制約最適化ソルバー SCOP
テキストインターフェイス

Solver for COnstraint Programming：スコープ

LOG OPT Co., Ltd.

SCOPとは

- 大規模な制約計画問題を高速に解くためのソルバー
- 茨木先生（京都大学名誉教授）と野々部先生（法政大学）の開発したメタヒューリスティクスを基礎
- トライアルバージョン（１５変数まで）
  http://www.logopt.com/scop.htm
- 簡易モデリング言語によるデータ入力と，ライブラリ呼び出しによる利用が可能（C++, .Visual Basic, C#, Pythonなどから呼び出し可能）
- 研究の普及を主目的としているため安価

制約計画とは

数理計画とは異なり組合せ最適化に特化したソルバー

制約充足問題（constraint satisfaction problem）を対象

- 変数（variable）: 最適化によって決めるもの．制約充足問題では，変数は，与えられた集合（領域）から1つの要素を選択
- 領域（domain）: 変数ごとに決められた変数のとりえる値の集合
- 制約（constraint）: 幾つかの変数が同時にとることのできる値に制限を付加するための条件

重み付き制約充足問題

- SCOPで対象とする制約充足問題の拡張
- 単に制約を満たす解を求めるだけでなく，制約からの逸脱量の重み付き和（ペナルティ）を最小化
  - 逸脱を許さない制約（ハードな制約，絶対制約）
  - 逸脱を許す制約（ソフトな制約，考慮制約）
- 目的関数も，ソフトな制約として処理可能

仕事の割当の例(1)

- 3人の作業員 A,B,C を3つの仕事 0,1,2 に割り当てる.
- 割当費用＝

	0	1	2
A	15	20	30
B	7	15	12
C	25	10	13

変数の定義

variable　変数名 in { 領域 }

領域　＝　値1, 値2, 値3, ...

variable　A in {0, 1, 2}
variable　B in {0, 1, 2}
variable　C in {0, 1, 2}

作業員 A,B,Cに対して,
割当可能な仕事を領域と定義

制約の定義

制約名: weight= 制約の重み
type = 制約の種類
[制約本体]

制約の重み = 正数（ソフトな制約を定義）
inf（無限大;ハードな制約を定義）

制約の種類 = linear（線形制約）,
alldiff（相異制約）, quadratic（2次制約）

線形制約

制約名: weight= 制約の重み
type = linear

[制約本体] =
係数1 (変数名1 , 値1)
係数2 (変数名2 , 値2), ...
<= (>=, =) 右辺定数

	0	1	2
A	15	20	30
B	7	15	12
C	25	10	13

```
obj: weight=1  type=linear
  15 (A, 0) 20(A, 1) 30(A, 2)
   7 (B, 0) 15(B, 1) 12(B, 2)
  25 (C, 0) 10(C, 1) 13(C, 2)
 <=0
```

目的関数を表す線形制約
割当費用の和が0以下になるように規定

重みは1であるので，ソフトな制約

相異制約

制約名: weight= 制約の重み
type = alldiff

[制約本体] =
変数名1 , 変数名2 , ... , 最後の変数名;

constraint: weight=inf type=alldiff A B C;

作業員に割り当てられる仕事がすべて異なることを規定
重みはinf（無限大）なので，ハードな制約

SCOPによる求解

scop < ex1-scop.dat

```
# reading data ... done: 0.00(s)

penalty = 1/32 (hard/soft), time = 0.00(s), iteration = 0
# improving the initial solution greedily
penalty = 0/47 (hard/soft), time = 0.00(s), iteration = 0
# start tabu search
penalty = 0/37 (hard/soft), time = 0.01(s), iteration = 2
# penalty = 0/37 (hard/soft)
# cpu time = 0.01/0.05(s)
# iteration = 2/100

[best solution]
A: 0
B: 2
C: 1

penalty: 0/37 (hard/soft)

[Violated constraints]
obj: 37
```

37=15+12+10（万円）の費用

	0	1	2
A	15	20	30
B	7	15	12
C	25	10	13

仕事の割当の例(2)

- ５人の作業員 A,B,C,D,E を3つの仕事 0,1,2 に割り当てる.
- 仕事の必要人数＝(１，２，２)人
- 割当費用＝

	0	1	2
A	15	20	30
B	7	15	12
C	25	10	13
D	15	18	3
E	5	12	17

変数の定義

variable　変数名 in { 領域 }

領域　＝　値1, 値2, 値3, ...

```
variable  A in {0, 1, 2}
variable  B in {0, 1, 2}
variable  C in {0, 1, 2}
variable  D in {0, 1, 2}
variable  E in {0, 1, 2}
```

作業員 A,B,C,D,Eに対して,
割当可能な仕事を領域と定義

線形制約（1）

制約名: weight= 制約の重み
type = linear

[制約本体] =
係数1 (変数名1 , 値1)
係数2 (変数名2 , 値2), ...
<= (>=, =) 右辺定数

	0	1	2
A	15	20	30
B	7	15	12
C	25	10	13
D	15	18	3
E	5	12	17

```
constraint0: weight=inf  type=linear
    1(A,0) 1(B,0) 1(C,0) 1(D,0) 1(E,0) >=1
```

