

# Ein Benders Ansatz zur Sportligaplanung

Seminar Complex Scheduling

Vortrag gehalten von  
Ingo Holzenkamp



## Formulierung des Problems

- Betrachtung von double round robin tournaments
  - 2n Mannschaften
  - Jeder gegen Jeden einmal Heim- und Auswärts
  - Ein Spiel pro Spieltag
- Für jedes Team wird ein home-away Pattern eingeführt
  - Array bestehend aus 0 und 1
  - Ist Eintrag an Position t eine 0, so spielt man auswärts
- Die Menge aller Patterns wird als Pattern Set bezeichnet
  - Ist zulässig, wenn entsprechender timetable existiert
- Timetable
  - ist eine Matrix, Reihe für jedes Team, Spalte für jeden Spieltag
  - Eintrag (i,s) liefert den Gegner von i am Spieltag s



## Überblick

- Formulierung des Problems
- Die Methodik
- Das Erstellen von Patterns
- Die Pattern Sets
- Überprüfung der Gültigkeit und Benders Cuts
- Der Algorithmus
- Ergebnisse mit dem Computer
- Das constant Traveling Tournament Problem
- Zusammenfassung



## Formulierung des Problems

- Alternierender Wechsel zwischen Heim- und Auswärtsspielen erwünscht
- Nicht immer möglich
- Zwei Heim- bzw. Auswärtsspiele bezeichnet man als Break
- In Tabelle 2.1 sind diese unterstrichen
- Guter Spielplan minimiert die Anzahl an Breaks
- Aufeinanderfolgende Breaks sind unerwünscht
- Zulässiger Plan enthält höchstens zwei Patterns ohne Breaks

Slot	1	2	3	4	5	6	7	8	9	10
$p_1$	0	1	0	1	0	1	0	1	0	1
$p_2$	0	1	<u>1</u>	0	1	<u>1</u>	0	<u>0</u>	1	0
$p_3$	1	0	1	<u>1</u>	0	<u>0</u>	1	0	<u>0</u>	1
$p_4$	0	1	0	<u>0</u>	1	<u>1</u>	0	1	<u>1</u>	0
$p_5$	1	0	1	<u>0</u>	1	0	1	0	1	0
$p_6$	1	0	<u>0</u>	1	0	<u>0</u>	1	<u>1</u>	0	1

Tabelle 2.1: Home away pattern set

Slot	1	2	3	4	5	6	7	8	9	10
team1	6	3	5	2	4	6	3	5	2	4
team2	5	6	4	1	3	5	6	4	1	3
team3	4	1	6	5	2	4	1	6	5	2
team4	3	5	2	6	1	3	5	2	6	1
team5	2	4	1	3	6	2	4	1	3	6
team6	1	2	3	4	5	1	2	3	4	5

Tabelle 2.2: entsprechender timetable





## Formulierung des Problems

- Der Spielplan sollte darüber hinaus auch spezielle Wünsche der einzelnen Teams realisieren
- Teams möchten an best. Tagen unbedingt zu Hause spielen
  - Werder Bremen und Werder Bremen 2 teilen sich das Weserstadion
  - Besondere Ereignisse (zB. Freimarkt) sollen beachtet werden
- Einführung von place constraints
  - Besagen, ein Team an best. Tagen heim- bzw. auswärts spielen muss
  - Werden bei der kanonische-1-Faktorisierung allgemein nicht betrachtet
- Ein Spielplan sollte für ein mirrored tournament gespiegelt sein
  - Spiel am Tag t ergibt Rückspiel am Tag t+(2n-1) im anderen Stadion
  - Vorteil: Abstand dieser beiden Spiele ist maximal
  - Nachteil: place constraints lassen sich nicht unbedingt verwirklichen



Ein Benders Ansatz zur Sportligaplanung

5



## Formulierung des Problems

- Non mirrored tournaments können die place constraints besser realisieren
  - Nachteil: Neue Bedingung für den Mindestabstand k von zwei Spielen mit den selben Gegnern muss eingeführt werden
- Für die Formulierung setzen wir voraus
  - 2n Mannschaften, die jeweils an 4n-2 Spieltage einmal spielen
  - place constraints
  - Keine aufeinander folgende Breaks
  - Mirrored bzw. non mirrored tournaments (mit Abstandsbedingung)
  - T Menge der teilnehmenden Teams, S Menge der Spieltage
  - $I_i^0$  und  $I_i^1$  enthalten jew. die Spieltage, an denen i aufgrund der place constraints heim- bzw. auswärts spielen muss



