

Consulting Report: Phase III Scheduling Baseball League



Image Source: 12th Man webpage(<http://www.12thman.com/index.asp?main=baseball>)

Prepared for:
The College Station Little League Baseball

Prepared by:
ISEN Team8 Consulting Group
January 2006

Consulting Team:
Abhijeet Shinde
Naga Ramesh Kamiseti
Prashanth Potana

Disclaimer

This report has been generated solely for the purpose of solving the league scheduling problem faced by College Station Little League Baseball (here on referred to as client). All the programming codes, data, analysis and mathematical equations used and furnished to solve the given problem are strictly and independently generated by ISEN Team8 Consulting Group (here on referred to as consulting group). The AMPL code, solution and analysis given in the report can be used by the client for any purpose whatsoever at their own discretion. The Consulting Group agrees that no claim of copyright or further compensation on the usage of any part of this report's content will be raised.

Contents

List of Tables	5
Preface	6
Executive Summary.....	7
Problem Description	9
Approach I.....	10
Parameters.....	11
Sets.....	11
Decision variables	12
Objective Function.....	12
Constraints	13
<i>Constraint 1</i>	13
<i>Constraint 2</i>	13
Constraint 3:	13
<i>Constraint 4</i>	14
<i>Constraint 5</i>	14
AMPL.....	15
Results.....	18
What-If analysis	19
Manual for installing and using AMPL	25
Solver	25
AMPL License Information.....	25
Model and Data Files	25
Solver tools available	25
Schematic representation of AMPL working	26
Run AMPL, Load Solver(CPLEX), run Model and Data File and Display Output.....	26
Making changes to a Data file.....	28
Interpreting AMPL Solution	30
Approach II.....	31
Parameters.....	32
Decision Variables.....	33
Objective Function.....	33
Constraints	33
<i>Constraint 1</i>	33
<i>Constraint 2</i>	33

<i>Constraint 3</i>	34
<i>Constraint 4</i>	34
<i>Constraint 5</i>	34
<i>Constraint 6</i>	34
<i>Constraint 7</i>	35
<i>Constraint 8</i>	35
<i>Constraint 9</i>	35
<i>Constraint 10</i>	35
<i>Constraint 11</i>	35
AMPL.....	36
Results.....	39
Conclusion.....	40
References	41

List of Tables

Table 1: Block number details	8
Table 2: $H[k]$: number of holidays in block k	9
Table 3: One of the possible schedules.....	15
Table 4: Slot numbers for corresponding d & w	17
Table 5: Game vs Slot number matrix	25
Table 6: Game vs Calendar date matrix	25

List of Figures

Fig 1: Plot for Number of teams vs Minimum number of full 4-game blocks required.....	19
Fig 2: Plot for Number of game slots per block vs Total number of blocks required	22
Fig 3: Plot for Number of game slots in a Saturday block vs Total number of blocks required	23
Fig 4: Schematic diagram of AMPL working	27

Preface

The problem currently faced by the client is similar to a traditional round robin scheduling problem in sports environment. Round robin scheduling requires assigning different combinations of pairs of teams to a fixed number of available game slots adhering to all league requirements. Typical requirements include home/away game scheduling and breaks after a fixed number of games per team.

Generally, scheduling problems are addressed leveraging linear programming models. Linear programming (LP) (also called linear optimization) is a method to achieve the best outcome (such as maximum profit or lowest cost) in a mathematical model whose requirements are represented by linear relationships. Linear programming is a special case of mathematical programming^[1]. Linear Programming models are being widely used to solve optimization problems in industries as diverse as sports, banking, petroleum, and logistics. An integer programming problem is a mathematical optimization or feasibility program in which some or all of the variables are restricted to be integers^[2].

There are multiple software tools available for solving optimization problems. AMPL is one of the widely used programming language having a great similarity of its syntax to mathematical notation of optimization problems. This allows for a very concise and readable definition of problems in the domain of optimization^[3]. Out of the many solvers available on AMPL platform we chose CPLEX to meet the client's requirement.

Executive Summary

The management of Local College Station Little League Baseball - our client in this case - has requested a comprehensive solution and structured analysis for scheduling games among 9 teams for the upcoming Baseball Summer season of 2006. Their management team is currently facing a situation of strict adherence to timeline of the season taking into account holidays and other constraints such as giving breaks to teams between consecutive games. They have hired ISEN Team8 Consulting Group to analyze the current situation and address them back with any relaxation of underlying constraints if required or give a solution by scheduling all games in the given timeline. This report is an outcome of consolidated analysis generating a scheduling rubric for all 9 teams in the league adhering to all constraints put forth by the client.

The consulting group after carefully considering the problem context and the end objective of scheduling the league in given timeline finalized to use integer linear optimization model for generating the scheduling rubric. The consulting group considered flexibility as an important factor to be built in the model additional to scheduling the league to accommodate possible scope changes in the schedule of league.

The results of the given problem statement including AMPL code, data and model files are published towards the end of the report stating that the problem has feasible solution which is scheduling all 72 games into 74 available slots adhering to all the limitations of the client. One of possible scheduling rubric is provided (on page no. 15) to better interpret output of the model. The AMPL model and data file are collectively included in the report as per negotiated contract terms.

The requirements of given scheduling problem are fulfilled and further the consulting team has performed case based 'What-If' analysis to understand future scope of changes and corresponding set of activities to be undertaken. Following are the major outcomes from this analysis (for additional details refer to 'What-If' analysis section of the report).

- There are no critical game slots in the season calendar provided by the client which could hamper the schedule.
- The plot for number of full 4-game blocks required V/s number of teams is provided on page number 19.
- Planning to schedule game on a Friday and an additional game on Saturday doesn't reduce the duration of the league with 9 teams participating. However, this provides scope for adding another team to the league without extending the duration.
- Allowing a team to play more than once in a Monday-Thursday block doesn't reduce the duration of the league. However, if the client decides to incorporate this requirement, a provision has been given to amend the data file accordingly as per need.

- If maximum number of games that can be scheduled in a block is reduced, the duration of the league increases. A plot of number of blocks required V/s maximum number of games that can be scheduled in block is provided in page number 22.
- If maximum number of games that can be scheduled in a Saturday block is reduced, the duration of the league increases. A plot of number of blocks required V/s maximum number of games that can be scheduled in Saturday block is provided in page number 23.
- There is possibility of scheduling the league in the given duration in case the client requires games between any pair of teams not to be scheduled in consecutive blocks. The necessary modifications to the model and data file are provided on page number 23.

An alternative approach for solving the scheduling problem is provided as Approach II. The client can choose either of the two approaches provided as per their convenience and at their discretion.

Problem Description

The given problem requires to check for a feasible solution and then schedule games for a Baseball league for the upcoming Summer season from March 13, 2006 till May 31, 2006. There are 9 teams participating in the league this season and each team has to play every other team twice before the conclusion of league. The season has 80 calendar days out of which 50 days are available to schedule games and the rest 30 days are holidays. No game can be scheduled on a holiday. Fridays and Sundays are considered as mandatory holidays along with other calendar holidays.

Games are conducted every week on Monday through Thursday and on Saturday. No team can play more than once in any four-day (Monday to Thursday) time block or more than once on Saturday. At most four games can be scheduled in a time block from Monday to Thursday and on a Saturday. A team that has played in a given Monday to Thursday block can play on the following Saturday.

Summarizing, a total of 72 games (each team playing 8 games in entire tournament) are to be scheduled in 74 available game slots (10 Monday-Thursday blocks = 40 slots; 8 Saturdays = 32 slots; 2 slots in the last Monday-Thursday block due to a holiday on Monday and end of season on Wednesday). So, the objective for the consulting group is to check for a feasible solution to the problem and come up with a game schedule rubric.

A brief description of the constraints set by the League's management team and taken into consideration by the consulting group prior to solving the integer linear optimization model is as follows:

1. Each team must play against every other team twice.
2. Monday-Thursday time block can hold at most four games.
3. Saturday can hold at most 4 games (played simultaneously).
4. Each team can play at most one game in a Monday-Thursday time block.
5. Each team can play at most one game on a Saturday.
6. No game to be scheduled on a holiday.

Approach I

There are a total of 74 game slots available which are divided amongst 19 blocks numbered from 1 to 19. Each block is considered to have four game slots. The 19th block has 2 mandatory off-slots (color coded yellow in Table 1), one of holiday on Monday and other of Thursday (season ends on Wednesday) where any game is possible to be scheduled in each of the time blocks 1 to 19. The table below is created after excluding holidays.

