Sports Scheduling in Video Games

Richard Gibson

CMPUT 621 Project
University of Alberta

April 17, 2009

Outline

Outline

Outline

1. Background and motivating example
2. How I think the example should be

Outline

1. Background and motivating example
2. How I think the example should be
3. Scheduling and the n-queens problem

Outline

1. Background and motivating example
2. How I think the example should be
3. Scheduling and the n-queens problem
4. How I fix the example

Outline

1. Background and motivating example
2. How I think the example should be
3. Scheduling and the n-queens problem
4. How I fix the example
5. Summary and a final example schedule

Background and Motivation

Background and Motivation

- I like sports video games.

Background and Motivation

- I like sports video games.
- Many sports video games today include a league or season mode.

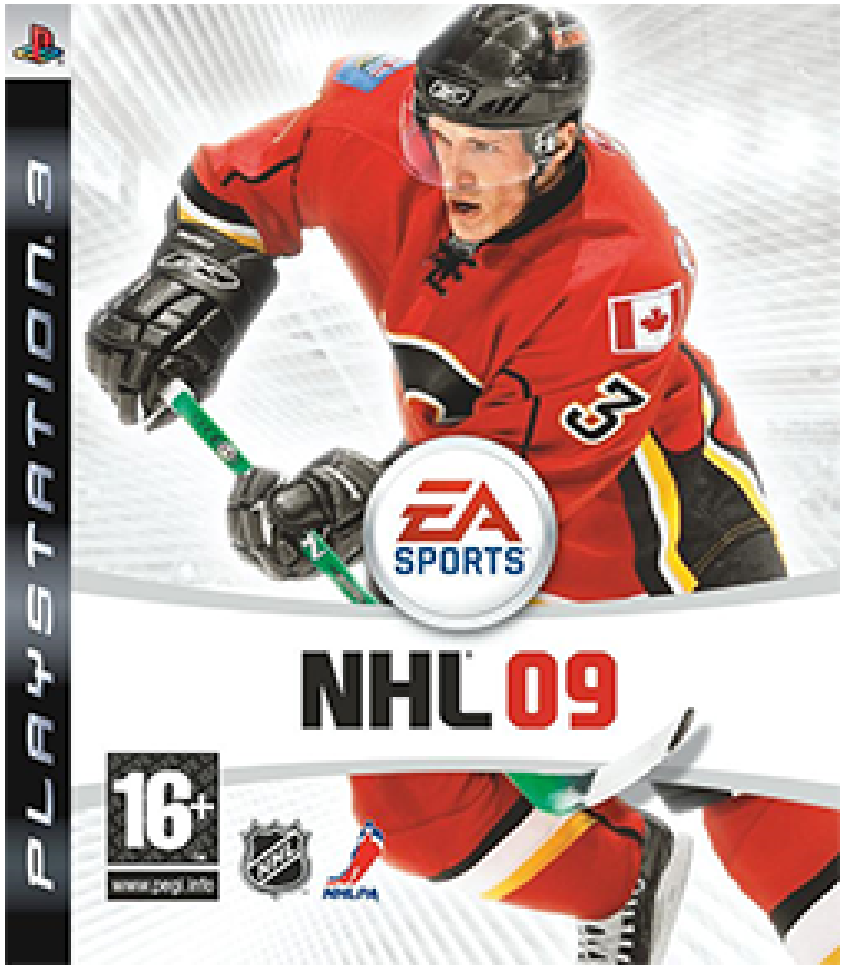
Background and Motivation

- I like sports video games.
- Many sports video games today include a league or season mode.
- However, most of these games do not allow the user (i.e. me) to adjust the league schedule to their liking.
  - Usually just a small number of hard-coded schedules to choose from.

Background and Motivation

- I like sports video games.
- Many sports video games today include a league or season mode.
- However, most of these games do not allow the user (i.e. me) to adjust the league schedule to their liking.
  - Usually just a small number of hard-coded schedules to choose from.
- My goal: Write a program that creates league schedules on-line specific to the user's (i.e. my) requests.

Example: NHL 09



Example: NHL 09

Western Conference

Northwest Division	Central Division	Pacific Division
Calgary	Chicago	Anaheim
Colorado	Columbus	Dallas
Edmonton	Detroit	Los Angeles
Minnesota	Nashville	Phoenix
Vancouver	St. Louis	San Jose

Eastern Conference

Northeast Division	Atlantic Division	Southeast Division
Boston	New Jersey	Atlanta
Buffalo	New York I	Carolina
Montreal	New York R	Florida
Ottawa	Philadelphia	Tampa Bay
Toronto	Pittsburgh	Washington

Example: NHL 09

Western Conference

Northwest Division	Central Division	Pacific Division
Calgary	Chicago	Anaheim
Colorado	Columbus	Dallas
Edmonton	Detroit	Los Angeles
Minnesota	Nashville	Phoenix
Vancouver	St. Louis	San Jose

Eastern Conference

Northeast Division	Atlantic Division	Southeast Division
Boston	New Jersey	Atlanta
Buffalo	New York I	Carolina
Montreal	New York R	Florida
Ottawa	Philadelphia	Tampa Bay
Toronto	Pittsburgh	Washington

Example: NHL 09

Example: NHL 09

- 6 games against divisional opponents, 4 games against other teams in same conference, 1 game against teams in other conference

