Jerry’s Fantasy Football Lineup Set Optimization

My Fantasy Football Model

In Fantasy Football the overall goal is to mix and match players from a set of games to great a lineup that scores the most *fantasy points* while the sum of each player’s given salary is under a fixed number.

There are several variations on lineup requirements, one is called *Captain Mode* in which you select only 6 players from a single game, you do not have any position requirements, and can select one player for the **Captain Slot**. This slot multiplies both the player’s salary and overall fantasy score by 1.5x.

Some contests allow for submitting multiple lineups (up to 20 in many cases). Submitting multiple lineups can increase your chances in finishing very high in the contest, in which prize money is much higher.

Create the Optimization Model

Start with basic constraints to create a set of lineups. An individual player can:
1. Not be selected in a lineup
2. Be selected for the captain slot (and 1.5x Salary)
3. Selected at regular cost (the player is put in a “FLEX” slot)

This model will list an individual player, salary, and estimated points separately from their captain mode counterpart. The data set *projections* has each players name, team, estimated points (DKpoints) and salary. Below is Stefon Diggs’ data for the captain and flex roster slots.

```
projections[which(projections$Name=="Stefon Diggs"),]
```

```
## # A tibble: 2 x 7
##   Name               ID Pos   ‘Roster Position‘ Team  DKpoints Salary
##   <chr>           <dbl> <chr> <chr>             <chr>    <dbl>  <dbl>
## 1 Stefon Diggs 20955369 WR    CPT               BUF       24.9  15900
## 2 Stefon Diggs 20955420 WR    FLEX              BUF       16.6  10600
```

Parameters for the model:

```
n = nrow(projections)             #### number of players-slots (46 in this case)
points = projections$DKpoints     #### estimated fantasy points
cost = projections$Salary         #### player salary
cap = 50000                       #### Salary cap

#### These statements identify, by index, players by position, if they have
#### captain salary/points,and team they play for
qb = which(projections$Pos=="QB")
rb = which(projections$Pos=="RB")
wr = which(projections$Pos=="WR")
te = which(projections$Pos=="TE")
```

```
dst = which(projections$Pos=="DST")
k = which(projections$Pos=="K")
flex = which(projections$`Roster Position`=="FLEX")
capt = which(projections$`Roster Position`=="CPT")
team1 = which(projections$Team==game[1])
team2 = which(projections$Team==game[2])
```

Decision Variables

There are i named players and since each player can occupy two potential slots there are $n = 2i$ player-slots. $player_{k,l} = 1$ if the k^{th} indexed player/roster position is selected in lineup l and 0 otherwise, $k = 1, \dots, n$, and $l = 1, \dots, nlineups$. In this example, $n = 46$.

Constraints for Each Lineup

Salary cap constraint for each lineup.

$$\sum_{k=1}^{n} cost_k * player_{k,l} \leq Cap, l = 1, \dots, nlineups$$

Can only have one captain in each lineup.

$$\sum_{k \in CAPT} player_{k,l} = 1, l = 1, \dots, nlineups$$

Need five other players to complete the lineup in the FLEX spots.

$$\sum_{k \in FLEX} player_{k,l} = 5, l = 1, \dots, nlineups$$

The following constraints are not required by the contest, but preferences to adhere to. No more than two quarterbacks in each lineup.

$$\sum_{k \in QB} player_{k,l} \leq 2, l = 1, \dots, nlineups$$

Similarly, I have constraints for the RB, WR, TE, K, and DST positions. I also do not want an entire lineup made of players from a single team.

$$\sum_{k \in Team_i} player_{k,l} \leq 5, l = 1, \dots, nlineups, i = 1, 2$$

Constraints for Each Player

A player cannot be assigned to a FLEX and Captain (CAPT) slot in the same lineup. Forgive the notation.

