2022/10/10 Mon.

問題解決

組合せ最適化と整数計画法
Sports Scheduling

堀田 敬介

スポーツ・スケジューリング

team/slot	1	2	3	4	5
A	B	C			
B	A	E			
C	D	A			
D	C	F			
E	F	B			
F	E	D			

- single round-robin tournament
  - 6チームの総当たり戦スケジュールをつくる
  - 全試合 Home vs Away で戦う
  - 全チーム総移動距離の最小化を目指す【目的1】

d_{ij}	A	B	C	D	E	F
A	0	5	4	6	6	8
B		0	5	9	4	7
C			0	4	2	4
D				0	6	6
E					0	3
F						0

- 最適化問題の定式化（Σ表記）
  - 0-1変数 $x_{ijs}=1$ … slot s で i vs j (i がHome)
  - 0-1変数 $x_{ijs}=0$ … slot s で i vs j (i がHome)をしない
  - 距離行列 distance matrix $\boldsymbol{D}=[d_{ij}]$

$$min. \sum_{s\in S}\sum_{i\in T}\sum_{j\in T} 2d_{ij}x_{ijs}$$

$$s.t. \sum_{i\in T/\{j\}} (x_{ijs}+x_{jis}) = 1\ (\forall j \in T, \forall s \in S)$$

$$\sum_{s\in S} (x_{ijs}+x_{jis}) = 1\ (\forall i,j \in T(i \neq j))$$

$$x_{ijs} \in \{0,1\}(\forall i,j \in T, \forall s \in S)$$

T={1,2,…,6} : team集合
S={1,2,…,5} : slot集合
Away teamは1試合毎に「出掛けて」「戻る」ことにする

スポーツ・スケジューリングをCPLEXで解く

- 新規プロジェクトの作成
  - ① [ファイル(F)]ー[新規(N)]ー[OPLプロジェクト]を選択
  - ② [プロジェクト名] を記入（例：SportsScheduling）し，3カ所にチェックする
    - ☑ デフォルトの実行構成の追加
    - ☑ モデルの作成
    - ☑ データの作成
  - ③ [終了]をクリック

プロジェクト名は自由だが，半角英数で何の問題を解こうとしているのかが分かる名前が良い

- プロジェクト内のいくつかの名前を変更
  - ✓ [構成1] → [config1]　※日本語を英語に変更しないと実行時エラーになる
  - ✓ モデルファイル [SportsScheduling.mod] → [srt-di.mod]
  - ✓ データファイル [SportsScheduling.dat] → [srt-diex1.dat]

- モデルファイル・データファイルを記述し保存（次ページ参照）
- [config1]にモデルファイルとデータファイルをセットし，解く

スポーツ・スケジューリングをCPLEXで解く

- モデルファイル（srt-di.mod）の中身の記述

```
int i_max = ...;// チーム数の添え字の最大値
range I = 1..i_max;
range S = 1..i_max-1;// slot数 = team数-1

float d[I,I] = ...;// 距離行列 (size: IxI)

dvar int+ x[I,I,S] in 0..1;// 0-1変数(size: IxIxS)

minimize
  sum(s in S) sum(i in I) sum(j in I) 2*d[i,j]*x[i,j,s];
subject to{
  forall(s in S) {// 各slot において
    forall(i in I) {// 各チーム i は
      sum(j in I:i!=j) (x[i,j,s] + x[j,i,s]) == 1;
    };// 自分以外の全チーム j とHome/Away のどちらかで 丁度1回対戦する
  };
  forall(i in I) {// 各チーム i は
    forall(j in I:i!=j) {// 自分以外の全チーム j と各々
      sum(s in S) (x[i,j,s] + x[j,i,s]) == 1;
    };// どこかのslotで Home/Away のどちらかで 丁度1回対戦する
  };
  forall(i in I) {// 自分とは対戦しないので
    sum(s in S) x[i,i,s] == 0;// その変数を全て0に
  };
};
```

※この変数定義では，
[i vs i] in s を含むので…

※自チームとは戦わない，
というこの式が必要

スポーツ・スケジューリングをCPLEXで解く

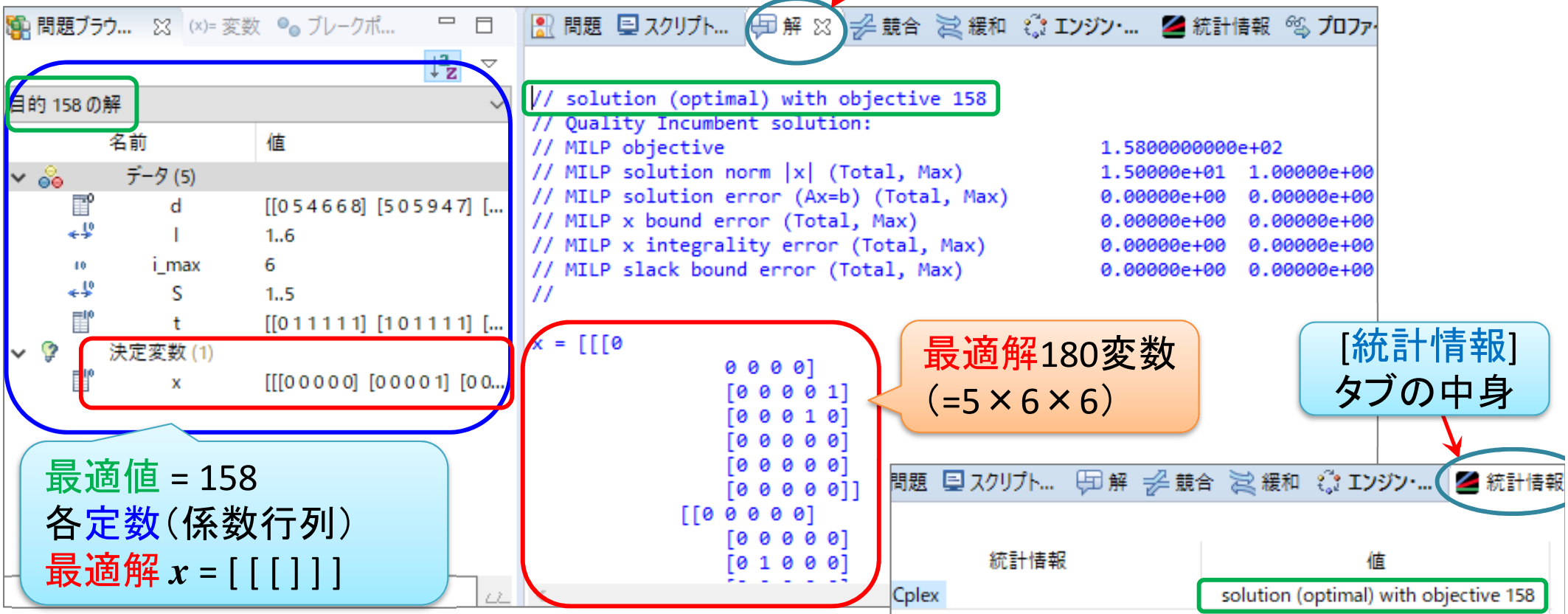
- データファイル（srt-diex1.dat）の中身の記述

```
i_max = 6;// team数最大値

d = [
[0 5 4 6 6 8]
[5 0 5 9 4 7]
[4 5 0 4 2 4]
[6 9 4 0 6 6]
[6 4 2 6 0 3]
[8 7 4 6 3 0]
];// 距離行列（往路・復路の距離が同じなら，対称行列になる）
```

