【第22回SCSR発表会資料】

ローカルサーチ法による次世代数理計画法システム：LocalSolver4.0とその適用事例

2014年5月27日

宮崎　知明（MSI株式会社）

はじめに

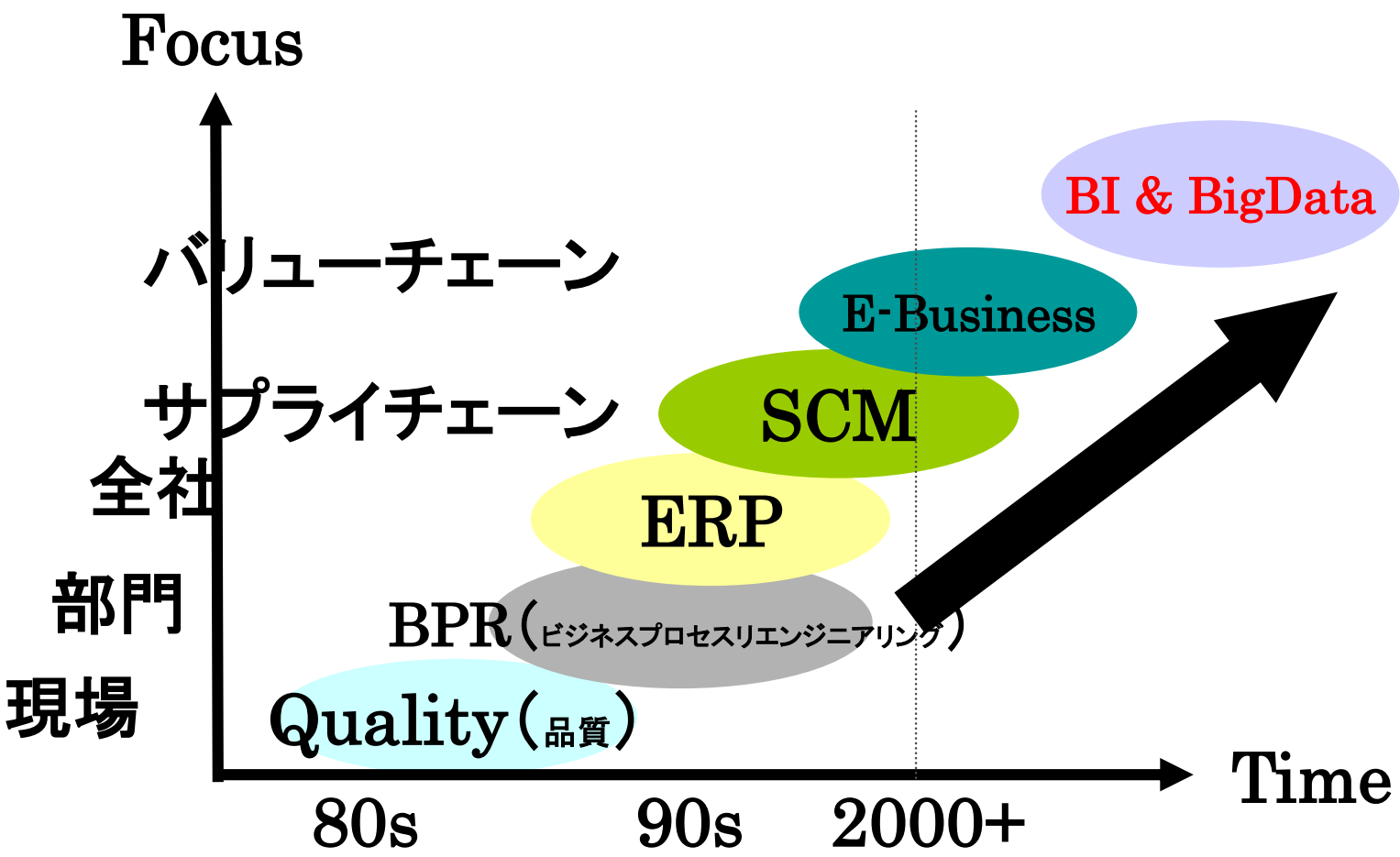
- SCMの浸透により、大規模最適化問題の解決がますます重要になっているが、数理計画法システムの限界で苦労しているのが、現状。
- 従来の方法（MIP：混合整数計画法）では解けない**大規模組合せ 最適化問題（800万以上のパターンの組合せから最適な組み合わせを求める）**を実用的に解くことができるようなソフトが出現。
- 従来の数理計画法システムの良い点と最新のIT技術を活用した**近傍探索による全く新しい統合数理計画法システム（LocalSolver)** が実現された。
- どんな現実問題をも直接モデル化できる時代（**非線形制約、論理制約、非線形目的関数**）になった。

