NBA Scheduling

Kancharla, Chinepalli,
Moessner, Sun, & Zhu

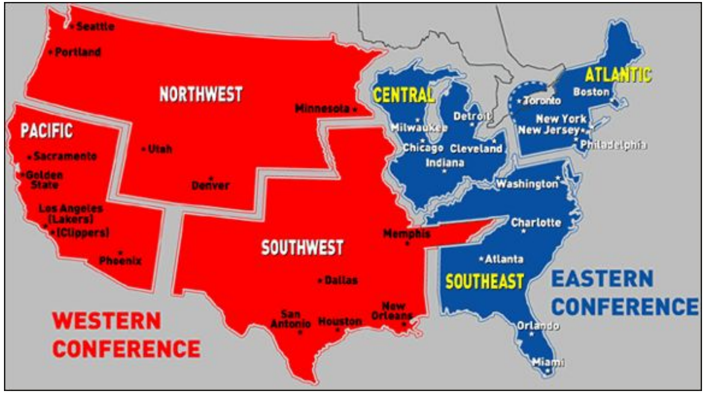
Summary

The issue of scheduling NBA games is one of enormous complexity; optimizing schedules can dramatically decrease costs and increase revenues

1. Why this problem is interesting
2. Primary objectives
3. Our scaled down systems
4. Minimizing distance
5. Minimizing injuries
6. Combining objectives
7. Other possible objectives
8. Limitations and extensions

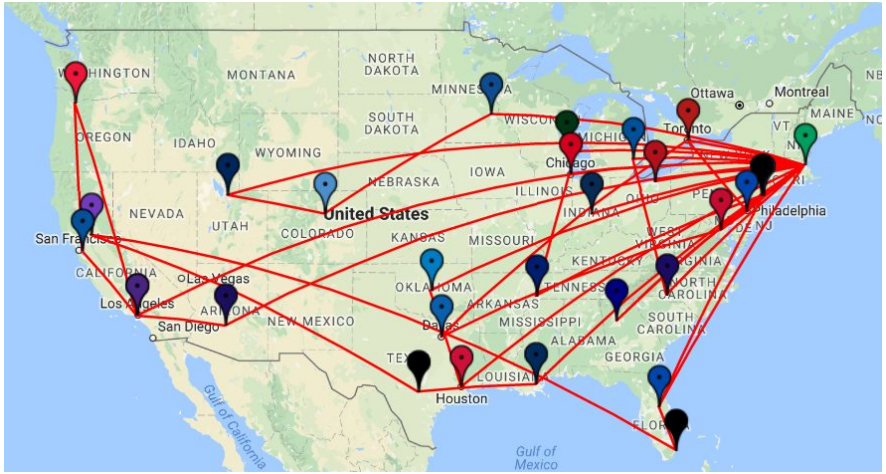
Why this problem is interesting

1. Scheduling NBA games is a large problem; there are 32 teams across the nation; the season is 168 days; each team plays 82 games; there are 1312 total games
2. The ‘optimal’ schedule changes based on what one wishes to prioritize
3. The number of factors in the system is massive; some games seek prime television times, some stadiums cannot be used at all times, specific matchups are sought at specific times, etc.
4. Most constraints are absolute



Breakdown of the NBA Conferences

Travel map of Boston Celtics for 2015-2016 Season



Primary Objectives

- Most of the possible objectives discussed thus far are quite difficult to model
- Some objectives, such as optimizing revenue from TV broadcasts, are more reasonable to consider after optimizing other objectives
- Iteratively adding constraints can make developing a feasible solution difficult
- The most important features are quite difficult to impose: slack and adaptability

1. Minimize the number of consecutive games that a team will play; i.e. minimize the number of times that a team plays 2 games in 2 days
    a. This is known to reduce the likelihood of player injuries
    b. This is known to positively impact game and television turnout
2. Minimize the distance that a team travels
    a. Actual cost of travel is negligible (<2m)
    b. Teams are significantly more likely to lose if they travel more
    c. Crossing time zones amplifies this

Our scaled down system (a)

- Length of season: 168 days ↠ 30 days
- Teams: 30 teams ↠ 8 teams
- Games per team: 82 games ↠ 14 games
- Total games: 1312 games ↠ 56 games
- Teams are geographically distributed in both
- Constraints and objectives readily carry over