Ein Benders Ansatz zur Sportligaplanung

6



## Formulierung des Problems

Nun lassen sich folgende Variable einführen:

$$h_i^s = \begin{cases} 1 & , \text{wenn Team } i \text{ am Spieltag } s \text{ heimwärts spielt} \\ 0 & , \text{sonst} \end{cases}$$

$$b_i^s = \begin{cases} 1 & , \text{wenn Team } i \text{ am Spieltag } s \text{ ein Break hat} \\ 0 & , \text{sonst} \end{cases}$$

$x_{ij}$  enthält den Spieltag, an dem Team i heimwärts gegen Team j spielt



Ein Benders Ansatz zur Sportligaplanung

7



## Formulierung des Problems

Das eigentliche Problem besteht in der Minimierung der Breaks

$$\min \sum_{i \in T} \sum_{s \in S} b_{is}$$

mit den zusätzlichen Bedingungen

$$\text{sequence}(1, 2, 3, \text{all}(s \in S) h_{is}, 1, 2n-1), i \in T$$

- erfordert genau 2n-1 Heimspiele
- keine Entstehung von aufeinander folgenden Breaks
- $(b_i^s = 1) \Leftrightarrow (h_{i, s-1} = h_{is}), i \in T, s \in S \setminus \{1\}$
- definiert die Breaks



Ein Benders Ansatz zur Sportligaplanung

8





## Formulierung des Problems

$$b_{i1} = 0, i \in T$$

- Am ersten Spieltag sind keine Breaks erlaubt

$$\sum_{i \in T} h_{is} = n, s \in S$$

- An jedem Spieltag s müssen genau n Teams Heimspiele absolvieren

$$h_{is} = 1, i \in T, s \in I_i^1$$

$$h_{is} = 0, i \in T, s \in I_i^0$$

- Einhaltung der place constraints

$$(h_{is} = 0) \vee (h_{js} = 1) \Rightarrow (x_{ij} \neq s), i, j \in T, i \neq j$$

- i spielt nicht heimwärts gegen j, wenn i auswärts und j heimwärts spielt



## Formulierung des Problems

$$\text{alldifferent}(\text{all}(j \in T \setminus i)x_{ij}, \text{all}(j \in T \setminus i)x_{ji}), i \in T$$

- Team i darf an einem Spieltag s nur einmal spielen

$$(x_{ij} - x_{ji} < -k) \text{ oder } (x_{ij} - x_{ji} > k), i, j \in T, i < j$$

- Zwischen zwei Spielen mit den selben Gegnern müssen k Spieltage liegen



## Formulierung des Problems

$$\min \sum_{i \in T} \sum_{s \in S} b_{is}$$

$$\text{sequence}(1, 2, 3, \text{all}(s \in S)h_{is}, 1, 2n - 1), i \in T$$

$$(b_i^s = 1) \Leftrightarrow (h_{i, s-1} = h_{is}), i \in T, s \in S \setminus \{1\}$$

$$b_{i1} = 0, i \in T$$

$$\sum_{i \in T} h_{is} = n, s \in S$$

$$h_{is} = 1, i \in T, s \in I_i^1$$

$$h_{is} = 0, i \in T, s \in I_i^0$$

$$(h_{is} = 0) \vee (h_{js} = 1) \Rightarrow (x_{ij} \neq s), i, j \in T, i \neq j$$

$$\text{alldifferent}(\text{all}(j \in T \setminus i)x_{ij}, \text{all}(j \in T \setminus i)x_{ji}), i \in T$$

$$(x_{ij} - x_{ji} < -k) \text{ oder } (x_{ij} - x_{ji} > k), i, j \in T, i < j$$



## Die Methodik

Der Ansatz von Benders zerlegt das Problem in die vier Komponenten

- Generiere gültige home-away Patterns
- Suche nach Pattern Sets, die sich durch die home-away Pattern ergeben
- Überprüfe, ob die Pattern Sets gültig sind
- Finde einen Spielplan und die Ausrichtung der Teams

