2015-08-05

BlzLP.dll, BlzGA.dll 取扱い説明書 (ver1.1)

目次：

概要：

BlzLP.dll（以下 LP） および BlzGA.dll (以下 GA)は、一次元資材割付問題用のソルバーです。

LP は線形計画法を利用したソルバーで、小～中規模の問題で最適解を得ることが出来ます。
パターン数の少ない解を求めるアルゴリズムを実装しており、高品質な解を得ることが出来ます。得られた解は最適解であることが保証され、パターン数の少ない解が得られることから、通常は LP を利用することを推奨いたします。

GA はヒューリスティックを利用したソルバーで、最適解を得る保証はありませんが LP では解けない大規模な問題を解くことが可能です。また、GA では与えられた材料の総長さがピースの延べ長さに満たない場合でも、与えられた材料に対する最大歩留りの解を探索することが出来ますので、大規模な問題の部分問題を解いて問題規模の縮小を図りたい場合にも利用することが出来ます。

LP と GA は問題の入出力に共通のインターフェースを持っており、一部の機能を除いて同様にアクセスすることが出来ますので、問題の規模と用途に応じて使い分けることが容易です。

本バージョンより LP,GA ともにアルゴリズムを見直し、高速化と解の高品質化を実現しております。

使い方：

構成ファイルは、BlzLP.dll, BlzGA.dll, blzlp.nes の 3 つで, DLL は任意のフォルダに配置できます。

BlzLP.dll の動作には lpsolve55.dll が必要です。lpsolve55x64.dll 及び lpsolve55x86.dll は BlzLP.dll と同じフォルダに配置してください。LP が初回実行時に利用環境を調査して、環境に応じた Dll を改名コピーし、lpsolve55.dll を作成します。

ライセンスファイル（blzlp.nes）は、任意のポートに挿入された USB メモリのルートフォルダに配置してください。ライセンスは LP,GA 共通です。

LPSolver, GASolver クラス：

名前空間：BlzLP, BlzGA

クラス名：LPSolver, GASolver

BlzLP.dll および BlzGA.dll は 、.NetFramework4.0 以降で動作するマネージ DLL です。C++(CLR)、C#、VB.Net、F# から利用することが出来ます。プロジェクトフォルダに DLL を置いてプロジェクトに追加・参照してください。DLL のプロパティーは、常時出力としてください。

LP は実行時に lpsolve55.dll （ネイティブ DLL）を参照します。lpsolve55.dll は、実行環境に応じて BlzLP.dll が、lpsolve55x86.dll と lpsolve55x64.dll よりリネームコピーしますので、両 DLL を BlzLP.dll と同じフォルダに配置してください。両 DLL をプロジェクトに追加し、プロパティーを常時出力としてください。

コンストラクタ：

LP: void LPSolver()

GA: void GASolver()

ex. BlzLP.LPSolver lpSolver = new BlzLP.LPSolver();

ex. BlzGA.GASolver gaSolver = new BlzGA.GASolver();

関　数：

ピース設定：

共通: bool SetPieces(int cutLoss, string[] p_marks, int[] p_length, int[] p_quantity)

戻り値：定義に成功すると True を返します。
　　入力が不正あるいはライセンスが見つからない場合は False が返ります。

引　数：切代、ピース摘要文字列配列、ピース長さ整数配列、ピース数量整数配列をとります。

int　　cutLoss;　　～ 切代を設定します。

string[] p_marks;　　～ 符号を設定した配列を与えます。

int[]　　p_length;　　～ 長さを設定した配列を与えます。

int[]　p_quantity;　　～ 数量を設定した配列を与えます。

bool　isSuccess = Solver.SetPieces(cutloss, p_mark, p_length, p_quantity);

材料設定：

LP:　int　SetMaterials(int[] m_length, int[] m_quantity) ;

GA：bool SetMaterials(int[] m_length, int[] m_quantity);

戻り値：LP～　問題の規模を示す整数を返します(材料設定に失敗すると 0 を返します)。

GA～　材料設定の成否を示す bool 値を返します。

引　数：利用可能な材料の長さ、数量を整数の配列で渡します。

端材など利用可能な数量に制約のある長さの材料には数量を、定尺材など利用可能な数量に制約のない材料の数量は「0」を設定します。

例：長さ 5500 の定尺に制約が無く、長さ 1200 の端材が 3 本利用できる場合を示します。

```
int[]    m_length      = {1200,5500};
int[]    m_quantity    = { 3, 0 };
LP:  int     size = lpSolver.SetMaterials( m_length, m_quantity );
GA:  bool    success = gaSolver. SetMaterials( m_length, m_quantity );
```