Our scaled down system (b)

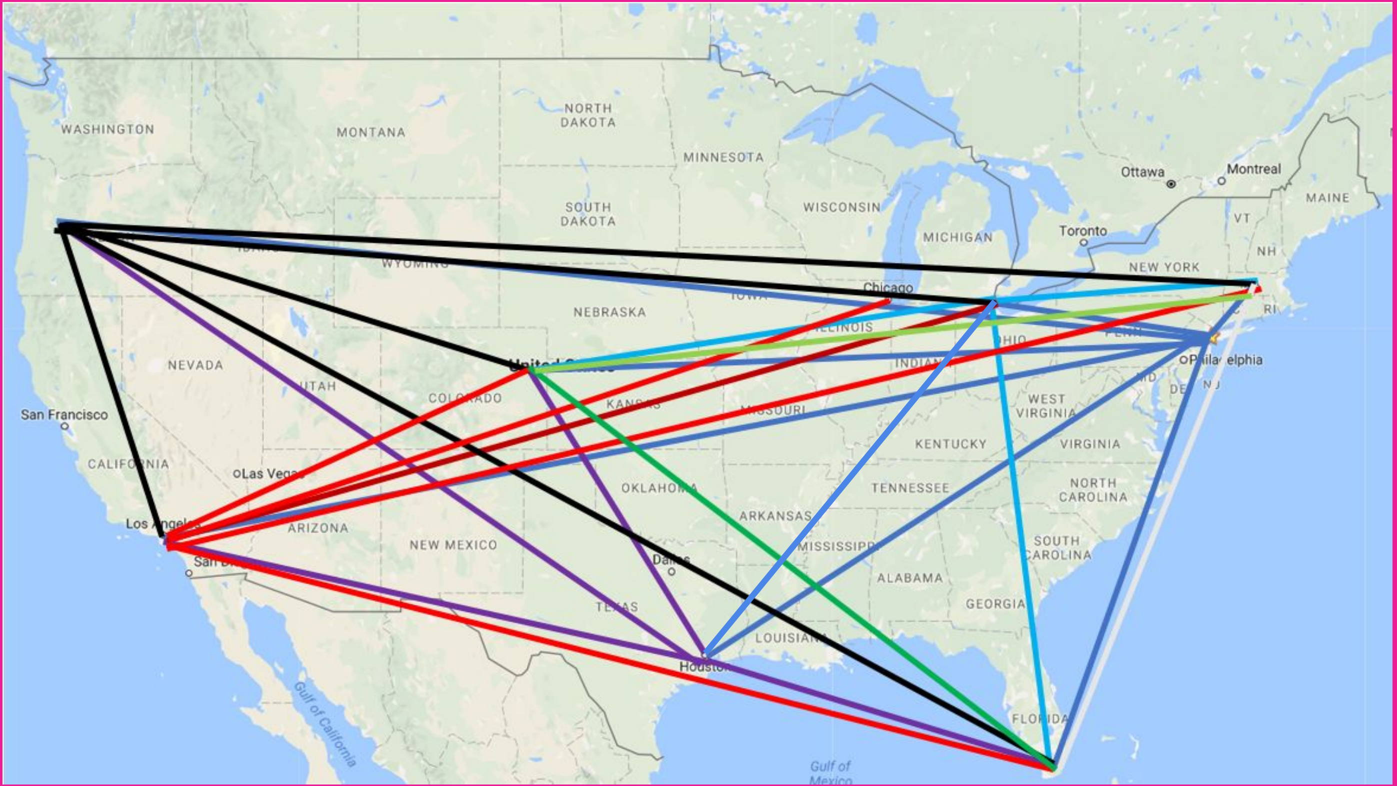
- Length of season: 168 days ↣ 12 days
- Teams: 30 teams ↣ 4 teams
- Games per team: 82 games ↣ 6 games
- Total games: 1312 games ↣ 12 games
- Teams are geographically distributed in both
- Constraints and objectives readily carry over

Minimizing the distance of each team is an NP-Hard problem which requires complete enumeration; 8.9 E 12 cases!