SSをCPLEXで解く

➢ 計算結果の確認1



※1対1の1回総当たり戦で，Home-Away 単純往復なので，目的関数値（総移動距離）は必ず距離行列の全要素の和に一致する
（なのでこの目的関数に意味は無い）

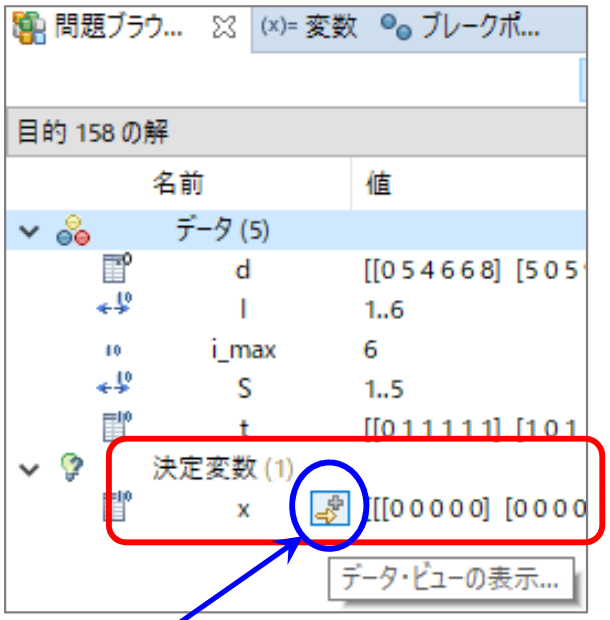
→ minimaxにして全チーム移動距離均等化に
（※最大移動距離最小化という意味での均等）

SSをCPLEXで解く

➢ 計算結果の確認2

※ 1回クリック = 昇順ソート
2回クリック = 降順ソート
3回クリック = 元の順に戻る

[値]を2回クリックし降順ソートする



このボタンをクリックし
データ・ビューを表示
※マウスをこの位置に近づける
とボタンが表示される

$x_{134} = 1$
team1(H) vs 3(A) in slot4

データ・ビュー

ssdi.mod | ssdiex1.dat | x の値

I (サイズ 6)	I (サイズ 6)	S (サイズ 5)	値
1	1	1	0
1	1	2	0
1	1	3	0
1	1	4	0
1	1	5	0
1	2	1	0
1	2	2	0
1	2	3	0
1	2	4	0
1	2	5	1
1	3	1	0
1	3	2	0
1	3	3	0
1	3	4	1
1	3	5	0
1	4	1	0
1	4	2	0
1	4	3	0
1	4	4	0
1	4	5	0
1	5	1	0
1	5	2	0
1	5	3	0
1	5	4	0

ssdi.mod | ssdiex1.dat | x の値

I (サイズ 6)	I (サイズ 6)	S (サイズ 5)	値
6	5	2	1
6	4	5	1
6	3	3	1
6	2	4	1
6	1	1	1
5	4	4	1
5	2	1	1
5	1	3	1
4	2	3	1
4	1	2	1
3	5	5	1
3	4	1	1
2	3	2	1
1	3	4	1
1	2	5	1
6	6	5	0
6	6	4	0
6	6	3	0
6	6	2	0
6	6	1	0
6	5	5	0
6	5	4	0
6	5	3	0
6	5	1	0

SSをCPLEXで解く

➢ 計算結果の確認3

データ･ビュー

ssdi.mod | ssdiex1.dat | x の値

I (サイズ 6)	I (サイズ 6)	S (サイズ 5)	値
6	5	2	1
6	4	5	1
6	3	3	1
6	2	4	1
6	1	1	1
5	4	4	1
5	2	1	1
5	1	3	1
4	2	3	1
4	1	2	1
3	5	5	1
3	4	1	1
2	3	2	1
1	3	4	1
1	2	5	1
6	6	5	0
6	6	4	0
6	6	3	0
6	6	2	0
6	6	1	0
6	5	5	0
6	5	4	0

1行目をクリック

[Shift]キーを押しながら
この行[値=1の最終行]
クリック

[値=1]の行（15行=3試合×5slot）が全選択されたので，この状態で [Ctrl]+[c]（コピー）

Excelシートに貼り付け（[Ctrl]+[v]）

結果

	対戦表	3	3	3	3	3
	team/slot	1	2	3	4	5
1	A	F	D	E	C	B
2	B	E	C	D	F	A
3	C	D	B	F	A	E
4	D	C	A	B	E	F
5	E	B	F	A	D	C
6	F	A	E	C	B	D

