

スケジューリング最適化コンポーネント

OptSeq

ユーザズガイド

LOG OPT Co., Ltd.

ご注意

- このソフトウェアおよびマニュアルの著作権は LOGOPT 社にあります。
- このソフトウェアおよびマニュアルの一部または全部を無断で複製することはできません。
- このソフトウェアおよびマニュアルを運用した結果の影響については、一切責任を負いかねますのでご了承ください。
- このマニュアルに記載されている事柄は、将来予告なしに変更することがあります。

Windows NT, Windows 95 (98), Access, Excel, Visual Basic, Project は、米国 Microsoft Corporation の商標です。

1 はじめに

この説明書はスケジューリングコンポーネント OptSeq(以下 OptSeq と略します) について書かれたものです。OptSeq は汎用スケジューリングソルバを内蔵しており、ActiveX コンポーネントとして実装されています。OptSeq は限られた時間内で得られる最良の解を出力します。OptSeq では、スケジュールの要素として、作業 (activity) と資源 (resource) を扱います。スケジューリングの種類によって、作業は仕事 (job) やタスク (task) とも呼ばれます。また、資源は機械 (machine) や人 (person) とも呼ばれます。OptSeq では計画期間として 0 から始まる整数を扱います。なお、この説明書ではプログラム例を説明する際に VBA コードを用いています。

本説明書の構成は以下の通りです。2 節では個々のメソッドの詳細を説明します。3 節では、スケジューリング問題を解く際に、それぞれのメソッドを使う順番を説明します。4 節では、実際にマイクロソフトエクセルやマイクロソフトプロジェクトのマクロにおける使用例を示します。5 節では、その他の注意事項について説明します。

2 メソッド

OptSeq はインターフェースとしてメソッドのみを用意しています。プロパティの設定はできません。メソッドは主に、動作設定メソッド、問題入力メソッド、求解メソッド、解出力メソッドに分類できます (表 1 参照)。以下に個々のメソッドの詳細を示します。

表 1: メソッドの分類

分類	メソッド
動作設定	setMaxCpuTime, setHorizon
問題入力	addResource, addActivity, addRes2Act, setPrecedence
求解	solve
解出力	getStartTime

2.1 short setMaxCpuTime(double cpu_time)

引数: 最大計算時間 (秒)

返値: 正常終了 (0), 最大計算時間が範囲外 (-10)

機能: 最大計算時間を秒単位で設定します。設定できる最大計算時間の範囲は 1~3600 です。設定しない場合、最大計算時間はデフォルト値の 600 秒です。

例: `ret = 問題.setMaxCpuTime(60)` , 「問題」の解を 1 分以内に得たい

2.2 short setHorizon(long horizon)

引数: 計画期間

返値: 正常終了 (0), 計画期間が範囲外 (-10)

機能: 最大計画期間を設定します。設定できる計画期間の範囲は 1~2147483647(INT_MAX) です。設定しない場合、最大計画期間はデフォルト値の 1000 です。

例: `ret = 問題.setHorizon(365)` , 「問題」の計画期間は 365 日 (日単位の計画の場合)

2.3 short addResource(long amount)

引数: 資源量

返値: 資源 ID(> 0), メモリオーバー (-1), 資源量が範囲外 (-10)

機能: 問題に資源を加えます。正常に資源を加えることができた場合、返値として資源 ID を返します。設定できる資源量の範囲は 1~2147483647(INT_MAX) です。資源 ID は同一問題の中で固有の自然数です。

例: `資源 ID = 問題.addResource(3)` '3 人で作業 (人が作業する計画の場合)
'この後、この 3 人は「資源 ID」として識別

2.4 short addActivity(long time)

引数: 作業に要する処理時間

返値: 作業 ID(> 0), メモリオーバー (-1), 処理時間が範囲外 (-10)

機能: 問題に仕事を加えます。正常に仕事を加えることができた場合、返値として仕事 ID を返します。設定できる処理時間の範囲は 0~2147483647(INT_MAX) です。(注意: 今後のプログラム改版で範囲が小さくなる可能性があります。) 仕事 ID は同一問題の中で固有の自然数です。