目次

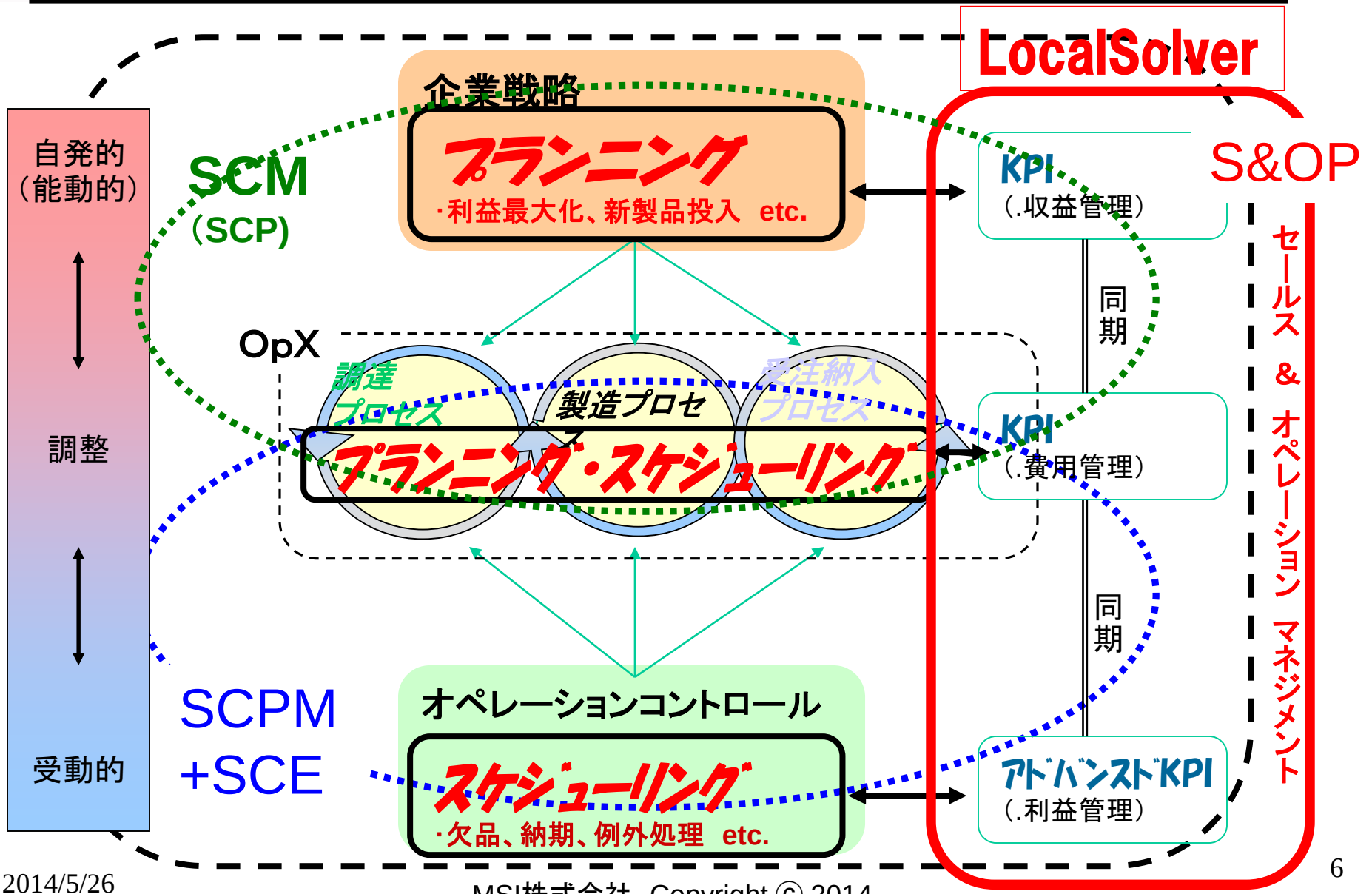
1. 大規模最適化の歩み

SCM市場動向

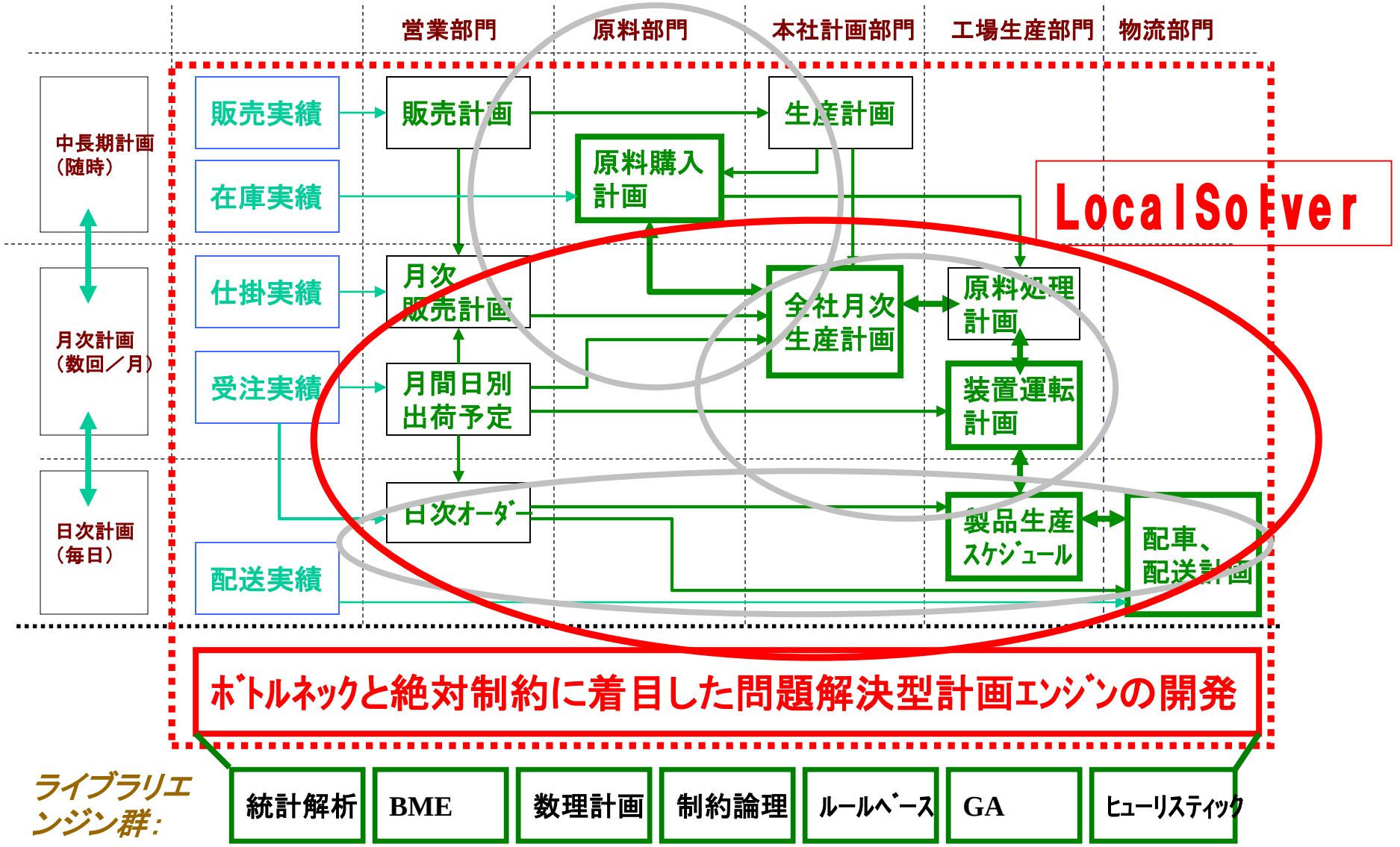
ソリューションの進化



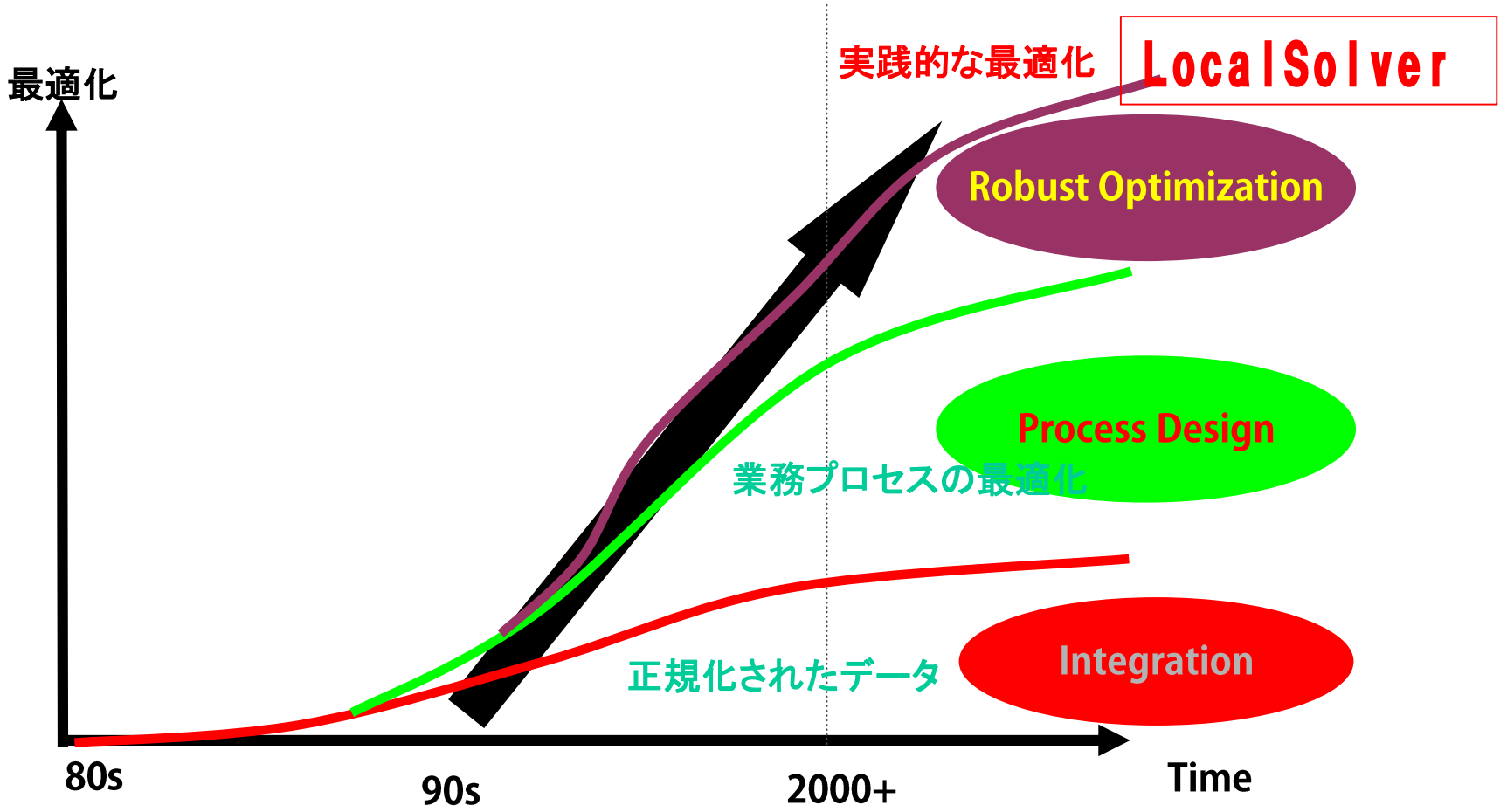
SCMソリューションの概念



問題解決型アプローチの実現へ



大規模最適化の持つ意味
―管理可能な範囲が増大―



技術の進化により実践的な全体最適への道が開けつつある

大規模最適化問題とは

※過去の事例

- **毎週数百万トンの鉄鋼生産をスケジューリング**
  新日鉄、POSCO、USINOR他
- **毎日500万個以上の荷物を配達**
  UPS，FEDEX他
- **毎日170,000人のクルーをスケジューリング**
  AmericanAirline，UnitedAirline，DELTA，BA他

- 石油化学での原価配賦計算(数万行)
- 日本全国のタンクローリの配車日程計画
- 日本全国のキャリアカーの配車日程計画

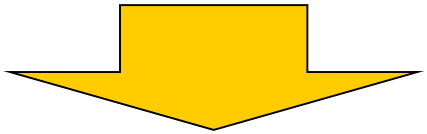
２．SCM事例（従来手法）

※当日プレゼンします。

ー最適化事例における成果ー

過去の事例

- **ロジスティクスコストが10%削減**
  企業合併に伴う拠点再配置による生産コストと物流コストの削減
- **原料費が2%削減**
  石油化学での原料調達の業務プロセス改善（月次計画40日以上を1週間で）
- **部品調達の物流コストが5%削減**
  自動車メーカでのサプライヤ – 物流センタ – 工場間



計画業務の見直しと数理最適化技術の導入により
最低でも、2～10%のコスト削減が期待できる

3．LocalSolverの必要性

何故、今、LocalSolverか？（現状）

大規模最適化の現状

課題レベル	課題
出来ないこと	大規模組合せ最適化問題が解けない
適用の課題	時間がかかる
	大規模なシステム資源が必要（CPU，メモリ等）
運用の課題	モデルチューニングがタイムリーに出来ない
	モデルの変更が簡単にできない

LocalSolverは、仏国のInnovation 24社が開発した他社に先駆けてローカルサーチ法をベースとした数理計画システムのソフトウェアである。