Example: NHL 09

- 6 games against divisional opponents, 4 games against other teams in same conference, 1 game against teams in other conference
- Plus at most 28 playoff games after this regular season

Example: NHL 09

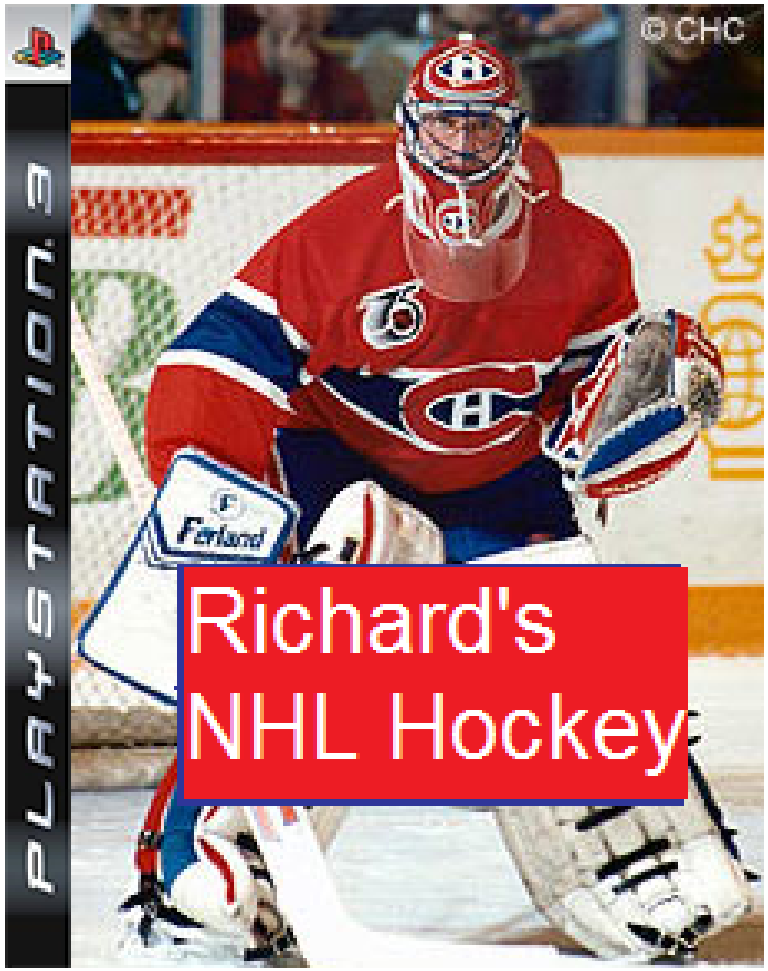
- 6 games against divisional opponents, 4 games against other teams in same conference, 1 game against teams in other conference
- Plus at most 28 playoff games after this regular season
- Say, 25 minutes per game = Up to about 46 hours of playing time

Example: NHL 09

- 6 games against divisional opponents, 4 games against other teams in same conference, 1 game against teams in other conference
- Plus at most 28 playoff games after this regular season
- Say, 25 minutes per game = Up to about 46 hours of playing time
- THIS IS TOO LONG! (for me)

Richard's NHL Hockey

Now, if I was to make a hockey video game:



Richard's NHL Hockey

Western Conference

Northwest Division	Central Division	Pacific Division
Calgary	Chicago	Anaheim
Colorado	Columbus	Dallas
Edmonton	Detroit	Los Angeles
Minnesota	Nashville	Phoenix
Vancouver	St. Louis	San Jose

Eastern Conference

Northeast Division	Atlantic Division	Southeast Division
Boston	New Jersey	Atlanta
Buffalo	New York I	Carolina
Montreal	New York R	Florida
Ottawa	Philadelphia	Tampa Bay
Toronto	Pittsburgh	Washington

Campbell Conference

Smythe Division


Calgary


Edmonton


Los Angeles


Phoenix


San Jose


Vancouver

Norris Division


Chicago


Columbus


Dallas


Detroit


Nashville


St. Louis

Prince of Wales Conference

Adams Division


Boston


Buffalo


Carolina


Montreal


Ottawa

Toronto

Patrick Division


New Jersey


New York I

New York R


Philadelphia


Pittsburgh


Washington

Richard's NHL Hockey

Richard Conference

West Division	East Division
 Calgary	 Montreal
 Edmonton	 Ottawa
 Vancouver	 Toronto

Richard's NHL Hockey

Richard Conference

West Division	East Division
Calgary	Montreal
Edmonton	Ottawa
Vancouver	Toronto

Richard's NHL Hockey

Richard Conference

West Division	East Division
Calgary	Montreal
Edmonton	Ottawa
Vancouver	Toronto

Number of games against divisional opponents =

Number of games against other teams in same conference =

Number of games against inter-conference opponents = 0

Richard's NHL Hockey

Richard Conference

West Division	East Division
Calgary	Montreal
Edmonton	Ottawa
Vancouver	Toronto

$$\begin{aligned} \text{Number of games against divisional opponents} &= 4 \\ \text{Number of games against other teams in same conference} &= 2 \\ \text{Number of games against inter-conference opponents} &= 0 \\ \text{Total number of games} = 4 \times 2 + 2 \times 3 &= 14 \end{aligned}$$

Away Home

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

Away Home

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

- About 6 hours of playing time to complete regular season!

Away Home

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

- About 6 hours of playing time to complete regular season!
- Question: Can we use answer-set programming to create these NHL hockey schedules on-line?

