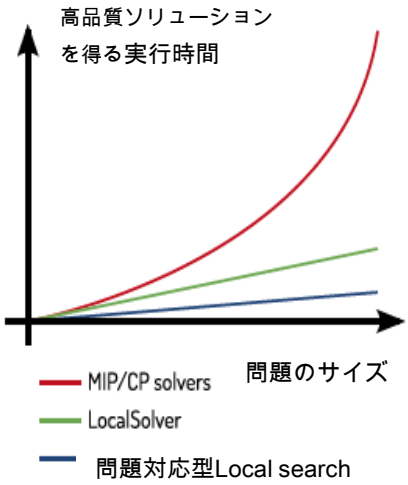
LocalSolver

mathematical programming by local search

史上初　ローカルサーチ法による
次世代-数理計画法システム



LocalSolverは、最適化計算分野における一流の専門家による５年間のR＆Dプロジェクトにより完成した、フランス発の次世代-数理計画法システムです。
LocalSolverはローカルサーチ技術をベースとして、使いやすいモデリング言語と実行環境を備えており、大規模組合せ最適化問題をユーザがモデルを純粋に定義するだけで、複雑なチューニングをしなくても、数秒から数分で高品質な解を得ることができます。

- 非線形0-1計画問題をも解くことが可能
- 目標計画法としても利用可能
- 1000万の意思決定変数まで実行可能
- 革新的なモデリング言語
- C++、Java、.NETで簡単に利用できるAPIs
- 問題に忠実なモデリングと実行環境

Localsolverはビジネスおよび産業で生じる大規模な実世界の組み合わせ最適化問題を解くのに特に最適です。従来のツリー探索を基にした手法では、このような数千万の変数を伴う問題を解くことは現実的には不可能でした（MIPまたはCP）。

短時間で高品質ソリューションを提供

Thanks to LocalSolver, our Siemens project team could record dramatic improvements on a very combinatorial problem, for which the best available mathematical programming solvers could hardly find a solution in several hours

Denis Montaut
Chairman & CEO of Eurodecision

車両投入計画（Renault）
ルノー社はLocalSolverを使用し従来では現実的に利用できなかった車両投入順序計画を数分で最適に近い解を見つけ出しています。

機械の配置転換（Google Challenge）
グーグル社の問題で、LocalSolverは5分間で100マシーンに10,000工程の割り当てを最適化する解をみつけることができました。LocalSolverは80チームの国際競争で、たった1日でモデリングから最適化実行までを実現できたとして認められた唯一のソルバーです。

- 非線形割当て：車両投入計画、頻度割当て
- パッキング＆カバーリング：メディアプランニング、機械スケジューリング、グラフ分割等
- 設備配置：ロジスティック　クラスタリング、通信網の最適化
- 人員スケジューリング、グループプランニング、看護師スケジューリング等

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
Web-site：www.msi-jp.com/localsolver/

Customers & partners

DECIDE　刃之砺 EdgeStone　EURODECISION OPERATIONAL RESEARCH　HEURISIS　mbio　MSI Management Solutions Integration

Innovation 24（フランス開発元）、MSI株式会社（日本配給元）

何故、LocalSolver？

汎用の数理計画法システムが使用している分枝限定法と異なり、LocalSolverはローカルサーチ法をベースとして使用します。

混合整数計画法または制約論理プログラミングは分枝限定法をベースにしています。分枝限定法では解空間の中で、解を決定する整数変数を繰り返し使用します。

分枝限定法では、解の探索が指数関数的に増えていくので、小規模または中規模の問題に限定されます。分枝限定法では整数変数を拡大させることができません。
たとえ実行時間に制限を設けなくても、実際の計算機環境の制限で終了してしまい、ヒューリスティックなアプローチよりもソリューションに対する保証ができません。

何千もの0-1意思決定変数を含む実世界の問題の多くでは、分枝限定法で数時間または数日の計算時間を費やしてもなお、探索領域を絞り込むことができず、実行可能解の発見に失敗するケースが多く発生します。

対照的にローカルサーチ法では、は目的関数を改善するように、"ムーブ"と呼ばれる計算コストをかけない局所的な変更によるイタレーションを行います。
このため、短時間の実行時間で高品質のソリューションを得ることが可能なため、ローカルサーチ法は専門家の間では高く評価されています。(数分または数秒)

