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 —

宮崎　知明

1960 年代に，数理計画法システムが商用化され，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．当時，LP 計算は大型計算機センターで利用されるのが普通であったが，ホストコンピュータが自社に導入されるようになり，LP 計算がベンチマークの主役になったケースもある．その当時の LP 計算では，ホストコンピュータの CPU 使用率が 90%以上に達することもあった．1000 式の LP モデルの実行に数時間かかることもあり，今とは隔世の感がある．本稿では，数理計画法システムの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なる可能性を秘めている LocalSolver の概要を紹介する．

キーワード：数理計画法システム，LP：線形計画法，MIP：混合整数計画法

1. 1970 年代から 1980 年代

1970 年代は，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， 80 桁のカードに一行一行パンチして，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，数理計画法システムの性能がホストコンピュータ選定の決め手となることもあった．

当時日本では，国策として IBM 互換路線をとることを目的とし，ホストコンピュータメーカ 6 社（富士通，日立，日本電気，東芝，三菱電機，沖電気）で 3 グループに分かれ，開発を行った経緯がある．

当時，ホストベンダはすべての分野で自社開発を行っており，顧客の資産移行は，共通言語からコンパイルし直しをすることで行った．

みやざき　ともあき
MSI 株式会社　技術顧問（数理モデリング研究所長）
〒 261–7102　千葉県千葉市美浜区中瀬 2–6　WBG マリブウエスト 2 階

1.1 数理計画法システム

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 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，非線形最適化の研究を行い OSIV NLPS として製品化した．

当時の数理計画法システムは，基本的には以下の機能をもち，長時間の実行にも耐えられるようになっていた．

・MPS 形式データ入力処理
・入力された問題ファイルの変更処理
・Matrix ファイル，逆行列ファイルの作成
・行列のスケーリング
・解の一時保存，再実行処理
・解の出力処理
・感度分析機能（レンジング，パラメトリック）
・MG：Matrix Generation　モデル作成
・RG：Report Generation　報告書作成

また，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．

1.2 当時の数理計画法システム（MPS/X の例）

当時の数理計画法システムは，独自の簡易言語プログラムをもち，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，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 = ‘‘INPUT’’ ：入力データデックの名前
APBNAME = ‘‘PROBLEM’’ ：問題ファイル名
CALL INPUT ：MPS 形式データの入力
AOBJ = ‘‘OBJ’’ ：目的関数名
ARHS = ‘‘RHS’’ ：RHS 名
AMINMAX = ‘‘MAX’’ ：最大化指定
CALL SETUP ：Matrix ファイルなどの準備
CALL SCALE ：スケーリング
CALL CRASH ：有効基底作成
CALL OPTIMIZE：シンプレックス法による最適化
CALL SOLUTION：解出力
STOP
END
```

1.3 精度との戦い

当時のホストコンピュータは，メモリが十分でなく（1 MB もない）インコア処理ができない状況であった．また，IBM 互換機のハードは，倍精度計算の場合，16 進数の 64 ビットであったため，仮数部は 16 進で 14 桁であった．これはつまり，$1.0\times10^{-10}/1.0\times10^{+10}$ では，有効桁がなくなってしまうことを意味する．また，数理計画法システムの最適化計算では，掃き出し計算が必要であり，ピボットとして選択した数値で割り算する必要が生じる．

このため，倍精度演算での桁落ちを防ぎ，計算精度を維持するために，以下の機能を備えた．

・マトリクスのスケーリング
　行中の最大係数が 1 となるよう式を最大係数で割るなど
・各種トレランス指定による倍精度計算の制御
　・絶対ゼロトレランス (FABSZT)
　・相対トレランス (FRELZT)
　・ピボットトレランス (FMPIVOT)　　など
・逆行列作成し直しタイミングの指定

富士通の MPS/X では，アセンブラで倍精度計算を行い，計算精度を維持する工夫を行った（内積計算時，プラス要素とマイナス要素を別々に計算し，最後に合算するなど）.

1.4 スピード競争

当時，スピードを妨げる最大の要因はマトリクス情報と逆行列情報をファイルでもつ必要があり，インコア処理ができないことであった．IBM 互換機の開発により，メモリもある程度自由に使えるようになり，マトリクスと逆行列をメモリに格納し，インコア計算を行えるようにした（MPS/X-HLP の名前で商品化）．この結果，IBM MPSX/370 と互角に性能比較をすることが可能となった.

MPS/X-HLP で実現したインコア処理を以下に示す：

・独自のハッシュ関数による行名，列名のハッシュ化（8 バイト）
・マトリクス係数のハッシュ化
・指数部だけでのスケーリング
・指数部だけでの演算シミュレーション
・事前解析によるモデルの縮小

1.5 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 などが提唱した 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，列名などの変数名は，アルファニューメリックで 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 セクションなどの順番で 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

MPS 形式データの簡単な例を以下に示す.