Scheduling Problem

Our scheduling problem is different from most common sports scheduling problems.

Scheduling Problem

Our scheduling problem is different from most common sports scheduling problems.

- Most prior sports scheduling research concerns round-robin schedules; however, our problem is more general.

Scheduling Problem

Our scheduling problem is different from most common sports scheduling problems.

- Most prior sports scheduling research concerns round-robin schedules; however, our problem is more general.
- Real-life professional sports schedules attempt to minimize travel distances; however, optimization is not an issue for us (we just want to find a feasible solution).

Scheduling Problem

Our scheduling problem is different from most common sports scheduling problems.

- Most prior sports scheduling research concerns round-robin schedules; however, our problem is more general.
- Real-life professional sports schedules attempt to minimize travel distances; however, optimization is not an issue for us (we just want to find a feasible solution).
- Real-life sports schedules often take days or even months to construct; however, efficiency is important for us as video gamers will not want to wait.

Scheduling Problem

The minimum requirements for our scheduling problem are:

Scheduling Problem

The minimum requirements for our scheduling problem are:

- Each of the n teams play the same number of games (determined by the user):
  - x games against divisional opponents
  - y games against other conference opponents
  - z games against inter-conference opponents

  (typically $x \geq y \geq z$)

Scheduling Problem

The minimum requirements for our scheduling problem are:

- Each of the n teams play the same number of games (determined by the user):
  - x games against divisional opponents
  - y games against other conference opponents
  - z games against inter-conference opponents

  (typically $x \geq y \geq z$)
- Each team plays an equal number of home games and away games.

Scheduling Problem

The minimum requirements for our scheduling problem are:

- Each of the n teams play the same number of games (determined by the user):
  - x games against divisional opponents
  - y games against other conference opponents
  - z games against inter-conference opponents

  (typically $x \geq y \geq z$)
- Each team plays an equal number of home games and away games.
- No team plays more than 1 game per day.

Scheduling Problem

The minimum requirements for our scheduling problem are:

- Each of the n teams play the same number of games (determined by the user):
  - x games against divisional opponents
  - y games against other conference opponents
  - z games against inter-conference opponents

  (typically $x \geq y \geq z$)
- Each team plays an equal number of home games and away games.
- No team plays more than 1 game per day.
- Schedule does not take long to build (say, at most 5 minutes).

Scheduling Problem

The minimum requirements for our scheduling problem are:

- Each of the n teams play the same number of games (determined by the user):
  - x games against divisional opponents
  - y games against other conference opponents
  - z games against inter-conference opponents

  (typically $x \geq y \geq z$)
- Each team plays an equal number of home games and away games.
- No team plays more than 1 game per day.
- Schedule does not take long to build (say, at most 5 minutes).

Simplifications:

Scheduling Problem

The minimum requirements for our scheduling problem are:

- Each of the n teams play the same number of games (determined by the user):
  - x games against divisional opponents
  - y games against other conference opponents
  - z games against inter-conference opponents

  (typically $x \geq y \geq z$)
- Each team plays an equal number of home games and away games.
- No team plays more than 1 game per day.
- Schedule does not take long to build (say, at most 5 minutes).

Simplifications:

- All divisions (conferences) must be of equal size.

Scheduling Problem

The minimum requirements for our scheduling problem are:

- Each of the n teams play the same number of games (determined by the user):
  - x games against divisional opponents
  - y games against other conference opponents
  - z games against inter-conference opponents

  (typically $x \geq y \geq z$)
- Each team plays an equal number of home games and away games.
- No team plays more than 1 game per day.
- Schedule does not take long to build (say, at most 5 minutes).

Simplifications:

- All divisions (conferences) must be of equal size.
- x, y, z must be even.

Scheduling Problem

The minimum requirements for our scheduling problem are:

- Each of the n teams play the same number of games (determined by the user):
  - x games against divisional opponents
  - y games against other conference opponents
  - z games against inter-conference opponents

  (typically $x \geq y \geq z$)
- Each team plays an equal number of home games and away games.
- No team plays more than 1 game per day.
- Schedule does not take long to build (say, at most 5 minutes).

Simplifications:

- All divisions (conferences) must be of equal size.
- x, y, z must be even.
- $n \leq 30$ and each team plays at most 82 games.

Scheduling and the Queens Problem

Example:

Richard Conference

West Division	East Division
 Calgary	 Montreal
 Edmonton	 Ottawa
 Vancouver	 Toronto

$$\begin{aligned} x &= 4 \\ y &= 2 \\ z &= 0 \end{aligned}$$

Scheduling and the Queens Problem

Example:

Richard Conference

West Division	East Division
 Calgary 2	 Montreal
 Edmonton 2	 Ottawa 4
 Vancouver 2	 Toronto 4

$$
\begin{aligned}
x &= 4 \\
y &= 2 \\
z &= 0
\end{aligned}
$$

Scheduling and the Queens Problem

Example:

Richard Conference

West Division	East Division
 Calgary 4	 Montreal 2
 Edmonton	 Ottawa 2
 Vancouver 4	 Toronto 2

$$\begin{aligned} x &= 4 \\ y &= 2 \\ z &= 0 \end{aligned}$$

Scheduling and the Queens Problem

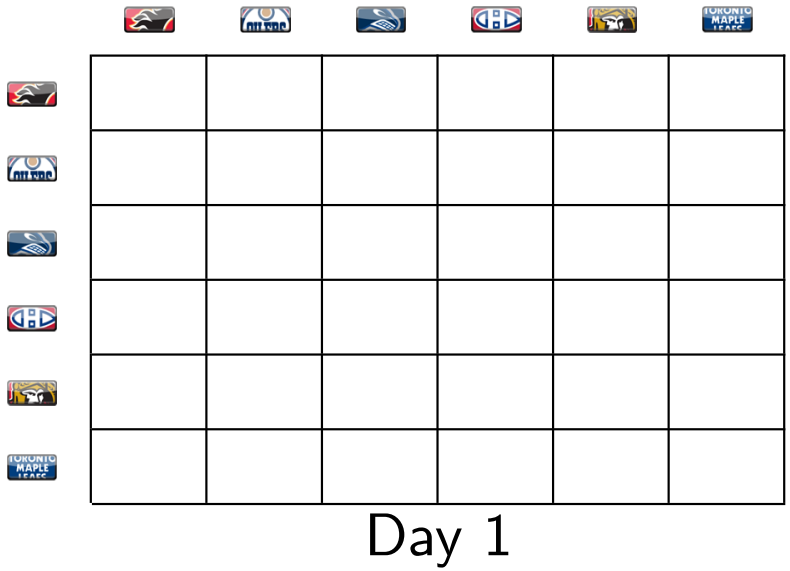
Example:

Richard Conference

West Division	East Division
 Calgary 2/2	 Montreal 1/1
 Edmonton	 Ottawa 1/1
 Vancouver 2/2	 Toronto 1/1

$$x = 4$$
$$y = 2$$
$$z = 0$$

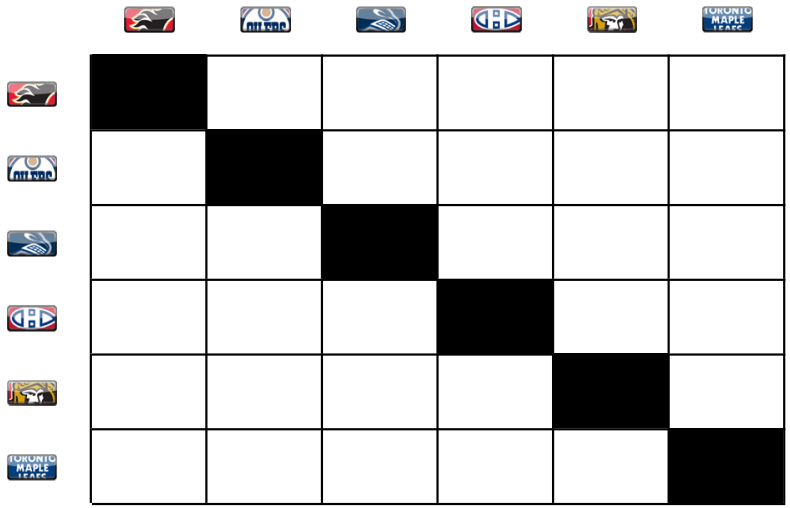
Away/Home



Day 1

Day 1 games:

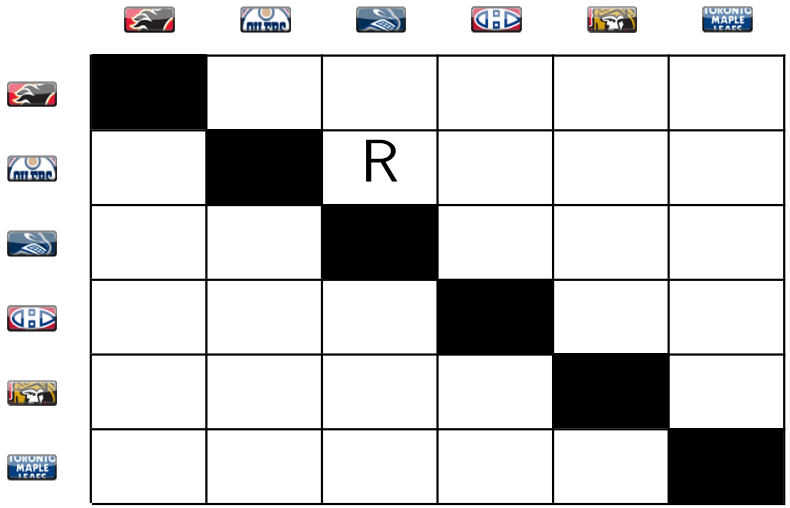
- Place rooks on $n \times n$ board to denote games



Day 1

Day 1 games:

- Place rooks on $n \times n$ board to denote games
- No team plays against itself

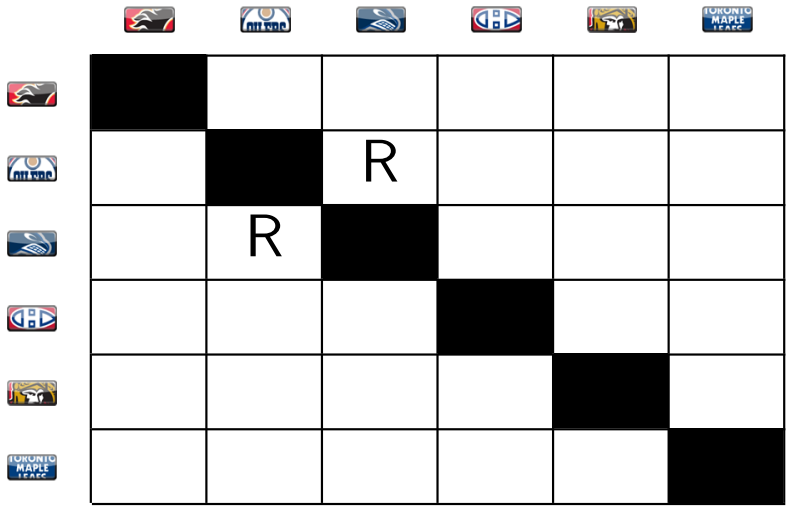


Day 1

Day 1 games:

vs.