割付実行：

LP:　int　Solve(TimeSpan limit, bool once, out answer , 0　)

GA:　void Solve(TimeSpan limit, bool once)

戻り値：　LP～　得られた解のパターン数を返します。

GA～　なし

引　数：　制限時間を TimeSpan で渡します。

共通：　once = true とすると、制限時間（limit）内であっても最適解を得た時点で終了します。

once = false とすると、制限時間内は解析を継続し、より高品質な解を探索します。

LP:　int　patternCount = lpSolver.Solve(limit, true,out answer , 0);

短時間の解析でも解が得られたならば最適解となります。最低 1 回は解析を行うため、解を得るのに limit 以上の時間を要する場合があります。制限時間内に最適解が得られなければ、0 を返します。

長時間の解析を行うと、得られる解のパターン数が減少します。

※～ver1.1 より、パターン数の削減を行う処理が追加され、初回の解析は最適本数を得れば良く、反復探索の必要はなくなりました。したがって、once = true に固定出来ます。

GA:　gaSolver.Solve(limit, true);

長時間の解析を行うと、最適解に近づき、延長さ最適解を得た後は解のパターン数が減少します。

制限時間になると探索を終了します。

パターン数の圧縮：（LP のみ）

```
int  Reduce(int TargetPatternCount, TimeSpan reduce_limit)
```

戻り値：　　得られた解のパターン数を返します。

引　数：　　パターン数の目標（制限）値を int で渡します。

制限時間を TimeSpan で渡します。

注　　：　　目標値は、初期解で得られたパターン数から１ずつ減じてください。

解の取得：

共通：　　AnswerClass　GetAnswer()

戻り値：　　得られた解（AnswerClass）を返します。

例：　AnswerClass　answer = solver.GetAnswer();

GA の場合、制限時間内に最適解が得られなくても、終了時点で最良の解を返します。

解の文字列化：

共通：　　string AnswerToString(AnswerClass answer, string DELIMITER) ;

戻り値：　　得られた解を文字列で返します。

引　数：　　文字列化する解(AnswewClass)と、解に使用する区切り文字を渡します。

例：

```
string DELIMITER = “¥t”;
string　strAnswer = solver.AnswerToString( answer, DELIMITER );
```

Public 変数

規模取得状態の取得（LP のみ）：

LP:　float progress = lpSolver.Progress;

戻り値：　　問題の規模の計算終了割合を 0～1 までの数値で返します。

（※：lpSolver.SetMaterials() を非同期で実行させ、別スレッドから読み出します）

延長さ最適解の正否（GA のみ）：

GA:　bool isOptimal = gaSolver.isOptimal;

戻り値：　　延長さ最適解を得た場合は true そうでなければ false を返す。

AnswerClass クラス：

名前空間：BlzLP,BlzGA

変　数：

public AnswerToatalClass	Total	使用する材料・本数の情報
public List<ArrangementClass>	ArrangeList	組合せパターン情報
public ExtraArrangementClass	Extra	余分(不足)ピース情報

ArrangementClass クラス：

名前空間：BlzLP、BlzGA

変　数：

public int	MaterialLength	材料の長さ
public int	RequiredCount	同じ組み合わせの切断本数
public int	Margin	残寸法
public List<int>	PieceLengthList	パターンが含むピース長さ
public List<int>	PieceCountList	パターンが含むピース本数
public List<string>	PieceMarkList	パターンが含むピース符号

AnswerTotalClass クラス：

名前空間：BlzLP、BlzGA

変　数：

public int[]	MaterialLengthArray	使用する材料の長さ
public int[]	MaterialCountArray	使用する材料の本数