Date	Day	Block No.	Date	Day	Block No.
03/13/2006	Monday	1	04/29/2006	Saturday	11
03/14/2006	Tuesday		05/01/2006	Monday	12
03/15/2006	Wednesday		05/02/2006	Tuesday	
03/16/2006	Thursday		05/03/2006	Wednesday	
03/18/2006	Saturday	2	05/04/2006	Thursday	13
03/20/2006	Monday	3	05/06/2006	Saturday	
03/21/2006	Tuesday		05/08/2006	Monday	14
03/22/2006	Wednesday		05/09/2006	Tuesday	
03/23/2006	Thursday		05/10/2006	Wednesday	
03/25/2006	Saturday	4	05/11/2006	Thursday	15
03/27/2006	Monday	5	05/13/2006	Saturday	
03/28/2006	Tuesday		05/15/2006	Monday	16
03/29/2006	Wednesday		05/16/2006	Tuesday	
03/30/2006	Thursday		05/17/2006	Wednesday	
04/01/2006	Saturday	6	05/18/2006	Thursday	17
04/03/2006	Monday	7	05/20/2006	Saturday	
04/04/2006	Tuesday		05/22/2006	Monday	18
04/05/2006	Wednesday		05/23/2006	Tuesday	
04/06/2006	Thursday		05/24/2006	Wednesday	
04/17/2006	Monday	8	05/25/2006	Thursday	19
04/18/2006	Tuesday		05/29/2006	Monday	
04/19/2006	Wednesday		05/30/2006	Tuesday	
04/20/2006	Thursday		05/31/2006	Wednesday	
04/22/2006	Saturday	9	06/01/2006	Thursday	
04/24/2006	Monday	10			
04/25/2006	Tuesday				
04/26/2006	Wednesday				
04/27/2006	Thursday				

Table 1: Block number details

Parameters

A parameter is a quantity whose value is selected for the circumstances and in relation to which other variable quantities may be expressed^[4].

1. **B**: Total number of available time blocks. A block is either a group of four days in a week from Monday to Thursday; or a Saturday. Here, $B = 19$
2. **G**: Maximum number of games possible in a Monday – Thursday block without considering holidays. Here, $G = 4$.
3. **GS**: Maximum number of games possible in a Saturday block without considering holidays. Here, $GS = 4$.
4. **N**: Number of teams participating in the league. Here, $N = 9$.
5. **R**: Number of times each team must receive every other team. Here, $R=1$
6. **M**: Maximum number of games a team can play in Monday - Thursday block. Here, $M=1$
7. **MS**: Maximum number of games a team can play in Saturday block. Here, $MS=1$
8. **H[k]**: Number of off-slots in block k. Here, in block number 19, ($H[19] = 2$).

Block Number	Number of off-slots
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	2

Table 2: $H[k]$: number of off-slots in block k

Sets

1. **MT**: Set of Block numbers having Monday-Thursday defined as block.
Here $MT = \{1, 3, 5, 7, 8, 10, 12, 14, 16, 18, 19\}$
2. **S**: Set of Block numbers having Saturday defined as block.
Here, $S = \{2, 4, 6, 9, 11, 13, 15, 17\}$

Decision variables

Decision variables are unknown quantities that need to be determined in order to solve our objective.

$$1. X_{i,j,k} = \begin{cases} 1, & \text{if team } i \text{ plays with team } j \text{ in block } k, \text{ for } i, j \in [1, N], k \in [1, B] \\ 0, & \text{otherwise} \end{cases}$$

$X_{i,j,k}$ is a binary variable and takes the value of 1 when team i plays with team j in a particular block k . It takes a value of 0 for all other possible combinations.

2. $Y_{i,j}$ displays a matrix where i^{th} row and j^{th} column corresponds to the block number in which team i plays with team j .

Objective Function

Maximize: Z

Maximize Z: 0

Here, we choose objective function as maximize 0. Since, we are not trying to optimize anything and it is a scheduling problem, selecting 0 as objective function will suffice our purpose.

Constraints

Constraints are limiting hurdles, restrictions or other mandatory compliance factors that are to be strictly adhered to while achieving our business goals.

Constraint 1:

Each team must receive every other team exactly R times in league. We considered that team i playing with team j and team j playing with team i are two different games. Hence team i should receive team j R times and team j should receive team i R times. This should happen for all the 9 teams.

$$\sum_{k=1}^B X_{i,j,k} = R, \text{ for } i, j \in [1, N], i \neq j$$

Constraint 2:

A team can play at most a definite times in a block. (Also, as explained in constraint 1, team i playing with team j is different compared to team j playing with team i)

Constraint 2a

In Monday- Thursday block, each team can play at most M times

$$\sum_{\substack{t=1 \\ t \neq i}}^N X_{i,t,k} + \sum_{\substack{t=1 \\ t \neq i}}^N X_{t,i,k} \leq M \text{ for } k \in [1, B], i \in [1, N]$$

Constraint 2b

In a Saturday block, each team can play at most MS times

$$\sum_{\substack{t=1 \\ t \neq i}}^N X_{i,t,k} + \sum_{\substack{t=1 \\ t \neq i}}^N X_{t,i,k} \leq MS \text{ for } k \in [1, B], i \in [1, N]$$

Constraint 3:

The number of games possible in a block is equal to the difference of maximum number of games possible and the number of off slots in a block. Off slots are the slots in which a game cannot be scheduled (as per our formulation there are 2 off slots in block 19).

Constraint 3a

In Monday- Thursday block, number of games possible is equal to the difference of maximum number of games possible and the number of off slots in a block

$$\sum_{i=1}^N \sum_{j=1}^N X_{i,j,k} \leq (G - H[k]) \quad \text{for } k \in [1, B]$$

Where G is the maximum number of possible game in Monday-Thursday block and H[k] is the parameter defined in Table 2.

Constraint 3b

In Saturday block, number of games possible is equal to the difference of maximum number of games possible (GS) and the number of off slots in a block

$$\sum_{i=1}^N \sum_{j=1}^N X_{i,j,k} \leq (GS - H[k]) \quad \text{for } k \in [1, B]$$

Where GS is the maximum number of possible game in Saturday block and H[k] is the parameter defined in Table 2.

Constraint 4:

$Y_{i,j}$ is used for displaying the results of block numbers in which games are scheduled.

$$Y_{i,j} = \sum_{k=1}^B (k \times X_{i,j,k}) \quad \text{for } i, j \in [1, N], i \neq j$$

$Y_{i,j}$ equals the value of k when team i plays with team j in block k . For this combination of i, j , k , $X_{i,j,k}$ equals 1 and 0 for all other values of k . $Y_{i,j}$ gets the value of the block number k in which the teams i and j played with each other.

Constraint 5:

$X_{i,j,k}$ is defined as binary variable which ensures the program assigns a value of either 0 or 1 as previously described in decision variables for any value of i, j and k

$$X_{i,j,k} \text{ is binary}$$

AMPL

[Following is the AMPL model file:](#)

```
# Parameters used in the model

param B;           # Total number of blocks
param G;           # Maximum number of games possible in a Monday-Thursday block excluding holidays
param GS;          # Maximum number of games possible in a Saturday block excluding holidays
param N;           # Number of teams participating in the league
param R;           # Number of times each team must receive every other team
param M;           # Maximum number of games a team can play in a Monday-Thursday block
param MS;          # Maximum number of games a team can play in a Saturday block
param H{k in 1..B}; # Number of off-slots in block k
set MT;            # Set of Block numbers having Monday-Thursday defined as block
set S;             # Set of Block numbers having Saturday defined as block

# Variables used in the model

var X{i in 1..N,j in 1..N,k in 1..B} binary; # X takes a value of 1 when team i receives team j in block k
var Y{i in 1..N,j in 1..N:i!=j}; # defined for creating a schedule display matrix and Y takes block number

# Objective Function

maximize Z: 0;

subject to

# Each team must receive every other team exactly R times
Constraint1{i in 1..N,j in 1..N:i!=j}: sum{k in 1..B} X[i,j,k]=R;

# In Monday- Thursday block, each team can play at most M times
Constraint2a{k in MT, i in 1..N}: sum{t in 1..N:t!=i} X[i,t,k] + sum{t in 1..N:t!=i} X[t,i,k]<=M;

# In Saturday block, each team can play at most MS times
Constraint2b{k in S, i in 1..N}: sum{t in 1..N:t!=i} X[i,t,k] + sum{t in 1..N:t!=i} X[t,i,k]<=MS;

# The game capacity of each block cannot be exceeded

## For Monday- Thursday Block
Constraint3a{k in MT}: sum{i in 1..N,j in 1..N} X[i,j,k]<=(G-H[k]);

## For Saturday Block
Constraint3b{k in S}: sum{i in 1..N,j in 1..N} X[i,j,k]<=(GS-H[k]);

# Y displays block number in which a game is scheduled
Constraint4{i in 1..N,j in 1..N:i!=j}: Y[i,j] = sum{k in 1..B}(k * X[i,j,k]);
```