```
NAME                INPUT
ROWS
  N   OBJ
  L  RUBBER
  L   POWER
COLUMNS
    COLX     OBJ         5.0
    COLX     RUBBER     10.0
    COLX     POWER       8.0
    :
RHS
    RHS     RUBBER     100.0
    RHS     POWER      200.0
ENDATA
```

1.6 MPS/X が扱う問題と MPS/X の使い方

当時の数理計画法システム（MPS/X など）は以下の問題を扱うことができた.

LP：線形計画問題
MIP：混合整数計画問題
SEP：折線近似問題

当時は，1000 式を超える LP 問題や 100 整数変数を超える MIP 問題を解くには数時間から数日必要とすることが一般的であった．

このため，最適化計算を途中で中断し，再開したりするための，以下の機能が備わっていた．

・BASISIN，BASISOUT
　カード形式の基底を入力し，現在の基底解を再現 (BASISIN)，現在の基底解を保存し (BASISOUT)，次回の再開に備える機能
・SAVE，RESTORE
　現在の最適化情報を MPS/X の問題ファイル (PBFILE) に保存 (SAVE) したり，保存した最適化情報から最適化計算を再開 (RESTORE) したりする機能
・REVISE，MODIFY
　一度最適解を求めた問題を部分的に修正する機能．REVISE, MODIFY と SAVE/RESTORE, BASISIN/BASISOUT などを併用すれば，新しい問題を効率よく解くことができる．

また, 最適解の事後分析用に, RANGE, PARAOBJ, PARARHS，PARARIM の機能を備えていた．RANGE は現在の最適解（最適基底）を崩さずに，目的関数，制約値などをどれだけ変化させることができるかを示す．一方，PARAOBJ，PARARHS，PARARIM は変更範囲を指定することで，最適解がどう変化するかを示すことができた．

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．

大きく変化したのは，数理計画問題が UNIX，PC で簡単に解けるようになったこと，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利用，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．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 万倍速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，コンピュータ環境が PC (Windows) および 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．以下にこの時代の進化を述べる．

2.1 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）

1980 年代後半から，ホストコンピュータはサーバ，スーパーコンピュータへと進化していった．それは，ベクトル処理と並列処理である．富士通の数理計画法システムも当時最新のロジックをもつ MPS/X-HLP と SCICONIC/F（最新の MIP ソルバ）を融合させ，当時最速の性能をもつ AMPS シリーズとして，製品化していった．

シンプレックス法の計算では，列決定の dj 計算が内積計算となる．このため，ベクトル型のスーパーコンピュータで実装をした（AMPS/VP として商品化）．

結果としては，内積計算部分の性能改善は 90%以上の性能向上であった（dj 計算部分を一桁速くすることができた）が，全体としての効果は，20～30%の改善であった．

次に，並列型のスーパーコンピュータへの実装を試みた．MIP の解法である B&B（分枝限定法）への適用を試行した．並列型スーパーコンピュータでの試行結果は 100 台の並列マシンで 64 倍の性能を出すことができ，並列度を上げると最適化計算性能は，並列度と線形で比例することが実証できた．ただ，当時はビジネス利用での並列スーパーコンピュータが浸透せず，商品化は断念したことを覚えている．MIP の問題は，実数の値をもつ整数変数を上限の整数に固定した二つの子問題を生成しながら，最適化を進めることになる．子問題の数は，理論上は 2 の n 乗個となるため，超並列でもあれば別であるが，線形の性能向上では，実用的にはならなかった．