例: `作業 ID = 問題.addActivity(5)` '処理に 5 単位だけ時間がかかる' この後、この作業は「作業 ID」として識別

2.5 short setRes2Act(short actID, short resID, long amount)

引数: 作業 ID, 作業に用いる資源の ID, 作業に用いる資源の量

返値: 正常終了 (0), メモリオーバー (-1), 対応する作業 ID なし (-11), 対応する資源 ID なし (-12), 資源量が範囲外 (-10)

機能: 作業に必要な資源を加えます。作業の処理に用いる資源の種類と量を設定します。一つの作業に対して、複数の資源を加えることができます。設定できる資源量の範囲は 1~2147483647(INT_MAX) です。(注意: 今後のプログラム改版で範囲が小さくなる可能性があります。)

例: 人間のみで完了できる作業の場合

人 ID = `問題.addResource(3)` '3 人が作業可能
作業 ID = `問題.addActivity(5)` '5 日で完了 (日単位の計画の場合)
`問題.addActivityMode(作業 ID, 人 ID, 2)` '2 人必要

例: 人間と機械の両方が必要な作業の場合

人 ID = 問題.addResource(3) '3人が作業可能
機械 ID = 問題.addResource(5) '5台が作業可能
作業 ID = 問題.addActivity(5) '5日で完了(日単位の計画の場合)
問題.addActivityMode(作業 ID, 人 ID, 2) '2人必要
問題.addActivityMode(作業 ID, 機械 ID, 4) '4台必要

2.6 short setPrecedence(short act1, short act2)

引数: 先行する作業の ID, 後続する作業の ID

返値: 先行順序関係 ID (> 0), メモリーオーバー (-1), 対応する作業 ID なし (-11)

機能: 先行順序関係を加えます。ここで指定された先行順序関係は絶対制約となります。先行する仕事が終了しないと、後続する仕事は開始できません。正常に先行順序関係を加えることができた場合、返値として先行順序関係 ID を返します。先行順序関係 ID は同一問題の中で固有の自然数です。

例: 作業 ID1 = 問題.addActivity(7)
作業 ID2 = 問題.addActivity(5)
先行順序関係 ID = 問題.setPrecedence(作業 ID1, 作業 ID2)

2.7 short solve(void)

引数: なし

返値: 正常終了 (0), 異常終了 (< 0)

機能: 問題を解きます。より正確には、タブーサーチを用いて実行可能解を得ます。

備考: 現在テスト中です。

例: ret = 問題.solve

2.8 getStartTime(short actID)

引数: 作業 ID

返値: 作業の開始時間 (> 0), 対応する作業 ID なし (-11)

機能: 作業の開始時間を得ます。

例: 作業 ID = 問題.addActivity(7)
作業の開始時間 = 問題.getStartTime(作業 ID)

3 メソッドを用いる順番

本コンポーネントにおいて、正しい解を得るためにはメソッドを正しい順番で使う必要があります。そうでない場合には異常終了してしまう場合がありますので、必ず正しい順番で使ってください。

以下にメソッドを用いる順番を述べます。大まかに分けると、動作設定、問題入力、求解、解出力の順番で使ってください。まず最初に、動作設定のメソッド `setMaxCpuTime`, `setHorizon` を使ってください。`setHorizon` は省略しても構いませんが `setMaxCpuTime` は必ず使ってください。次に問題入力のメソッド `addResource`, `addActivity`, `addRes2Act`, `setPrecedence` を使ってください。問題入力の4つのメソッドを使う順番は任意で構いませんが、他のメソッドの入力で必要となるものは先に指定してください。例えば、`addResource` で資源を指定しなくても `addActivity` で作業を加えることはできますが、`addRes2Act` で作業が使う資源を設定する前には `addResource` で資源の使用可能量が設定されている必要があります。また、`addActivity` で定義されていない作業間に `setPrecedence` で先行順序関係を定義することはできません。求解と解出力はそれぞれメソッドが一つしかないので、それぞれ順番に使ってください。`solve` で求解する前に `getStartTime` で解出力はできないので注意してください。`solve` 求解する前に `getStartTime` で解出力しようとするると異常終了してしまいます。