Following is the AMPL data file:

```

param B:= 19; # Total number of blocks
param G:= 4; # Maximum number of games possible in a monday-Thursdays block
param GS:= 4; # Maximum number of games possible in a saturday block
param N:= 9; # Number of teams participating in the league
param R:= 1; # Number of times each team must receive every other team
param M:= 1; # Maximum number of games a team can play in a monday to thursday block
param MS:= 1; # maximum number of games a team can play on a saturday
set MT= 1 3 5 7 8 10 12 14 16 18 19; # Set of Block numbers having Monday-Thursdays defined as block
set S= 2 4 6 9 11 13 15 17; # Set of Block numbers having Saturday defined as block

param H:= 1 0 # Number of off-slots in block k
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
14 0
15 0
16 0
17 0
18 0
19 2;

```

Date	Day	Block No.
05/29/2006	Monday	19
05/30/2006	Tuesday	
05/31/2006	Wednesday	
06/01/2006	Thursday	

Following is the AMPL output file:

```

sw: ampl
ampl: reset;
ampl: reset;
ampl: option solver cplex;
ampl: model little_league_baseball_model.txt;
ampl: data little_league_baseball_data.txt;
ampl: solve;
CPLEX 12.6.3.0: optimal integer solution; objective 684
1175 MIP simplex iterations
0 branch-and-bound nodes
No basis.
ampl: display Y;
Y [*,*]

```

	1	2	3	4	5	6	7	8	9
1	.	17	14	2	3	8	4	9	11
2	18	.	7	14	15	10	9	11	5
3	16	4	.	10	13	3	11	12	6
4	6	3	5	.	1	4	7	15	9
5	12	8	2	11	.	17	6	10	4
6	15	2	9	18	16	.	13	7	14
7	1	12	15	8	14	5	.	16	17
8	13	1	18	17	5	6	2	.	8
9	7	16	1	13	18	12	10	3	.

Game Schedule Matrix (i vs j in block k)

:=

Teams 1 to 9 playing away games

Team 5 plays home game with team 7 in block no. 6

Teams 1 to 9 playing home games

Results

This formulation of the problem as integer linear model, gives the results as shown in table 3. The variable Y_{ij} represents the block number in which the team i plays with team j . Hence, this model allows for flexibility to schedule the game i vs j in any slot for a given block k as specified by Y_{ij} instead of assigning a fixed slot number. For example, as per *Table 2*, team 1 plays with team 2 in block 18. Hence, we can schedule the game 1 vs 2 on either Monday, Tuesday, Wednesday or Thursday between 22nd-25th May.

This kind of flexibility is not valid for games scheduled in blocks 2,4,6,9,11,13,15,17 because these blocks represent Saturdays and on Saturdays all games will have to happen simultaneously i.e. not in a consecutive order. The result shown in *Table 3* is one of the possible schedules for the given problem. This sample did not schedule any games in block 19 because we set our objective function to minimize the duration of the league and two possible extra slots are available. So, our model forced the schedule to fit into the first 18 blocks successfully there by ending the season on 25th May, 2006.

Game vs Slot Assignment Rubric ($Y_{i,j}$)					Mon-Thu Game		1,3,5,7,8,10,12,14,16,18		
					Saturday Game		2,4,6,9,11,13,15,17		
i \ j	1	2	3	4	5	6	7	8	9
1	-	17	14	2	3	8	4	9	11
2	18	-	7	14	15	10	9	11	5
3	16	4	-	10	13	3	11	12	6
4	6	3	5	-	1	4	7	15	9
5	12	8	2	11	-	17	6	10	4
6	15	2	9	18	16	-	13	7	14
7	1	12	15	8	14	5	-	16	17
8	13	1	18	17	5	6	2	-	8
9	7	16	1	13	18	12	10	3	-

Table 3: One of the possible schedule

What-If analysis

The consulting team has provided a set of scenarios and their possible outcomes using 'What-If' analysis. This helps the client understand the scope of solution model for the given scheduling problem and also run test cases for any anticipated changes in future scheduling requirements.

1. Are there any critical days or game slots so that, if those slots were not available, then it would become impossible to finish the tournament on time without adding other game slots?

Answer: There are **NO** critical game slots in the season calendar provided by the client which could hamper the schedule.

Rationale: In the schedule given by the client consider Monday-Thursday as one block and Saturday as another block. Both these blocks have 4 game slots and no team can play more than one game in each block due to additional requirements. These two conditions in synergy makes them look equivalent when it comes to assigning games. So irrespective of the position of the block in the given calendar each of them has equal priority. This scenario corresponds to no criticality across all the first 18 blocks.

The last two additional game slots in block 19 act as a buffer to the scheduling problem due to coverage of the minimum requirement of 72 game slots (9x8 games) by the first 18 full 4-game blocks which makes last two game slots complimentary in nature.

2. What is the number of full 4-game blocks that would be needed to complete the tournament as a function of the total number of teams?

Analysis: The plot between number of teams participating and the min number of full 4-game blocks needed to complete the tournament is displayed below.

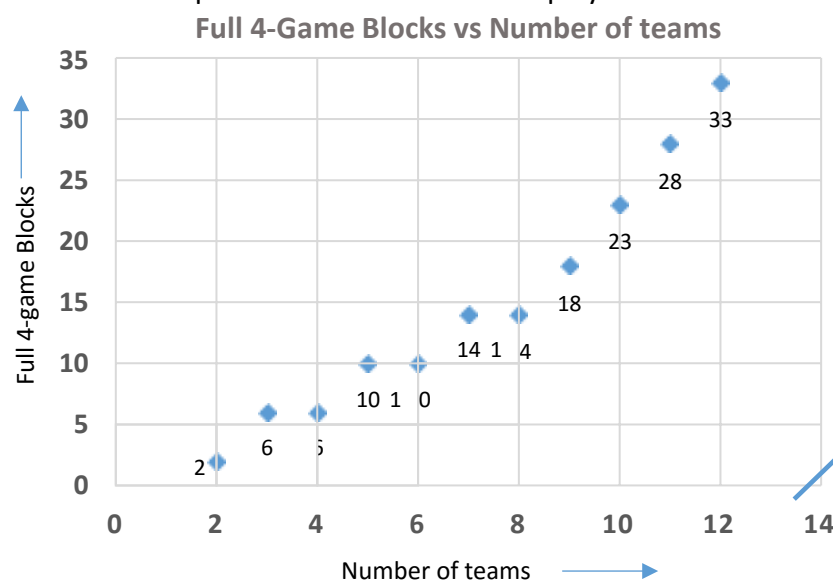


Fig 1: Plot for Number of teams vs Minimum number of full 4-game blocks required

This is achieved by adding an objective function which calculates the number of 4 game blocks required for conducting the league with any number of teams. Following is a new model file with above objective function.

```
# Parameters used in the model

param B;          # Total number of blocks
param N{k in 1..B}; # Maximum number of games allowed in a block
param T;          # Number of teams participating in the league
param H;          # Number of times each team must receive every other team
param G;          # Maximum number of games a team can play in a block

# Variables used in the model

var X{i in 1..T,j in 1..T,k in 1..B} binary; # X takes a value of 1 when team i receives team j in block k
var Y{i in 1..T,j in 1..T:i!=j}; # defined for creating a schedule display matrix and Y takes block number

# Objective Function

minimize duration: ceil((T*(T-1))/(if floor(T/2)<G then (max(floor(T/2),G)) else min(floor(T/2),G)));

subject to

# Each team must receive every other team exactly H times
Constraint1{i in 1..T,j in 1..T:i!=j}: sum{k in 1..B} X[i,j,k]=H;

# the game capacity of eachblock cannot be exceeded
Constraint2{k in 1..B}: sum{i in 1..T,j in 1..T:i!=j} X[i,j,k]<=N[k];
```

3. What if Friday slots are opened for scheduling games and Saturdays can host an extra game? Can I reduce the current block count from 19 if so what is the minimum no. of blocks required?