2.2 IT の進化による数理計画法システムの発展（GUI の進化）

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．Windows，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，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 カ月以上の期間が必要であった．数理計画問

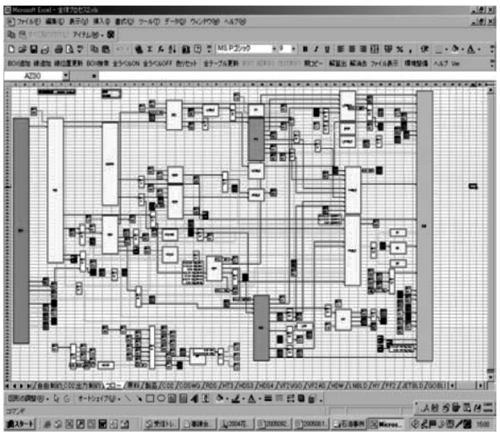


図 1　Excel ベースのモデリングのイメージ

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，フローでモデルを定義できるよう，AMPS/VI：AMPS/Visual Interface（UNIX 版），FRISolver（PC 版）として商品化を行った．

物の流れを「箱―線―箱―線―箱―線…」でグラフィカルにモデルを定義していくことで，LP 問題の定式化を実現したものである．図 1 に Excel ベースのモデリングのイメージを示す．

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 1 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 カ月以上かかっていたモデルの構築を 1 カ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　サードパーティソフトの台頭

1990 年代に入り，技術計算のダウンサイジングがおき，UNIX，PC (Windows) がソフトウェア環境として，デファクトスタンダードになったことで，商用化された数理計画法システムが多数出現した．商用化された数理計画法システムには，最適化計算機能だけでなく，開発環境，モデリング環境，結果の利用環境を備えている場合が多い．表 1 に 2000 年代初頭の標準的な数理計画法システムを示す．

表 1　主な数理計画法システムの例（2003 年当時）

	DASH	ILOG	数理システム	GNU
数理計画	Xpress-MP	CPLEX	NUOPT	GLPK
制約論理	Xpress-CP	Solver	（検討中）	－
MIP機能	○	○	○	△
モデリング記法	Mosel	Concert	Simple	－
ビジュアル開発環境	Xpress-IVE	OPL Studio	NUOPTGUI	－
有償・無償	有償	有償	有償	無償
備考	デファクト	デファクト	非線形に強い	保守なし

表 1 に示した数理計画法システムベンダは現在ではすべて変わっているが，商品は現在も続いている．DASH 社は FICO に，ILOG 社は IBM に，数理システムは NTT データ数理システム社へと変わっている．また，最近では，CPLEX 開発メンバが Gurobi を商品化している．とくに，Xpress-MP は，ダウンサイジングの先駆者であり，日本でも MSI 株式会社が 26 年間にわたり，提供を続けている．この中で，GLPK は，オープンソフトウェアである．

2.4　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，MPS 形式のほかに，LP 形式がデファクトスタンダード化された．ただし，LP 形式データは，数理計画法ソフトごと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 形式データがカラムワイズ (column-wize) であるのに対し，ロウワイズ (row-wize) 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
・変数などの属性宣言

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．

