Sample(ver1.1) コード解説

共通変数定義：

問題定義配列変数：

```
int cutloss;                    // 切代
int[] p_length;                 // ピース長さの配列
int[] p_quantity;               // ピース要数の配列
string[] p_mark;                // ピース摘要の配列
int[] m_length;                 // 材料長さの配列
int[] m_quantity;               // 材料数量の配列
```

p_length, p_quantity, p_mark 配列変数は同じ長さの配列としてください。
摘要保存用配列変数 p_mark は、摘要不要の場合でも同じ長さの配列を生成してください。
同様に、m_length, m_quantity 配列も同じ長さの配列としてください。使用数量に制約の無い材料には「０」を代入してください。

ex：`int len = p_length.Length;`
`p_mark = new string[len];` null 文字が割り当てられた 長さ= len の配列を生成。

`m_length = { 5500, 6000 }` 長さ5500 と 6000 の材料を使用する場合
`m_quantity = { 0 , 1 }` 長さ 5500 の材料に制約がなく、6000 の材料が１本使用できる。

解析条件設定変数：

```
bool Once;                      // true = 最適解を得たら終了 :
                                        false = 制限時間内は探索継続
TimeSpan limit;                 // 制限時間
```

LP ： once = true とすると最適解を得た時点で解析を終了します。
limit = 1 など制限時間に小さな値を設定しても最低１回の解析を行うため規模の大きな問題の場合、制限時間内で終了する保証はありません。
GA ： once = true とすると延長さから算出される最小本数への割付が完了した時点で終了します。
GA は制限時間で探索を終了し、その時点で得られた最良解を保持します。

問題定義変数への数値代入：

```
ReadFileConsole(out cutloss, out p_length, out p_quantity,
                out p_mark, out m_length, out m_quantity);
```

Sample コードでは、種々の問題を試行しやすいようテキストファイルに定義された問題を、問題定義変数に読み込む例を記述しています。

ソルバー生成：

```
LPSolver lpsolver = new LPSolver();    LP のインスタンス化
GASolver gasolver = new GASolver();    GA のインスタンス化
```

問題定義：

```
lpsolver.SetPieces(cutloss, p_mark, p_length, p_quantity);
gasolver.SetPieces(cutloss, p_mark, p_length, p_quantity);
```

LP,GA インスタンスにピース変数を設定します。

```
int size = lpsolver.SetMaterials(m_length, m_quantity);
```

LP のインスタンスに材料の変数を設定すると問題規模を返します。規模が大きくなると処理時間を要します。また、規模が過大な場合は「０」を返します。

```
gasolver.SetPieces(cutloss, p_mark, p_length, p_quantity);
```

GA のインスタンスに材料変数を設定しま（GA では数量制約の無い材料は延べ長さの+6%となる本数から順次本数を減少させて探索します。組合せ最適解が+6%を超える場合割付不能解が発生します）。

解析条件設定：

```
once = false;
limit = TimeSpan.ParseFromSecond("1");
```

上のコードでは、1 秒間継続して解析を行い、得られた解のうち最良のものを保持します。
パターン数の少ない解が要求される場合、問題規模に応じて制限時間を長めに設定します。
解の品質が要求されない場合は once = true とすると、最短時間で解析を終了できます。LP の場合、必ず最適解を返すので見積時の計算において有効です。

解析・探索実行：

```
BlzLP.AnswerClass answer;
int pattern_count = lpsolver.Solve(limit, Once,out answer,0);
```

LP のインスタンスが与えられた解析条件で解析を行い、得られた解のパターン数を返します。

```
gasolver.Solve(limit, Once);
```

GA のインスタンスが与えられた解析条件で探索を行います。

パターン数圧縮：(LP のみ)

```
int TrgPatternCount = 10;
TimeSpan limit = TimeSpan.FromSeconds( 1 );
int ReducedCount = lpsolver.Reduce( TrgPatternCount , limit );
```

与えられた目標パターン数以下の解を探索し、得られた解のパターン数を返します。
limit には単位探索時間を指定します。（問題サイズにより、総探索時間は変わります）
この処理は長時間を要します。サンプルコードでは、BlzLP による探索を別スレッドで行い、
タイマーイベントで LPSolver.Progress を読み込みプログレスバーに反映しています）

解の取得：

```
BlzLP.AnswerClass lp_answer = lpsolver.GetAnswer();
```

LP のインスタンスから解を取得します。

```
BlzGA.AnswerClass ga_answer = gasolver.GetAnswer();
int pattern_count = ga_answer.ArrangeList.Count;
```

GA のインスタンスから解を取得し、取得した解からパターン数を取得します。

LP,GA ともに解は AnswerClass で戻ります。AnswerClass の詳細は「BlzLP&GA DLL 取り扱い説明書」を参照してください。注：Extra クラスの意味が LP と GA では異なります。

問題例

この Sample Solution には、３つの例題を添付しています。

1.wiki.txt：

Cutting Stock Problem の英語版 Wiki（ http://en.wikipedia.org/wiki/Cutting_stock_problem ）に記載されている問題です。小規模な問題ですが、解の品質を評価するのに適した問題です。LP で解析すると１秒以内でパターン数１０の解を得ることが出来ます（ GA 短時間では 10 パターンへ到達不可 ）。

2.middle.txt

LP が対象とする標準的な規模の問題です。解析時間によって得られる解のパターン数がどのように変化するか、試行することが出来ます。また、GA と比較すると適正規模の問題では LP の方がより高速に高品質な解を得られることが分ります。

3.bpp14.txt

BPP/CSP の難題として有名な Hard28 の一つで、大規模かつ最適解を得るのが困難な問題です。Gurobi をサーバー上で利用する VectorPacking(http://vpsolver.dcc.fc.up.pt/demo/vbp#)にて bpp14 を解きますと約 60 秒で Objective：62 と解を得ますが、最適解は 61 本ですので、高額な商用ソルバーであっても、必ずしも最適解を得られるわけではないことが分ります。LP が利用できない規模の問題ですが、GA では１秒の探索時間で同じく 62 本の解を得ることが出来ます。