Analysis: Opening up additional game slots such as on Fridays and an additional game on Saturdays doesn't reduce the duration of the league with 9 teams participating. This is because of the presence of odd number of teams which leads to a team sitting idle in every fully scheduled game block hosting 8 teams (4 matches = 4 teams pairs).

However this provides scope for adding another team to the league without extending the duration i.e total number of game slots are increased from current 74 ($18 \times 4 + 2$) to 92 ($18 \times 5 + 2$). Adding a 10th team to the league will not affect the duration of league as it requires 90 games slots (10×9)

Data File Modifications: Number of slots in each block increased to 5 i.e. change values of parameters G and GS from 4 to 5

```
param B:= 19;
param G:= 4;
param GS:= 4;
param N:= 9;
param R:= 1;
param M:= 1;
param MS:= 1;
```

Change from 4 to 5



4. What if a team can play more than once in a Monday-Thursday block? Will it reduce the duration of league?

Analysis: Allowing a team to play more than once in a Monday-Thursday block doesn't reduce the duration of the league. This is because the number of games to be played remains the same and since all game slots have an equal priority the allotment of games to a particular block doesn't affect the overall duration of league.

However if the client decides to incorporate this requirement, a provision has been given to amend the data file accordingly as per their need.

Data File Modifications: Change value of parameters M from 1 to 2 as per client's need

```
param B:= 19;
param G:= 4;
param GS:= 4;
param N:= 9;
param R:= 1;
param M:= 1;
param MS:= 1;
```

Change from 1 to 2

Sample Output

Y	[*,*]	1	2	3	4	5	6	7	8	9	:=
1	.	12	17	6	9	7	15	5	8		
2	4	.	2	14	19	16	7	13	10		
3	11	6	.	13	3	18	9	12	18		
4	16	11	8	.	5	3	17	2	9		
5	2	14	4	15	.	1	16	6	17		
6	7	17	10	14	14	.	11	15	13		
7	13	1	16	10	7	1	.	12	10		
8	18	9	5	19	11	4	8	.	1		
9	8	15	3	5	3	12	6	18	.		

Team 9 plays twice in block 3

5. What if the number of game slots per block is reduced?

Analysis: If maximum number of games that can be scheduled in a block is reduced, the duration of the league increases. The plot between number of game slots per block and total number of blocks required to schedule the tournament is displayed next.

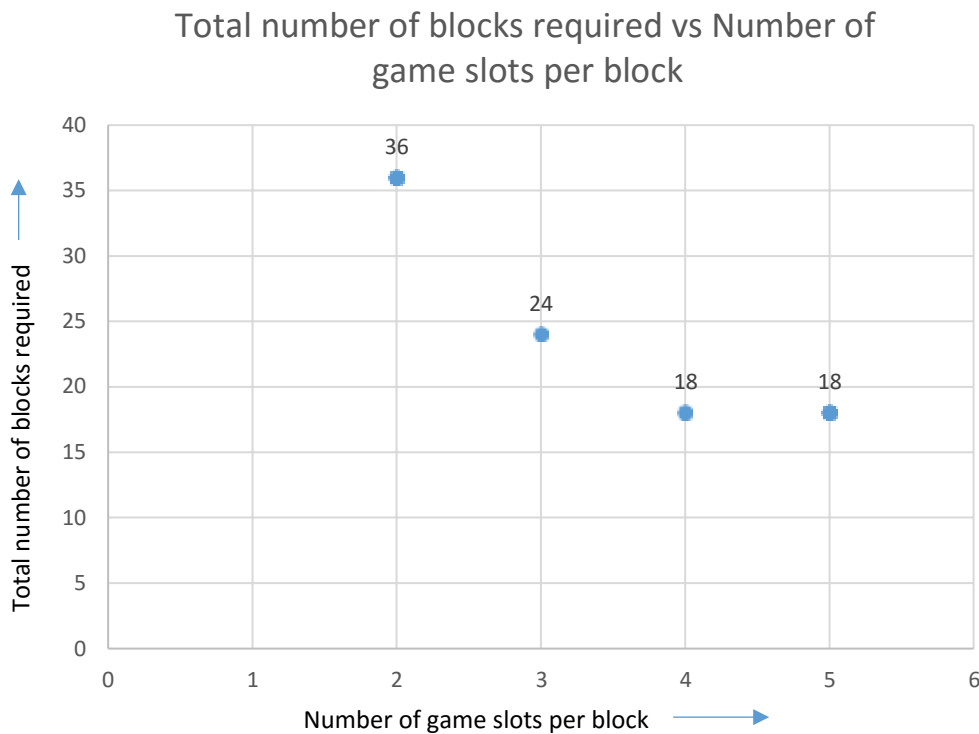


Fig 2: Plot for Number of game slots per block vs Total number of blocks required

Data File Modifications: Number of slots in each block reduced to 2 or 3 i.e. change values of parameters G and GS from 4 to 2 or 4 to 3

```
param B:= 19;
param G:= 4;
param GS:= 4;
param N:= 9;
param R:= 1;
param M:= 1;
param MS:= 1;
```

Change from 4 to 2 or 3

}

6. What if the number of game slots in blocks corresponding to Saturdays are reduced?

Analysis: If maximum number of games that can be scheduled in blocks corresponding to Saturdays are reduced, the duration of the league increases. The plot between number of game slots per Saturday block and total number of blocks required to schedule the tournament is displayed next

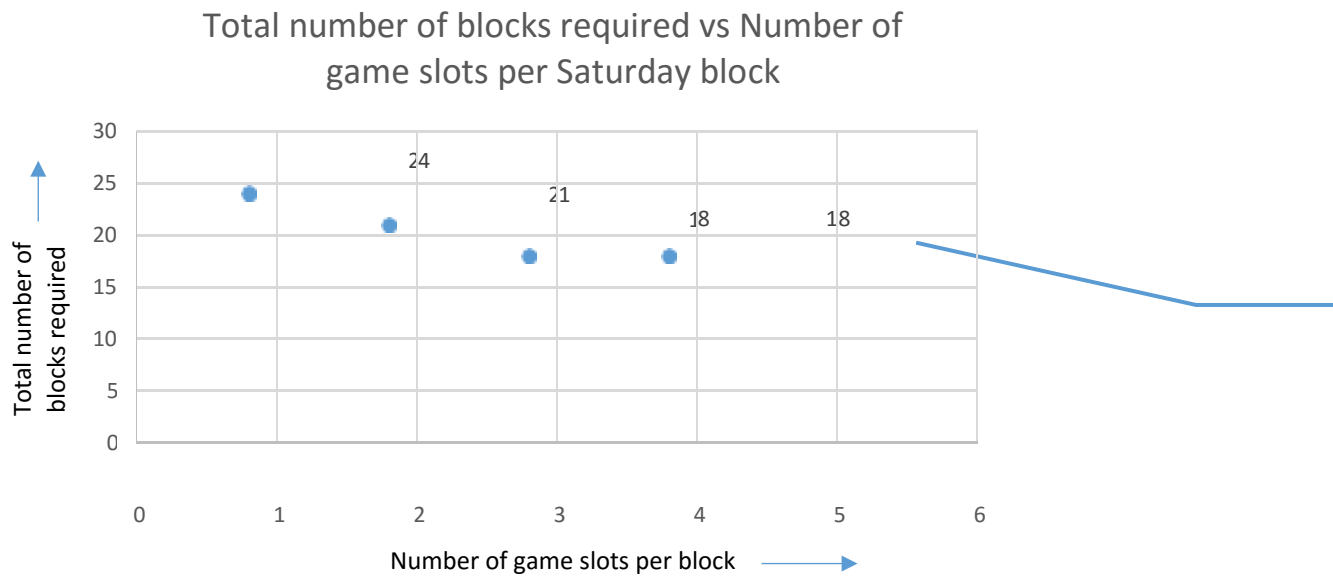


Fig 3: Plot for Number of game slots in a Saturday block vs Total number of blocks required

Data File Modifications: Change value of parameter Gs from 4 to 2 or 3 as per client's need

```
param B:= 19;
param G:= 4;
param GS:= 4;
param N:= 9;
param R:= 1;
param M:= 1;
param MS:= 1;
```

→ Change from 4 to 2 or 3

7. What if a pair teams should not play with in consecutive blocks? OR In other words, we need a separation of minimum one game block between a team playing 2nd time with same team?

Analysis: There is possibility of scheduling the league in the given duration in case the client requires games between any pair of teams not to be scheduled in consecutive blocks.