LocalSolverは、数理最適化分野で広く利用されているMIP（混合整数計画法）、CP(制約論理プログラミング)が持つ上記問題を解決することを狙ってる。

何故、今、LocalSolverか？（技術的背景）

従来方式（分岐限定法）では実現不可能な世界

現在適用されている最適化（数理計画法）は、混合整数計画法（MIP)または制約プログラミング（CP）が多くを占めており、分岐限定法をベースにしている。
分岐限定法は、整数変数を一つ一つ固定して子問題、孫問題と枝わかれさせながら、解空間を探索していくため最適解を見つける過程においては解の組合せが親子関係で決まってしまい、枝別れをして探索していくため、最適解探索時間が指数関数的に増加する。
また、目的関数は一つしか定義できないため、複数目的がある場合の目的計画法的な最適化モデルの記述ができない。

従来方式では理論的には実現可能だが現実的でない世界

何千規模の0-1意思決定変数を含む現実社会の大規模モデルを解こうとすると長時間走行または長時間走行しても解が発見できないことに遭遇する。
費用対効果を度外視したシステムリソース（CPU，メモリなど）を潤沢に投入でも、ソリューションとしての保証ができないことが発生する。

従来方式の運用面（継続的な高い安全性と信頼性）の課題

従来方式では、モデルとして連立方程式を構築する必要があり、モデルの検証に時間がかかるだけでなく、環境変化に対応したタイムリーなモデルチューニングが困難である。

何故、今、LocalSolverか？（経済的背景）

拡大する適用シーン

数理計画法（最適化技術）は、今まで生産現場での最適な生産計画策定や物流分野などで活用され、多くの成果をもたらしている。しかし、近年のクラウドの進展にともなう大量データ（ビックデータ）の存在や、グローバル化に伴う製造・物流拠点の増加などにより、最適化を検討する際の制約条件や与件が複雑化しており、**従来の最適化手法やシステムでは、与件の多さにより計算爆発を起こし最適解が得られず、多くの企業では事業機会を逃している**。

- 製造（生産計画、裁断計画、ブレンディング）
- 物流、流通（拠点配置計画、配送計画、倉庫管理）
- 流通・サービス(要員シフトの最適化)
- 電力（スマートグリッド、ユニットコミットメント（発電機起動停止計画））
- 運輸（乗務員スケジューリング、荷役計画、バースアロケーション、機材メンテナンス計画）
- 金融（ポートフォリオマネージメント）他多数