- Place rooks on $n \times n$ board to denote games
- No team plays against itself

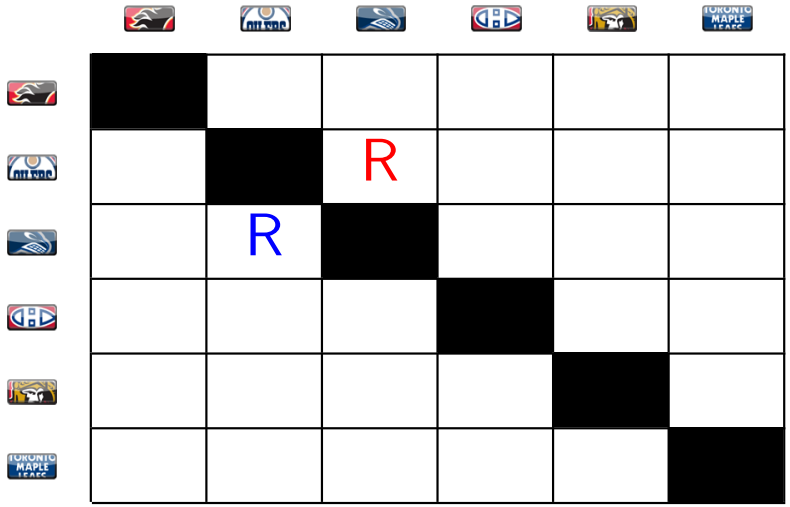


Day 1

Day 1 games:

vs.

- Place rooks on $n \times n$ board to denote games
- No team plays against itself
- Placements are "symmetric"

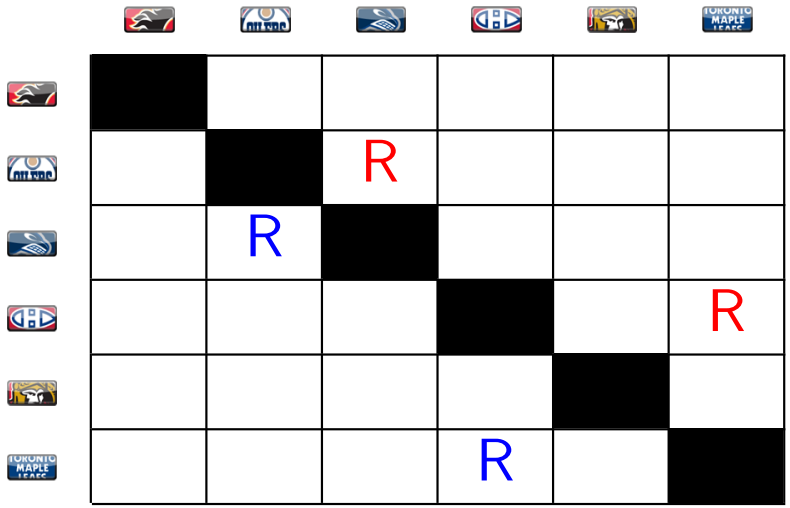


Day 1

Day 1 games:

 at 

- Place rooks on $n \times n$ board to denote games
- No team plays against itself
- Placements are "symmetric"
- Colour the rook blue if the row team is away, red if at home



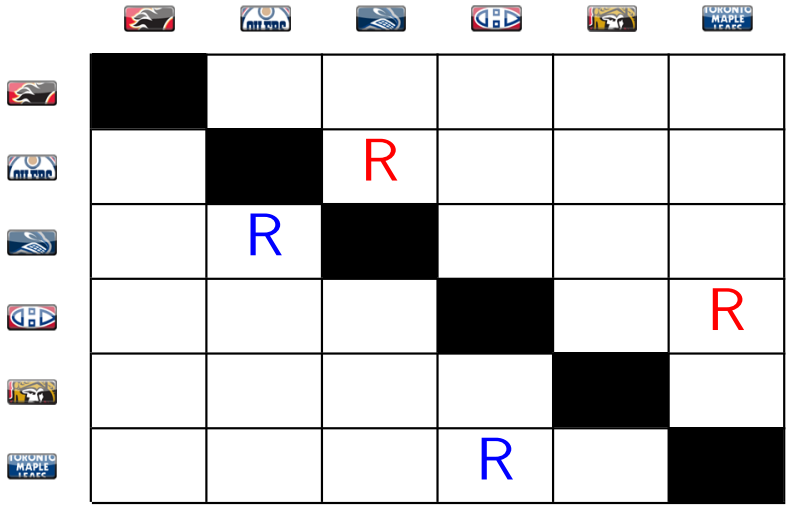
Day 1

Day 1 games:

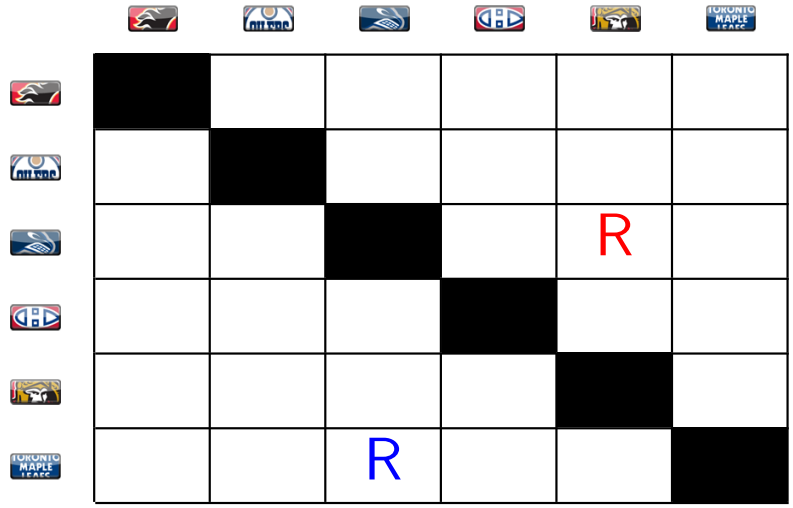
at

at

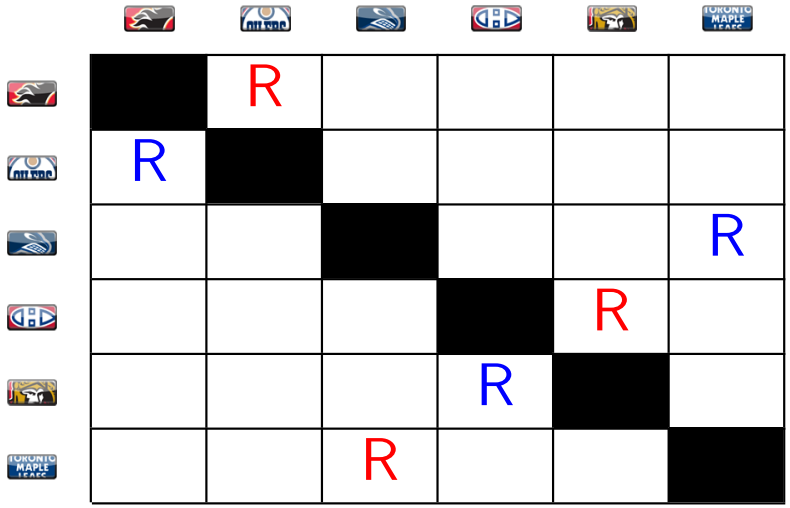
- Place rooks on $n \times n$ board to denote games
- No team plays against itself
- Placements are "symmetric"
- Colour the rook blue if the row team is away, red if at home
- No two rooks may attack each other, since no team may play more than 1 game per day



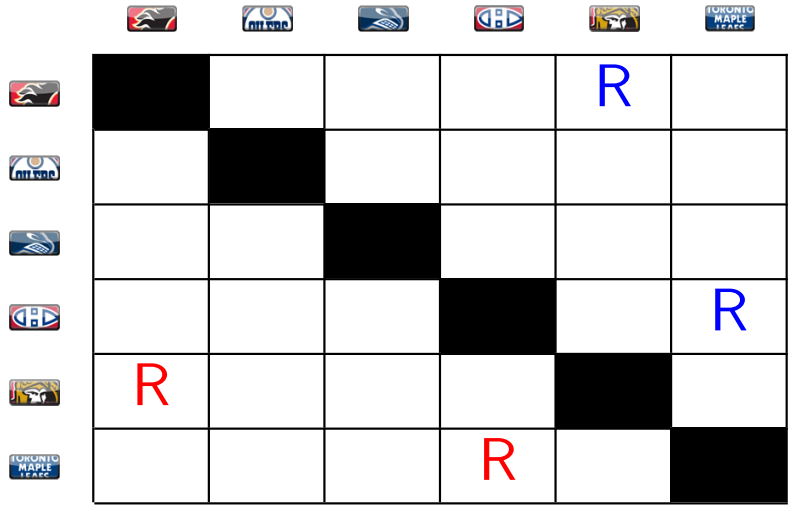
Day 1



Day 2



Day 3



Day 4

. . .

- Each square receives exactly $\frac{x}{2}$, $\frac{y}{2}$ or $\frac{z}{2}$ rooks of each colour respectively.

Minimum Encoding

Blocked n-queens Optimal Encoding

```
% Each row has exactly 1 queen
1{queen(R,C) : num(C)}1 :- num(R).
```

Minimum Encoding

Blocked n-queens Optimal Encoding

```
% Each row has exactly 1 queen
1{queen(R,C) : num(C)}1 :- num(R).
```

Schedule Builder

```
% No team plays more than one game per day
0{game(T1, T2, D, L) : isTeam(T2) : T1 ≠ T2 : location(L)}1 :-
    isTeam(T1), day(D).
```

Minimum Encoding

Blocked n-queens Optimal Encoding

```
% Each row has exactly 1 queen
1{queen(R,C) : num(C)}1 :- num(R).
```

Schedule Builder

```
% No team plays more than one game per day
0{game(T1, T2, D, L) : isTeam(T2) : T1 ≠ T2 : location(L)}1 :-
    isTeam(T1), day(D).

% Each team plays x games against divisional opponents
x/2{game(T1, T2, D, L) : day(D)}x/2 :-
    team(T1, Conf, Div),
    team(T2, Conf, Div),
    location(L),
    T1 ≠ T2.
```

(Similar for y and z)

Problems

This encoding has some problems.