関　数：

public int	TotalCount()	使用する材料の総本数を返す

ExtraArrangementClass クラス：

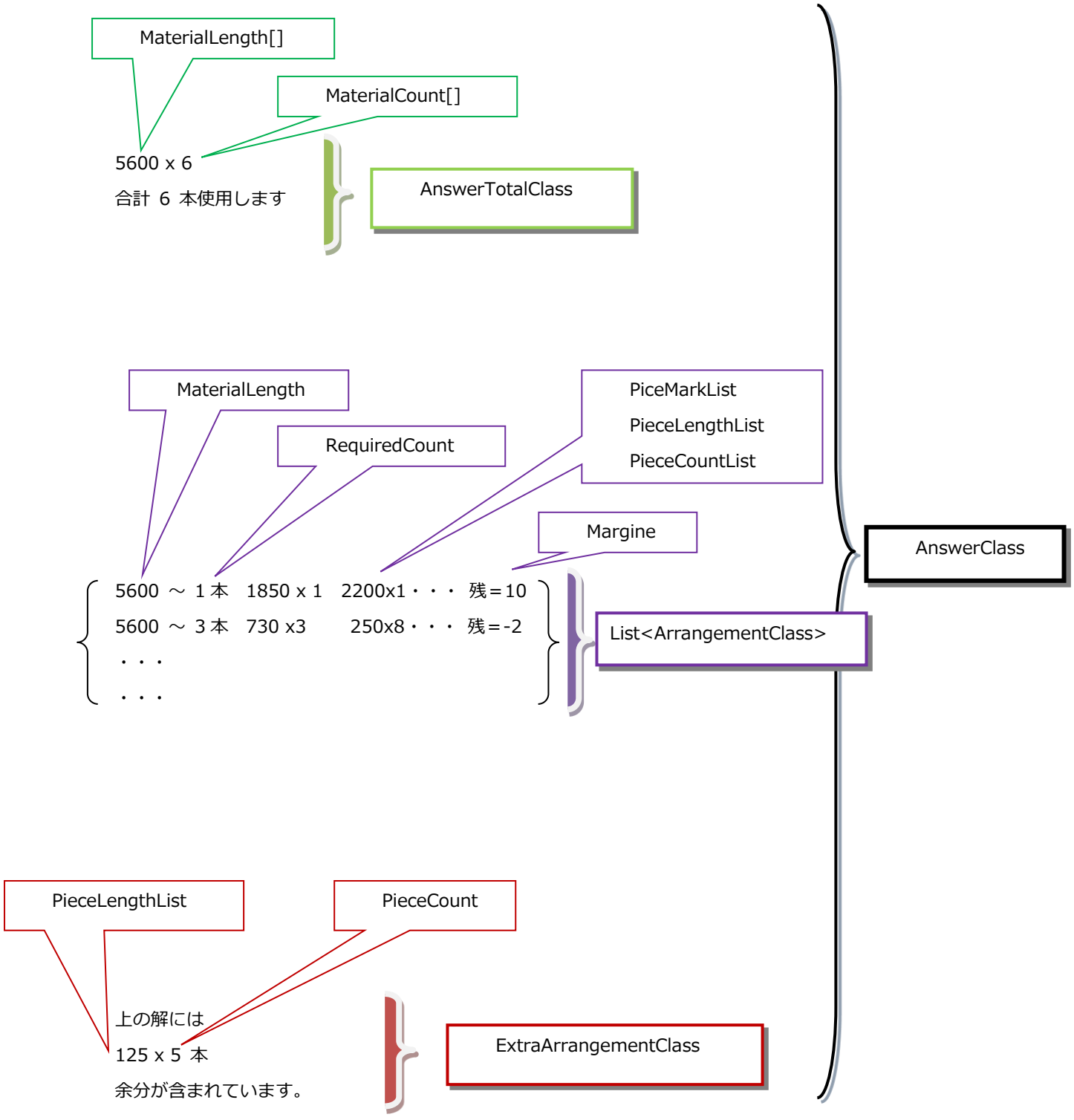
名前空間：BlzLP、BlzGA

変　数：

public List<int>	PieceLengthList	余分に含まれるピース長さ(※)
public List<int>	PieceCountList	余分に含まれるピース本数(※)
public List<string>	PieceMarkList	余分に含まれるピース符号(※)

※：BlzGA においては割付不能となったものを意味します。

Answer クラス図：


MaterialLength[]
MaterialCount[]
5600 x 6
合計 6 本使用します
AnswerTotalClass
MaterialLength
RequiredCount
PiceMarkList
PieceLengthList
PieceCountList
Margine
AnswerClass
5600 ～ 1 本 1850 x 1 2200x1・・・ 残＝10
5600 ～ 3 本 730 x3 250x8・・・ 残＝-2
・・・
・・・
List<ArrangementClass>
PieceLengthList
PieceCount
上の解には
125 x 5 本
余分が含まれています。
ExtraArrangementClass

ver 1.1　の主要な変更点について：

LP の変更点：

使用パターン数圧縮メソッドの追加：

ver1.1 より、パターン数の小さい解を探索する Reduce メソッドが追加されました。

実務における LP、GA の利用法について：

材料切断実務の工程：

鋼材・木材・その他 長尺材から製品を切り出す作業の工程は下記のようになります。

見積時 ⇒ 1. ピースのリストアップ → 2. 材料必要数量の算出
受注後 ⇒ 3. 切断指示書作成 → 4. 材料手配

上記工程のうち、2. と 3. の工程で資材切り出し問題を解く必要があります。

見積時の計算：

見積時の計算の目的は、延長さの計算で得られる材料の必要数量に、組合せを考慮した材料の必要数量が収まるかを確認すること（＝**損失発生の防止**）ですので、端材を考慮せず単一定尺材への割付を計算します。

