2026/7/21 Tue.

問題解決技法入門

6. Scheduling
Sports Scheduling

堀田 敬介

演習

日付(曜日)時限	学籍番号	氏　　名
R8年　　月　　日(　　)　　限	C　　R 1 1	

問)6チーム(A, B, C, D, E, F)の総当たり対戦スケジュールをつくれ

スロット チーム＼	1	2	3	4	5	Home	Away	Break	移動ルート 距離の和	距離 計
A										
B										
C										
D										
E										
F										
							計		max. - min.	

SPORTS SCHEDULING

考慮したい条件：Home-Away

- 対戦は必ず一方がHomeでもう一方がAwayとする

※競技によっては
Home v.s. Home や
Away v.s. Away　もありうる

チーム＼スロット	1	2	3	4	5	Home	Away
A	D						
B	F						
C	E						
D	A						
E	C						
F	B						

※各行（チーム）で〇が付いている箇所では，そのチームがAwayで戦う
（例）slot1 の[A vs. D] は，Aの行に〇があるので，AがAwayでDがHome

SPORTS SCHEDULING

考慮したい条件：Home-Away

- 対戦は必ず一方がHomeでもう一方がAwayとする

※競技によっては
Home v.s. Home や
Away v.s. Away　もありうる

チーム＼スロット	1	2	3	4	5	Home	Away
A	D	E	C	B	F	2	3
B	F			A			
C	E		A				
D	A						
E	C	A					
F	B				A		

※各行（チーム）で〇が付いている箇所では，そのチームがAwayで戦う
（例）slot1 の[A vs. D] は，Aの行に〇があるので，AがAwayでDがHome

SPORTS SCHEDULING

考慮したい条件：Home-Away, Break

※競技によっては
Home v.s. Home や
Away v.s. Away　もありうる

- 対戦は必ず一方がHomeでもう一方がAwayとする
- あるチームのHome-Awayパターンの中に，HH, AA のように，HomeやAwayが2回連続する場合，Breakという

チーム＼スロット	1	2	3	4	5	Break
A	A	A	H	A	H	1
B	H	A	A	H	H	2
C	A	A	A	H	A	2
D	H	H	H	A	H	2
E	H	H	A	H	A	1
F	A	H	H	A	A	2
						10

Home-Away table

【Home-Away table 作成時の注意】

1. 同じパターンのチームは2つ作れない

team A：　HAHAH
team B：　HAHAH

（なぜか？）

2. 各スロットでHとAの数が違ってはダメ

H
A
A
H
H
H

（なぜか？）

- Break合計を最小化
- Breakの偏りを最小化

（※ Home-Away table 注1,2を満たしても，スケジュールが組めるとは限らない）
（※与えられたHome-Away tableでスケジュールができるかどうかの判定はNP困難）

SPORTS SCHEDULING

考慮したい条件：Home-Away, Break

※競技によっては
Home v.s. Home や
Away v.s. Away　もありうる

- 対戦は必ず一方がHomeでもう一方がAwayとする
- あるチームのHome-Awayパターンの中に，HH, AA のように，HomeやAwayが2回連続する場合，Breakという

チーム＼スロット	1	2	3	4	5	Break
A	A	A	H	A	H	1
B	H	A	A	H	H	2
C	A	A	A	H	A	2
D	H	H	H	A	H	2
E	H	H	A	H	A	1
F	A	H	H	A	A	2
						10

Home-Away table

※6チームの場合，
Breakの合計は
4以上20以下

【Break数最小のパターン】
1. HAHAH　(Break 0)
2. AHAHA　(Break 0)

【Break数最大のパターン】
1. HHHHH　(Break 4)
2. AAAAA　(Break 4)

※それぞれ2つしかない

よって

SPORTS SCHEDULING

※coe値の定義は，強豪チームに限ったものではない
※最終日の翌日は初日と定義する

考慮したい条件：coe（carry-over effect）

- 強豪チーム（A）と対戦し疲弊したチーム（B）と次に戦うチームは有利だろう
- ***d*日目 [team *i* v.s. team *k*]，*d*+1日目 [team *j* v.s. team *k*] のとき，「team *i* が team *j* に carry-over effect を与える」と定義

チーム＼スロット	1	2	3
A（強豪）	B	C	D
B	A	D	C
C	D	A	B
D	C	B	A

*1*日目 [A v.s. B]
*2*日目 [D v.s. B]（強豪Aと対戦後でBは疲弊中）
→ A が Dにcoe を与えた（c_{AD}=1）
*2*日目 [A v.s. C]
*3*日目 [B v.s. C]（強豪Aと対戦後でCは疲弊中）
→ A が Bにcoe を与えた（c_{AB}=1）
*3*日目 [A v.s. D]
*1*日目 [C v.s. D]（強豪Aと対戦後でDは疲弊中）
→ A が Cにcoe を与えた（c_{AC}=1）