Distances between our teams

	New York Knicks	Miami Heat	Portland Trailblazers	Houston Rockets	Denver Nuggets	Los Angeles Lakers	Chicago Bulls	Boston Celtics
New York Knicks	XXXXXXXX	1093	2441	1419	1627	2448	1712	198
Miami Heat		XXXXXXXX	2705	968	1724	2335	1192	1259
Portland Trailblazers			XXXXXXXX	1834	982	827	1755	2535
Houston Rockets				XXXXXXXX	879	1371	939	1605
Denver Nuggets					XXXXXXXX	830	917	1767
Los Angeles Lakers						XXXXXXXX	1744	2593
Chicago Bulls							XXXXXXXX	850
Boston Celtics								XXXXXXXX

All possible travel arcs



Attempt to minimize distance via local optimization

Pairing Knicks	Route	Distance
1	NY-MA-IL-NY-CO-OR-NY-FL-NY-LA-NY-TX-NY-NY	17730
2	NY-IL-NY-NY-FL-TX-NY-MA-NY-LA-OR-NY-CO-NY	12205
3	NY-FL-NY-TX-NY-IL-NY-MA-NY-OR-NY-LA-NY-CO	14785
4	TX-LA-NY-NY-NY-IL-CO-NY-OR-NY-MA-FL-NY-NY	16893
5	MA-NY-IL-NY-CO-NY-FL-NY-OR-NY-LA-NY-TX-NY	18658