Problems

This encoding has some problems.

- Problem 1: Using smodels, the maximum 30 team, 82 game case takes over 2.5 hours to create a schedule.

Problems

This encoding has some problems.

- Problem 1: Using smodels, the maximum 30 team, 82 game case takes over 2.5 hours to create a schedule.
- Solution: Use Clasp: only takes 20 seconds to create a schedule of the same size!

More problems

Problem 2: Ex. 6 team, 14 game schedule for Montreal:

Away Home

1 	2 	3 	4 	5 	6 	7 
8 	9 	10 	11	12	13 	14 
15 	16 	17	18	19	20	21
22	23	24	25	26	27	28

More problems

Problem 2: Ex. 6 team, 14 game schedule for Montreal:

Away Home

1 	2 	3 	4 	5 	6 	7 
8 	9 	10 	11	12	13 	14 
15 	16 	17	18	19	20	21
22	23	24	25	26	27	28

- Games are bunched together; this is bad if players get fatigued from playing too frequently.

Extra Constraint 1

So let's add two extra constraints:

Extra Constraint 1

So let's add two extra constraints:

No team plays more than G games per M consecutive days

```
{game(T1, T2, D, L) : isTeam(T2) : T1 ≠ T2 : day(D) : location(L) :
Dmin ≤ D : D ≤ Dmax}G :-
  isTeam(T1),
  day(Dmin),
  day(Dmax),
  Dmin < Dmax,
  Dmax − Dmin == M−1,
  atMost(G, M).

atMost(2,3).
atMost(3,5).
atMost(5,8).
```

Extra Constraint 2

So let's add two extra constraints:

Extra Constraint 2

So let's add two extra constraints:

Each team must play at least G games per M consecutive days

```
G{game(T1, T2, D, L) : isTeam(T2) : location(L) : day(D) : Dmin ¡= D :
D ¡= Dmax : T1 ≠ T2} :-
  isTeam(T1),
  day(Dmin),
  day(Dmax),
  Dmin < Dmax,
  Dmax − Dmin == M−1,
  atLeast(G, M).

atLeast(1,7).
```

Schedule with Extra Constraints

Ex. 6 team, 14 game schedule for Montreal:

Away Home

1	2 	3 	4	5	6 	7 
8	9	10 	11	12	13 	14
15 	16 	17	18 	19	20 	21 
22	23	24 	25 	26	27 	28

Schedule with Extra Constraints

Ex. 6 team, 14 game schedule for Montreal:

Away Home

1	2 	3 	4	5	6 	7 
8	9	10 	11	12	13 	14
15 	16 	17	18 	19	20 	21 
22	23	24 	25 	26	27 	28

- Games are now nicely spread out.

Schedule with Extra Constraints

Ex. 6 team, 14 game schedule for Montreal:

Away Home

1	2 	3 	4	5	6 	7 
8	9	10 	11	12	13 	14
15 	16 	17	18 	19	20 	21 
22	23	24 	25 	26	27 	28

- Games are now nicely spread out.
- Problem 3: Games against a particular team may be bunched.

Extra Constraint 3

So let's add a third extra contraint:

Extra Constraint 3

So let's add a third extra contraint:

No two teams play each other more than G times per M consecutive days

```
{game(T1, T2, D, L) : location(L) : day(D) : Dmin ≤ D : D ≤ Dmax}G :-
  isTeam(T1),
  isTeam(T2),
  T1 ≠ T2,
  day(Dmin),
  day(Dmax),
  Dmin < Dmax,
  Dmax − Dmin == M−1,
  diverse(G, M).

diverse(2,7).
diverse(3,14).
```

Schedule With Another Extra Constraint

Ex. 6 team, 14 game schedule for Montreal:

Away Home

1	2 	3 	4	5	6 	7
8 	9	10	11 	12	13 	14
15	16 	17 	18	19	20 	21
22 	23 	24	25 	26	27 	28 

Schedule With Another Extra Constraint

Ex. 6 team, 14 game schedule for Montreal:

Away Home

1	2 	3 	4	5	6 	7
8 	9	10	11 	12	13 	14
15	16 	17 	18	19	20 	21
22 	23 	24	25 	26	27 	28 

- Games against each team are now spread out nicely.

Schedule With Another Extra Constraint

Ex. 6 team, 14 game schedule for Montreal:

Away Home

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

- Games against each team are now spread out nicely.
- Problem 4: Away games and home games are bunched together.

Away/Home Bunching Problem

- How can we "unbunch" home games and away games?

Away/Home Bunching Problem

- How can we "unbunch" home games and away games?
- Maybe try something similar to the previous constraints:

Away/Home Bunching Problem

- How can we "unbunch" home games and away games?
- Maybe try something similar to the previous constraints:

No team can play more than G consecutive games at one location

```
{hasGameAt(T1, D1, L1) : day(D1) : Dmin ≤ D1 : D1 ≤ Dmax}G :-
  0{hasGameAt(T1, D2, L2) : day(D2) : Dmin ≤ D2 : D2 ≤ Dmax}0,
  isTeam(T1),
  location(L1),
  location(L2),
  L1 ≠ L2,
  day(Dmin),
  day(Dmax),
  Dmin + G ≤ Dmax,
  consecLoc(G).

consecLoc(4).
```

Away/Home Bunching Problem

- This extra constraint causes the program to run out of memory!