- coe行列

$$(c_{ij}) = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ & & 0 & \\ & & & 0 \end{bmatrix} \end{array}$$

*1*日目 [B v.s. A] *2*日目 [C v.s. A] →BがCにcoe を与えた（c_{BC}=1）
*2*日目 [B v.s. D] *3*日目 [A v.s. D] →BがAにcoe を与えた（c_{BA}=1）
*3*日目 [B v.s. C] *1*日目 [D v.s. C] →BがDにcoe を与えた（c_{BD}=1）

- coe値 $= \Sigma_{ij} c_{ij}^2$ $\{\geq 2n(2n-1)\}$

※チーム数2*n*とすると，coe値が最小となるのは，
非対角要素が全て1のとき，即ち
$\forall i, j (i \neq j), c_{ij} = 1$のときで2*n*(2*n*-1)　balanced schedule

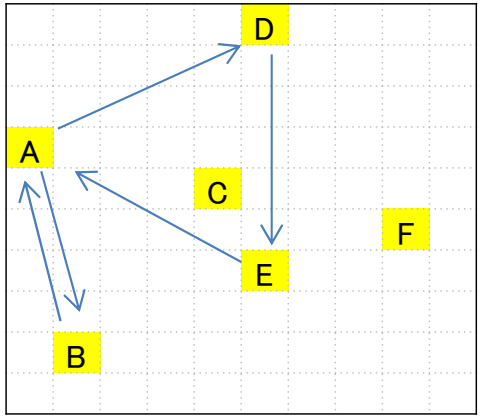
SPORTS SCHEDULING

考慮したい条件：総移動距離 （巡回トーナメントの場合）

- 各チームの移動距離の総和を最小化する

チーム＼スロット	1	2	3	4	5	移動ルート 距離の和	距離 計
A	D	E	C	B	F	AD + DE + EA + AB + BA = 4 + 3 + 3 + 1 + 1	12
B							
C							
D							
E							
F							

team A の移動の様子



2チーム間の距離

	B	C	D	E	F
A	1	2	4	3	4
B		1	5	4	3
C			2	1	2
D				3	5
E					1

注）スロット5にAwayで対戦したチームは最後に本拠地に戻るのを忘れないように

SPORTS SCHEDULING

考慮したい条件：その他

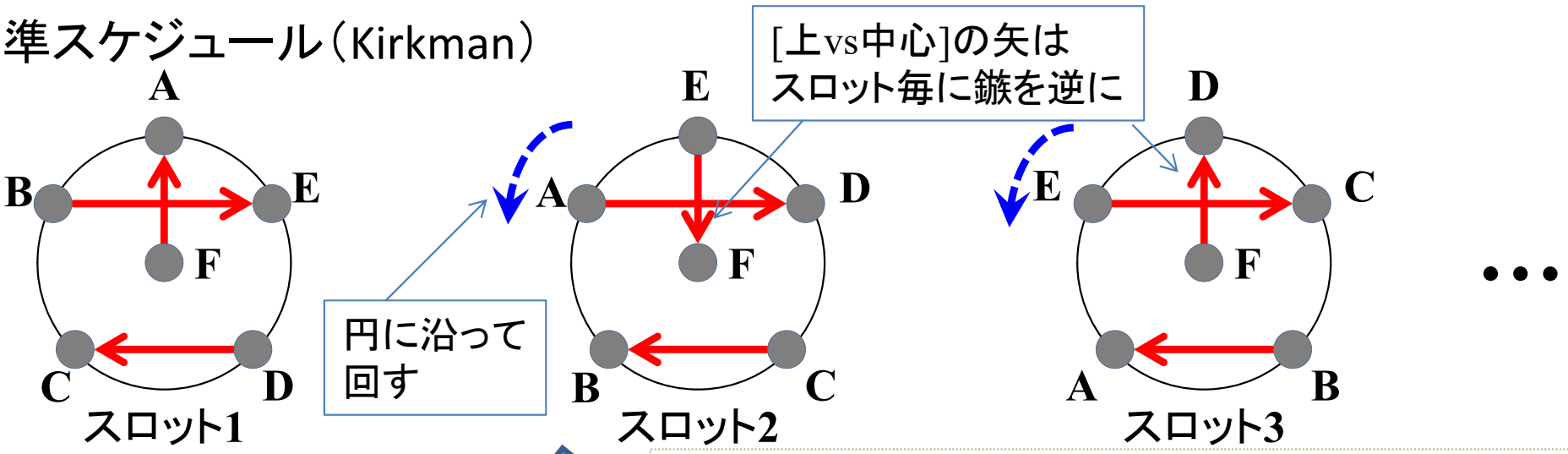
- TV放映権
- 会場（Home/本拠地）の都合
- 次の試合日は連続する日か？　それとも何日か後か？
- 優勝争いは最終日までもつれて欲しい
- 様々な条件における，チーム間の公平性
- etc.