4 コンテナプログラムの書き方

4.1 具体的に数値を入れる例

以下に3作業1資源のプロジェクトスケジューリングを入力する場合の例を示します。問題例として、「作業1(処理時間5), 作業2(処理時間8), 作業3(処理時間2)があり, 作業2と作業3は作業1の終了後でなければ開始できない(図1参照)」。という例を用います。この問題をコンテナとしてプログラミングした例が図2です。

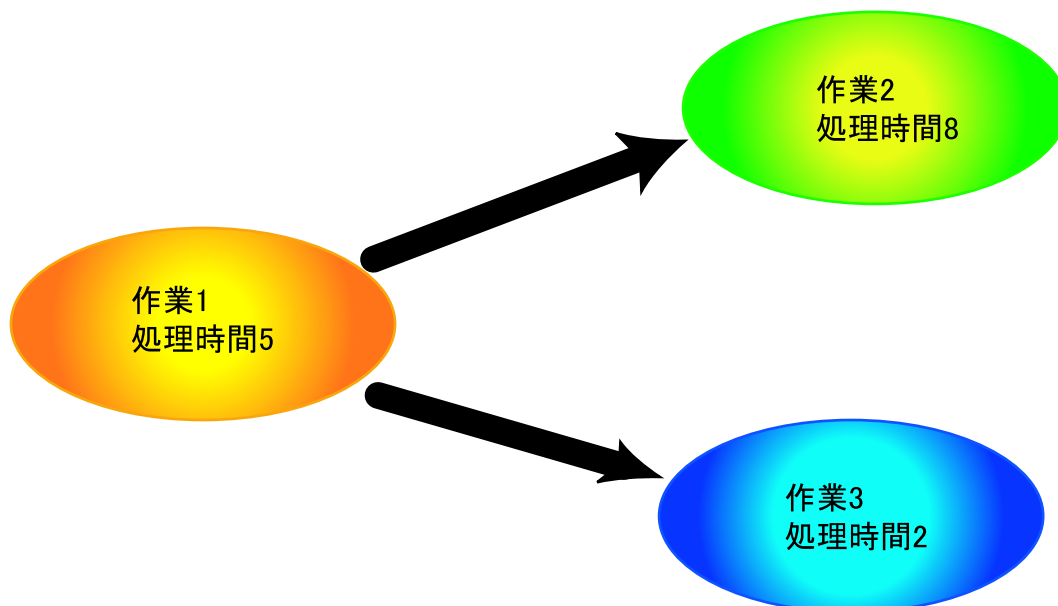


図 1: 3 作業 1 資源の問題例

```

Dim opt_seq As OptSeq
Set opt_seq = New OptSeq
Dim 資源量 As Integer
Dim 資源 ID As Integer
Dim 作業 ID(3) As Integer
Dim 開始時刻 (3) As Integer

opt_seq.setMaxCpuTime(60)
資源量 = 1
' 資源を追加
資源 ID = opt_seq.addResources(資源量)
' 作業を追加
作業 ID(1) = opt_seq.addActivity(5)
opt_seq.addRes2Act(作業 ID(1), 資源 ID, 資源量)
作業 ID(2) = opt_seq.addActivity(8)
opt_seq.addRes2Act(作業 ID(2), 資源 ID, 資源量)
作業 ID(3) = opt_seq.addActivity(2)
opt_seq.addRes2Act(作業 ID(3), 資源 ID, 資源量)
' 先行順序を追加
opt_seq.setPrecedence(作業 ID(1), 作業 ID(2))
opt_seq.setPrecedence(作業 ID(1), 作業 ID(3))
' 求解
opt_seq.solve
' 作業開始時刻を取得
開始時刻 (1) = opt_seq.getStartTime(作業 ID(1))
開始時刻 (2) = opt_seq.getStartTime(作業 ID(2))
開始時刻 (3) = opt_seq.getStartTime(作業 ID(3))