何故、今、LocalSolverか？（経済と技術的背景）

新たな適用分野（現場とそこに携わる人間を支援する最適化）

ビックデータ活用を中心としたIT技術の向上がもたらす社会、企業の構造変化的に対応する仕組みが必要となりつつある。

経営は本社機構が中心の統制型管理に現場主体の緻密な制御の組み合わせが（**月次計画と日次制御の融合**）進みつつあり、この変化に対して現場を支援する仕組みが必要である。

溢れる情報から現場業務能力を低下させることなく、本来の接客サービスなどの業務に専念できる時間を確保できる環境整備とも言える。

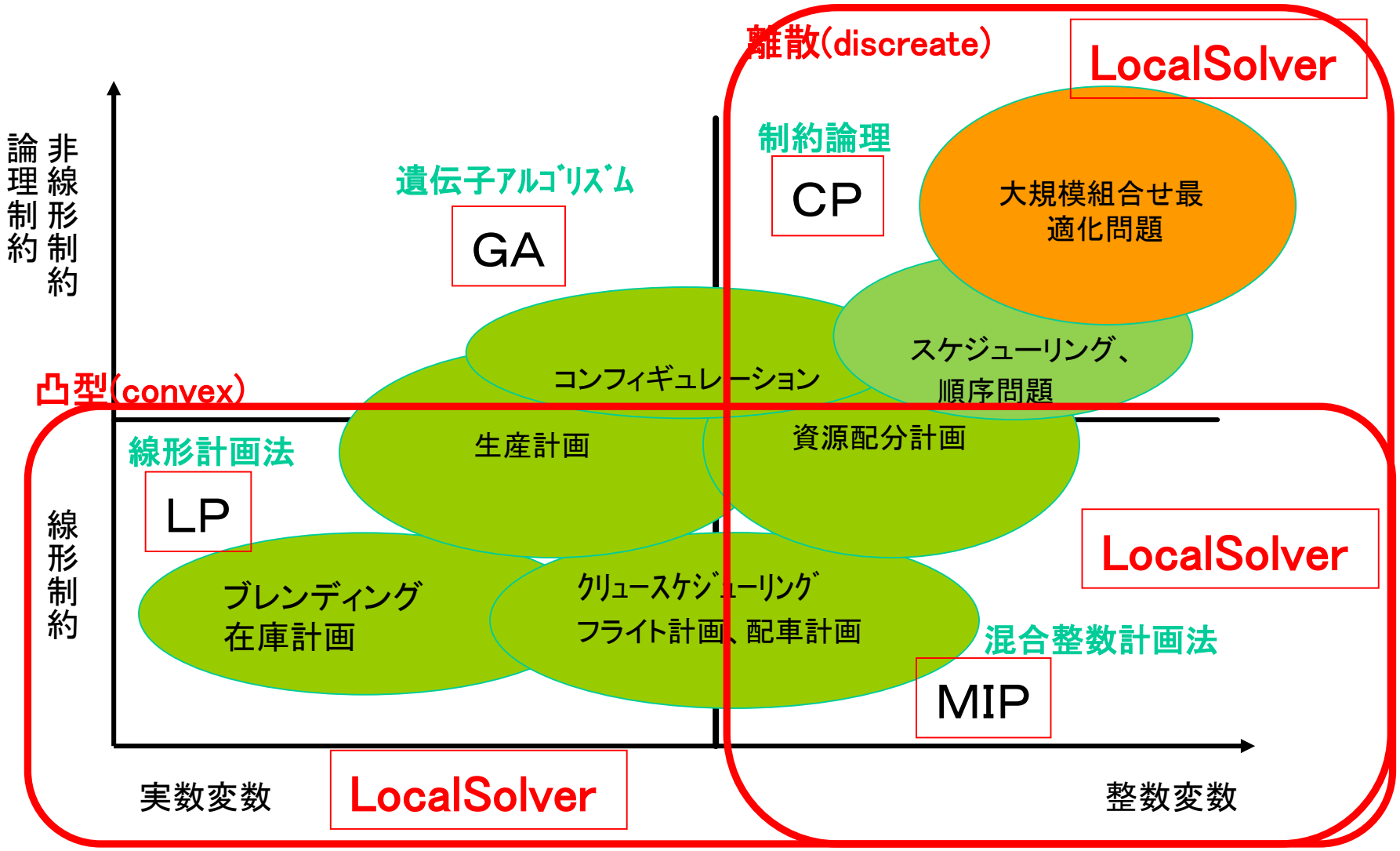
- 現場に新鮮で有益な情報を提供するためには**複雑で瞬時の判断**が求められるが、それを現場で使いこなすため(人間がボトルネックにならない)には、現場のさまざまな**意思決定（最適化）の支援**が最も重要となる。
- ビックデータ活用技術により最適化モデルの変数に投入するパラメタデータの入手がタイムリーに可能となり、**ビックデータ活用をもとにした意思決定**が必要である。
- センシング＆アラータ技術が発達し、**意思決定の時間短縮**が必要（警報音が響く職場環境で右往左往）。

また、**属人化の危機**（熟練者が退職などで伝承が困難）に対応するため、**熟練者の知識を最適化モデルとして管理・運用する仕組み**が重要である。

離散凸解析からから見たLocalSolver

問題の条件	解法	特徴	問題規模	システム
離散（ディスクリート）&凸型（コンベックス）	B＆B法＋シンプレックス法	子問題を生成して解く	整数変数の数に制約あり	汎用ソフト MIP CP
離散（ディスクリート）	メタヒューリスティックス（局所探索）	0－1変数のタブサーチ（専用アルゴリズムが多い）	**100万以上の0－1整数変数の処理が可能**	個別ソフト **汎用ソフト LocalSolver**
凸型（コンベックス）	線形計画法（シンプレックス法）	線形な連立方程式を解く	100万式、変数でも計算可能	汎用ソフト LP
凸型（コンベックス）	専用解法（線形/非線形）	専用アルゴリズム	プログラミングで決まる	NLP TLP 等

―LocalSolverの適用範囲―



LocalSolverで拡がるソリューション分野

従来、大規模問題では、探索空間が離散的であるもしくは離散的なもので表現できる問題は実用的に解けなかったが解くことができるようになった。

集合，順序，割当て，グラフ，論理，整数など離散的な構造を持つ。
多くの場合、**混合整数計画問題として定式化**できる.