SPORTS SCHEDULING

✓ チーム数は偶数限定でよい
（なぜか？奇数の時はどうする？）

簡便な（1重）総当たりリーグ戦の作り方

- 基準スケジュール（Kirkman）



チーム＼スロット	1	2	3	4	5	B
A	F	D	B	E	C	0
B	E	C	A	D	F	1
C	D	B	E	F	A	1
D	C	A	F	B	E	1
E	B	F	C	A	D	1
F	A	E	D	C	B	0

※表は，背景色付きがAway　＝その行のチームがAwayで戦う
例えば，team A の slot2 は [A(Away) vs. D(Home)]

【基準スケジュールの作り方】

- ✓ スロット1で，1チームを中心に，残りを円周上に配置
- ✓ 中心と上を結び，残りは全て上から順に横線を引く
- ✓ 横線の鏃は上から順に交互にする（全スロット共通）
- ✓ 矢線で結ばれたチームどうしが戦う（鏃側がHome ）
- ✓ スロット2は，図のように円周に沿って全teamを一つ移動させる（以降同じ．中心teamは動かさない）
- ✓ 縦線の鏃は，スロット毎に逆にする（なぜか？）

【基準スケジュールの妥当性と性質[H/A, break]】

- ✓ Fが全teamと丁度1回戦うのは自明．他teamは円周上に奇数team & 左に居る時と右に居る時は円周上team listの一つ飛ばし対戦，円最下段で左→右移行時のみ連続対戦より
- ✓ team FはA/Hが交互に，teamAはH/Aが交互になるのは自明．それ以外のteamはH/A交互＆break1回(HH or AA)になる

SPORTS SCHEDULING

例）6チームの場合

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			

（1重）総当たり戦スケジューリングの最適化モデル例

- single round robin tournament problem
  - チーム数 $|T|$ は偶数とし，全チームと1回対戦
  - 全試合 Home vs Away で戦う

※$|T|$偶数として一般性を失わない

- 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 の対戦を i のHome で実施する
    ＝0 … slot s で i vs j の対戦をしない
  - 定式化：制約条件（1重総当たりリーグとなる条件式）

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

← どのteam j も，各slot s で，H/Aどちらかで，自分以外のどこか他のteam i と1回戦う

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

← どの2team i, j も，全slot s のどこかで，H/Aどちらかで，丁度1回戦う

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

SPORTS SCHEDULING

- 演習）12チーム{a, b, c,…, l} の1重総当たり戦スケジュール

- Google colaboratory で整数計画ソルバー gurobi を利用して解く
  - Google アカウントにログインし，google drive へ移動する
  - [新規]ー[その他]ー[Google colaboratory] を選択

制限付き
無償利用

  - [コード]欄に以下を記述し，実行（▸クリック）して gurobi をインストール

```
▶ !pip install gurobipy
```

    - ✓ インストールが終了する（以下のメッセージが出る）まで静かに待つ

```
...
Successfully installed gurobipy-13.0.2
```

    - ✓ 既にインストール済みの場合は，以下のメッセージが出る

```
Requirement already satisfied:...
```

  - 次に[+コード] をクリックして [コード]欄を追加し，次ページの python コードを入力して，実行（▸クリック）する

SPORTS SCHEDULING

➢ 演習）12チーム{a, b, c,…, l} の1重総当たり戦スケジュール

```
Team = ['a','b','c','d','e','f','g','h','i','j','k','l'] # チーム
I,J,S = range(len(Team)),range(len(Team)),range(len(Team)-1)

from gurobipy import *
md = Model("SportsScheduling") # モデル設定:スポーツ・スケジューリング
x = {}
for i in I:
  for j in J:
    for s in S:
      x[i,j,s] = md.addVar(vtype="B") # 0-1変数 x[i,j,s] =1:Slot s で [H]i vs j[A], =0:対戦せず
for s in S:          # どのslotにおいても…
  for i in I:        # どのTeam i も…
    md.addConstr(x[i,i,s] == 0)  # 制約0:自チームとは対戦しない
    md.addConstr(quicksum(x[i,j,s] + x[j,i,s] for j in J if i!=j) == 1) # 制約1:Team j(≠i)と対戦
for i in I:          # Team i について…
  md.addConstr(quicksum(x[i,j,s] for j in J for s in S) <= len(Team)*(2/3)) # 制約2:HomeGameは2/3
  for j in J:        # Team jと…
    if j!=i:         # ただし Team i とは異なる Team j について
      md.addConstr(quicksum(x[i,j,s] + x[j,i,s] for s in S) == 1) # 制約3:どこかのslotで i vs j 1回
md.setObjective(quicksum(x[0,j,s] for j in J for s in S), GRB.MAXIMIZE) # 目的:チーム'a'のHG数最大

md.optimize()      # 最適化実行

if md.Status == GRB.OPTIMAL:
  print("最適解:")   # 最適解を表示
  for s in S:
    print("Slot %s:" % (s+1), end=" ")
    for i in I:
      for j in J:
        if x[i,j,s].X == 1:
          print("[H]%s vs %s[A], " % (Team[i],Team[j]), end=" ")
    print()
```