Heat	FL-NY-MA-FL-FL-IL-FL-CO-TX-FL-FL-OR-LA-FL	14692

IP for Optimizing NY-Knicks Only

Min CnXn: Cn is the distance between two cities

XN_T+XN_O+XN_L+XN_C+XN_I+XN_M+XN_F+XT_N+XO_N+XL_N+XC_N+XM_N +XF_N=7 //bNY=7

XN_F+XF_M+XF_I+XF_C+XF_L+XF_O+XF_T+XF_N+XM_F+XI_F+XC_F+XL_F+XO_F+XT_F=1//bFL=1

XM_F+XM_N+XM_I+XM_C+XM_L+XM_O+XM_T+XN_M+XF_M+XI_M+XC_M+XL_M+XO_M=1

XN_I+XF_I+XM_I+XI_C+XI_L+XI_O+XI_T+XI_N+XI_F+XI_M+XC_I+XL_I+XO_I+XT_I=1

XC_N+XC_F+XC_M+XC_I+XC_L+XC_O+XC_T+XN_C+XF_C+XM_C+XI_C+XL_C+XO_C+XT_C=1

XL_N+XL_F+XL_M+XL_I+XL_C+XL_O+XL_T+XN_L+XF_L+XM_L+XI_L+XC_L+O_L+XT_L=1

XO_N+XO_F+XO_M+XO_T+XO_C+XO_L+XO_T+XN_O+XF_O+XM_O+XI_O+XC_O+XL_O+XT_O=1

XT_N+XT_FXT_M+XT_I+XT_C+XT_L+XT_O+XO_T+XL_T+XC_T+XI_T+XM_T+XF_T+XN_T=1

XN_T+XT_F+XF_O+XF_C+XF_I+XF_M+XF_L<=3

XN_T+XT_O+XO_F+XO_C+XO_I+XO_M+XO_L<=3

XN_T+XT_C+XC_F+XC_O+XC_I+XC_M+XC_L<=3

XN_T+XT_I+XI_F+XI_C+XI_F+XI_M+XI_L<=3

XN_T+XT_M+XM_F+XM_C+XM_O+XM_I+XM_L<=3

XN_T+XT_L+XL_F+XL_O+XL_C+XL_M+XL_I<=3

XN_F+XF_T+XT_C+XT_O+XT_I+XT_L+XT_M<=3

XN_F+XF_O+XO_C+XO_I+XO_M+XO_L+XO_T<=3

XN_F+XF_C+XC_T+XC_O+XC_I+XC_M+XC_L<=3

XN_F+XF_I+XI_T+XI_C+XI_F+XI_M+XI_L<=3

XN_F+XF_M+XM_T+XM_C+XM_O+XM_I+XM_L<=3

XN_F+XF_L+XL_T+XL_O+XL_C+XL_M+XL_I<=3

XN_M+XM_F+XF_O+XF_C+XF_I+XF_T+XF_L<=3

XN_M+XM_O+XO_F+XO_C+XO_I+XO_T+XO_L<=3

XN_M+XM_C+XC_F+XC_O+XC_I+XC_T+XC_L<=3

XN_M+XM_I+XI_O+XI_C+XI_F+XI_T+XI_L<=3

XN_M+XM_L+XL_F+XL_O+XL_C+XL_T+XL_I<=3

XN_M+XM_T+XT_C+XT_O+XT_I+XT_L+XT_F<=3

XN_C+XC_F+XF_O+XF_T+XF_I+XF_M+XF_L<=3

XN_C+XC_O+XO_F+XO_T+XO_I+XO_M+XO_L<=3

XN_C+XC_T+XT_F+XT_O+XT_I+XT_M+XT_L<=3

XN_C+XC_I+XI_F+XI_T+XI_F+XI_M+XI_L<=3

XN_C+XC_M+XM_F+XM_T+XM_O+XM_I+XM_L<=3

XN_C+XC_L+XL_F+XL_O+XL_T+XL_M+XL_I<=3

XN_I+XI_T+XT_C+XT_O+XT_O+XT_L+XT_M<=3

XN_I+XI_O+XO_C+XO_I+XO_M+XO_L+XO_T<=3

XN_I+XI_C+XC_T+XC_O+XC_T+XC_M+XC_L<=3

XN_I+XI_T+XT_O+XT_C+XT_F+XT_M+XT_L<=3

XN_I+XI_M+XM_T+XM_C+XM_O+XM_O+XM_L<=3

XN_I+XI_L+XL_T+XL_O+XL_C+XL_M+XL_O<=3

XN_O+XO_F+XF_M+XF_C+XF_I+XF_T+XF_L<=3

XN_O+XO_M+XM_F+XM_C+XM_I+XM_T+XM_L<=3

XN_O+XO_C+XC_F+XC_M+XC_I+XC_T+XC_L<=3

XN_O+XO_I+XI_F+XI_C+XI_M+XI_T+XI_L<=3

XN_O+XO_L+XL_F+XL_M+XL_C+XL_T+XL_I<=3

XN_O+XO_T+XT_C+XT_M+XT_I+XT_L+XT_F<=3

Xi is binary for all i.

There must be 7 home games:

XN_T+XN_O+XN_L+XN_C+XN_I+XN_M+XN_F+XT_N+XO_N+XL_N+XC_N+XM_N +XF_N=7 //bNY=7

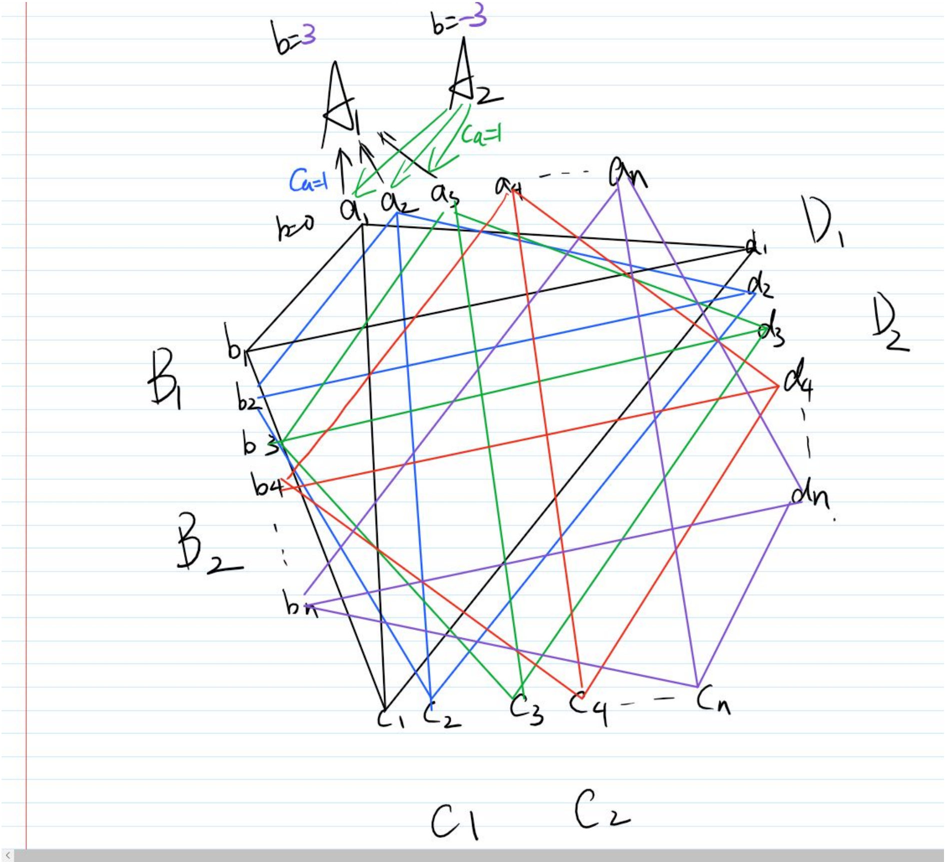
Each city must be visited once:

XN_F+XF_M+XF_I+XF_C+XF_L+XF_O+XF_T+XF_N+XM_F+XI_F+XC_F+XL_F+XO_F+XT_F=1//bFL=1

No more than 3 consecutive away games:

XN_C+XC_F+XF_O+XF_T+XF_I+XF_M+XF_L<=3

General Graph For Distance Minimization for 4 team scale.



A1: Demand is 3 at node A for 3 home games

A2: Supply is 3 for other notes for away games

an: stand for date of the games

Each real arch has a different cost which is the same as distance.

Solving for the minimum distance

$$\min \sum_{i \in T} \sum_{d \in \{0,\ldots,D\}} C_{id}$$

$$\sum_{k \in \{0,1\}} (G_{i(d+k)} = 0) \leq 1 \forall i \in T, d \in \{1,\ldots,D-1\}$$

no team play consecutive home games

$$\sum_{k \in \{0,\ldots,3\}} (G_{i(d+k)} \neq 2) \leq 3 \forall i \in T, d \in \{1,\ldots,D-3\}$$

length of home days

$$\sum_{k \in \{0,\ldots,4\}} (G_{i(d+k)} = 2) \geq 1 \forall i \in T, d \in \{1,\ldots,D-4\}$$

length of off days

$$\sum_{m \in \{0,\ldots,k\}} (G_{i(d+m)} = 0) = 0 \Rightarrow \sum_{m \in \{0,\ldots,k\}} (G_{i(d+m)} = 1) \leq 3 \forall i \in T, k \in \{0,\ldots,D\}, d \in \{1,\ldots,D-k\}$$

length of away games will be equal or less than three

$$\sum_{d \in D} (G_{id} = 1 \,\&\, O_{id} = j) = 1 \forall i, j \in T, i \neq j$$

$$\sum_{d \in D} (G_{id} = 0 \,\&\, O_{id} = j) = 1 \forall i, j \in T, i \neq j$$

make sure a double round robin game

$$G_{id} = 0 \Rightarrow V_{id} = i \forall i \in T, d \in D$$

stay at home on home game day

$$G_{id} = 1 \Rightarrow V_{id} = O_{id} \forall i \in T, d \in D$$

stay at opponent venue on road game day

$$G_{id} = 2 \Rightarrow V_{id} = V_{i(d-1)} \forall i \in T, d \in D$$

stay at the previous venue on off days

Solving for the minimum distance

Bao (2009) states that this problem is not a trivial task to solve.

Assumptions:

- Each team starts at its home city and get back to its home city after the final game.
- Team will not travel back to its home city during break between two games.
- No other constraints.