```

図 2: 3 作業 1 資源の問題のコンテナプログラムの例

4.2 マイクロソフトエクセルを用いた場合

以下にジョブショップスケジューリングをエクセルマクロで用いる例を図 3(変数の宣言とエクセルシートからの問題の読み込み), 図 4(問題入力と求解と解出力) に示します。

4.3 マイクロソフトプロジェクトを用いた場合

以下にジョブショップスケジューリングをマイクロソフトプロジェクトマクロで用いる例を図 5(変数の宣言とプロジェクトのオブジェクトからの問題の読み込み), 図 6(問題入力と求解と解出力) に示します。 図 7 は計算後のマイクロソフトプロジェクトの画面です。

```

Dim 資源数 As Integer
Dim 資源 As Integer
Dim スケジューラー As Optseq
Dim 仕事数 As Integer
Dim 仕事 As Integer
Dim ret As Integer
Dim i As Integer
Dim 仕事 ID() As Integer
Dim 処理時間 () As Integer
Dim 資源 ID() As Integer
Dim 後続仕事 ID() As Integer

Set スケジューラー = New Optseq

' 問題をシートから読み込み
仕事数 = Sheet1.Cells(1, 2)
資源数 = Sheet1.Cells(2, 2)
ReDim 仕事 ID(仕事数) As Integer
ReDim 処理時間 (仕事数) As Integer
ReDim 資源 ID(仕事数) As Integer
ReDim 後続仕事 ID(仕事数) As Integer
For i = 1 To 仕事数
    仕事 ID(i) = Sheet1.Cells(4 + i, 1)
    処理時間 (i) = Sheet1.Cells(4 + i, 2)
    資源 ID(i) = Sheet1.Cells(4 + i, 3)
    後続仕事 ID(i) = Sheet1.Cells(4 + i, 4)
Next i

```

図 3: 変数の宣言とエクセルシートからの問題の読み込み

5 その他注意事項

以下にその他の注意事項を述べます。

本コンポーネントはスケジューリングの解を得る方法として、タブーサーチを用いています。よって得られる解は最適なものとは限りません。一般に、実務的なスケジューリング問題において最適な解を得ることは、計算時間の意味で、ほとんど不可能であると言われていています。タブーサーチは限られた時間内で上質の解を得る手法として今もっとも注目されています。

本コンポーネントは指定された計画期間内で作業の順序を入れ換えることによって、スケジューリングの解を求めています。計画期間が作業の処理時間に対して十分大きくないと、解を得ることはできません。あらかじめ計画期間を大きめに設定してください。

```

    ' 問題をスケジューラーに入力
    スケジューラー.setMaxCpuTime (3)
    For 資源 = 1 To 資源数
        スケジューラー.addResource (1)
    Next 資源
    For 仕事 = 1 To 仕事数
        ret = スケジューラー.addActivity(処理時間(仕事))
        ret = スケジューラー.setRes2Act(仕事 ID(仕事), 資源 ID(仕事), 1)
    Next 仕事
    For 仕事 = 1 To 仕事数
        If 後続仕事 ID(仕事) <> 0 Then ret = スケジューラー.setPrecedence(仕事 ID(仕事), 後
    続仕事 ID(仕事))
    Next 仕事

    ' 求解
    スケジューラー.solve

    ' 解をシートに書き込み
    For 仕事 = 1 To 仕事数
        Sheet1.Cells(4 + 仕事, 5) = スケジューラー.getStartTime(仕事 ID(仕事))
    Next 仕事