Home-Away table

H	A	team/slot	1	2	3	4	5
2	3	A	A	A	A	H	H
1	4	B	A	H	A	A	A
2	3	C	H	A	A	A	H
2	3	D	A	H	H	A	A
3	2	E	H	A	H	H	A
5	0	F	H	H	H	H	H

Excel上で加工

※ここでは手で copy & paste してるが，データファイル[***.dat]に，結果をExcelシートに貼り付ける命令を書いても可（CPLEXに解の copy & paste を命じる）

スポーツ・スケジューリング

team/slot	1	2	3	4	5
A	B	C			
B	A	E			
C	D	A			
D	C	F			
E	F	B			
F	E	D			

- single round-robin tournament
  - 6チームの総当たり戦スケジュールをつくる
  - 全試合 Home vs Away で戦う
  - 全チーム移動距離の均等化を目指す【目的2】
    （※移動距離が最大のチームを最小化）
- 最適化問題の定式化（Σ表記）
  - 0-1変数 x_{ijs}＝1 … slot s で i vs j (i がHome)
  - 0-1変数 x_{ijs}＝0 … slot s で i vs j (i がHome)をしない

$$min. \max_{j \in T} \sum_{s \in S} \sum_{i \in T} 2 d_{ij} x_{ijs}$$

$$s.t. \sum_{i \in T/\{j\}} (x_{ijs} + x_{jis}) = 1 \ (\forall j \in T, \forall s \in S)$$

$$\sum_{s \in S} (x_{ijs} + x_{jis}) = 1 \ (\forall i, j \in T (i \neq j))$$

$$x_{ijs} \in \{0,1\} (\forall i, j \in T, \forall s \in S)$$

$$min. dist$$

$$s.t. \sum_{s \in S} \sum_{i \in T} 2 d_{ij} x_{ijs} \leq dist$$

$$\sum_{i \in T/\{j\}} (x_{ijs} + x_{jis}) = 1 \ (\forall j \in T, \forall s \in S)$$

$$\sum_{s \in S} (x_{ijs} + x_{jis}) = 1 \ (\forall i, j \in T (i \neq j))$$

$$x_{ijs} \in \{0,1\} (\forall i, j \in T, \forall s \in S)$$

スポーツ・スケジューリングをCPLEXで解く

- モデルファイル[srt-di.mod] を copy & paste して修正
  - ① モデルファイル [srt-di.mod] を選択（クリック）
  - ② [Ctrl]+[c] を押して copy
  - ③ [Ctrl]+[v] を押して paste
  - ④ 名前 [srt-di2.mod] を入力
  - ⑤ モデルファイル [srt-di2.mod] をダブルクリックして開き中身を修正
    （※修正部分は次ページ参照）

- モデルファイルを入れ替えて解く
  - ① 実行構成　[config1] のモデルファイルを入れ替えて解く
    1. [config1] 内の [srt-di.mod] を削除
    2. [config1] 内に [srt-di2.mod] を drag&drop

    （※データファイルは同じもの [srt-diex1.dat] を使う）

スポーツ・スケジューリングをCPLEXで解く

- モデルファイル（srt-di2.mod）の中身の記述

```
int i_max = ...;// チーム数の添え字の最大値
range I = 1..i_max;
range S = 1..i_max-1;// slot数 = team数-1
float d[I,I] = ...;// 距離行列 (size: IxI)
dvar int+ x[I,I,S] in 0..1;// 0-1変数(size: IxIxS)
dvar float dist;// minimax上界用変数

minimize
  dist;
subject to{
  forall(i in I) {// 各チーム i の総移動距離を計算し，distで上から抑える
    sum(s in S) sum(j in I) 2*d[i,j]*x[i,j,s] <= dist;
  };
  forall(s in S) {// 各slot において
    forall(i in I) {// 各チーム i は
      sum(j in I:i!=j) (x[i,j,s] + x[j,i,s]) == 1;
    };// 自分以外の全チーム j とHome/Away のどちらかで 丁度1回対戦する
  };
  forall(i in I) {// 各チーム i は
    forall(j in I:i!=j) {// 自分以外の全チーム j と各々
      sum(s in S) (x[i,j,s] + x[j,i,s]) == 1;
    };// どこかのslotで Home/Away のどちらかで 丁度1回対戦する
  };
  forall(i in I) {// 自分とは対戦しないので
    sum(s in S) x[i,i,s] == 0;// その変数を全て0に
  };
};
```

追加

修正

追加

SSをCPLEXで解く

➢ 計算結果の確認（例：ex1）

ssdi.mod　ssdiex1.dat　ssdi2.mod　x の値

↓I (サイズ 6)	↓I (サイズ 6)	↓S (サイズ 5)	↓値.
6	4	3	1
6	1	4	1
5	6	5	1
5	2	4	1
4	5	1	1
4	3	4	1
4	2	5	1
3	6	1	1
3	5	2	1
2	6	2	1
2	3	3	1
2	1	1	1
1	5	3	1
1	4	2	1
1	3	5	1
6	6	5	0
6	6	4	0
6	6	3	0
6	6	2	0
6	6	1	0
6	5	5	0
6	5	4	0
6	5	3	0
6	5	2	0
6	5	1	0
6	4	5	0

[値=1]で降順ソート後のデータ･ビュー

1行目をクリック

[Shift]キーを押しながらこの行[値=1の最終行]クリック

[値=1]の行（15行=3試合×5slot）が全選択されたので，この状態で [Ctrl]+[c]（コピー）

Excelシートに貼り付け（[Ctrl]+[v]）

結果

	対戦表	3	3	3	3	3
	team/slot	1	2	3	4	5
1	A	B	D	E	F	C
2	B	A	F	C	E	D
3	C	F	E	B	D	A
4	D	E	A	F	C	B
5	E	D	C	A	B	F
6	F	C	B	D	A	E