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。

- **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 (乗務員・看護師の勤務表，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)

MIPLIBでのベンチマーク結果

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
dc11	open	37,297	11,100,000	21,300,000	1,840,402	unknown
ds-big	open	6,020	9,844	62,520	5,256	unknown
ex1010-pi	open	25,200	249	251	247	unknown
ivu06-big	open	1,812,044	* 479	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
siena1	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

□最少化問題
□実行CPU時間：5分、
□PC： Intel Core i7-820QM (4 cores, 1.73 GHz, 6 GB RAM, 8 MB cache)

4. LocalSolver4.0 概要

LocalSolverの概要

- **Bouyguesは、フランスで最も大きい**
- **会社の1つ--収入は330億€/年**

- **Innovation 24は、ビジネス分析及び**
- **最適化を実現するBouyguesの子会社**

LocalSolverは、Innovation 24が開発、

- **提供を始めた統合数理計画法システム**

LocalSolver4.0の新機能

- バイナリ(bool)意思決定変数及び**連続(float(下限、上限))** 意思決定変数
- 実行可能解探索の改良

- 改良された前処理の改良と推論による下限値計算を追加
- 局所的かつ複雑な領域内での新イタレーションの追加（LP,MIP用）
- MIPのテクニックを活用した大規模近傍探索

適用問題: supply chain optimization,
unit commitment,
portfolio optimization,
numerical optimization arising in engineering (ex: mechanics)

http://msi-jp.com/localsolver/

LocalSolver開発経緯

- ブイグ社の最適化部門による2000年からの実績
（汎用化を試行2000-2005年）
- フランスの**ORチーム（准教授2人及び実践家4人）**による
2007年から研究開発プロジェクトで成果（国の支援）
- MIP（混合整数計画法）では解けない**大規模組合せ 最適化問題（800万以上の0-1整数変数）**を実用的に解く
- メタヒューリスティク解法（局所探索解法）をベースにシンプレックス法をも取り込んだ**全く新しい汎用解法**を実現
- 非線形制約、非線形目的関数まで拡張した問題を解く:
Mixed-variable non-convex programming
- 最新のIT技術、数理最適化技術を活用
（関数型プログラミング言語、問題解析機能、並列処理等）

LocalSolver研究経緯

局所探索による汎用化を試行

逐次改善法を実現

- 現在のソリューションの近傍を探索
- 小さい探索空間から徐々に必要な探索空間を拡大
  → 解空間全体から効率的に探索

局所探索は大規模最適化では、標準的な考え方

- 多くの教科書でも採用されているアルゴリズム
- 多くの一般的なメタヒューリスティックアプローチの利点を吸収
- 最悪の場合でも効率的でないことを結果として出力
- 現実的な時間で実践的な最適な解を提供

LocalSolver研究成果

2000-2005: 研究開発

- Prologiaに対しOR工学を適用: 従業員スケジューリング
- ROADEF2005に挑戦: ルノーの自動車組み立てラインスケジューリング(B.EstellonとK.Nouiouaが最初の賞を受賞)

貢献（成果）

- 大規模組合せ最適化問題に対する方法論を提案
- 大規模混合整数問題の最適化に対する方法論を提案
- 局所探索をベースとして大規模組合せ最適化解法を提案
- 大規模かつ混合変数問題（非凸最適化問題）に対するソルバーを提供

LocalSolver研究成果（理論及び実践でのノウハウの蓄積）

何故局所探索（近傍探索）か

列挙して最適化するのが絶望的なとき

- 大規模な順列組み合わせ問題
- 緩和か推論が何ももたらさない時(例えば、線形緩和は非常に断片的)
- 緩和か推論を計算するのにコストがかかるとき

クライアントニーズに適合

- 良質の解（目的計画法：目的順に求解、準最適解であっても着実に）
- 速いこと:イタレーション時間が問題サイズに線形で比例すること

  → 短い実行時間+大規模モデルが解けること

LocalSolver研究成果（アルゴリズムの特長）

適切な探索領域の設定

- 探索空間の確立を高め、拡大していく
- 制約条件からではなく、目的関数に基づいて探索
- 現実の問題は良い探索空間を持つモデルになりやすい

局所探索:「基本」に帰って

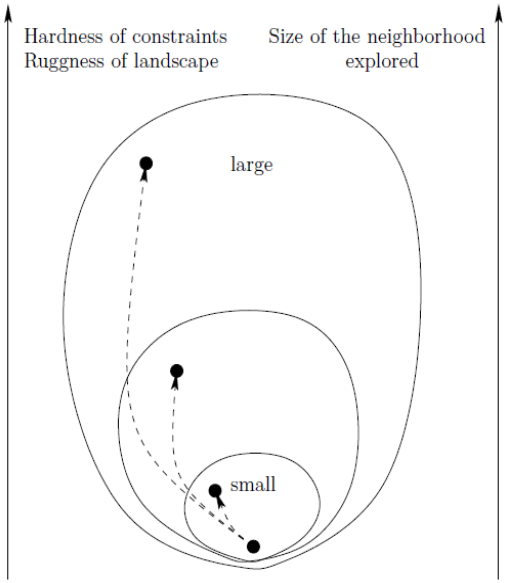
- 「メタ」に焦点を合わせない
- MOVE（イタレーション）の速さと効率性に集中
- クライアントからのフィードバックとテストが重要

→理論の専門技術とコンピュータ活用技術で高性能な近傍探索を実現

LocalSolver研究成果
（従来手法をハイブリッドに活用）

グローバル探索としての近傍探索

- 小さい近傍から探索を広げていく
- 近傍探索からダイナミックに探索領域を調整：縮小、拡大、特化
- 大規模領域探索にツリー探索(MIP、CP)を導入
- 完全な近傍と正確な探索では最適解に到達



汎用モデリング言語：LSP（概要）

LSP言語は、大規模組合せ最適化問題をモデル化し、モデルの検証及び解の検証を行うフェーズでの試行錯誤を行うのに最適な環境を提供することを目的として開発されている。

LSP言語は、最新の**関数型プログラミング言語**である。
関数型プログラミング言語の特長は、**型推論**を備えた言語であるため、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を推定するため、データの種類（型）をプログラマが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が、軽量言語では実現できないコンパイラによるプログラムのチェックが可能になる点にある。

汎用モデリング言語：LSP（利点）

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。従来比、1/10から1/3の開発量）
- バグを抑えやすい（コンパイラが型の間違い等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(マルチスレッド）

- 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及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（エディタ、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順番に複数設定することができる
  （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）

生産性の高いモデルの記述（LSP言語）

インタプリタ型言語が搭載されており、トライ＆エラー方式で高い生産性でモデルの記述と検証が可能である。モデルは、テキストエディタを使用してLSPファイル（xxx.lsp)として作成する。
モデルは、以下の五つの関数セクションからなる。

セクション	記述	機能
入力制御	function input() { }	外部ファイルからデータを取り込み。それをもとにモデルに投入する変数の値を制御
モデル定義	function model() { }	最適化モデルを定義
出力制御	function output() { }	最適化の結果を業務システムなどとの連携を目的として外部ファイルに出力
表示制御	function display() { }	最適化走行中に標準出力に変数の値などを表示
パラメタ制御	function param() { }	local-search の実行前にモデルをパラメタ化

```
FUNCTION MODEL(){
// O-1 decisions
   X_1 <-BOOL(); X_2 <-BOOL(); X_3<-BOOL(); X_4<-
BOOL(); X_5 <-BOOL(); X_6 <-BOOL(); X_7<-BOOL(); X_8
<-BOOL();
//weight constraint
  KnapsackValues <-10*x_1+ 60*x_2+ 30*x_3+ 40*x_4+
30*x_5+ 20*x_6+ 20*x_7+ 2*x_8;
  Constraint kNAPSACKwEIGHT<=102;
//maximize  Value
  KnapsackValues <-1*x_1+ 10*x_2+ 15*x_3+ 40*x_4+
60*x_5+ 90*x_6+ 100*x_7+ 15*x_8;
  Maximize KaapsackValue;
}
```

```
function output() {
  println("Selected Products:");
  for [i in 0..nbProducts-1 : getValue(x[i]) == 1]
    println("#"+i+" ("+value[i]+")");
}
```