一般的にはローカルサーチ法を現実世界の問題に適用するのは、簡単ではありません。アルゴリズムの知識とコンピュータプログラミング技術の双方を必要とするため、専門的なアルゴリズム部分で"ムーブ"をどう評価するかが非常に困難です。

LocalSolverはIローカルサーチ法をベースとした史上初の数理計画法システムです。

LocalSolverは自律的かつ革新的な"ムーブ"及び計算コストができるだけ小さくなるように高度に最適化された目的関数評価の機能をもちます。

LocalSolverでは、100万以上の意思決定変数をもついくつかのケースで、数分の計算時間で高品質なソリューションが求まります。

LocalSolverはまた、ハイパーグラフ理論を利用した推論テクニックで最適解を示唆したり、実行不可能性を示唆したりします。

製品概要

ビジネスニーズを実現するソルバー

LocalSolverは数秒で高品質ソリューションを提供します。LocalSolverは非凸および非平滑領域で数百万の変数を持つモデルを解くことができます。このような問題は、現在の数理計画法システムでは解くことができません。
LocalSIverは推論テクニックを使用し、最適性または実行不可能性を示唆します。

手法の理解が不要なソルバー

LocalSolverは純粋なモデリングと実行環境をもちます。 LocalSolverはシンプルな数理モデリング形式を基盤としています。さらに、解法に関して複雑なチューニングを行う必要がありません。

革新的モデリング言語

LocalSolverは強力モデリング言語があります。LocalSolverのモデリング言語を使えば、大規模組合せ最適化問題でも短時間でプロトタイピングすることができます。

LocalSolverのプログラミング言語(LSP)は効率的なプログラミングスタイルを提供します。ダイナミックかつ暗黙的な変数宣言、コンパクトなシンタックスのループ宣言等。
Localsolverの関数は、モデリングだけでなくC++、C#、JAVAでの開発を容易にすることができます。

LSP言語システム開発の目的はプロトタイプ作業をするときの負担を出来る限り小さくすることです。
出来あがったLSPモデルは既存のモデリング言語で記述したモデルよりも、理解し易く読みやすい内容となるでしょう。

目的指向言語での開発を用意にするAPIs

ユーザーのビジネスアプリケーション システムにLocalSolverを組み込むために、C++、java、.NET向けに使用し易いオブジェクト指向プログラミングインターフェースを提供しています。

LocalSolverが提供するAPIは数が少なく使いやすい形になっています。
LSPからLocalSolverが提供するAPIを使えば簡単に目的指向言語に移行できます。
ユーザーは自身の数理最適化モデルを自然な形で作成することに専念するだけで良いのです。
ユーザーはLSPで作成した問題を分解する必要も、ソルバーのチューニングの変更を行う必要もありません。数分間で有効な組合せ最適化問題を解くために、追加的なコードを記述する必要さえもありません。

信頼性、ロバスト性およびポータブルで提供

信頼性およびロバスト性なしでは効率性は意味を持ちません。
私たちは抜本的かつ継続的に製品を拡張していきます。
クライアントへ最高品質の製品を提供するだけでなく、最高のサービスを保証し、さらに開発者による迅速かつ丁寧なサポートを保証致します。

LocalSolverはフルにポータブルな環境での提供をいたします。
現在、3つのプラットホーム(Windows、Mac OS、Linux)および2つのアーキテクチャ(×86、×64)でご利用頂けます。

```
function model() {
  x[1..nbItems] <- bool();
  knapsackWeight <- sum[i in 1..nbItems](weights[i] * x[i]);
  constraint knapsackWeight <= knapsackBound;
  knapsackValue <- sum[i in 1..nbItems](values[i] * x[i]);
  maximize knapsackValue;

  // secondary objective: minimize product of min and max values
  knapsackMinValue <- min[i in 1..nbItems](x[i] ? values[i] : infinity);
  knapsackMaxValue <- max[i in 1..nbItems](x[i] ? values[i] : 0);
  knapsackProduct <- knapsackMinValue * knapsackMaxValue;
  minimize knapsackProduct;
}
```

LSPで記述した複数の目的(非線形条件)を持つナップサック問題の例