Der Algorithmus wechselt zwischen den vier Teilproblemen

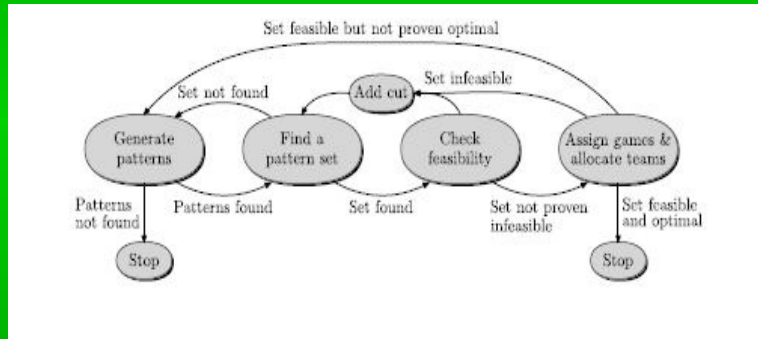
Erhält man ungültige Pattern Sets, so werden dem Problem zusätzlich Benders Cuts hinzugefügt





# Die Methodik

Der Algorithmus lässt sich graphisch auch wie folgt darstellen:



# Das Generieren der Patterns

Zur Erstellung der Patterns wird ein CP Modell verwendet.

Die Patterns sollen

- c Breaks enthalten
- die vorherige sequence Bedingung erfüllen
- eine 0 an der ersten Stelle besitzen

Einführung der neuen Variable

$$h_s = \begin{cases} 1 & \text{, wenn das Pattern an der Stelle } s \text{ eine } 1 \text{ enthält} \\ 0 & \text{, sonst} \end{cases}$$



# Das Generieren der Patterns

Das CP Modell lässt sich nun folgendermaßen beschreiben

$$sequence(1,2,3,all(s \in S)h_s,1,2n-1),i \in T$$

$$h_1 = 0$$

$$\sum_{s=1}^{|S|-1} (h_s = h_{s+1}) = c$$

$$h_s + h_{s+2n-1} = 1, \forall s \in \{1, \dots, 2n-1\}$$

$$h_s \in \{0,1\}, s \in S$$



# Die Pattern Sets

Um aus den erstellten Patterns ein gültiges Pattern Set, welches eine Teilmenge der erstellten Patterns besitzt, zu bekommen, muss die Bedingung, dass an jedem Spieltag n Teams Heimspiele absolvieren, erfüllt sein.

Man will jetzt Patterns Sets finden, bei denen

- Allen Teams mindestens ein Pattern zugeordnet werden kann
- Jedes Team ein unterschiedliches Pattern erhält
- Die place constraints eingehalten werden

Einführung der Variablen LB (Einsparung von Rechenzeit) und UB (Überprüfung der Optimalität)





## Die Pattern Sets

Weitere benötigte Variablen:

$P$  Menge der erstellten Patterns

$p \in P$  mit  $p_j = \begin{cases} 1 & \text{, wenn Pattern } j \text{ in der Teilmenge enthalten ist} \\ 0 & \text{,sonst} \end{cases}$

$c_j$  enthält die Anzahl der Breaks von Pattern  $j$

$h_{js} = \begin{cases} 1 & \text{, wenn Pattern } j \text{ am Spieltag } s \text{ ein Heimspiel hat} \\ 0 & \text{,sonst} \end{cases}$

$P_i = \{ j \in P \mid h_{js} = 1 \forall s \in I_i^1 \wedge h_{js} = 0 \forall s \in I_i^0 \}$  zum Abspeichern der

Patterns, die den place constraints von Team  $i$  genügen

Ein Benders Ansatz zur Sportligaplanung

17



## Die Pattern Sets

- Verwendung eines IP Modells

$$\min \sum_{j \in P} c_j p_j$$

$$\sum_{j \in P} p_j = 2n$$

genau  $2n$  Teams in Teilmenge enthalten

$$\sum_{j \in P} h_{js} p_j = n, s \in S$$

an jedem Spieltag genau  $n$  Heimspiele

$$\sum_{j \in P_i} p_j \geq 1, i \in T$$

alle Teams können einem Pattern zugewiesen w

$$\sum_{j \in P} c_j p_j \geq LB$$

steuern die Anzahl an Breaks

$$\sum_{j \in P} c_j p_j \leq UB$$

$$p_j \in \{0, 1\}, j \in P$$

Ein Benders Ansatz zur Sportligaplanung

18



## Überprüfung der Gültigkeit und Benders Cuts

Falls man nun alle Spiele in Spieltage einteilen und allen Teams ein Pattern zuweisen kann, so ist das Pattern Set gültig