```

図 4: 問題入力と求解と解出力

```

Dim 資源数 As Integer
Dim 資源 As Integer
Dim スケジューラー As Optseq
Dim 仕事数 As Integer
Dim 仕事 As Integer
Dim ret As Integer
Dim i As Integer
Dim 仕事 ID() As Integer
Dim 処理時間 () As Integer
Dim 資源 ID() As Integer
Dim 後続仕事 ID() As Integer
Dim 開始日 As Date

Set スケジューラー = New Optseq

' 問題をプロジェクトから読み込み
仕事数 = ActiveProject.Tasks.Count
資源数 = ActiveProject.Resources.Count
ReDim 仕事 ID(仕事数) As Integer
ReDim 処理時間 (仕事数) As Integer
ReDim 資源 ID(仕事数) As Integer
ReDim 後続仕事 ID(仕事数) As Integer
For i = 1 To 仕事数
    仕事 ID(i) = ActiveProject.Tasks.UniqueID(1)
    処理時間 (i) = ActiveProject.Tasks(i).Duration
    資源 ID(i) = ActiveProject.Tasks(i).Resources(1).UniqueID
    後続仕事 ID(i) = 0
    If ActiveProject.Tasks(i).SuccessorTasks.Count > 0 Then 後 続 仕 事
ID(i) = ActiveProject.Tasks(i).SuccessorTasks(1).UniqueID
Next i

```

図 5: 変数の宣言とエクセルシートからの問題の読み込み

```

' 問題をスケジューラーに入力
スケジューラー.setMaxCpuTime (3)
For 資源 = 1 To 資源数
    スケジューラー.addResource (1)
Next 資源
For 仕事 = 1 To 仕事数
    ret = スケジューラー.addActivity(処理時間(仕事))
    ret = スケジューラー.setRes2Act(仕事 ID(仕事), 資源 ID(仕事), 1)
Next 仕事
For 仕事 = 1 To 仕事数
    If 後続仕事 ID(仕事) <> 0 Then ret = スケジューラー.setPrecedence(仕事 + 1, 後続仕事 ID(仕事) + 1)
Next 仕事

' 求解
ret = スケジューラー.solve

' 解をシートに書き込み
開始日 = DateValue(ActiveProject.ProjectStart)
For 仕事 = 1 To 仕事数
    ActiveProject.Tasks(仕事).Start = 開始日 + スケジューラー.getStartTime(仕事 + 1)
Next 仕事

```

図 6: 問題入力と求解と解出力

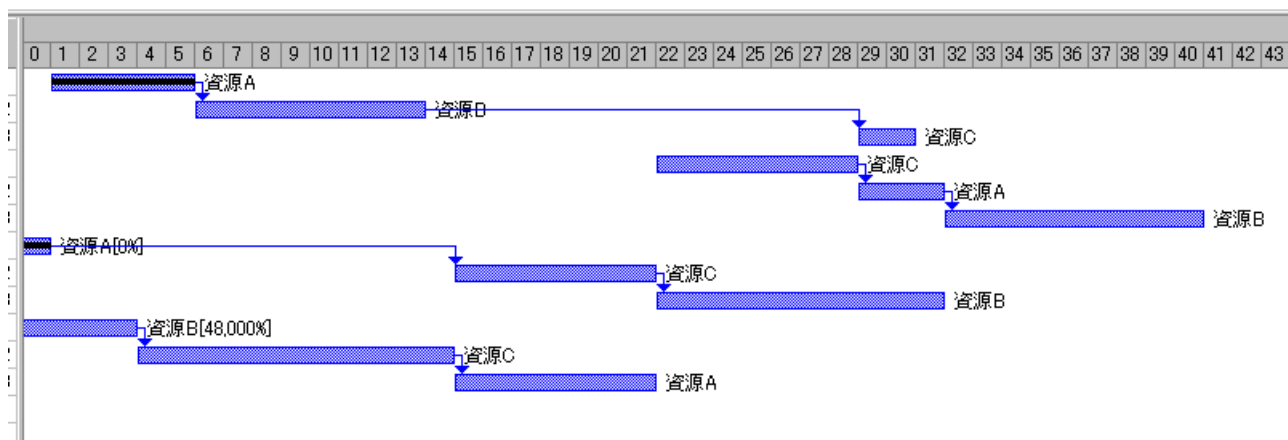


図 7: マイクロソフトプロジェクトの画面