新しいソリューションを可能とする新数理計画法システム LocalSolver

01606110　MSI 株式会社　＊宮崎　知明　MIYAZAKI Tomoaki
MSI 株式会社　石村　猛　ISHIMURA　Takeshi

１．はじめに

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出来なかったのが現状である。

新数理計画法システム LocalSolver は、解探索に組合せ爆発が起こらないよう、局所探索と既存の解法を組み合わせたシステムであり、１０００万以上の０－１整数変数及び実数変数を扱うことができる。

また、Localsolver は大規模最適化問題を汎用的かつコンパクトに定式化できるモデル記述言語（LSP 言語）を備えている。本稿では、LocalSolver のモデル記述言語である LSP 言語によるモデリングの考え方を紹介するとともに、大規模最適化問題への適用事例を示す。

LocalSolver の適用範囲を図１に示す。

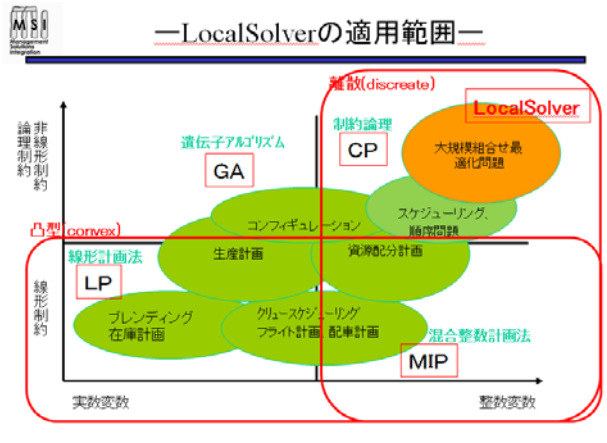


図 1. LocalSolver の適用範囲

現実世界の多くの問題は大規模最適化問題となる。

- 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（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）

２．LSP 言語による定式化

LocalSolverは、0-1意志決定変数(bool)及び上下限を持つ意思決定変数(float)からなるモデルを超高速に試行しながら最適化を目指すことができる。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る。

以下の手順でモデリングを行う。

１）意志決定変数(bool 変数及び float 変数)を定義する。

２）bool 変数及び float 変数を使い、制約条件、目的関数を定義する。この時、MIP 問題のように、線形制約に拘る必要はなく、制約条件、目的関数ともに、論理表現、非線形表現で記述できる。

※選ばれた意思決定変数の組み合わせで、制約、目的関数が必要とする数量（生産量等）が決まるよう、意思決定変数を定義する。

ナップサック問題を例に、LSP による定式化を示す。

品物が、８品あり、それぞれの重さと価値以下に定義する。

重さ：10、60、30、40、30、20、20、2　kg

価値：1、10、15、40、60、90、100、15　円

ナップサックには最大 102kg まで品物を入れることができ、価値が最大になる

よう、どの品物を選べばよいか、その時の価値はいくらになるかが問題である。

```
/********** toy.lsp **********/
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;
}
```

LSP 言語は、モデリング及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（型等）を推定するため、データの型等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。

３．事例

LocalSolver による事例をいくつか紹介する。

１）SCM 問題

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

実行結果

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）。

LocalSolver：

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、最適化**

２）要員配置問題

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。

実行結果（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：準最適解まで約２１０秒

既存 MIP　　：６００秒以上

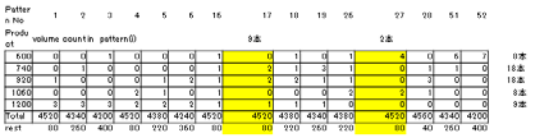
３）裁断計画

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



定式化として、裁断パターンと裁断パターンの使用回数を Bool 変数として定義する。定義した bool 変数で、目的関数、制約条件（製品数量）を定義する。

以下に MIP プログラムとの実行結果比較をしめす。

実行結果

製品種類数：１８、総製品数量：５０４
裁断パターン候補数：７０９２１
（意志決定変数：　３９９３７２）
LocalSolver 3.1：31秒で　8パターン、要求差=17
市販汎用ソフト　：4060秒で　10パターン、要求差＝22

４．おわりに

３０年前には殆ど実現出来なかった大規模組合せ最適化問題に対して、実践的なアプローチが実現できる時代になったと考える。「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, 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) 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.

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