Anderenfalls werden durch Benders Cuts die aktuelle und die dazu ähnlichen Lösungen verworfen

Benders Cuts erhält man aus zweifachen Schlussfolgerungen (inference duals)

Wenn ein Teilproblem lösbar ist, dann besagt inference duals, dass das gesamte Problem nicht lösbar ist

Damit werden ungültige Lösungen aus dem zu bearbeiteten Lösungsraum geworfen

Ein Benders Ansatz zur Sportligaplanung

19



## Überprüfung der Gültigkeit und Benders Cuts

**Zuweisung der Teams zu den Patterns des Pattern Sets  $P^C$**

- Sei  $G(A, B)$  ein bipartiter Graph
- Knoten  $i$  aus  $A$  und  $j$  aus  $B$  sollen verbunden werden, wenn
  - Team  $j$  mit den place constraints von Team  $i$  vereinbar ist
  - Für Team  $j$  ein passendes Pattern in  $P^C$  enthalten ist
 Also wenn  $j \in P_i \cap P^C$  gilt
- Das Zuweisen der Teams entspricht dann der Zuordnung der beiden Knotenmengen  $A$  und  $B$
- Das Theorem von Hall gibt dafür eine notwendige und hinreichende Bedingung

Ein Benders Ansatz zur Sportligaplanung

20





# Überprüfung der Gültigkeit und Benders Cuts



Hall's Theorem: Sei  $G=(A,B)$  ein bipartiter Graph, dann existiert eine Zuordnung von A und B genau dann, wenn

$$|\Gamma(X)| \geq |X|, \forall X \subseteq A$$

- In diesem Fall ist  $\Gamma(X) = \bigcup_{i \in X} (P_i \cap P^c)$
- Durch die ungarische Methode zum Lösen des Zuordnungsproblems wird eine Menge X an Teams gefunden, die das Theorem von Hall nicht erfüllen.
- Ist X nicht leer, so kann der Benders Cut  $\sum_{j \in P_i} p_j \geq |X|$  benutzt werden
- Ist X leer, so hat man eine gültige Lösung dieses Problems gefunden



# Überprüfung der Gültigkeit und Benders Cuts



## Das Erhalten unterschiedlicher Patterns

- Die Verwendung von ähnlichen bzw. fast gleichen Patterns gestaltet sich für die Erstellung eines Spielplans schwierig
- Miyashiro entwickelte die notwendige Bedingung

$$\sum_{s \in S} \left( \min \left\{ \sum_{j \in \bar{P}} h_{js}, \sum_{j \in \bar{P}} (1 - h_{js}) \right\} \right) \geq |\bar{P}| (|\bar{P}| - 1)$$

welche für alle Teilmengen  $\bar{P}$  eines beliebigen Pattern Sets erfüllt sein muss

Die maximale Anzahl an Spielen, die durch die Patterns aus  $\bar{P}$  pro Spieltag gespielt werden können, muss mindestens der Anzahl an eigentlichen Spielen entsprechen



# Überprüfung der Gültigkeit und Benders Cuts



## Das Erhalten unterschiedlicher Patterns

- Es lässt sich ein IP Modell entwerfen:
  - Für eine gegebene Anzahl an Teilmengenelementen wird eine Teilmenge an Teams gefunden, die die linke Seite minimiert
  - Somit muss nicht jede Teilmenge überprüft werden
- Einführung der Variablen

$$\alpha_j = \begin{cases} 1 & \text{,wenn Pattern } j \text{ in der Teilmenge enthalten ist} \\ 0 & \text{,sonst} \end{cases}$$

$$\delta_s = \begin{cases} 1 & \text{,Anzahl an Heimspielen } \beta_s \text{ wird gezählt} \\ 0 & \text{,Anzahl an Auswärtsspielen } \beta_s \text{ wird gezählt} \end{cases}$$