LSPによるモデリング

- 意志決定変数の定義
  - ・**０－１整数変数の型を持つ意志決定変数**を定義
  - ・この意志決定変数の組合せが一つの解となる
  - ・探索は意志決定変数の組合せで行われる

- 制約式、目的関数の定義
  - ・**意志決定変数を使って制約式、目的関数を表現**する。
  - ・式の表現に、算術演算子、論理演算子等が使用でき、**非線形表現**で記述可能である。

- 外部入出力及び、表示
  - ・数値データの外部入力、結果の外部出力機能により、**データと問題定義を完全に分離**できる。このため、どんなに大きなモデルでも、モデル本体を簡潔に表現可能である。

モデル定義（LSP言語）

予約語	機能	摘要
bool()	意思決定変数（0-1整数変数）	
float（下限、上限）	意思決定変数（実数）	
constraint	制約条件（実行可能性の判定に使用）	
Minimize	目的関数（最小）	定義順に最適化が適用
maximize	目的関数（最大）	定義順に最適化が適用
<-	内部変数の明示的宣言	モデルの見易さ
//	注釈	

モデルの記述例（ナップザック問題）

積める重量が102Kgの袋に、8個のアイテムを積める場合にアイテムの価値の合計価値を最大化させる組み合わせを求める。

番号	アイテム	価値	重量
1	X_0	1	10
2	X_1	10	60
3	X_2	15	30
4	X_3	40	40
5	X_4	60	30
6	X_5	90	20
7	X_6	100	20
8	X_7	15	2

```
function model() {
  // 0-1 decisions
  x_0 <- bool(); x_1 <- bool(); x_2 <- bool(); x_3 <- bool();
  x_4 <- bool(); x_5 <- bool(); x_6 <- bool(); x_7 <- bool();

  // 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;
}
```

バイナリ意志決定変数

内部変数（整数）

ユーザーはLSP言語でモデルを記述するだけ！！

出力制御

```
function output() {
  println("Selected Products:");
  for [i in 0..nbProducts-1 : getValue(x[i]) == 1]
    println("#"+i+" ("+value[i]+")");
}
```

モデルの実行

モデルは、DOSコマンドプロンプトを立ち上げ、LSPファイルの格納されているフォルダにカレントディレクトリを設定して実行する。

Cd C:¥localsolver_3_0¥example¥model¥
Localsolver model.lsp lsTimeLimit=1

model.lsp　　：　実行するLSPファイル
lsTimeLimit=1　：　実行時間を1秒に設定

モデルの実行結果（1）

モデルが実行され結果がログとして表示される。

```
    :
[1sec,38611 itr386111 ] : obj(280),mov=820062,inf=39.55,acc=43.5%,imp=14
386111 iterations,820062 moves performed in 1 seconds
Feasible solution: obj(280)
```

Obj(280)　　：目的関数の値（解）
mov=820062 ：探索回数
inf=39.55　　：探索回数のうち、実行不可能解の占める割合
acc=43.5%　：探索回数のうち、実行可能解の占める割合
imp=14　　：探索回数のうち、解が完全された回数

時間指定（この例では1秒を設定）などの設定により探索が打ち切られた場合、最適解の検出、実行不可能解の検出の状態を表示します。
Infeasible：実行不可能解
feasible　：実行可能解
Optimal　：最適解

価格の合計は、実行可能解として280円が検出される。

モデルの実行結果（2）

出力制御部の記述により、実行可能解などの状態や結果の内訳などを表示することができる。

```
Run output...
Selected Products:
#2 (15)
#4 (60)
#5 (90)
#6 (100)
#7 (15)
```

```
function output() {
  println("Selected Products:");
  for [i in 0..nbProducts-1 : getValue(x[i]) == 1]
    println("#"+i+" ("+value[i]+")");
}
```

価格の合計の実行可能解として検出した280円の内訳は、
X_2(30,15), X_4(30,60), X_5(20,90), X_7(2,15)の五個で
その重さは102Kgであることが分かりる。

出力制御部の記述により、状態や結果の内訳などをファイルに出力（ファイル連携）したり、アプリケーションプログラムに値を通知することもできる。

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:¥Documents and Settings¥miyazaki>cd C:¥localsolver_3_0¥examples¥toy
C:¥localsolver_3_0¥examples¥toy>localsolver toy.lsp lsTimeLimit=1
LocalSolver 3.0 (Win32, build 20121129)
Copyright (C) 2012 Bouygues SA, Aix-Marseille University, CNRS.
All rights reserved (see Terms and Conditions for details).
Run model...
Close model...
Run solver...
Model:
  expressions = 38, operands = 50
  decisions = 8, constraints = 1, objectives = 1
Param:
  time limit = 1 sec, no iteration limit
  seed = 0, nb threads = 2, annealing level = 1
Objectives:
  Obj 0: maximize, no bound
Phases:
  Phase 0: time limit = 1 sec, no iteration limit, optimized objective = 0
Phase 0:
[0 sec, 0 itr]: obj = (0), mov = 0, inf < 0.1%, acc < 0.1%, imp = 0
[1 sec, 386111 itr]: obj = (280), mov = 820062, inf = 39.5%, acc = 43.8%, imp =14
386111 iterations, 820062 moves performed in 1 seconds
Feasible solution: obj = (280)
C:¥localsolver_3_0¥examples¥toy>
```

事例（MultiObjective Knapsack問題）