例えば、長さ 3000 の製品 11 本を、長さ 5500 の定尺材から切り出す場合、延べ長さから材料必要本数を計算すると 3000 x 11 ÷ 5500 = 6 本 となりますが、長さ 3000 の製品は 5500 の定尺材から 1 本しか切り出せないため、実際には 11 本 の定尺材が必要となります（実際の問題では同様の危険がより複雑で判り難く隠ぺいされています）。

在庫の端材寸法は切断作業の度に変わるので、見積時の計算では端材を考慮するべきではありません。見積もりの提出から受注までにはタイムラグがあり、その間にも切断作業は行われますので、見積時の計算で端材を考慮しても、その端材が受注後に残っているとは限らないからです。端材を考慮した切断作業のための割付計算は受注後に行うので、見積時の計算で得られた解は切断指示書として利用されるわけではなく、パターン数など解の品質を考慮する必要もありません。これらの理由から、最適解を得た時点で解析を終了するよう Solve の引数 once = true として単一定尺材への割付を計算すれば、必要な材料の数量を不足なく計算するという所期の目的（**損失発生の防止**）を最短時間で達成することが出来ます。

受注後の計算：

受注後の計算の目的は、材料数量と段取りの最小化（＝**利益の最大化**）ですので、端材を含めて計算し材料数量が最小で、パターン数が少なく段取り替え工数の少ない切断指示書を作成します。

LP を利用して、許容される時間内で継続して計算を行いパターン数の少ない高品質な最適解を求めます。Solve の引数 limit に計算時間、once=false として時間内で可能な限り高品質な解を求める事が出来ます。ただし、端材など材料の種類を増やすと問題規模が急激に増大するため、LP の能力に対して問題規模が過大となる場合があります。その場合、利用可能な端材に対する割り付けをあらかじめ GA で求めて問題規模を縮小し、これを LP で解くことにより、パターン数の少ない高品質な解を得ることが出来ます。GA では 述べ長さに対して材料長さが小さい場合でも歩留りの良い割付の探索が可能です。

Simple Sample：

Set Parameters:

```
int[] p_length = { 1380, 1520, 1560, 1710, 1820, 1880}; // ピース長さ
int[] p_quantity = { 22, 25, 12, 14, 18, 18 };          // ピース要数
string[] p_mark = { "A", "B", "C", "D", "E", "F" };     // ピース摘要
int[] m_length = { 4500, 5500 };                        // 材料長さ
int[] m_quantity = { 1, 0 };                            // 材料数量制約
int cutloss = 0;                                        // 切代
TimeSpan limit = TimeSpan.FromSeconds(10);              // 制限時間
bool isOnce = false;                                    // 探索条件（反復/停止）
```

GA :

```
BlzGA.GASolver gaSolver = new BlzGA.GASolver ();

gaSolver.SetPieces(cutloss, p_mark, p_length, p_quantity);
gaSolver.SetMaterials(m_length, m_quantity);
gaSolver.Solve( limit, isOnce );
BlzGA.Answer gaAnswer = gaSolver.GetAnswer();
```

LP :

```
BlzLP.LPSolver lpSolver = new BlzGA.LPSolver ();
string version = LPSolver.ver;
lpSolver.SetPieces(cutloss, p_mark, p_length, p_quantity);
int size = lpSolver.SetMaterials(m_length, m_quantity);
BlzLP.AnswerClass answer;
int patternCount = lpSolver.Solve( limit, isOnce,out answer,0 );
BlzLP.Answer lpAnswer = lpSolver.GetAnswer();
int TrgCount= patternCount - 1;
TimeSpan reduce_limit = TimeSpan.FromSeconds(1);
int reduceCount = lpsolver.Reduce(TrgCount, reduce_limit);       // パターン数圧縮
answer = lpsolver.GetAnswer();
```

Answer:

```
for ( int  i = 0; i < answer.Total.MaterialLengthArray.Length; i ++)
{
        Console.WriteLine( answer.Total.MaterialLengthArray[i].ToString + " x " +
                                answer.Total.MaterialCountArray[i].ToString + "本必要です" );
}
```