# Überprüfung der Gültigkeit und Benders Cuts



## Das Erhalten unterschiedlicher Patterns

$$\min \sum_{s \in S} \beta_s$$
$$\sum_{j \in P^c} \alpha_j = Z \quad \text{genau } Z \text{ Patterns werden benutzt}$$

$$\beta_s - \sum_{j \in P^c} h_{js} \alpha_j + Z(1 - \delta_s) \geq 0, s \in S$$

$$\beta_s - \sum_{j \in P^c} (1 - h_{js}) \alpha_j + Z\delta_s \geq 0, s \in S$$

$$\alpha_j, \delta_s \in \{0, 1\}, j \in P^c, s \in S$$

$$\beta_s \in \mathbb{Z}_+, s \in S$$

die Anzahl an Heim- bzw. Auswärtsspielen darf nicht grösser als  $\beta_s$  sein





# Überprüfung der Gültigkeit und Benders Cuts



## Das Erhalten unterschiedlicher Patterns

- Mit Hilfe dieses Problems kann eine notwendige Bedingung (pattern diversity condition) für die Gültigkeit der Pattern Sets formuliert werden:

Für ein  $P^C$  und eine Anzahl  $Z$  an Teilmengen ist  $P^C$  genau dann gültig, wenn die optimale Lösung für die Erhaltung unterschiedlicher Patterns nicht kleiner als  $Z(Z-1)$  ist

Ist die Bedingung nicht erfüllt, so kann der Benders Cut  $\sum_{j \in P, a_j=1} p_j \leq Z - 1$  benutzt werden

Dies lässt sich auch auf non-mirrored tournaments anwenden



# Überprüfung der Gültigkeit und Benders Cuts



## Trennung der Spiele

- Für die Erstellung eines non-mirrored tournaments mit einem Parameter  $k$  lassen sich zwei weitere notwendige Bedingungen

Teilmengen mit zwei Patterns:

- Für  $k > 0$  kann es sein, dass durch zwei Patterns die geforderten Spiele nicht realisierbar sind

## The pair separation condition:

Seien  $P^C$  und zwei Pattern  $i$  und  $j$  gegeben. Dann ist das Pattern Set genau dann gültig, wenn es die Bedingung

$$(s_{ij}^l - s_{ji}^f > k) \vee (s_{ji}^l - s_{ij}^f > k) \quad S_{ij}^f \text{ Enthält den erstmöglichen Spieltag für das Spiel } i : j$$

erfüllt. Anderenfalls wird der Benders Cut  $p_i + p_j \leq 1$  benutzt.



# Überprüfung der Gültigkeit und Benders Cuts



## Trennung der Spiele

Beispiel:

	$s_{21}^f$	$s_{12}^f$	$s_{12}^l$	$s_{21}^l$						
1	1	0	1	1	0	0	1	0	1	0
2	1	1	0	0	1	0	1	0	1	0
Slot	1	2	3	4	5	6	7	8	9	10

Für  $k > 1$  ergibt sich anhand der pair separation condition

$$s_{12}^l - s_{21}^f = 4 - 2 = 2 \leq k \quad s_{21}^l - s_{12}^f = 5 - 3 = 2 \leq k$$

und somit ein ungültiges Pattern Set für alle  $k > 1$



# Überprüfung der Gültigkeit und Benders Cuts



## Trennung der Spiele

Einführung des CP Modells (GAM), um zu überprüfen, ob sich durch die Teilmenge  $\bar{P}$  (mit mehr als zwei Patterns) die erforderte Anzahl an Spielen realisieren lässt

$$(h_{is} = 0) \vee (h_{js} = 1) \Rightarrow (x_{ij} \neq s), i, j \in P, i \neq j, s \in S$$

$$\text{alldifferent}(\text{all}(j \in \bar{P} \setminus i)x_{ij}, \text{all}(j \in \bar{P} \setminus i)x_{ji}), i \in \bar{P}$$

$$(x_{ij} - x_{ji} < -k) \vee (x_{ij} - x_{ji} > k), i, j \in \bar{P}, i < j$$

$$x_{ij} \in S, i, j \in \bar{P}, i \neq j$$





# Überprüfung der Gültigkeit und Benders Cuts



## Trennung der Spiele

### The multiple pattern separation condition:

Sei  $P^C$  gegeben und  $\bar{P}$  eine daraus entnommene Teilmenge.  
 $P^C$  ist genau dann gültig, wenn  $\bar{P}$  laut der Formulierung (GAM) eine gültige Lösung ist.

Andernfalls wird der Benders Cut

$$\sum_{j \in \bar{P}} p_j \leq |\bar{P}| - 1$$

benutzt.