```
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;
}
```

算術演算子：prod, min, max, and, or, if-then-else, …

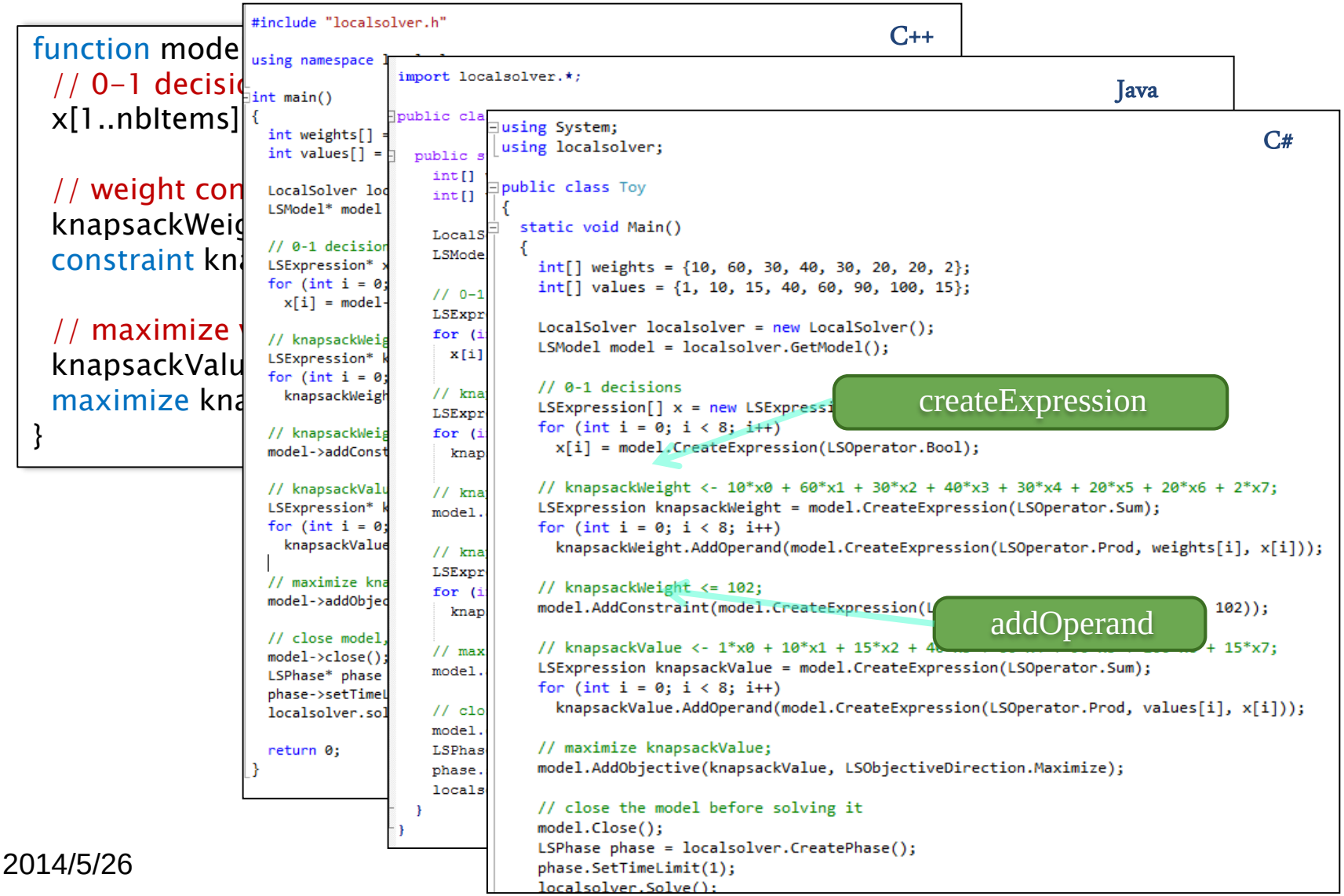
予約語

演算子の例

モデリングで利用可能なおもな演算子

算術演算子			論理演算子	関係演算子
sum	sub	prod	not	==
min	max	abs	and	!=
div	mod	sqrt	or	<=
log	exp	pow	xor	>=
cos	sin	tan	if	<
floor	ceil	round	array + at	>

Modeling APIs



開発言語比較

モデル名	lsp	C++	lsp比	Java	lsp比	C#	lsp比	備考
toy	17	49	2.9	46	2.7	48	2.8	ナップサック問題(トイモデル)
car_sequencing	77	211	2.7	211	2.7	246	3.2	車両投入順序問題
knasack	44	126	2.9	128	2.9	144	3.3	ナップサック問題
maxcut	44	138	3.1	137	3.1	149	3.4	裁断計画問題
multiobj_knapsack	93	172	1.8	182	2.0	191	2.1	ナップサック問題(非線形最適化)
pmedian	66	154	2.3	154	2.3	169	2.6	OR-LIB。2点間の輸送コスト最小
平均比較			**2.6**		**2.6**		**2.9**	
socialgolfer	69							CSPLIB, problem 010
steel_mill_slab_design	99							CSPLIB, problem 038
google_machine	231							Google問題(Googleがスポンサ)
table_layout	243							表配置問題
nurse_rostering	505							看護師スケジューリング

5. LocalSolver 事例

日本の事例（SCMモデル:超大規模モデル）

１．問題概要

　顧客要求に合わせて、予め決められた工場ー顧客間の配送便に間に合うように、いつどこ向けになにを生産するかを決める問題（複数工場の生産スケジュール）

２．モデル概要

　ー種類の0-1整数の意思決定変数（Bool変数）で、すべての制約を表現した。また、複数の目的関数を設定し、現実的な解法を実現した。

日本の事例（SCMモデルの実行結果比較）

実行結果（1） 既存システム

全国規模を一つのモデルにすると、実用時間内で解くことができなかった（実行CPU時間を5分程度にするのに、全国規模のモデルを **20分割**にする必要があった)。

実行結果（2） LocalSolver

LocalSolverでは、**5～6分で全国規模**を一つのモデルで解くことができた。

- Bool変数: **219万以上**
- 複数の目的関数を重要な順番で設定し、順番に最適化計算を実行。

事例（SCMモデルの実行結果2）

Phase 0:
[0 sec, 0 itr]: obj = (75593, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0), mov = 0, inf < 0.1%, acc < 0.1%, imp = 0

[1 sec, 27226 itr]: obj = (64967, 0, 6534, 0, 0, 0, 29280, 2838.45, 0, 519, 1059, 28.5621, 10132.2),
mov = 48265, inf = 55%, acc = 45%, imp = 17073

[2 sec, 65843 itr]: obj = (46593, 0, 23231, 0, 0, 0, 39820, 3763.47, 0, 1287, 1145, 30.6949, 27794.2),
mov = 124782, inf = 50.3%, acc = 49.6%, imp = 51706

[7 sec, 237135 itr]: obj = (11, 0, 51402, 0, 22, 0, 46130, 4122.05, 0, 2702, 1161, 31.1616, 73535.1),
mov = 477135, inf = 60.2%, acc = 38.9%, imp = 146374

[8 sec, 280000 itr]: obj = (11, 0, 51402, 0, 22, 0, 46130, 4122.05, 0, 2702, 1161, 31.1616, 73535.1),
mov = 560000, inf = 64.6%, acc = 34.4%, imp = 146374

[9 sec, 320000 itr]: obj = (10, 0, 52006, 0, 25, 0, 46150, 4125.65, 0, 2713, 1160, 31.1471, 73522.5),
mov = 640000, inf = 67.9%, acc = 31.1%, imp = 146375

[10 sec, 360000 itr]: obj = (10, 0, 52006, 0, 25, 0, 46150, 4125.65, 0, 2713, 1160, 31.1471, 73522.5),
mov = 720000, inf = 70.4%, acc = 28.5%, imp = 146376
360000 iterations, 720000 moves performed in 10 seconds
Feasible solution: obj = (10, 0, 52006, 0, 25, 0, 46150, 4125.65, 0, 2713, 1160, 31.1471, 73522.5)

日本の事例（人員配置（アルバイト配置計画））

1．意志決定変数の定義（Bool変数）

X(人員、日付、時刻、業務) ＝ {0, 1}

ここまでは装置のスケジューリングとあまり変わらないが・・・

2．人のスケジューリング特有の制約

・最大勤務時間

・最小勤務時間←**あまり短いとアルバイトは出てこない**

・１日１シフト←**勤務の途中に何もしない時間を入れない**

を導入。

日本の事例（人員配置計画の実行結果比較）

実行結果（1） 小モデル

人員：20名、1週間、1時間毎の16タイムスロット、3業務

(意志決定変数： 20 x 7 x 16 x 3 = 9520)

LocalSolver 3.1 ：最適解まで約20秒

既存MIP ：最適解まで約30秒

実行結果（2） 業務モデル

人員：83名、1週間、15分毎の54タイムスロット、5業務

(意志決定変数： 83 x 7 x 54 x 5 = 156870)

LocalSolver3.1：最適解まで約210秒

既存MIP ：600秒以上

（制約内容等は異なっているが・・・）

日本の事例(裁断計画)

5000mmの幅をもつフィルム等のロールから、色々な幅を持つ複数製品のロールを要求本数にできるだけ近くなるよう、裁断パターンとパターンの使用回数を求める問題である。

目的関数(複合目的関数)

1. パターン数最小(＊1000)
2. 要求数量とのギャップ最小(＊100)