This requires a special modification in the existing model file whereas the previous 'What-If' questions can be answered by making changes in the data file only. The necessary modifications to the model and data file are provided below.

Additional Constraint to be added in current Model file:

```
Constraint6{i in 1..N,j in 1..N,k in 1..(B-1):i!=j}: X[i,j,k] <= (1-X[j,i,k+1]);
```

Sample Output:

```

Y  [*,*]
:  1    2    3    4    5    6    7    8    9    :=
1  .    3    6    2   12    7   15   10   17
2  9    .   11   17   14    1   12    6    7
3  8   13    .   16    5   15   10   14    9
4  11  10    7    .   13    8    1    9   19
5  1    4   17   15    .    9    3    2   18
6  18    5   12    3    6    .   16   11   14
7  5    2    4   14    8   13    .   17   11
8  4    8    3    5   16   19    7    .   15
9  13   16    1    4   10    2    6   12    .
;
```

For example, as seen in above sample output, teams 1 and 2 compete each other in block 3 and 9.

Manual for installing and using AMPL

AMPL is a tool specifically designed for mathematical programming. It is one of the most widely used programming languages having a great similarity of its syntax to mathematical notation of optimization problems. This allows for a very concise and readable definition of problems in the domain of optimization.

Because of this simple and easy to use interface the consulting team has used it in solving the given scheduling problem and strongly suggests the client to use the same for any of their current and future optimization projects.

Solver

Solver is the actual number crunching algorithm that takes input from AMPL i.e the mathematical equations and gives an optimal output that best matches the client's requirements. An integration between AMPL with SOLVER helps in modelling language for describing optimization data, variables, objectives, and constraints giving an optimal output to large scale optimization problems.

AMPL License Information

The consulting team will not be able to share AMPL software resources used for solving the scheduling problem instead the client can purchase a full license from the following official site <http://ampl.com/try-ampl/buy-ampl/>

Model and Data Files

A model file is a simple text file which contains all the input parameters, variables and constraints used for solving the optimization problem. It is given as an input file to AMPL.

A data file is a simple text file which contains parameters used in the Model file along with their numerical values which are given in the form of scalars or multi-dimensional vectors depending on the type of parameter used. It is given as an input file to AMPL in conjunction with the corresponding Model file. The previously loaded Solver tool optimizes the objective value for the given set of conditions in the Model file using the values loaded in the Data file.

Solver tools available

There are many Solvers available on AMPL platform such as CPLEX, GUROBI, EXPRESS etc. but among all CPLEX was chosen by the consulting team because of the nature of the optimization problem which resembles an Integer Linear Programming model. Also, CPLEX is also one of the most widely used Solver tool for large scale optimization problems because of its efficiency and robustness.

Schematic representation of AMPL working

AMPL reads the model from the **.mod** file, data from the **.dat** file and puts them together into a format that the solver understands. Then, it hands over this problem instance to the solver, which in turn, solves the instance, and hands back the solution to AMPL.

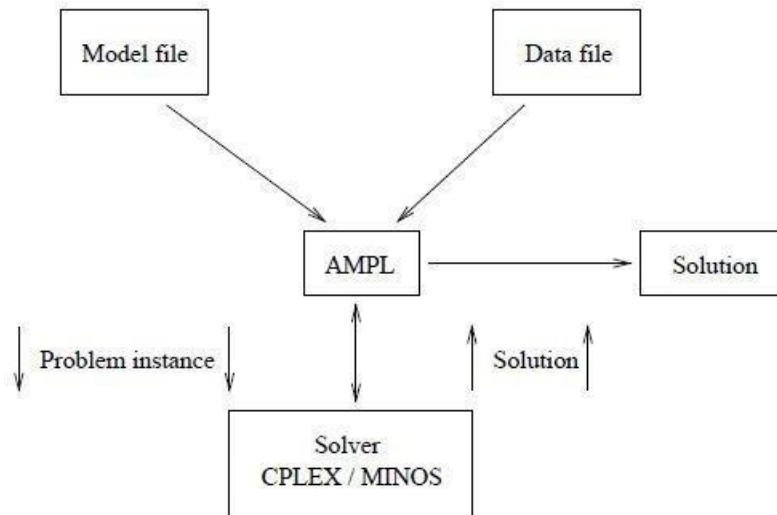


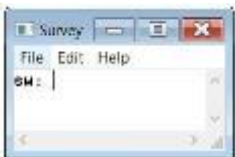


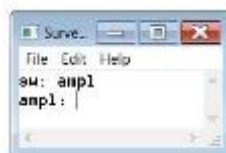
Fig 4: Schematic diagram of AMPL working

Run AMPL, Load Solver(CPLEX), run Model and Data File and Display Output

Once the AMPL software and Solver tool are installed on the system, copy the given model and data file provided by the consulting team and paste both of them in the same folder where the AMPL software and Solver tool are installed. Follow the below mentioned steps to run the model and data files.

STEP 1: Double click on **sw.exe**    # opens a command dialogue box

STEP 2: Type **"ampl"** (without the quotes) and hit enter. You should get the prompt: **ampl:**



STEP 3: The commands illustrated below are for running the model and data files of the client's baseball scheduling problem. The not-bolded text is what the computer outputs for you. The text after the symbol # are comments from the consulting team:

```
ampl: option solver cplex;
ampl: reset;
ampl: reset;
ampl: model p2t08mod.txt;
ampl: data p2t08dat.txt;
```



```
# selects CPLEX as the solver
# don't forget to reset each time; each error
# don't forget to reset each time; each error
# inputs model file
# inputs data file
```

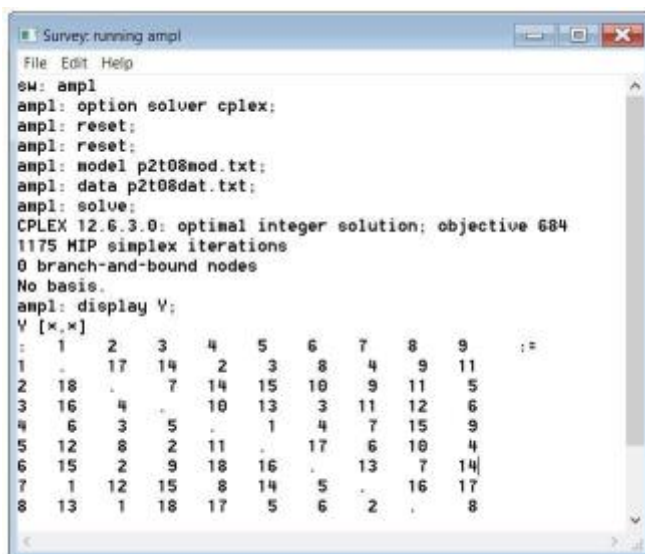
AMPL may complain after the command "**model p2t08mod.txt**" if there are any syntax errors in the model file. If so, then you need to modify your model, save the file again, and start from the reset; command above.

AMPL may complain after the command "**data p2t08dat.txt;**" if there are any syntax errors in the data file or if there is some conflicting information between the model and the data file. If so, then you need to modify either your model and or your data file, save the file again, and start from the reset; command above.

STEP 4: Post resolving all the error queries in model and data file, reset the program using commands mentioned above and proceed to solve the problem and display the output using further commands mentioned below

```
ampl: solve;                                # invokes CPLEX to optimize
CPLEX 12.6.3.0: optimal integer solution; objective 684
1175 MIP simplex iterations
0 branch-and-bound nodes
No basis.
```

```
ampl: display Y;                            # AMPL will display optimal variable values
```



Making changes to a Data file

The Integer Linear Programming model built for solving the client's problem is given a flexibility to enable the client to make changes in the model for meeting their future requirements at their own discretion such as increasing the team count, adding extra holidays in the calendar etc. This can be achieved by making a few changes (circled in red) to the **Data** file as illustrated by a few test cases mentioned below. After every change to the Data file save it and re-run the AMPL model as indicated in the former section – How to run AMPL and none of these require a change in Model file.

Model file should never be changed unless the client is faced with a situation which can't be solved by this model suggested by the consulting team. In that case, a new model has to be prepared based on the new requirements.