$$player_{k,l}^{CAPT} + player_{k,l}^{FLEX} \leq 1, k = 1, \dots, i$$

A common consideration for these contests is *exposure*, where you want to limit the number of times you put a player in a set of lineups. If a player is in every lineup and scores very little points then all lineups are probably going to do poorly.

$$\sum_{l=1}^{nlineups} player_{k,l}^{CAPT} + player_{k,l}^{FLEX} \leq exposure, k = 1, \dots, i$$

Lineup Ordering and Uniqueness

At this point lineups can be repeated as long as the exposure constraint is not violated. Requiring each lineup to be pairwise unique can add a number of constraints and auxiliary variables. The workaround I use is to require the total points to strictly decrease for each lineup, making the first have the highest expected points, and so on.

$$\sum_{k=1}^{n} points_k * player_{k,l} - \sum_{k=1}^{n} points_k * player_{k,l+1} \geq \epsilon, l = 1, \ldots, nlineups - 1$$

The problem with this – it is possible (and somewhat likely) that two successive lineups can have the same total points.

Objective Function

An obvious choice to to maximize the total points over all lineups.

$$MaxZ = \sum_{k=1}^{n} \sum_{l=1}^{nlineups} points_k * player_{k,l}$$

I have also tested maximizing the points for the last lineup.