Home-Away table

H	A	team/slot	1	2	3	4	5
3	2	A	A	H	H	A	H
3	2	B	H	H	H	A	A
2	3	C	H	H	A	A	A
3	2	D	H	A	A	H	H
2	3	E	A	A	A	H	H
2	3	F	A	A	H	H	A

最適値 = 28
（※最大移動距離のteamの移動距離）

Excel上で加工

演習

- single round-robin tournament
  - 8チームの総当たり戦スケジュールをつくる
  - 全試合 Home vs Away で戦う

  ✓ データファイル [srt-diex2.dat] を作成

データファイル（srt-diex2.dat）

```
i_max = 8;// team数最大値

d = [
[0 3 5 5 3 6 10 7]
[3 0 3 3 3 4 7 4]
[5 3 0 6 6 3 6 5]
[5 3 6 0 3 4 6 2]
[3 3 6 3 0 6 9 5]
[6 4 3 4 6 0 4 3]
[10 7 6 6 9 4 0 4]
[7 4 5 2 5 3 4 0]
];// 距離行列
```

- 2つの目的それぞれで解いてみよう
  - 全チーム総移動距離の最小化を目指す【目的1】
    - ✓ モデルファイル [srt-di.mod] で解く
  - 全チーム移動距離の均等化を目指す【目的2】

    （※移動距離が最大のチームを最小化）
    - ✓ モデルファイル [srt-di2.mod] で解く

スポーツ・スケジューリング

team/slot	1	2	3	4	5
A	B	C			
B	A	E			
C	D	A			
D	C	F			
E	F	B			
F	E	D			

- single round-robin tournament
  - 6チームの総当たり戦スケジュールをつくる
  - 全試合 Home vs Away で戦う
  - Home-Away数の均等化をしたい【目的3】
    （※Home or Away数が最大のチームを最小化）
- 最適化問題の定式化（Σ表記）
  - 0-1変数 x_{ijs}＝1 … slot s で i vs j (i がHome)
  - 0-1変数 x_{ijs}＝0 … slot s で i vs j (i がHome)をしない

$$min. \max_{j \in T} \left\{ \sum_{s \in S} \sum_{i \in T} x_{ijs}, \sum_{s \in S} \sum_{i \in T} x_{jis} \right\}$$

$$s.t. \sum_{i \in T/\{j\}} (x_{ijs} + x_{jis}) = 1 \ (\forall j \in T, \forall s \in S)$$

$$\sum_{s \in S} (x_{ijs} + x_{jis}) = 1 \ (\forall i, j \in T (i \neq j))$$

$$x_{ijs} \in \{0,1\} (\forall i, j \in T, \forall s \in S)$$

$$min. nha$$

$$s.t. \sum_{s \in S} \sum_{i \in T} x_{ijs} \leq nha, \sum_{s \in S} \sum_{i \in T} x_{jis} \leq nha \ (\forall j \in T)$$

$$\sum_{i \in T/\{j\}} (x_{ijs} + x_{jis}) = 1 \ (\forall j \in T, \forall s \in S)$$

$$\sum_{s \in S} (x_{ijs} + x_{jis}) = 1 \ (\forall i, j \in T (i \neq j))$$

$$x_{ijs} \in \{0,1\} (\forall i, j \in T, \forall s \in S)$$

スポーツ・スケジューリングをCPLEXで解く

- モデルファイル[srt-di2.mod] を copy & paste して修正
  - ① モデルファイル [srt-di2.mod] を選択（クリック）
  - ② [Ctrl]+[c] を押して copy
  - ③ [Ctrl]+[v] を押して paste
  - ④ 名前 [srt-ha.mod] を入力
  - ⑤ モデルファイル [srt-ha.mod] をダブルクリックして開き中身を修正
    （※修正部分は次ページ参照）

- モデルファイルを入れ替えて解く
  - ① 実行構成　[config1] のモデルファイルを [srt-ha.mod] にして解く

  - ② データファイルは6チーム（例1）と8チーム（例2）それぞれで解いてみよう
    - ✓ 例1）6チームのデータファイル [srt-diex1.dat]
    - ✓ 例2）8チームのデータファイル [srt-diex2.dat]

スポーツ・スケジューリングをCPLEXで解く

- モデルファイル（srt-ha.mod）の中身の記述

```
int i_max = ...;// チーム数の添え字の最大値

range I = 1..i_max;
range S = 1..i_max-1;// slot数 = team数-1

float d[I,I] = ...;// 距離行列 (size: IxI)

dvar int+ x[I,I,S] in 0..1;// 0-1変数(size: IxIxS)
dvar float nha;// minimax上界用変数

minimize
  nha;
subject to{
  forall(i in I) {// チームi毎の Home数とaway数を計算
    sum(s in S) sum(j in I) x[i,j,s] <= nha;// Home試合数をnhaで上から抑える
    sum(s in S) sum(j in I) x[j,i,s] <= nha;// Away試合数をnhaで上から抑える
  };
  forall(s in S) {// 各slot において
    forall(i in I) {// 各チーム i は
      sum(j in I:i!=j) (x[i,j,s] + x[j,i,s]) == 1;
    };// 自分以外の全チーム j とHome/Away のどちらかで 丁度1回対戦する
  };
  forall(i in I) {// 各チーム i は
    forall(j in I:i!=j) {// 自分以外の全チーム j と各々
      sum(s in S) (x[i,j,s] + x[j,i,s]) == 1;
    };// どこかのslotで Home/Away のどちらかで 丁度1回対戦する
  };
  forall(i in I) {// 自分とは対戦しないので
    sum(s in S) x[i,i,s] == 0;// その変数を全て0に
  };
};
```