Solving for the minimum distance

Our attempts: Complete enumeration of game order/schedule

```
g = np.empty((13,3))
t = 0
for i in range(1,5):
    for j in range(1,5):
        if i != j:
            t = t + 1
            g[t,1] = i
            g[t,2] = j

d = np.array([0,1093,1627,2448,1093,0,1724,2335,1627,1724,0,830,2448,2335,830,0])
dis = np.reshape(d,(4,4))

i = np.zeros(13)
soli = np.zeros(13)
soltravel = np.zeros(5)
min = sys.maxint

for i1 in range(1,13):
    current1 = [i1]
    i[1] = i1
    for i2 in range(1,13):
        if not i2 in current1:
            current2 = current1 +[i2]
            i[2] = i2
```

```
total = 0
team = np.zeros(5)
travel = np.zeros(5)
for j in range(1,5):
    team[j] = j

for j in range(1,13):
    home = g[i[j],1]
    away = g[i[j],2]
    if team[home] != home:
        travel[home] = travel[home] + dis[team[home]-1,home-1]
        total = total + dis[team[home]-1,home-1]
        team[home] = home
    if team[away] != home:
        travel[away] = travel[away] + dis[team[away]-1,home-1]
        total = total + dis[team[away]-1,home-1]
        team[away] = home

for j in range(1,5):
    if team[j] != j:
        travel[j] = travel[j] + dis[team[j]-1,j-1]
        total = total + dis[team[j]-1,j-1]

if total < min:
    min = total
    soli = i
    soltravel = travel
```

Solving for the minimum distance

Ongoing attempt: Decomposition method

First step: Optimal tour for each team

Second step: Optimal tour selection

$$\sum_{i \in Ut} i * x_i = 1 \forall t \in T \tag{4.34}$$

$$\sum_{i \in Ut} (v_{id} \neq i) * x_i + \sum_{i \notin Ut} (v_{id} = i) * x_i \leq 1 \forall t \in T, d \in D \tag{4.35}$$

$$\sum_{m \in \{0,\ldots,3\}} \sum_{i \in Ut} (v_{i(d+m)} \neq i) * x_i + \sum_{m \in \{0,\ldots,3\}} \sum_{i \notin Ut} (v_{i(d+m)} = i) * x_i \leq 3 \forall t \in T, d \in D-3 \tag{4.36}$$

$$\sum_{m \in \{0,\ldots,4\}} \sum_{i \in Ut} (v_{i(d+m)} = i) * x_i + \sum_{m \in \{0,\ldots,4\}} \sum_{i \notin Ut} (v_{i(d+m)} \neq i) * x_i \leq 4 \forall t \in T, d \in D-4 \tag{4.37}$$

$$\sum_{m \in \{0,1\}} \sum_{i \notin Ut} (v_{i(d+m)} = i) * x_i \leq 1 \forall t \in T, d \in \{1,\ldots,D-1\} \tag{4.38}$$

$$x_i \in \{0,1\} \tag{4.39}$$

$$v_{td} \in T \forall t \in T, d \in D \tag{4.40}$$

Theoretical framework for minimizing injuries

- As seen, optimizing travelling distance is (very) NP-Hard
- Injuries are far more detrimental to performance than travel

Minimize: $$\sum_{i=1}^{|T|} \sum_{d=1}^{|D|} \max\left[0, y_{id} * y_{i(d+1)}\right]$$

Subject to constraints:

$$y_{id} = \begin{cases} 1, & if\ \exists\ j \in T/i \mid x_{ijd} = 1 \\ 0, & else \end{cases} \quad \forall\ i \in T$$

$$\sum_{d \in D} (x_{ijd} + x_{jid}) = 1\ \forall\ i.j \in T, i < j$$

$$\sum_{j \in \{T \setminus i\}} (x_{ijd} + x_{jid}) \leq 1\ \forall\ i \in T, d \in D$$

$$x_{ijd} \in \{0,1\}\ \forall\ i, j \in T, i \neq j, d \in D$$

- x are decision variables; y are auxiliary
- T is the set of all teams; $|T|$ is the size
- D is the set of all days; $|D|$ is the length
- Obviously, problem is NP-Hard