仕事0に作業員が1人以上割り当てることを表す線形制約
重みはinfであるので，ハードな制約

線形制約（2）

```
constraint1: weight=inf  type=linear
    1(A,1) 1(B,1) 1(C,1) 1(D,1) 1(E,1) >=2

constraint2: weight=inf  type=linear
    1(A,2) 1(B,2) 1(C,2) 1(D,2) 1(E,2) >=2

obj: weight=1  type=linear
  15 (A, 0) 20(A, 1) 30(A, 2)
   7 (B, 0) 15(B, 1) 12(B, 2)
  25 (C, 0) 10(C, 1) 13(C, 2)
  15 (D, 0) 18(D, 1)  3(D, 2)
   5 (E, 0) 12(E, 1) 17(E, 2)
 <=0
```

仕事1に作業員が2人以上

仕事2に作業員が2人以上

目的関数を表す線形制約
費用の和が0以下になるように規定

重みは1であるので，ソフトな制約

SCOPによる求解

```
scop  <ex2-scop.dat
# reading data ... done: 0.00(s)

penalty = 1/52 (hard/soft), time = 0.01(s), iteration = 0
# improving the initial solution greedily
penalty = 0/57 (hard/soft), time = 0.01(s), iteration = 0
# start tabu search
penalty = 0/50 (hard/soft), time = 0.03(s), iteration = 2
# penalty = 0/50 (hard/soft)
# cpu time = 0.03/0.05(s)
# iteration = 2/100

[best solution]
A: 1
B: 2
C: 1
D: 2
E: 0

penalty: 0/50 (hard/soft)

[Violated constraints]
obj: 50
```

50 (=20+12+10+3+5) 万円

	0	1	2
A	15	20	30
B	7	15	12
C	25	10	13
D	15	18	3
E	5	12	17

仕事の割当の例(3)

- ５人の作業員 A,B,C,D,E を3つの仕事 0,1,2 に割り当,必要人数は(1,2,2)人（前の例と同じ）
- 作業員AとCは仲が悪いので，異なる仕事に割り振る
- 割当費用＝

	0	1	2
A	15	20	30
B	7	15	12
C	25	10	13
D	15	18	3
E	5	12	17

2次制約（1）

制約名: weight= 制約の重み
　　type　= quadratic

　　[制約本体] =
係数1 (変数名1 , 値1) (変数名2 , 値2)
係数2 (変数名3 , 値3) (変数名4 , 値4)
　...
　<= (>=, =) 右辺定数

```
quad: weight=100 type=quadratic
  1 (A,0) (C,0)   1 (A,1) (C,1)  1 (A,2) (C,2)  <=0
```

仕事AとCが同じ仕事に割り当てられることを禁止する制約
重みは100のソフトな制約

SCOPによる求解

```
scop  <ex3-scop.dat
# reading data ... done: 0.00(s)

penalty = 1/52 (hard/soft), time = 0.00(s), iteration = 0
# improving the initial solution greedily
penalty = 0/157 (hard/soft), time = 0.00(s), iteration = 0
# start tabu search
penalty = 0/60 (hard/soft), time = 0.01(s), iteration = 2
penalty = 0/52 (hard/soft), time = 0.01(s), iteration = 8
# penalty = 0/52 (hard/soft)
# cpu time = 0.01/0.03(s)
# iteration = 8/100

[best solution]
A: 0
B: 2
C: 1
D: 2
E: 1

penalty: 0/52 (hard/soft)

[Violated constraints]
obj: 52
```

52 (=15+12+10+3+12) 万円

	0	1	2
A	15	20	30
B	7	15	12
C	25	10	13
D	15	18	3
E	5	12	17

SCOPの使用法

コマンドプロンプト（Windows），コンソール（Unix）上で実行

scop < 入力ファイル名

scop -（オプション）<入力ファイル名

scop -help でオプションのヘルプを表示

SCOPのオプション

-noadjust：タブー探索中のペナルティ重み自動調整機能を無効化

-display : ログの出力レベルを設定 (0,1,2,3；数が大きいほど詳細情報表示)
（規定値＝0）

-interval : 解移動情報を何反復ごとに出力するか設定（規定値＝0）

-iteration : 最大反復回数を設定（規定値＝∞）

-seed ：乱数系列の種を設定，任意の正整数入力可（規定値＝1）

-target :目標とするペナルティ値を設定（既定値＝0）

-time :最大計算時間 (秒)（既定値＝600）

-initial：初期解設定

SCOPのデータ入力（まとめ）

1. 変数, 領域の定義
   variable 変数名 in { 値1, 値2, ... }
2. 目標値の設定
   target = 目標値
3. 制約の記述
   制約名: weight = 制約の重み
   type = 制約の種類　制約本体

必ず1,2を3の前に行う！

初期解設定方法

- 手順1. 結果ファイル下記の部分を切り取り，新しいファイルに保存（これが初期解ファイルである．）

  [best solution]

  …

  …

  penalty

  …

  切り取る部分（変数の値）

- 手順2. コマンドプロンプトで下記のように入力．

  >scop -initial 初期解ファイル名 < データファイル名

応用事例

- 時間割作成
  授業を時限と教室に割り当てる問題.
  SCOPを用いたソルバー;国際コンペティション
  (ITC-2007: International Timetabling Competition)でファイナリスト(全種目)
- スタッフスケジューリング
  スタッフの期ごとのシフトを決定する問題.
  SCOPを用いて看護婦スケジューリングなどの複雑な実際問題を解決
- 資材置き場への製品の配置
  ビンパッキング問題とグラフ彩色問題の融合問題.
  SCOPを用いて容易にモデル化