタレーションでの変数の入れ替えは，DAG グラフで示すように，現在の解位置と隣接する変数を選ぶことを基本としている．図 7 に DAG グラフでのモデル表現のイメージを示し，選択する変数候補を示す．

LocalSolver は実行可能解を見つけた後，小さな範囲で探索を開始し，徐々に範囲を広げていくような探索を行う．図 8 に探索のイメージを示す．

LocalSolver は，モデルの解析により，前処理で実行可能解を見つけて，実行可能性と単調性を保証しながら最適化を行う．

意思決定変数の入れ替えで，目的関数，制約条件の

値を差分計算し，目的関数の単調性を保証することで，最適化を行っている．このため，目的関数，制約条件を線形式にする必要はなく，自由に定式化が可能であり，最適解を見つけることが可能である．

## 4. おわりに

1970 年代には，顧客に OR チームができ，盛んに OR 手法を適用しようとしていたことが懐かしく感じられる．40 年前にはほとんど実用化できなかった大規模組合せ最適化問題に対して，OR 手法の知識が無くても，数理的知識がさえあれば，実践的な汎用アプローチが実現できる時代になったと考える．

今や，ビッグデータ + IoT で AI（自動化）が叫ばれれ，第 4 次産業革命と呼ばれつつある．データの精度も向上し，リアルタイムで解を導き出すことが求められ，かつ，答えられる時代を迎えていると考える．「実学に役立つ OR」として，人間と機械の調和を実現して日本の産業界の再生の一助となれば幸いである．

### 参考文献


[1] 宮崎知明，“数理計画法システム進化の歴史と今後の方向―All-In-One ソルバに近づく LocalSolver―，”オペレーションズ・リサーチ：経営の科学，**61**, pp. 35–42, 2016.

[2] MSI, https://www.msi-jp.com/xpress/（2020 年 2 月 20 日閲覧）

[3] FICO, https://www.fico.com/en/products/optimization/（2020 年 2 月 20 日閲覧）

[4] 渡辺展男，“数理計画ソフトウェアにおけるモデルと計算手続きの記述について，”情報科学研究，**38** 別冊，2017.

[5] IBM，http://www.ibm.com/software/commerce/optimization/cplex-optimizer/（2020 年 2 月 20 日閲覧）

[6] Gurobi, https://www.gurobi.com/（2020 年 2 月 20 日閲覧）

[7] NTT データ数理システム，https://www.msi.co.jp/（2020 年 2 月 20 日閲覧）

[8] 山下浩，“数理計画のためのモデリングツールの開発，”オペレーションズ・リサーチ：経営の科学，**50**, pp. 243–246, 2005.

[9] MSI, https://www.msi-jp.com/localsolver/（2020 年 2 月 20 日閲覧）

[10] LocalSolver, https://www.localsolver.com/（2020 年 2 月 20 日閲覧）

[11] 宮崎知明，山本邦雄，藤村茂，三竹春子，“新しい最適化システム All-In-One Solver の誕生―LocalSolver―，”OR 学会春季研究発表会予稿集，2020.

[12] 宮崎知明，“数理計画法システム最新動向―次世代数理計画法システム：LocalSolver 紹介，”スケジューリングシンポジウム予稿集，2014.

[13] T. Benoist, B. Estellon, F. Gardi, R. Megel and K. Nouioua, “LocalSolver 1.x: A black-box local-search solver for 0-1 programming, ” *4OR, A Quarterly Journal of Operations Research*, **9**, pp. 299–316, 2011.

[14] T. Benoist, J. Darlay, B. Estellon, F. Gardi and R. Megel, “LocalSolver 4.0 Hybrid Math Programming,” Presentation slides at INFORMS, 2014.