R Code Using ompr Package

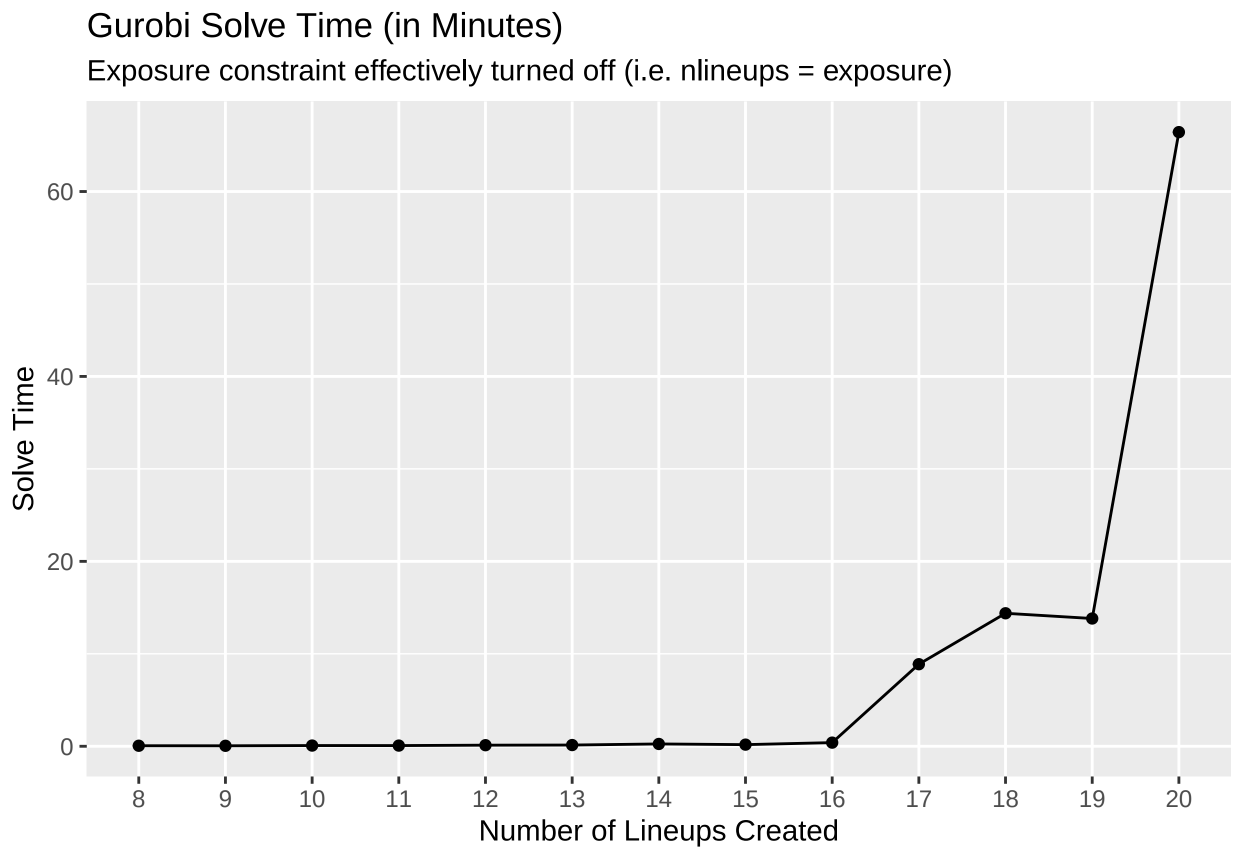
This is included to show how to create the model in R. The *pipe operator* (%>%) links one line to the next and is very popular in R, especially R data scientists.

```
n.lineups = 20
exposure = n.lineups - 3
capt.mult = MIPModel() %>%
  add_variable(player[i,l], i = 1:n, l = 1:n.lineups, type = "binary") %>%
  set_objective(sum_expr(points[i] * player[i,l], i = 1:n, l = 1:n.lineups)) %>%
  #add_variable(r, type = "continuous", lb = 0) %>%
  #set_objective(r, "max") %>%
  #add_constraint(r <= sum_expr(points[i] * player[i,l], i = 1:n), l = n.lineups) %>%
  add_constraint(sum_expr(points[i] * player[i,l], i = 1:n) -
                   sum_expr(points[i] * player[i,(l+1)], i = 1:n) >= 0.01, l = 1:(n.lineups-1)) %>%
  add_constraint(sum_expr(cost[i] * player[i,l], i = 1:n) <= cap, l = 1:n.lineups) %>%
  add_constraint(sum_expr(player[i,l], i = qb) <= 2, l = 1:n.lineups) %>%
  add_constraint(sum_expr(player[i,l], i = dst) <= 1, l = 1:n.lineups) %>%
  add_constraint(sum_expr(player[i,l], i = rb) <= 3, l = 1:n.lineups) %>%
  add_constraint(sum_expr(player[i,l], i = wr) <= 3, l = 1:n.lineups) %>%
  add_constraint(sum_expr(player[i,l], i = te) <= 2, l = 1:n.lineups) %>%
  add_constraint(sum_expr(player[i,l], i = k) <= 2, l = 1:n.lineups) %>%
  add_constraint(sum_expr(player[i,l], i = flex) == 5, l = 1:n.lineups) %>%
  add_constraint(sum_expr(player[i,l], i = capt) == 1, l = 1:n.lineups) %>%
  add_constraint(sum_expr(player[i,l], i = team1) <= 5, l = 1:n.lineups) %>%
  add_constraint(sum_expr(player[i,l], i = team2) <= 5, l = 1:n.lineups)

for (t in unique(projections$Name)) {
  id = which(projections$Name==t)
  if (length(id)==2) {
    capt.mult = capt.mult %>%
      #### contraint to manke sure each named player is used once in a lineup
      add_constraint(sum_expr(player[i,l], i = id) <= 1, l = 1:n.lineups) %>%
      #### overall exposure contraint across all lineups
      add_constraint(sum_expr(player[i,l], i = id, l = 1:n.lineups) <= exposure)
  } else {
    print(t)
  }
}
```