3. ロス部分(4600mmからの差)。

Valid Roll width		3600 ～ 4600		
Products	width	Request	minimum	maximum
1	600	10	8	12
2	740	15	12	18
3	920	20	16	24
4	1060	5	4	6
5	1200	10	8	12

Pattern No		1	2	3	4	5	6	16	17	18	19	26	27	28	51	52	
Product	volume	count in	pattern(i)						9本				2本				
600		0	0	1	0	0	0	1	0	1	0	1	4	0	6	7	8本
740		0	1	0	0	0	0	1	2	1	3	1	0	1	1	0	18本
920		1	0	0	0	1	2	1	2	2	1	1	0	3	0	0	18本
1060		0	0	0	2	1	0	1	0	0	0	2	2	1	0	0	8本
1200		3	3	3	2	2	2	1	1	1	1	0	0	0	0	0	9本
Total		4520	4340	4200	4520	4380	4240	4520	4520	4380	4340	4380	4520	4560	4340	4200	
rest		80	260	400	80	220	360	80	80	220	260	220	80	40	260	400	

日本の事例(裁断計画実行結果比較)

実行結果(1)　通常モデル

製品種類数:10、総製品数量:153、切断パターン候補数:12898

(意志決定変数:　75302)

LocalSolver 3.1 :11秒で4パターン、要求差＝3

既存MIP　　:400秒すぎても9パターン、要求差＝10

実行結果(2)　大規模モデル

製品種類数:18、総製品数量:504、切断パターン候補数:70921

(意志決定変数:　399372)

LocalSolver 3.1 :31秒で8パターン、要求差＝17

既存MIP　　:4060秒で10パターン、要求差＝22

フランスの事例（Car sequencing in enault's plants）

車両投入計画（塗装及びアセンブル）

- 従来はラインに沿って部品を配置する計画
- 同一色塗装の連続制約
- 二次的目的として色の変更を最小限にしたい

大規模な最適化問題

- **1300台の投入順**を決める → **400 000** バイナリ意志決定変数が必要
- MIP or CP ソルバーでは数時間の実行でも実行可能解さえ見つからない
- LocalSolver は**数秒**で実用的な精度を持つ解を Renault に提供

Car Sequencing at Renault（比較結果）

色に関する自動車配列問題の結果を示す。LP緩和解（上限は3001（万））。

LocalSolver V3.0（式数：80850, オペランド数：295212、意志決定変数：44184, 制約条件：636, 目的関数：3）

- **1秒で目的関数：16122**の解を見つける
- 6秒で3478に達する
- **1分後に、3125**に達する（この例で最も有名なものは3109である）

Guroubi　V5.1（式数: 29845, 列数：57331, 非零要素数：442093、連続変数：15777, バイナリ変数：41554）

- LP緩和解を求めるのに、87561イタレーション, 312 秒かかる
- B&Bで最初の実行可能解をみつけるのに**647秒**かかる
- 1766秒後に解27720を見つける
- **60分後でも**コストはまだ25000を越えている

大規模組合せ最適化問題に関しては、Branch&Boundに基づく探索では、探索が本質的に制限されるため、解くのが難しいのが実体である。
たとえヒューリスティクティクサーチ等の発見的探索法を使用するとしても限界がある。
※かなり良い線形緩和の問題（従来のMIP用ベンチマークデータ）では、従来型でもある程度実用的な時間で解を見つけることは可能である。

LocalSolver研究成果
（更なるアーキテクチャの統合に向けて）

大規模混合変数、非凸最適化問題に対して、
既存の最適化技術(LS, LP/MIP, CP/SAT, NLP, ...)を統合した数理計画法システムを目指す。

Feasibility search

Optimization

Infeasibility proof

Lower bound

Preprocessing	**Neighborhood Search**	**Moves**		
Model rewriting Structure detection Constraint inference Variable elimination Domain reduction	Simulated annealing Restarts Randomization Learning	**Combinatorial**	**Continuous**	**Mixed**
		Small Compound Large	Small Compound Large	Small Compound Large
	Divide & Conquer	**Propagation**	**Relaxation**	
	Tree search Interval branching	Discrete propagation Interval propagation	Dual linear relaxation Dual convex relaxation	

参照文献

(1)T. Benoist, B. Estellon, F. Gardi, R. Megel, K. Nouioua (2011).
・「LocalSolver 1.x: a black-box local-search solver for 0-1 programming」、*4OR, A Quarterly Journal of Operations Research* 9(3), pp. 299-316. Springer.

(2)MSI株式会社
・「http://msi-jp.com/localsolver/」ホームページ

(3) Frédéric Gardi(2013)
・「Habilitation Thesis in Computer Science:Toward a mathematical programming solver based on local search」、Université Pierre et Marie Curie (Paris 6) - Faculté d'Ingénierie (UFR 919) École Doctorale Informatique, Télécommunications et Électronique de Paris (EDITE).

(4) Frédéric Gardi(2013)
・「Toward a mathematical programming solver based on local search」、13th December 2013 ORO Workshop, Nantes.

有難う御座いました。

LocalSolverについてのお問い合わせ

MSI株式会社（日本配給元）
〒261-7102千葉市美浜区中瀬2-6 WBGマリブウエスト2階
Tel:043-297-8841 Fax:043-297-8836
Eメール：localsolver@msi-jp.com