※このモデルでは「距離行列」は必要ないが，データファイルを流用したいので残してある

修正

修正

修正

SSをCPLEXで解く

➢ 計算結果の確認（例：ex1）

[値=1]で降順ソート後のデータ･ビュー

ssha.mod　sshaex1.dat　x の値

I (サイズ 6)	I (サイズ 6)	S (サイズ 5)	値.
6	4	4	1
6	3	1	1
6	1	5	1
5	6	2	1
5	4	1	1
5	1	4	1
4	3	5	1
4	2	2	1
3	5	3	1
3	2	4	1
2	6	3	1
2	5	5	1
2	1	1	1
1	4	3	1
1	3	2	1
6	6	5	0
6	6	4	0
6	6	3	0
6	6	2	0
6	6	1	0
6	5	5	0
6	5	4	0
6	5	3	0
6	5	2	0

←1行目をクリック

[Shift]キーを押しながら
この行[値=1の最終行]
クリック

[値=1]の行（15行=3試合×5slot）が全選択されたので，この状態で [Ctrl]+[c]（コピー）

結果

	対戦表	3	3	3	3	3
	team/slot	1	2	3	4	5
1	A	B	C	D	E	F
2	B	A	D	F	C	E
3	C	F	A	E	B	D
4	D	E	B	A	F	C
5	E	D	F	C	A	B
6	F	C	E	B	D	A

Home-Away table

H	A	team/slot	1	2	3	4	5
2	3	A	A	H	H	A	A
3	2	B	H	A	H	A	H
2	3	C	A	A	H	H	A
2	3	D	A	H	A	A	H
3	2	E	H	H	A	H	A
3	2	F	H	A	A	H	H

最適値 = 3
（※Home/Away各試合数最大teamの試合数）

Excel上で加工

Excelシートに貼り付け（[Ctrl]+[v]）

スポーツ・スケジューリング

team/slot	1	2	3	4	5
A	B	C			
B	A	E			
C	D	A			
D	C	F			
E	F	B			
F	E	D			

- single round-robin tournament
  - 6チームの総当たり戦スケジュールをつくる
  - 全試合 Home vs Away で戦う
  - Break数を最小化したい【目的4】
- 最適化問題の定式化（Σ表記）
  - 0-1変数 $x_{ijs}=1$ … slot s で i vs j (i がHome)で対戦，$x_{ijs}=0$ … 対戦しない
  - 0-1変数 $y_{is}=1$ … team i がslot s→s+1 でBreak，$y_{is}=0$ … Breakではない

$$min. \sum_{s\in S}\sum_{i\in T} y_{is} \quad s.t. \sum_{j\in T}(x_{ijs}+x_{ijs+1}) \le y_{is}+1 (\forall i \in T)$$

← team i が slot間 s→s+1 でHomeのBreakがあるか？

$$\sum_{j\in T}(x_{jis}+x_{jis+1}) \le y_{is}+1 (\forall i \in T)$$

← AwayのBreakがあるか？

$$\sum_{i\in T/\{j\}}(x_{ijs}+x_{jis}) = 1 (\forall j \in T, \forall s \in S)$$

$$\sum_{s\in S}(x_{ijs}+x_{jis}) = 1 (\forall i,j \in T(i \ne j))$$

$$x_{ijs} \in \{0,1\}(\forall i,j \in T, \forall s \in S)$$

$$y_{is} \in \{0,1\}(\forall i \in T, \forall s \in S)$$

スポーツ・スケジューリングをCPLEXで解く

- モデルファイル[srt-ha.mod] を copy & paste して修正
  - ① モデルファイル [srt-ha.mod] を選択（クリック）
  - ② [Ctrl]+[c] を押して copy
  - ③ [Ctrl]+[v] を押して paste
  - ④ 名前 [srt-ha2.mod] を入力
  - ⑤ モデルファイル [srt-ha2.mod] をダブルクリックして開き中身を修正
    （※修正部分は次ページ参照）

- モデルファイルを入れ替えて解く
  - ① 実行構成　[config1] のモデルファイルを [srt-ha2.mod] にして解く
  - ② データファイルは6チーム（例1）と8チーム（例2）それぞれで解いてみよう
    - ✓ 例1）6チームのデータファイル [srt-diex1.dat]
    - ✓ 例2）8チームのデータファイル [srt-diex2.dat]

スポーツ・スケジューリングをCPLEXで解く

- モデルファイル（srt-ha2.mod）の中身の記述

追加
修正
修正
修正

```
int i_max = ...;// チーム数の添え字の最大値
range I = 1..i_max;
range S = 1..i_max-1;// slot数 = team数-1
range SB = 1..i_max-2;// slot数-1
float d[I,I] = ...;// 距離行列 (size: IxI)
dvar int+ x[I,I,S] in 0..1;// 0-1変数(size: IxIxS)
dvar int+ y[I,SB] in 0..1;// 0-1変数(size: IxS)

minimize
  sum(s in SB) sum(i in I) y[i,s];
subject to{
  forall(i in I) {// チームi毎のBreak数を計算
    forall(s in SB) {
      sum(j in I) (x[i,j,s]+x[i,j,s+1]) <= y[i,s]+1;// s→s+1でHomeのBreakがあるか
      sum(j in I) (x[j,i,s]+x[j,i,s+1]) <= y[i,s]+1;// s→s+1でAwayのBreakがあるか
    };
  };
  forall(s in S) {// 各slot において
    forall(i in I) {// 各チーム i は
      sum(j in I:i!=j) (x[i,j,s] + x[j,i,s]) == 1;
    };// 自分以外の全チーム j とHome/Away のどちらかで 丁度1回対戦する
  };
  forall(i in I) {// 各チーム i は
    forall(j in I:i!=j) {// 自分以外の全チーム j と各々
      sum(s in S) (x[i,j,s] + x[j,i,s]) == 1;
    };// どこかのslotで Home/Away のどちらかで 丁度1回対戦する
  };
  forall(i in I) {// 自分とは対戦しないので
    sum(s in S) x[i,i,s] == 0;// その変数を全て0に
  };
};
```