Solving the program

```
# First constraint is that each team has to play the other team on exactly one day,
# i.e. for some given team i and some given team j, the sum of xi_j_d over all d has to be 1
for i in range(num_teams):
    for j in range(num_teams):
        if (not i == j):
            variables = [] # list that will contain all variables for given i and j and then over all days
            for k,v in d.items():
                if((v[0] == i+1) and (v[1] == j+1)):
                    variables.append(m.getVarByName(k))
            variables_sum = quicksum(variables)
            m.addConstr(variables_sum == 1)
```

```
# Second contraint is that on a given day, a given team can only play at most once
# Third constraint deals with the yi_d variables and simply extends second constraint
for i in range(num_teams):
    for x in range(num_days):
        variables = []
        for k,v in d.items():
            if((v[0] == i+1) and (v[2] == x+1)):
                variables.append(m.getVarByName(k))
        variables_sum = quicksum(variables)
        m.addConstr(variables_sum <= 1)
        y_var = 'y' + str(i+1) + '_' + str(x+1)
        y_var = m.getVarByName(y_var)
        m.addConstr(y_var - variables_sum == 0)
```

Solution

Day 1 LA vs Denver Portland vs Chicago	2 Portland vs NY Chicago vs Miami	3 Boston vs Houston	4 Denver vs Chicago Boston vs LA	5 NY vs Denver Portland vs Boston	6 Houston vs NY Denver vs LA	7 Denver vs Boston Miami vs Portland Chicago vs LA
8 NY vs Portland	9 Houston vs Boston MIami vs Chicago	10 LA vs Chicago Houston vs Portland	11 LA vs NY Boston vs Chicago	12 NY vs Miami Denver vs Houston Chicago vs Boston	13 Portland vs Miami Chicago vs Houston	14 LA vs Houston
15 Miami vs NY	16 Houston vs Chicago	17 Chicago vs Denver	18 Chicago vs Portland Miami vs LA	19 Boston vs NY Miami vs Houston	20 NY vs Houston LA vs Portland Boston vs Denver	21 Houston vs Denver Chicago vs NY
22 Boston vs Miami Houston vs LA	23 NY vs Boston Denver vs Portland	24 Portland vs LA Denver vs NY Miami vs Boston	25 Denver vs Miami	26 Portland vs Denver LA vs Miami NY vs Chicago	27 LA vs Boston	28 Houston vs Miami
29 Boston vs Portland NY vs LA	30 Portland vs Houston Miami vs Denver					

Other possible constraints

- Ensure that no team plays three games in a row:

$$\sum_{k\in\{0,1,2\}} \sum_{j\in\{T\setminus i\}} \left(x_{ij(d+k)} + x_{ji(d+k)}\right) < 3 \;\forall\, i \in T, d \in \{1, \cdots, D-2\}$$

- Ensure that no team plays 4 games in 5 days:

$$\sum_{k\in\{0,1,2,3,4\}} \sum_{j\in\{T\setminus i\}} \left(x_{ij(d+k)} + x_{ji(d+k)}\right) < 4 \;\forall\, i \in T, d \in \{1, \cdots, D-4\}$$

- Arena availability (TV prime-time restriction):

$$x_{ijd} \leq v_{id} \;\forall\, i, j \in T, i \neq j, d \in D$$

Combined objective program

- Adding constraints to the program is not very difficult
- Attaining multiple objectives is hard because:
    - Some arbitrary scaling parameter is needed to compare ‘apples and oranges’
    - We can no longer solve the problem quickly
- More than minimizing total distance traveled, it is more important to ensure that each team travels comparable distances
    - This problem requires iterating through complete enumerations
    - It is interesting to note that outcomes can be manipulated by skewing travel times
- From a revenue perspective, it is more important to optimize game dates and matchups than minimize traveling distance

Concluding remarks

1. The scheduling of NBA games is one of unexpected complexity
2. The problem is NP-Hard for most objectives (e.g. minimize injuries)
3. It is possible to develop an adaptable linear program to achieve the above
4. Combining objectives will likely require complete enumeration to find the global optimum
5. Most locally optimizing measures will not actually work in this context
6. It is possible for the optimum solution to be very ‘unfair’ and therefore not actually optimal

Limitation and extensions

- Computational limits for us are far more restrictive than in professional scheduling practice
- Problem scales exponentially with number of teams and cities added
- Some methods employed can likely be extended to the full problem given more computing power, others aren’t suited to scale
- Active research into tabu-search as a feasible solution for sports league scheduling
- Numerous real constraints in the full problem greatly reduces the search space, making finding the optimum easier