Die Anzahl an Teilmengen mit wachsender Kardinalität steigt exponentiell

- Einführung einer oberen Grenze maxCard\_GAM



# Überprüfung der Gültigkeit und Benders Cuts



## Die Spielzuweisungen

- Hat man ein Pattern Set gefunden, dass alle notwendige Bedingungen erfüllt
  - So hat man schon eine Zuweisung der Teams zu den Patterns
  - eine gültige Spielzuweisung ist aber nicht unbedingt gegeben
- Dieses lässt sich mit Hilfe der in (GAM) formulierten Bedingungen überprüfen
- Ist (GAM) nicht erfüllt, so wird wie bei Trennen der Spiele der Benders Cut

$$\sum_{j \in \bar{P}} p_j \leq |\bar{P}| - 1$$

benutzt



# Die Algorithmen



```
procedure Generate Patterns
1  if(c > 2n-1)
2  Stop
3  else
4  Find all solutions to (PSM)
5  Update nbPatterns
6  Let c = c+1
7  if (nbPatterns < 2n) then
8  Generate Patterns
9  else
10 Find Pattern Set
end procedure
```

Algorithmus 7.1 : Methode *GeneratePatterns* zum Generieren der Patterns



# Die Algorithmen



```
procedure Find Pattern Set
1  Solve (PSM)
2  if ((PSM) is feasible) then
3  Update P-C
4  Update LB
5  Check Feasibility
6  else
7  Generate Patterns
end procedure
```

Algorithmus 7.2 : Methode *FindPatternSet* sucht nach gültigen Pattern Sets





# Die Algorithmen

```

procedure Check Feasibility
1  cutAdded = 0; card = 2
2  Use the Hungarian method
3  if (for one X in T: Gamma(X) < X) then
4    Add (6.1) to (PSM), cutAdded = 1
5  if (cutAdded = 0) then
6    for all (i,j in P^C: i<j) do
7      if((6.10) is violated) then
8        Add (6.10) to (PSM), cutAdded = 1
9  while ((card < maxCard_GAM) & (cutAdded = 0)) do
10   card = card+1
11   for all (subsets P' in P^C: |P'| = card) do
12     if(cutAdded = 0) then
13       Solve (GAM) for P'
14       if ((GAM) is infeasible) then
15         Add (6.16) to (PSM), cutAdded = 1
16   if (cutAdded = 0) then
17     Find Timetable
18   else
19     Find Pattern Set
end procedure

```

Algorithmus 7.3 : Methode *CheckFeasibility* zum Überprüfen der Gültigkeit von Pattern Sets



# Die Algorithmen

```

procedure Find Timetable
1  Solve (GAM) for P^C
2  if ((GAM) is infeasible) then
3    Add (6.16) to (PSM)
4    Find Pattern Set
5  else
6    Let UB = LB - 2
7    Let LB = 2n - 3 + c
8    if (c <= UB - 2n + 3) then
9      for all (i in {c,..., max{2n - 1, UB - 2n + 3}})
10       Generate Patterns
11       Let c = 2n
12       Find Pattern Set
13   else
14     Stop, optimal solution found
end procedure

```

Algorithmus 7.4 : Methode *FindTimetable* zur Bestimmung einer optimalen Lösung



# Die Ergebnisse mit dem Computer

- Der beschriebene Algorithmus TGBA wird auf verschiedene Instanzen angewandt
  - Mirrored bzw non mirrored Instanzen
  - Instanzen mit und ohne place constraints
- Vergleich mit TPA (älterer Algorithmus von Trick, Nemhauser, Henz)
- Benutzt wurde ein Pentium 4 Prozessor mit 512 MB RAM und die ILOG OPL Studio Plattform zum Lösen der IP und CP Probleme
- Methoden CPLEX und Solver stehen dazu zur Verfügung
- Es wurde ein Zeitlimit von 1800 Sekunden vereinbart