注）# とその右側の緑色の文字は，人間用のコメント（後で見た時に何が書かれているか理解を助けるために残す文字）なので，記述する必要はない

目的は「チーム'a'のHomeGame数を最大にする」スケジュールを求めることにしている

SPORTS SCHEDULING

- 演習）12チーム{a, b, c,…, l}の1重総当たり戦スケジュール
  - 実行結果：「目的関数＝チームaのHomeGame数を最大に」の結果

```
Gurobi Optimizer version 13.0.2 build v13.0.2rc1 (linux64 - "Ubuntu 22.04.5 LTS")

・・・(中略)・・・

Optimal solution found (tolerance 1.00e-04)
Best objective 8.000000000000e+00, best bound 8.000000000000e+00, gap 0.0000%
最適解：
Slot 1: [H]a vs b[A],  [H]c vs k[A],  [H]d vs l[A],  [H]e vs g[A],  [H]f vs h[A],  [H]i vs j[A],
Slot 2: [H]a vs l[A],  [H]b vs k[A],  [H]d vs g[A],  [H]e vs c[A],  [H]f vs j[A],  [H]h vs i[A],
Slot 3: [H]a vs g[A],  [H]c vs h[A],  [H]e vs j[A],  [H]f vs k[A],  [H]i vs d[A],  [H]l vs b[A],
Slot 4: [H]b vs g[A],  [H]c vs d[A],  [H]f vs l[A],  [H]i vs e[A],  [H]j vs h[A],  [H]k vs a[A],
Slot 5: [H]a vs h[A],  [H]b vs e[A],  [H]f vs d[A],  [H]g vs j[A],  [H]k vs i[A],  [H]l vs c[A],
Slot 6: [H]a vs f[A],  [H]c vs g[A],  [H]i vs b[A],  [H]j vs d[A],  [H]k vs e[A],  [H]l vs h[A],
Slot 7: [H]a vs j[A],  [H]c vs b[A],  [H]d vs e[A],  [H]f vs g[A],  [H]h vs k[A],  [H]i vs l[A],
Slot 8: [H]b vs f[A],  [H]e vs a[A],  [H]h vs d[A],  [H]i vs c[A],  [H]j vs k[A],  [H]l vs g[A],
Slot 9: [H]c vs f[A],  [H]d vs k[A],  [H]e vs l[A],  [H]h vs g[A],  [H]i vs a[A],  [H]j vs b[A],
Slot 10: [H]a vs c[A],  [H]b vs d[A],  [H]e vs h[A],  [H]i vs f[A],  [H]j vs l[A],  [H]k vs g[A],
Slot 11: [H]a vs d[A],  [H]b vs h[A],  [H]e vs f[A],  [H]g vs i[A],  [H]j vs c[A],  [H]l vs k[A],
```

  - 目的関数（mod.setObjective(quicksum(x[0,j,s]…), GRB.MAXIMIZE) の行）内にある x[0,j,s] の 0 の部分を，1～11のどれかに変更して解き直す（実行し直す）と，そのチームのHomeGame数を最大にした最適解を得られる
    - 例1）x[3,j,s] として実行すれば，チーム’d’のHomeGame数を最大にした最適解が得られる
    - 例2）x[7,j,s] として実行すれば，チーム’h’のHomeGame数を最大にした最適解が得られる

参考文献

- R.V. Rasmussen, M.A. Trick, ``Round robin scheduling –a survey,'' European Journal of Operational Research 188 (2008) 617-636.
- R. Bao, ``Time relaxed round robin tournament and the NBA scheduling problem,'' Cleveland State University, Ph.D Thesis (2009).

- 松井知己,「スポーツスケジューリング　～トーナメント表作成問題における組合せ論」
- 宮代隆平, 松井知己,「スポーツスケジューリング ～未解決問題を中心に」オペレーションズ・リサーチ Vol.50, no.2 (2005) 119-124.

- 早野大介,「スポーツの試合日程が勝敗に及ぼす影響についての一考察 ～NBAを例として」文教大学 情報学部 卒業論文 (2013).

- 堀田敬介,「手を動かして学ぶ はじめてのオペレーションズ・リサーチ」近代科学社 (2025)