- 1) Increasing the run length of baseball season by changing the schedule end date from 05/31/2006 to 06/08/2006 (assuming existing holiday pattern of Friday and Sunday remains the same)

Following is the current AMPL data file:

```
param B:= 19; # Total number of blocks
param G:= 4; # Maximum number of games possible in a block
param N:= 9; # Number of teams participating in the league
param R:= 1; # Number of times each team must receive every other team
param M:= 1; # Maximum number of games a team can play in a block
```

```
param H:= 1 0 # Number of off-slots in block k
          2 0
          3 0
          4 0
          5 0
          6 0
          7 0
          8 0
          9 0
          10 0
          11 0
          12 0
          13 0
          14 0
          15 0
          16 0
          17 0
          18 0
          19 2
```

Adding an extra Sat (03/06) and a Mon-Thu block (06/05- 06/08) in the schedule increases the block count by 2

So change "param B:= 19;" to "param B:= 21"

Date	Day	Block No.
05/29/2006	Monday	19
05/30/2006	Tuesday	
05/31/2006	Wednesday	
06/01/2006	Thursday	

Defines Holidays/Off Slots in a particular block k

Off-slot 1

Off-slot 2

Parameter H defines the no. of off slots (or holidays) in each block. Since the no. of blocks has been increased from 19 to 21 without any change in regular holidays i.e Fri and Sun. The client needs to define this input to AMPL by adding two additional numbers/blocks 20, 21 to parameter H and their corresponding off slot count i.e 0 in both the additional blocks. Use these commands "20 0" and "21 0;"

- 2) Increasing the no. of teams participating from 9 to 11 and changing the maximum no. of games a team can play in a particular block k from 1 to 2.

```
param B:= 19; # Total number of blocks
param G:= 4; # Maximum number of games possible in a block
param N:= 9; # Number of teams participating in the league
param R:= 1; # Number of times each team must receive every other team
param M:= 1; # Maximum number of games a team can play in a block
```

Increasing the no. of games a team can play in a particular block k leads to change in count of parameter "M" from 1 to 2.

So change "param M:= 1;" to
"param M:= 2;"

Adding two extra teams leads to change in count of parameter "N" from 9 to 11.

So change "param N:= 9;" to
"param N:= 11;"

- 3) Increasing the maximum no. of games possible in a block from 4 to 5 and changing the no. of times each team must receive every other team from 1 to 2.

```
param B:= 19; # Total number of blocks
param G:= 4; # Maximum number of games possible in a block
param N:= 9; # Number of teams participating in the league
param R:= 1; # Number of times each team must receive every other team
param M:= 1; # Maximum number of games a team can play in a block
```

Adding two extra teams leads to change in count of parameter "N" from 9 to 11.

So change "param G:= 4;" to
"param G:= 5;"

Changing the condition that each team must receive every other team to 2 leads to change in count of parameter "R" from 1 to 2.

So change "param R:= 1;" to
"param R:= 2;"

Interpreting AMPL Solution

An example of scheduling matrix has been represented below to make the client understand and interpret results outputted by the AMPL model

Following is the AMPL output file:

```
sw: ampl
ampl: reset;
ampl: reset;
ampl: option solver cplex;
ampl: model little_league_baseball_model.txt;
ampl: data little_league_baseball_data.txt;
ampl: solve;
CPLEX 12.6.3.0: optimal integer solution; objective 684
1175 MIP simplex iterations
0 branch-and-bound nodes
No basis.
ampl: display Y;
Y [*,*]
```

	1	2	3	4	5	6	7	8	9
1	.	17	14	2	3	8	4	9	11
2	18	.	7	14	15	10	9	11	5
3	16	4	.	10	13	3	11	12	6
4	6	3	5	.	1	4	7	15	9
5	12	8	2	11	.	17	6	10	4
6	15	2	9	18	16	.	13	7	14
7	1	12	15	8	14	5	.	16	17
8	13	1	18	17	5	6	2	.	8
9	7	16	1	13	18	12	10	3	.

Game Schedule Matrix (i vs j in block k)

Teams 1 to 9 playing away games

Team 5 plays home game with team 7 in block no. 6

Teams 1 to 9 playing home games

Approach II

In this approach, all the available calendar days are divided into 74 game slots and all possible games are allocated into these slots instead of using time blocks. Details of slots are mentioned below.

Day	Date	week No	Day no	Slot No
Monday	3/13/2006	1	1	1
Tuesday	3/14/2006	1	2	2
Wednesday	3/15/2006	1	3	3
Thursday	3/16/2006	1	4	4
Friday	3/17/2006	1	*	*
Saturday	3/18/2006	1	5,6,7,8	5,6,7,8
Sunday	3/19/2006	1	*	*
Monday	3/20/2006	2	1	9
Tuesday	3/21/2006	2	2	10
Wednesday	3/22/2006	2	3	11
Thursday	3/23/2006	2	4	12
Friday	3/24/2006	2	*	*
Saturday	3/25/2006	2	5,6,7,8	13,14,15,16
Sunday	3/26/2006	2	*	*

Day	Date	week No	Day no	Slot No
Monday	4/10/2006	5	*	*
Tuesday	4/11/2006	5	*	*
Wednesday	4/12/2006	5	*	*
Thursday	4/13/2006	5	*	*
Friday	4/14/2006	5	*	*
Saturday	4/15/2006	5	*	*
Sunday	4/16/2006	5	*	*
Monday	4/17/2006	6	1	29
Tuesday	4/18/2006	6	2	30
Wednesday	4/19/2006	6	3	31
Thursday	4/20/2006	6	4	32
Friday	4/21/2006	6	*	*
Saturday	4/22/2006	6	5,6,7,8	33,34,35,36
Sunday	4/23/2006	6	*	*

Day	Date	week No	Day no	Slot No
Monday	5/8/2006	9	1	53
Tuesday	5/9/2006	9	2	54
Wednesday	5/10/2006	9	3	55
Thursday	5/11/2006	9	4	56
Friday	5/12/2006	9	*	*
Saturday	5/13/2006	9	5,6,7,8	57,58,59,60
Sunday	5/14/2006	9	*	*
Monday	5/15/2006	10	1	61
Tuesday	5/16/2006	10	2	62
Wednesday	5/17/2006	10	3	63
Thursday	5/18/2006	10	4	64
Friday	5/19/2006	10	*	*
Saturday	5/20/2006	10	5,6,7,8	65,66,67,68
Sunday	5/21/2006	10	*	*

Day	Date	week No	Day no	Slot No
Monday	3/27/2006	3	1	17
Tuesday	3/28/2006	3	2	18
Wednesday	3/29/2006	3	3	19
Thursday	3/30/2006	3	4	20
Friday	3/31/2006	3	*	*
Saturday	4/1/2006	3	5,6,7,8	21,22,23,24
Sunday	4/2/2006	3	*	*
Monday	4/3/2006	4	1	25
Tuesday	4/4/2006	4	2	26
Wednesday	4/5/2006	4	3	27
Thursday	4/6/2006	4	4	28
Friday	4/7/2006	4	*	*
Saturday	4/8/2006	4	*	*
Sunday	4/9/2006	4	*	*

Day	Date	week No	Day no	Slot No
Monday	4/24/2006	7	1	37
Tuesday	4/25/2006	7	2	38
Wednesday	4/26/2006	7	3	39
Thursday	4/27/2006	7	4	40
Friday	4/28/2006	7	*	*
Saturday	4/29/2006	7	5,6,7,8	41,42,43,44
Sunday	4/30/2006	7	*	*
Monday	5/1/2006	8	1	45
Tuesday	5/2/2006	8	2	46
Wednesday	5/3/2006	8	3	47
Thursday	5/4/2006	8	4	48
Friday	5/5/2006	8	*	*
Saturday	5/6/2006	8	5,6,7,8	49,50,51,52
Sunday	5/7/2006	8	*	*

Day	Date	week No	Day no	Slot No
Monday	5/22/2006	11	1	69
Tuesday	5/23/2006	11	2	70
Wednesday	5/24/2006	11	3	71
Thursday	5/25/2006	11	4	72
Friday	5/26/2006	11	*	*
Saturday	5/27/2006	11	*	*
Sunday	5/28/2006	11	*	*
Monday	5/29/2006	12	*	*
Tuesday	5/30/2006	12	2	73
Wednesday	5/31/2006	12	3	74

Holiday

Table 4: Slot numbers for corresponding d & w

Parameters

1. **N**: Number of teams participating in the league. Here, $N = 9$.
2. **W**: Number of weeks. Here, $W=12$.
3. **G**: Maximum possible number of games in a week. Here, $G = 8$.
4. **F**: Maximum possible number of games from Monday to Thursday or on a Saturday in a week. Here, $F=4$.
5. **P**: Number of times each team must receive every other team. Here, $P=1$
6. **R**: Maximum number of games possible on a day from Monday - Thursday or in a slot on Saturday. Here, $R = 1$.
7. **T**: Maximum number of games for a team on Monday - Thursday block or on a Saturday. Here, $T = 1$
8. **H[d,w]**: Gets the value 1 if the corresponding day is holiday otherwise 0
9. **K[d,w]**: Slot number ranging from 1-74 (possible number of game slots)
10. **U[d,w]**: Calendar Dates (MDDYY format) between which the entire league is scheduled

Decision Variables

1. $X_{i,j,d,w} = \begin{cases} 1, & \text{if team } i \text{ plays with team } j \text{ in slot } d \text{ of week } w, \\ & \text{for } i, j \in [1, N], d \in [1, B] \text{ } i \neq j \\ 0, & \text{otherwise} \end{cases}$

$X_{i,j,d,w}$ is a binary variable and takes the value of 1 when team i plays with team j in a slot d of week w . It takes a value of 0 for all other possible combinations.