# Die Ergebnisse mit dem Computer

Instance	Breaks	TPA	PGBA
np-mi-n04*	-	0.00	0.02
np-mi-n08	18	0.02	0.03
np-mi-n12	30	0.09	0.08
np-mi-n16	42	3.27	0.31
np-mi-n20	54	88.14	1.34
np-mi-n30	84	-	2.33
np-mi-n40	114	-	-
np-nm-k0-n04	2	0.02	0.02
np-nm-k0-n08	6	0.28	0.17
np-nm-k0-n12	10	29.78	1.41
np-nm-k0-n16	14	-	1.70
np-nm-k0-n28	26	-	129.00
np-nm-k0-n30	?	-	-
np-nm-k2-n04*	-	0.06	0.09
np-nm-k2-n06	10	555.11	15.14
np-nm-k2-n08	8	19.61	0.55
np-nm-k2-n10	10	91.03	0.89
np-nm-k2-n12	12	-	0.92
np-nm-k2-n20	20	-	14.75
np-nm-k2-n22	?	-	-

Instance	# Feasible Instances	# Solved by PGBA	# Solved by TPA	Avg. time	Avg. time
				TPA	PGBA
pl-mi-n12-p05	10	10	10	0.31	0.19
pl-mi-n12-p15	09	10	10	0.65	0.32
pl-mi-n12-p30	05	09	10	3.63	0.71
pl-mi-n16-p05	10	10	10	6.83	0.37
pl-mi-n16-p15	09	10	10	137.51	2.15
pl-mi-n16-p20	09	10	10	33.72	24.96
pl-mi-n16-p30	09	09	10	215.67	8.11
pl-nm-k0-n08-p05	10	10	10	0.14	0.20
pl-nm-k0-n08-p20	05	08	10	5.34	0.83
pl-nm-k0-n08-p30	03	10	10	8.65	0.77
pl-nm-k2-n08-p05	10	10	10	7.03	0.29
pl-nm-k2-n08-p15	08	09	10	66.59	0.86
pl-nm-k2-n08-p25	07	05	10	16.16	2.19
pl-nm-k2-n08-p30	03	10	10	69.79	1.07





## The distance Traveling Tournament Problem (TTP)

- Eingeführt von Easton, Nemhauser und Trick
- Ursprung in der Major Baseball League

Problem besteht in der Minimierung der Reisedistanz aller Teams

Eingabe:

- Die Anzahl an teilnehmenden Teams  $n$
- Eine  $n \times n$  Matrix  $D$  für die Entfernungen;  $D_{ij}$  Distanz von Stadion  $i$  zu Stadion  $j$
- Zwei Integer Parameter  $L$  und  $U$

Ausgabe:

- Spielplan mit minimierter Reisedistanz aller Teams
- Spielplan, bei dem die Anzahl an aufeinander folgenden Heim- und Auswärtsspielen zwischen  $L$  und  $U$  liegen



## The constant distance Traveling Tournament Problem (CTTP)

- Problem bei dem die Entfernungen der Spielstädten konstant sind
- Urrutia und Ribeiro zeigten, dass das Minimieren einer konstanten Reisedistanz äquivalent zum Maximieren der Breaks ist
- Miyashiro und Matsui zeigten, dass das Maximieren der Breaks äquivalent zum Minimieren der Breaks
- Bedingungen werden so geändert, dass wir Breaks maximieren
- Erlauben höchstens 3 aufeinander folgende Heim- bzw Auswärtsspiele



## The constant distance Traveling Tournament Problem (CTTP)

Teams	Distance LB	Distance PGBA	Breaks	Time
4	17	17	14	0.02
8	80	80	64	0.13
14	252	253	222	35.50
18	432	432	360	2.53
20	520	-	-	-

Tabelle 8.8: Ergebnisse der Instanzen für ein mirrored CTTP

Teams	Distance UB	Distance PGBA	Breaks	Time
4	17	17	14	0.02
8	80	80	64	0.94
14	253	252	224	32.63
16	331	327	306	43.42
18	423	-	-	-

Tabelle 8.9: Ergebnisse der Instanzen für ein non-mirrored CTTP



## Zusammenfassung

- Im Gegensatz zu vorherigen Methoden wurde diesmal die Anzahl an erstellten Patterns minimiert
- Benutzung von Bender Cuts zur Entfernung von ungültigen Pattern Sets aus dem Lösungsraum
- Der Algorithmus zeigte sich bei unterschiedlichen place constraints sehr stabil
- Mit wachsender Anzahl an Teams stieg die Rechenzeit zum Teil enorm (bis zu 30 min)
- Es lassen sich damit auch andere dazu ähnliche Problem wie das CTTP lösen





- Ende -

Fragen?