```
max : 3x + 4y;
x <= 10;
y >=  5;
row1: 2x + 5y <= 100;
row2: 3x + 2y <= 200;
int x;
```

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 LP 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，LP 形式データを生成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．次に Xpress のモデリング言語である Mosel と開発環境である Xpress-IVE を紹介する．

2.5 最近のモデリング環境

Xpress-Mosel と Xpress-IVE を例に最近のモデリング環境を紹介する．

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 od 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
```

Xpress-IVE は Window 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．

図 2 に Xpress-IVE のイメージを示す．

3. 数理計画法システムの最新動向

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．

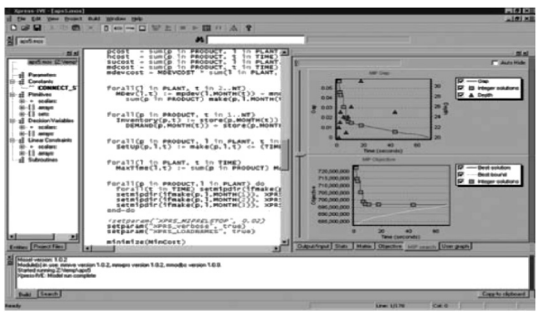


図 2 Xpress-IVE のイメージ

表 2 MIPLIB ベンチマーク比較

instances	Status	Variables	LocalSolver 3.1	Gurobi 5.5	Cplex 12.4	Optimum
opm2-z10-s2	hard	6,250	* -25,719	-19,601	-18,539	-33,826
opm2-z11-s8	hard	8,019	* -33,028	-21,661	-18,883	-43,485
opm2-z12-s14	hard	10,800	* -46,957	-11,994	-36,469	-64,291
opm2-z12-s7	hard	10,800	* -46,034	-12,375	-30,887	-65,514
pb6	hard	462	-62	-62	-62	-63
queens-30	hard	900	-38	-36	-39	-40
dcl1	open	37,297	11,100,000	21,300,000	1,840,402	unknown
de-big	open	6,020	9,864	62,520	5,256	unknown
exl010-pi	open	25,200	249	251	247	unknown
ivu06-big	open	1,812,044	* 475	9,416	678	unknown
ivu52	open	1,423,438	4,907	16,880	3,285	unknown
mining	open	753,404	* -65,720,600	902,969,000	no solution	unknown
pb-simp-nonunif	open	23,848	* 90	140	94	unknown
ramos3	open	2,187	* 223	274	267	unknown
rmine14	open	32,205	* -3469	-170	-968	unknown
rmine21	open	162,547	* -3657	-184	no solution	unknown
rmine25	open	326,599	* -3052	-161	no solution	unknown
sienal	open	13,741	256,620,000	315,186,152	54,820,419	unknown
sts405	open	405	342	342	354	unknown
sts729	open	709	648	648	665	unknown

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も実用化されている．本節では，All-In-One Solver を目指した次世代型の数理計画法システム：LocalSolver の概要を紹介する．

表 2 に MIPLIB のベンチマーク比較を示す．

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グループの研究所メンバ 4 人とマルセーユ大学の准教授 2 人による過去 10 年間の研究開発成果であり，2013 年からブイグ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いる．彼らは，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である．目的は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にある．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，既存の数理計画法システム (MIP) や制約論理システム (CP) 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のが現状である．

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にすることを，LocalSolver は狙っている．

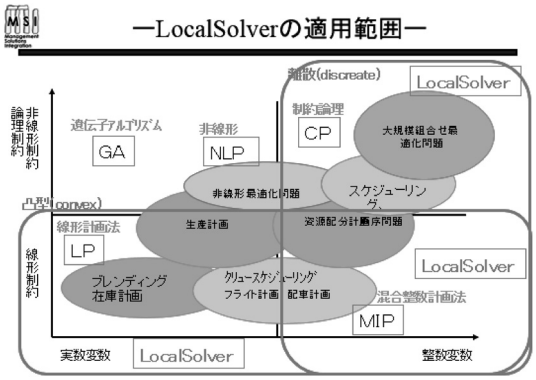


図 3　LocalSolver の適用範囲

・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（看護師などの勤務表，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）

大規模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であり，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．LocalSolver の適用範囲を図 3 に示す．

3.2　LocalSolver の解法

LocalSolver は，以下の問題（手法）を統一的に定義し，最適化することができる．

・LP（線形計画法）
・IP（整数計画法）
・MIP（混合整数計画法）
・NLP（非線形計画法）
・CP（制約論理計画法）

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 4 に LocalSolver

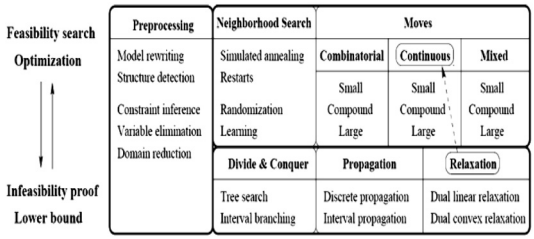


図 4　LocalSolver が取り入れている手法

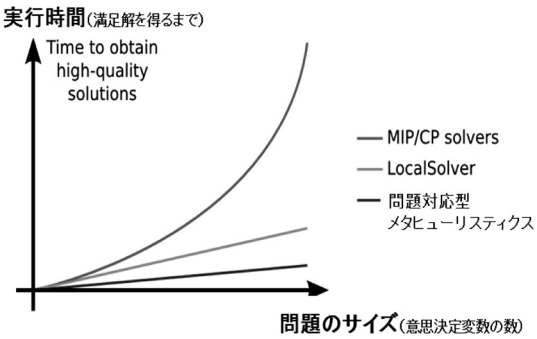


図 5　LocalSolver の計算時間の特徴

が取り入れている手法を示す．

LocalSolver の解法は，図 5 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 6 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 7 に探索のイメージを示す．