※このモデルでは「距離行列」は必要ないが，データファイルを流用したいので残してある

スポーツ・スケジューリングをCPLEXで解く

➢ 計算結果の確認（例：ex1）

0-1変数y[i,s] のデータ・ビュー

ssha2.mod　y の値

I (サイズ 6)	SB ...ズ 4)			
	1	2	3	4
1	0	0	0	0
2	0	0	1	0
3	0	0	0	0
4	0	0	1	0
5	0	1	0	0
6	0	1	0	0

最適値 = 4
（※Break数）

0-1変数x[i,j,s]について[値=1]で降順ソート後のデータ・ビュー

ssha2.mod　x の値　y の値

I (サイズ 6)	I (サイズ 6)	S (サイズ 5)	値.
6	5	4	1
6	4	1	1
5	4	3	1
5	3	5	1
5	2	2	1
4	2	5	1
4	1	2	1
3	6	2	1
3	4	4	1
2	6	3	1
2	3	1	1
2	1	4	1
1	6	5	1
1	5	1	1
1	3	3	1
6	6	5	0
6	6	4	0

1行目をクリック

[Shift]キーを押しながら
この行[値=1の最終行]
クリック

[値=1]の行（15行=3試合×5slot）が全選択されたので，この状態で [Ctrl]+[c]（コピー）

Excelシートに貼り付け（[Ctrl]+[v]）

結果

対戦表		3	3	3	3	3
	team/slot	1	2	3	4	5
1	A	E	D	C	B	F
2	B	C	E	F	A	D
3	C	B	F	A	D	E
4	D	F	A	E	C	B
5	E	A	B	D	F	C
6	F	D	C	B	E	A

Home-Away table

H	A	team/slot	1	2	3	4	5
3	2	A	H	A	H	A	H
3	2	B	H	A	H	H	A
2	3	C	A	H	A	H	A
2	3	D	A	H	A	A	H
3	2	E	A	H	H	A	H
2	3	F	H	A	A	H	A

Excel上で加工

スポーツ・スケジューリングをgurobiで解く

- 問題（ex1）を python & gurobi で記述（データ生成部分①）

①

```
# coding: Shift_JIS
from gurobipy import *

# ######## 例題設定 ########
def make_data_ 1():
        I = [1,2,3,4,5,6]
        S = [1,2,3,4,5]
        d = {(1,1):0,(1,2):5,(1,3):4,(1,4):6,(1,5):6,(1,6):8,
                    (2,1):5,(2,2):0,(2,3):5,(2,4):9,(2,5):4,(2,6):7,
                    (3,1):4,(3,2):5,(3,3):0,(3,4):4,(3,5):2,(3,6):4,
                    (4,1):6,(4,2):9,(4,3):4,(4,4):0,(4,5):6,(4,6):6,
                    (5,1):6,(5,2):4,(5,3):2,(5,4):6,(5,5):0,(5,6):3,
                    (6,1):8,(6,2):7,(6,3):4,(6,4):6,(6,5):3,(6,6):0,
        } # 距離
        return I,S,d
```

※4種の定式化（srt-di, srt-di2, srt-ha, srt-ha2）で共通で使う

スポーツ・スケジューリングをgurobiで解く

- srt-di定式化（例1）
  総移動距離最小化

1つのファイル「srt-di.py」に
①②③の順に記述して保存

②

```
# ####### 定式化 #######
def srtdi(I,S,d):
    mod = Model("single round-robin tournament:Min.TotalDistance")

    # 変数設定
    x = {}
    for i in I:
        for j in I:
            for s in S:
                if j != i:
                    x[i,j,s] = mod.addVar(vtype="B", name="x(%s,%s,%s)" % (i,j,s))
    mod.update()

    # 制約条件の設定
    for s in S:
        for i in I:
            mod.addConstr(quicksum(x[i,j,s]+x[j,i,s] for j in I if j!=i) == 1)
    for i in I:
        for j in I:
            if j != i:
                mod.addConstr(quicksum(x[i,j,s]+x[j,i,s] for s in S) == 1)

    # 目的関数の設定
    mod.setObjective(quicksum(2*d[i,j]*x[i,j,s] for (i,j,s) in x), GRB.MINIMIZE)
    mod.update()
    mod.__data = x
    return mod
```

③

```
# ####### 実行 #######
if __name__=="__main__":
    I,S,d = make_data_ex1()          # デー
    mod = srtdi(I,S,d)               # モデ
    mod.write("srt-diex1.lp")        # lpフ
    mod.optimize()                   # 最適
    print("¥n optimal value = ", mod.ObjVal
    mod.printAttr('X')               # 最適
    mod.write("srt-diex1.sol")       # 最適解をsolファイルに出力
```

スポーツ・スケジューリングをgurobiで解く

- 実行結果

  総移動距離最小化

```
gurobi> exec(open("srt-di.py").read())
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 10 physical cores, 20 logical processors, using up to 20 threads
Optimize a model with 60 rows, 150 columns and 600 nonzeros
Model fingerprint: 0x7220648d
Variable types: 0 continuous, 150 integer (150 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [4e+00, 2e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 158.0000000
Presolve removed 15 rows and 75 columns
Presolve time: 0.00s
Presolved: 45 rows, 75 columns, 225 nonzeros
Variable types: 0 continuous, 75 integer (75 binary)

Explored 0 nodes (0 simplex iterations) in 0.00 seconds (0.00 work units)
Thread count was 20 (of 20 available processors)

Solution count 1: 158

Optimal solution found (tolerance 1.00e-04)
Best objective 1.580000000000e+02, best bound 1.580000000000e+02, gap 0.0000%
```

```
optimal value =  158.0

    Variable            X
-----------------------
    x(1,2,3)            1
    x(1,5,1)            1
    x(1,6,5)            1
    x(2,4,1)            1
    x(2,5,5)            1
    x(3,1,2)            1
    x(3,2,4)            1
    x(3,4,5)            1
    x(4,1,4)            1
    x(5,3,3)            1
    x(5,4,2)            1
    x(5,6,4)            1
    x(6,2,2)            1
    x(6,3,1)            1
    x(6,4,3)            1
gurobi>
```