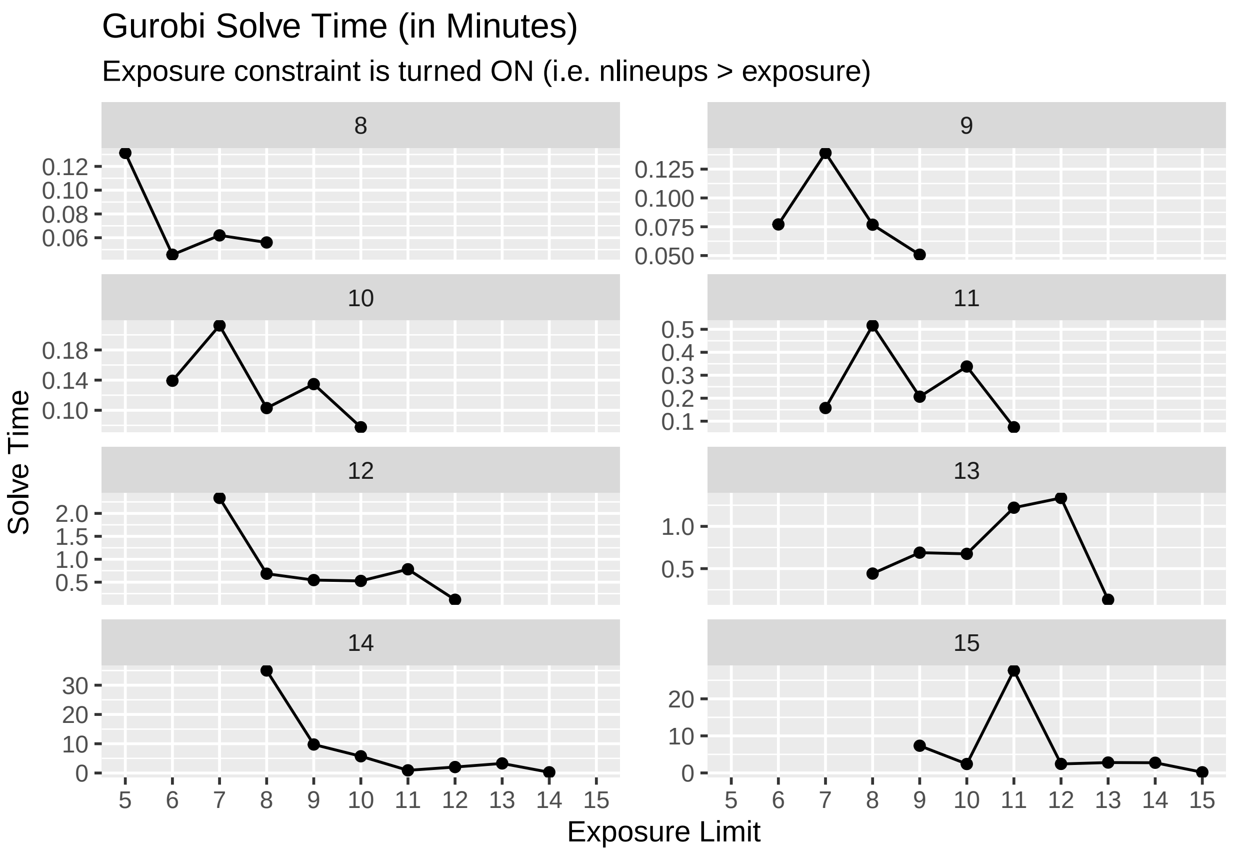
Model Solve Times

No Exposure Restriction



Different Exposure Values

Note that the y-axis varies for each block, which contains the number of lineups created. This jumps around more than I anticipated. Additionally to the chart below, the case of 20 lineups and a max exposure of 19 took 3.5 hours to solve and exposure of 18 didn't solve after 10 hours.


Gurobi Solve Time (in Minutes)
Exposure constraint is turned ON (i.e. nlineups > exposure)
Solve Time
Exposure Limit
8
9
10
11
12
13
14
15
0.12
0.10
0.08
0.06
0.125
0.100
0.075
0.050
0.18
0.14
0.10
0.5
0.4
0.3
0.2
0.1
2.0
1.5
1.0
0.5
1.0
0.5
30
20
10
0
20
10
0
5 6 7 8 9 10 11 12 13 14 15