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．

3.3　LocalSolver のモデリング言語（LSP）

LSP 言語モデルは，以下の要素から構成される．

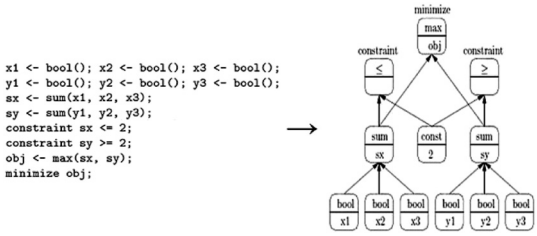


図 6　LocalSolver の変数選択のイメージ

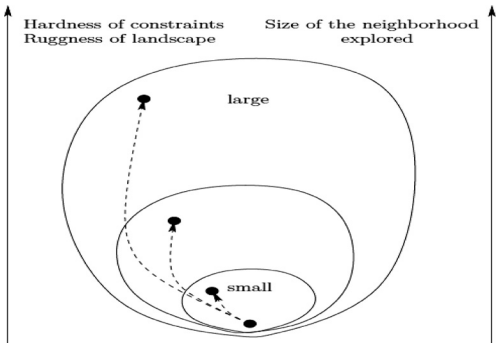


図 7　LocalSolver のイタレーションのイメージ

・**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，最新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（開発生産性がよい．従来に比べて，1/5 から 1/2 の開発量）
・バグを抑えやすい（コンパイラが型の間違いなどを自動的にチェックする）
・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，モデルの開発およ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 1000 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．

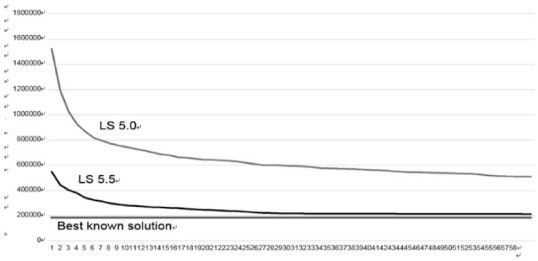
以下に list 関数を使った Traveling Salesman Problem：TSP 問題の例を示す．

TSP 問題の例

```
function model() {

  x <- list(N) ; // order n cities {0, ..., n-1} to visit
  constraint count(x) == N; // exactly n cities to visit
  minimize sum[i in 1..N-1]( Dist[ x[i-1] ][ x[i] ] )
                  + Dist[ x[N-1] ][ x[0] ] ; // minimize sum of traveled distances
}
```

List 関数でセット化した意思決定変数の効果を以下に示す．200 カ所以上を廻る TSP 問題の実例である．



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．

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．30 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，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）がさけばれ，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] 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.

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

[3] 中原啓一，三次衛，『システムエンジニアハンドブック』，オーム社，1982.

[4] 今野浩，『役に立つ一次式』，日本評論社，2005.

[5] http : / / jp . fujitsu . com / group / fri / downloads / develop/abst_moki.pdf

[6] https://www.msi-jp.com/xpress/

[7] http : / / www . fico . com / en / products / fico-xpress-optimization-suite

[8] http://www.ibm.com/software/commerce/optimization/cplex-optimizer/

[9] https://www.msi.co.jp/nuopt/

[10] http://www.gurobi.com/

[11] http://www.msi-jp.com/localsolver/

[12] 宮崎知明，大西真人，常盤晋吾，“SCM 最新動向と大規模最適化問題実用化への取組み，”日本 OR 学会春季研究発表会　アブストラクト集，2005.

[13] 宮崎知明，茂木恵美子，池ノ上晋，“プロセス産業の管理会計を支援する数理計画法の新しい活用，”日本 OR 学会春季研究発表会　アブストラクト集，2006.

[14] 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.

[15] 宮崎知明，“LocalSolver を中心とした数理計画法システムの最新動向，”日本 OR 学会春季研究発表会　アブストラクト集，2015.