スポーツ・スケジューリングをgurobiで解く

- srt-di2定式化（例1）
  移動距離均等化

1つのファイル「srt-di2.py」に
①②③の順に記述して保存

②

```
# ######## 定式化 ########
def srtdi2(I,S,d):
    mod = Model(“single round-robin tournament:Min.Max.Distance")

    # 変数設定
    x,dist = {},{}
    for i in I:
        for j in I:
            for s in S:
                if j != i:
                    x[i,j,s] = mod.addVar(vtype="B", name="x(%s,%s,%s)" % (i,j,s))
    dist = mod.addVar(vtype="C", name="dist")
    mod.update()

    # 制約条件の設定
    for s in S:
        for i in I:
            mod.addConstr(quicksum(x[i,j,s]+x[j,i,s] for j in I if j!=i) == 1)
    for i in I:
        mod.addConstr(quicksum(2*d[i,j]*x[i,j,s] for s in S for j in I if j!=i) <= dist)
        for j in I:
            if j != i:
                mod.addConstr(quicksum(x[i,j,s]+x[j,i,s] for s in S) == 1)

    # 目的関数の設定
    mod.setObjective(dist, GRB.MINIMIZE)
    mod.update()
    mod.__data = x,dist
    return mod
```

③

```
# ######## 実行 ########
if __name__=="__main__":
    I,S,d = make_data_ex1()          # テ
    mod = srtdi2(I,S,d)              # モ
    mod.write(“srt-di2ex1.lp")       # lp
    mod.optimize()                   # 最
    print("¥n optimal value = ", mod.Obj
    mod.printAttr('X')               # 最
    mod.write(“srt-di2ex1.sol")      # 最
```

スポーツ・スケジューリングをgurobiで解く

- 実行結果

  移動距離均等化

```
optimal value =  28.0

   Variable            X
-------------------------
    x(1,2,2)           1
    x(1,6,3)           1
    x(2,3,3)           1
    x(2,4,5)           1
    x(3,1,5)           1
    x(3,4,4)           1
    x(3,5,2)           1
    x(3,6,1)           1
    x(4,1,1)           1
    x(4,5,3)           1
    x(5,1,4)           1
    x(5,2,1)           1
    x(5,6,5)           1
    x(6,2,4)           1
    x(6,4,2)           1
       dist           28
gurobi>
```

```
gurobi> exec(open("srt-di2.py").read())
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 10 physical cores, 20 logical processors, using up to 20 threads
Optimize a model with 66 rows, 151 columns and 756 nonzeros
Model fingerprint: 0x73ad7550
Variable types: 1 continuous, 150 integer (150 binary)
Coefficient statistics:
  Matrix range     [1e+00, 2e+01]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 38.0000000
Presolve removed 15 rows and 0 columns
Presolve time: 0.00s
Presolved: 51 rows, 151 columns, 606 nonzeros
Variable types: 0 continuous, 151 integer (150 binary)

Root relaxation: objective 2.633333e+01, 137 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0   26.33333    0   11   38.00000   26.33333  30.7%     -    0s
H    0     0                       30.0000000   26.33333  12.2%     -    0s
     0     0   28.00000    0   35   30.00000   28.00000  6.67%     -    0s
H    0     0                       28.0000000   28.00000  0.00%     -    0s
     0     0   28.00000    0   35   28.00000   28.00000  0.00%     -    0s

Cutting planes:
  Cover: 1
  MIR: 1
  StrongCG: 1
  RLT: 1

Explored 1 nodes (365 simplex iterations) in 0.03 seconds (0.01 work units)
Thread count was 20 (of 20 available processors)

Solution count 3: 28 30 38

Optimal solution found (tolerance 1.00e-04)
Best objective 2.800000000000e+01, best bound 2.800000000000e+01, gap 0.0000%
```

スポーツ・スケジューリングをgurobiで解く

- srt-ha定式化（例1）

  Home-Away数均等化

1つのファイル「srt-ha.py」に
①②③の順に記述して保存

②

```
# ######## 定式化 ########
def srtha(I,S,d):
    mod = Model("single round-robin tournament:Min.Max.Num.H/A")

    # 変数設定
    x,nha = {},{}
    for i in I:
        for j in I:
            for s in S:
                if j != i:
                    x[i,j,s] = mod.addVar(vtype="B", name="x(%s,%s,%s)" % (i,j,s))
    nha = mod.addVar(vtype="C", name="nha")
    mod.update()

    # 制約条件の設定
    for s in S:
        for i in I:
            mod.addConstr(quicksum(x[i,j,s]+x[j,i,s] for j in I if j!=i) == 1)
    for i in I:
        mod.addConstr(quicksum(x[i,j,s] for s in S for j in I if j!=i) <= nha)
        mod.addConstr(quicksum(x[j,i,s] for s in S for j in I if j!=i) <= nha)
        for j in I:
            if j != i:
                mod.addConstr(quicksum(x[i,j,s]+x[j,i,s] for s in S) == 1)

    # 目的関数の設定
    mod.setObjective(nha, GRB.MINIMIZE)
    mod.update()
    mod.__data = x,nha
    return mod
```

③

```
# ######## 実行 ########
if __name__=="__main__":
    I,S,d = make_data_ex1()          # デー
    mod = srtha(I,S,d)               # モデ
    mod.write("srt-haex1.lp")        # lpフ
    mod.optimize()                   # 最適
    print("¥n optimal value = ", mod.ObjVal
    mod.printAttr('X')               # 最適
    mod.write("srt-haex1.sol")       # 最適
```