2. Y_{ij} displays a matrix where i^{th} row and j^{th} column corresponds to the slot number in which team i plays with team j .
3. Z_{ij} displays a matrix where i^{th} row and j^{th} column corresponds to the calendar date on which team i plays with team j .

Objective Function

The objective function in this approach is to assign all possible games in the given schedule.

$$\text{Maximize } L: 0$$

Constraints

Constraint 1:

Each team must receive every other team exactly once over the entire schedule meaning each team plays a home game and an away with every other team. This makes sure a combined total of 2 games(home+away) for any combination of i and j .

$$\sum_{d=1}^G \sum_{w=1}^W X_{i,j,d,w} = P \quad \text{for } i, j \in [1, N], i \neq j$$

Constraint 2:

As per the client's requirement, at most 1 game can be scheduled on each day from Monday to Thursday in a week. This ensures that at most 4 games can be scheduled between Monday to Thursday in any particular week w

$$\sum_{i=1}^N \sum_{j=1}^N \sum_{d=1}^F X_{i,j,d,w} \leq F \quad \text{for } w \in [1, W], i \neq j$$

Constraint 3:

As per the client's requirement at most 4 games can be scheduled on a Saturday in any particular week w .

$$\sum_{i=1}^N \sum_{j=1}^N \sum_{d=F+1}^G X_{i,j,d,w} \leq F \quad \text{for } w \in [1, W], i \neq j$$

Constraint 4:

On a any particular day from Monday to Thursday at most one game can be scheduled to play which means that all the slots(excluding holidays) corresponding to Monday, Tuesdays , Wednesdays and Thursdays can host at most one game.

$$\sum_{i=1}^N \sum_{j=1}^N X_{i,j,d,w} \leq R \quad \text{for } d \in [1, F], w \in [1, W], i \neq j$$

Constraint 5:

On any particular Saturday at most 4 games can be scheduled to play simultaneously which means that all the slots (excluding holidays) corresponding Saturdays can host at most one game.

$$\sum_{i=1}^N \sum_{j=1}^N X_{i,j,d,w} \leq R \quad \text{for } d \in [F + 1, G], w \in [1, W], i \neq j$$

Constraint 6:

As per the clients requirement each team can play at most one game (both home and away games combined) between Monday to Thursday in any particular week w .

$$\sum_{d=1}^F \sum_{\substack{j=1 \\ j \neq i}}^N X_{i,j,d,w} + \sum_{d=1}^F \sum_{\substack{j=1 \\ j \neq i}}^N X_{j,i,d,w} \leq T \quad \text{for } w \in [1, W], i \in [1, N]$$

Constraint 7:

As per the clients requirement each team can play at most one game (both home and away games combined) on a Saturday in any particular week w . This can also be explained from the fact that all the matches taking place on Saturday run simultaneously.

$$\sum_{\substack{t=1 \\ t \neq i}}^N X_{i,t,d,w} + \sum_{\substack{t=1 \\ t \neq i}}^N X_{t,i,d,w} \leq T \quad \text{for } d \in [F+1, G], w \in [1, W], i \in [1, N]$$

Constraint 8:

A game cannot be scheduled on a holiday. This makes sure that whenever there is a predefined holiday given as an input there won't be any game scheduled on that particular day.

$$H_{d,w} \leq 1 - X_{i,j,d,w} \quad \text{for } d \in [1, G], w \in [1, W], i, j \in [1, N], i \neq j$$

Constraint 9:

Display Constraint - Y displays the slot number in which team i plays with team j . The corresponding slot no. can be matched to a particular calendar as defined in *Table 4*.

$$Y_{i,j} = \sum_{d=1}^G \sum_{w=1}^W K_{d,w} \times X_{i,j,d,w} \quad \text{for } i, j \in [1, N], i \neq j$$

Constraint 10:

Display Constraint - Z displays the calendar date on which team i plays with team j .

$$Z_{i,j} = \sum_{d=1}^G \sum_{w=1}^W U_{d,w} \times X_{i,j,d,w} \quad \text{for } i, j \in [1, N], i \neq j$$

Constraint 11:

$X_{i,j,d,w}$ is defined as binary variable which ensures the program assigns a value of either 0 or 1 as previously described in decision variables for all possible combinations of i, j, d and w .

$$X_{i,j,d,w} \text{ is binary}$$

AMPL

[Following is the AMPL model file:](#)

```
# Parameters used in the model
param N;           # Number of teams participating in league
param W;           # Number of weeks
param G;           # Maximum possible number of games in a week
param F;           # Maximum possible number of games from Monday to Thursday or on a Saturday in a week
param P;           # Number of times each team must receive every other team
param R;           # Maximum number of games possible on a day from Monday - Thursday or in a slot on Saturday
param T;           # Maximum number of games for a team on Monday - Thursday block or on a Saturday
param H{d in 1..G,w in 1..W}; # gets the value 1 if the corresponding day is holiday otherwise 0
param K{d in 1..G,w in 1..W}; # slot number ranging from 1-74 (possible number of game slots)
param U{d in 1..G,w in 1..W}; # Date corresponding to a day and week
# Variables used in the model

var X{i in 1..N,j in 1..N,d in 1..G,w in 1..W:i!=j} binary; # X receives 1 when team i receives j on game day d of week w
var Y{i in 1..N,j in 1..N:i!=j}; # defined for creating a schedule display matrix and Y takes slot number
var Z{i in 1..N,j in 1..N:i!=j}; # defined for creating a schedule display matrix and Z takes corresponding date

# Objective Function

maximize Schedule: 0; # this is just a scheduling problem

subject to

# Team i receives team j exactly P times.
Constraint1{i in 1..N,j in 1..N:i!=j}: sum{w in 1..W,d in 1..G} X[i,j,d,w] =P;

# Maximum F games in a Monday to Thursday block in a week
Constraint2{w in 1..W}: sum{i in 1..N,j in 1..N,d in 1..F:i!=j} X[i,j,d,w]<=F;

# Maximum F games on a Saturday
Constraint3{w in 1..W}: sum{i in 1..N,j in 1..N,d in (F+1)..G:i!=j} X[i,j,d,w]<=F;

# Atmost R games in one day from Monday to Thursday
Constraint4{d in 1..F,w in 1..W}: sum{i in 1..N,j in 1..N:i!=j} X[i,j,d,w]<=R;

# Atmost R games in a slot on a Saturday
constraint5{d in (F+1)..G,w in 1..W}: sum{ i in 1..N,j in 1..N:j!=i} X[i,j,d,w]<=R;

# A team can play atmost T games from Monday - Thursday of a week
Constraint6{i in 1..N,w in 1..W}: sum{d in 1..F,j in 1..N:j!=i} X[i,j,d,w] + sum{d in 1..F,j in 1..N:j!=i} X[j,i,d,w]<=T;

# A team can play atmost T games on a Saturday
Constraint7{d in (F+1)..G, i in 1..N,w in 1..W}: sum{t in 1..N:t!=i} X[i,t,d,w] + sum{t in 1..N:t!=i} X[t,i,d,w]<=T;

# A game cannot be scheduled on a holiday
Constraint8{d in 1..G, w in 1..W,i in 1..N,j in 1..N:i!=j}: H[d,w]<= (1-(X[i,j,d,w]));

# Display Constraint - Y displays the slot number in which team i plays with team j
Displayslot{i in 1..N,j in 1..N:i!=j}: Y[i,j] = sum{d in 1..G,w in 1..W} (K[d,w]*X[i,j,d,w]);

# Display Constraint - Z displays the date on which team i plays with team j
DisplayDate{i in 1..N, j in 1..N:i!=j}: Z[i,j]=sum{d in 1..G, w in 1..W}X[i,j,d,w]*U[d,w];
```