Away/Home Bunching Problem

- This extra constraint causes the program to run out of memory!
- This problem has yet to be solved completely, and thus this constraint is not included in the solution.

Constraint Summary and Another Problem

So we have added 3 extra constraints that have worked nicely for small schedules, but:

Constraint Summary and Another Problem

So we have added 3 extra constraints that have worked nicely for small schedules, but:

Problem 5: While a 6 team, 14 game schedule can be found almost instantly, the 30 team, 82 game schedule runs out of memory.

Constraint Summary and Another Problem

So we have added 3 extra constraints that have worked nicely for small schedules, but:

Problem 5: While a 6 team, 14 game schedule can be found almost instantly, the 30 team, 82 game schedule runs out of memory.

- Should we throw away some of the constraints? But then the schedule will have other problems again. How can we fix this problem?

Constraint Summary and Another Problem

So we have added 3 extra constraints that have worked nicely for small schedules, but:

Problem 5: While a 6 team, 14 game schedule can be found almost instantly, the 30 team, 82 game schedule runs out of memory.

- Should we throw away some of the constraints? But then the schedule will have other problems again. How can we fix this problem?
- After much my-head-to-wall banging...

Solution to Problem 5

Solution: If we alter the constraints, for example, as follows:

No team plays more than G games per M consecutive days (old)

```
{game(T1, T2, D, L) : isTeam(T2) : T1 ≠ T2 : day(D) : location(L) :
Dmin ≤ D : D ≤ Dmax}G :-
  isTeam(T1),
  day(Dmin),
  day(Dmax),
  Dmin < Dmax,
  Dmax − Dmin == M−1,
  atMost(G, M).

atMost(2,3).
atMost(3,5).
atMost(5,8).
```

Solution to Problem 5

Solution: If we alter the constraints, for example, as follows:

No team plays more than G games per M consecutive days (new)

```
:- G+1{game(T1, T2, D, L) : isTeam(T2) : T1 ≠ T2 : day(D) :
  location(L) : Dmin ≤ D : D ≤ Dmax}
  isTeam(T1),
  day(Dmin),
  day(Dmax),
  Dmin < Dmax,
  Dmax − Dmin == M−1,
  atMost(G, M).

atMost(2,3).
atMost(3,5).
atMost(5,8).
```

Solution to Problem 5

Solution: If we alter the constraints, for example, as follows:

No team plays more than G games per M consecutive days (new)

```
:- G+1{game(T1, T2, D, L) : isTeam(T2) : T1 ≠ T2 : day(D) :
  location(L) : Dmin ≤ D : D ≤ Dmax}
  isTeam(T1),
  day(Dmin),
  day(Dmax),
  Dmin < Dmax,
  Dmax − Dmin == M−1,
  atMost(G, M).

atMost(2,3).
atMost(3,5).
atMost(5,8).
```

then a 30 team, 82 game schedule can be found in under 2 minutes!

Summary

We have a program that can build NHL schedules.

Summary

We have a program that can build NHL schedules.

- Number of teams and their arrangement into divisions and conferences can be set by the user.

Summary

We have a program that can build NHL schedules.

- Number of teams and their arrangement into divisions and conferences can be set by the user.
- Number of games a team plays against division, conference, and inter-conference opponents can be set by the user.

Summary

We have a program that can build NHL schedules.

- Number of teams and their arrangement into divisions and conferences can be set by the user.
- Number of games a team plays against division, conference, and inter-conference opponents can be set by the user.
- Do not have to wait too long for a schedule to be created.

Summary

We have a program that can build NHL schedules.

- Number of teams and their arrangement into divisions and conferences can be set by the user.
- Number of games a team plays against division, conference, and inter-conference opponents can be set by the user.
- Do not have to wait too long for a schedule to be created.
- Conclusion: Answer-set programming is a good way to make sports schedules for video games!

Summary

We have a program that can build NHL schedules.

- Number of teams and their arrangement into divisions and conferences can be set by the user.
- Number of games a team plays against division, conference, and inter-conference opponents can be set by the user.
- Do not have to wait too long for a schedule to be created.
- Conclusion: Answer-set programming is a good way to make sports schedules for video games!
- However, home games and away games may still end up bunched together...

Home-and-Home series

In the real NHL, schedules often include "home-and-home" series of 2 games between divisional opponents. So let's add the following constraint:

Home-and-Home series

In the real NHL, schedules often include "home-and-home" series of 2 games between divisional opponents. So let's add the following constraint:

Divisional teams play at least one home-and-home series

```
homeAndHome(T1, T2) :-
  team(T1, Conf, Div), team(T2, Conf, Div),
  game(T1, T2, D1, L1), game(T1, T2, D2, L2),
  day(D1), day(D2),
  location(L1), location(L2),
  T1 ≠ T2, L1 ≠ L2,
  abs(D2 − D1) == 1, x > 2.

:- team(T1, Conf, Div), team(T2, Conf, Div),
  x > 2, T1 ≠ T2,
  not homeAndHome(T1, T2).
```

Final Example Schedule

This gives us the following 6 team, 14 game schedule for Montreal:

Away Home

1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28

- Home-and-home series on days 13-14 and days 22-23.

Thanks for listening!