スポーツ・スケジューリングをgurobiで解く

- 実行結果

  Home-Away数均等化

```
optimal value =  3.0

   Variable            X
-------------------------
    x(1,2,5)           1
    x(1,4,3)           1
    x(1,6,4)           1
    x(2,4,1)           1
    x(2,5,2)           1
    x(2,6,3)           1
    x(3,1,2)           1
    x(3,2,4)           1
    x(3,5,3)           1
    x(4,3,5)           1
    x(4,6,2)           1
    x(5,1,1)           1
    x(5,4,4)           1
    x(6,3,1)           1
    x(6,5,5)           1
        nha            3
gurobi>
```

```
gurobi> exec(open("srt-ha.py").read())
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 10 physical cores, 20 logical processors, using up to 20 threads
Optimize a model with 72 rows, 151 columns and 912 nonzeros
Model fingerprint: 0x8aea16ec
Variable types: 1 continuous, 150 integer (150 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 4.0000000
Presolve removed 15 rows and 0 columns
Presolve time: 0.00s
Presolved: 57 rows, 151 columns, 762 nonzeros
Variable types: 0 continuous, 151 integer (150 binary)

Root relaxation: objective 2.500000e+00, 87 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    2.50000    0    7    4.00000    2.50000  37.5%     -    0s
H    0     0                       3.0000000    2.50000  16.7%     -    0s
     0     0    2.50000    0    7    3.00000    2.50000  16.7%     -    0s

Explored 1 nodes (248 simplex iterations) in 0.03 seconds (0.00 work units)
Thread count was 20 (of 20 available processors)

Solution count 2: 3 4

Optimal solution found (tolerance 1.00e-04)
Best objective 3.000000000000e+00, best bound 3.000000000000e+00, gap 0.0000%
```

スポーツ・スケジ

- srt-ha定式化（例1）
  Break数最小化

1つのファイル「srt-ha2.py」に
①②③の順に記述して保存

②

```
# ######## 定式化 ########
def srtha2(I,S,d):
    mod = Model(“single round-robin tournament:Min.Break")

    # 変数設定
    x,y = {},{}
    for i in I:
        for j in I:
            for s in S:
                if j != i:
                    x[i,j,s] = mod.addVar(vtype="B", name="x(%s,%s,%s)" % (i,j,s))
        for s in range(1,len(S)):
            y[i,s] = mod.addVar(vtype="B", name="y(%s,%s)" % (i,s))
    mod.update()

    # 制約条件の設定
    for s in S:
        for i in I:
            mod.addConstr(quicksum(x[i,j,s]+x[j,i,s] for j in I if j!=i) == 1)
    for i in I:
        for j in I:
            if j != i:
                mod.addConstr(quicksum(x[i,j,s]+x[j,i,s] for s in S) == 1)
        for s in range(1,len(S)):
            mod.addConstr(quicksum(x[i,j,s]+x[i,j,s+1] for j in I if j!=i) <= y[i,s]+1)
            mod.addConstr(quicksum(x[j,i,s]+x[j,i,s+1] for j in I if j!=i) <= y[i,s]+1)

    # 目的関数の設定
    mod.setObjective(quicksum(y[i,s] for (i,s) in y), GRB.MINIMIZE)
    mod.update()
    mod.__data = x,y
    return mod
```

③

```
# ######## 実行 ########
if __name__=="__main__":
    I,S,d = make_data_ex1()          # デ
    mod = srtha2(I,S,d)              # モ
    mod.write(“srt-ha2ex1.lp")       # lp
    mod.optimize()                   # 最
    print("¥n optimal value = ", mod.Obj
    mod.printAttr('X')               # 最
    mod.write(“srt-ha2ex1.sol")      # 最
```

スポーツ・スケジューリングをgurobiで解く

- 実行結果

  Break数最小化

```
optimal value =  4.0

   Variable            X
-----------------------
    x(1,2,1)           1
    x(1,3,4)           1
    x(1,5,3)           1
      y(1,3)           1
    x(2,3,2)           1
    x(2,6,5)           1
      y(2,3)           1
    x(3,4,1)           1
    x(3,5,5)           1
    x(3,6,3)           1
    x(4,1,5)           1
    x(4,2,3)           1
    x(4,5,2)           1
      y(4,2)           1
    x(5,2,4)           1
    x(5,6,1)           1
      y(5,2)           1
    x(6,1,2)           1
    x(6,4,4)           1
gurobi>
```

```
gurobi> exec(open("srt-ha2.py").read())
Gurobi Optimizer version 9.5.2 build v9.5.2rc0 (win64)
Thread count: 10 physical cores, 20 logical processors, using up to 20 threads
Optimize a model with 108 rows, 174 columns and 1128 nonzeros
Model fingerprint: 0x3ec9245c
Variable types: 0 continuous, 174 integer (174 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 1e+00]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 1e+00]
Found heuristic solution: objective 10.0000000
Presolve removed 15 rows and 0 columns
Presolve time: 0.00s
Presolved: 93 rows, 174 columns, 978 nonzeros
Variable types: 0 continuous, 174 integer (174 binary)

Root relaxation: objective 0.000000e+00, 87 iterations, 0.00 seconds (0.00 work units)

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0    0.00000    0   37   10.00000    0.00000   100%     -    0s
H    0     0                       6.0000000    0.00000   100%     -    0s
     0     0    0.00000    0   60    6.00000    0.00000   100%     -    0s
     0     0    1.50000    0   38    6.00000    1.50000  75.0%     -    0s
     0     0    1.50000    0   38    6.00000    1.50000  75.0%     -    0s
     0     0    1.50000    0   37    6.00000    1.50000  75.0%     -    0s
     0     2    2.00000    0   36    6.00000    2.00000  66.7%     -    0s
*   12     9               3       4.0000000    2.66667  33.3%  15.8    0s

Explored 24 nodes (1109 simplex iterations) in 0.05 seconds (0.03 work units)
Thread count was 20 (of 20 available processors)

Solution count 3: 4 6 10

Optimal solution found (tolerance 1.00e-04)
Best objective 4.000000000000e+00, best bound 4.000000000000e+00, gap 0.0000%
```