[Following is the AMPL data file:](#)

```

param N:=9; # Number of teams participating in league
param W:=12; # Number of weeks
param G:=8; # Maximum possible number of games in a week
param F:=4; # Maximum possible number of games from Monday to Thursday or on a Saturday in a week
param P:=1; # Number of times each team must receive every other team
param R:=1; # Maximum number of games possible on a day from Monday - Thursday or in a slot on Saturday
param T:=1; # Maximum number of games for a team on Monday - Thursday block or on a Saturday

param H: 1 2 3 4 5 6 7 8 9 10 11 12:= # gets the value 1 if the corresponding day is holiday otherwise 0
1 0 0 0 0 1 0 0 0 0 0 0 1
2 0 0 0 0 1 0 0 0 0 0 0 0
3 0 0 0 0 1 0 0 0 0 0 0 0
4 0 0 0 0 1 0 0 0 0 0 0 1
5 0 0 0 1 1 0 0 0 0 0 1 1
6 0 0 0 1 1 0 0 0 0 0 1 1
7 0 0 0 1 1 0 0 0 0 0 1 1
8 0 0 0 1 1 0 0 0 0 0 1 1;

param K: 1 2 3 4 5 6 7 8 9 10 11 12:= # slot number ranging from 1-74 (possible number of game slots)
1 1 9 17 25 0 29 37 45 53 61 69 0
2 2 10 18 26 0 30 38 46 54 62 70 73
3 3 11 19 27 0 31 39 47 55 63 71 74
4 4 12 20 28 0 32 40 48 56 64 72 0
5 5 13 21 0 0 33 41 49 57 65 0 0
6 6 14 22 0 0 34 42 50 58 66 0 0
7 7 15 23 0 0 35 43 51 59 67 0 0
8 8 16 24 0 0 36 44 52 60 68 0 0;

param U: 1 2 3 4 5 6 7 8 9 10 11 12:= # gets the date corresponding to game slot on a day and week
1 31306 32006 32706 40306 41006 41706 42406 50106 50806 51506 52206 52906
2 31406 32106 32806 40406 41106 41806 42506 50206 50906 51606 52306 53006
3 31506 32206 32906 40506 41206 41906 42606 50306 51006 51706 52406 53106
4 31606 32306 33006 40606 41306 42006 42706 50406 51106 51806 52506 60106
5 31806 32506 40106 40806 41506 42206 42906 50606 51306 52006 52706 60306
6 31806 32506 40106 40806 41506 42206 42906 50606 51306 52006 52706 60306
7 31806 32506 40106 40806 41506 42206 42906 50606 51306 52006 52706 60306
8 31806 32506 40106 40806 41506 42206 42906 50606 51306 52006 52706 60306;

```


Following is the AMPL output file:

```

ampl: model little_league_baseball_model_2.txt;
ampl: data little_league_baseball_data_2.txt;
ampl: solve;
CPLEX 12.6.3.0: optimal integer solution; objective 0
0 MIP simplex iterations
0 branch-and-bound nodes
No basis.

```

```

ampl: display Y;
Y [*,*]

```

	1	2	3	4	5	6	7	8	9
1	.	1	9	17	25	29	37	45	53
2	61	.	69	10	18	26	30	38	46
3	73	54	.	2	62	19	27	31	39
4	70	74	47	.	55	63	5	28	32
5	13	21	33	40	.	3	11	71	41
6	49	57	65	6	48	.	56	12	72
7	14	22	34	42	50	58	.	4	20
8	66	7	15	23	35	43	64	.	51
9	59	67	8	16	24	36	44	52	.

Game vs Slot Matrix (i vs j in a slot defined by d and w)

Teams 1 to 9 playing away games

Team 4 plays team 6 in slot no. 63

Teams 1 to 9 playing home games

```

ampl: display Z;
Z [*,*]

```

	1	2	3	4	5	6	7	8	9
1	.	31306	32006	32706	40306	41706	42406	50106	50806
2	51506	.	52206	32106	32806	40406	41806	42506	50206
3	53006	50906	.	31406	51606	32906	40506	41906	42606
4	52306	53106	50306	.	51006	51706	31806	40606	42006
5	32506	40106	42206	42706	.	31506	32206	52406	42906
6	50606	51306	52006	31806	50406	.	51106	32306	52506
7	32506	40106	42206	42906	50606	51306	.	31606	33006
8	52006	31806	32506	40106	42206	42906	51806	.	50606
9	51306	52006	31806	32506	40106	42206	42906	50606	.

Game vs Calendar Date Matrix (i vs j on a particular date)

Teams 1 to 9 playing away games

Denotes Month (May)

Denotes Year (2006)

5 06 06

Denotes Day (6th)

Team 9 plays team 8 on 6th May, 2006

Teams 1 to 9 playing home games

Results

This formulation of the problem as integer linear model gives the results as shown in table 5. The variable $Y_{i,j}$ represents the slot number in which team i plays with team j . For example, team 3 plays with team 4 on slot number 2 which corresponds to the date 14th March, 2006 as shown in Table 3. An addition of two more holidays can be included on a week day from Monday to Thursday. No further holidays can be added on a Saturday of any week as these will cut down the number of slots by 4 for every holiday. This model re adjusts itself to accommodate the additional two holidays and gives an output similar to the one shown in Table 5.

Schedule Rubric ($Y_{i,j}$)									
i \ j	1	2	3	4	5	6	7	8	9
1	-	1	9	17	25	29	37	45	53
2	61	-	69	10	18	26	30	38	46
3	73	54	-	2	62	19	27	31	39
4	70	74	47	-	55	63	5	28	32
5	13	21	33	40	-	3	11	71	41
6	49	57	65	6	48	-	56	12	72
7	14	22	34	42	50	58	-	4	20
8	66	7	15	23	35	43	64	-	51
9	59	67	8	16	24	36	44	52	-

Table 5: Game vs Slot number matrix

The table below shows the output in which variable $Z_{i,j}$ represents the date on which team i plays with team j . It is similar to the schedule rubric reflected in the above Table 5 except that the corresponding slot numbers are replaced with their corresponding calendar dates which makes it easier for the client instead of matching the slot numbers with Table 4.

Schedule Rubric ($Z_{i,j}$) With Calendar Dates									
i \ j	1	2	3	4	5	6	7	8	9
1	-	31306	32006	32706	40306	41706	42406	50106	50806
2	51506	-	52206	32106	32806	40406	41806	42506	50206
3	53006	50906	-	31406	51606	32906	40506	41906	42606
4	52306	53106	50306	-	51006	51706	31806	40606	42006
5	32506	40106	42206	42706	-	31506	32206	52406	42906
6	50606	51306	52006	31806	50406	-	51106	32306	52506
7	32506	40106	42206	42906	50606	51306	-	31606	33006
8	52006	31806	32506	40106	42206	42906	51806	-	50606
9	51306	52006	31806	32506	40106	42206	42906	50606	-

Table 6: Game vs Calendar date matrix

The model provided in the Approach II is comparatively less flexible since it schedules games as per the calendar provided with the problem statement.

Conclusion

Two different models along with their respective approaches are provided for solving the scheduling problem raised by Little League Baseball College Station. Both the models have suggested that a feasible solution exists for scheduling the entire league in compliance with requirements set forth by the client. Each model has its own advantages in terms of flexibility. In approach 1, the model outputs a block number for a game and the end user is required to choose one of the four possible slots in that block whereas in approach 2, the model outputs a slot number or a corresponding calendar date for a particular game. In approach 2, the objective function has been chosen to maximize 0 which is used just for a scheduling assignment and has no direct relation to optimize other dynamics of the league whereas in approach 1, the objective function has been chosen to minimize the duration of the league which in turn might reduce the cost of conducting the league. Since the problem is limited to scheduling both these objective functions are valid. The end user is free to choose an approach that benefits them the most.

References

1. *Linear Programming* retrieved from https://en.wikipedia.org/wiki/Linear_programming
2. *Integer Programming* retrieved from https://en.wikipedia.org/wiki/Integer_programming
3. *AMPL* retrieved from <https://en.wikipedia.org/wiki/AMPL>
4. *Parameter* retrieved from <https://en.oxforddictionaries.com/definition/parameter>
5. AMPL guide provided by Prof. Erick Moreno Centeno as part of course material for ISEN620