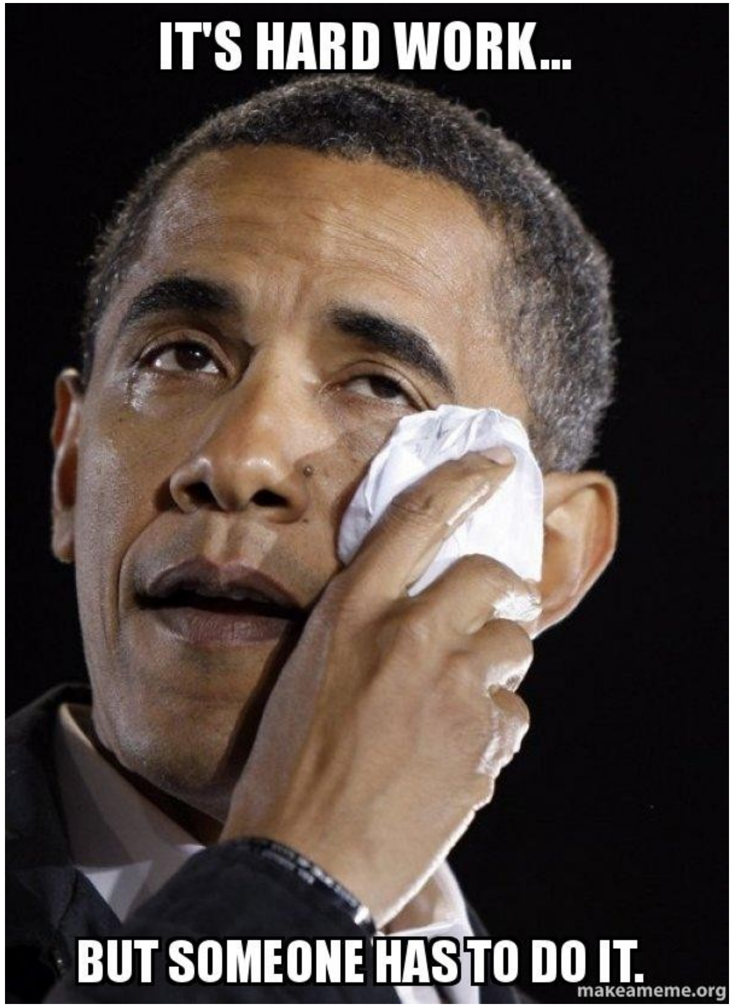
Learning To Schedule Sport Tournaments from Tensor Data

Tom De Bièvre

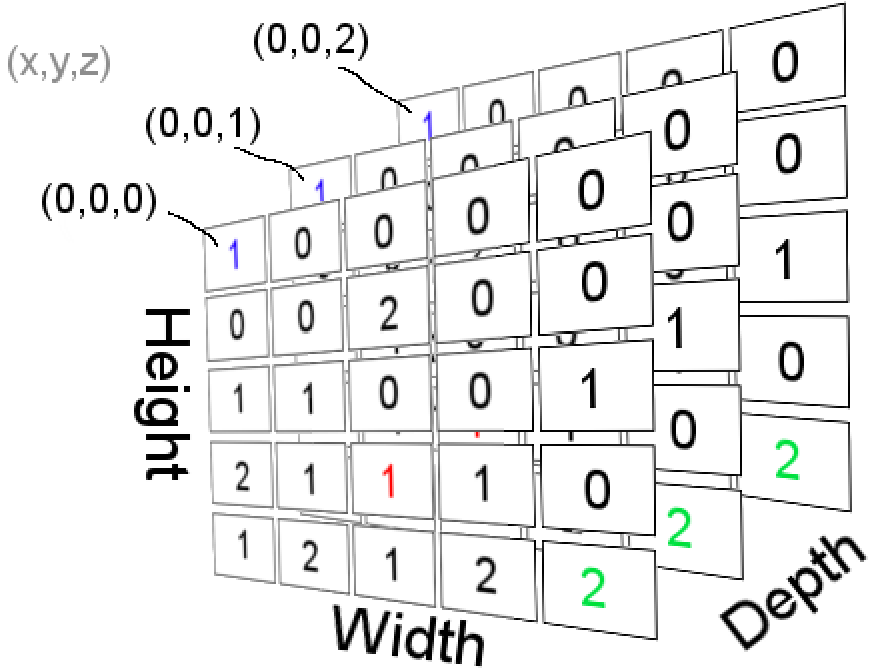
Context

- Many real world problems require constraints
- Sport scheduling is really complex and laboursome
- Major League Baseball season 2018-2019
    - 2 leagues
    - Each league consists of 3 divisions
    - 162 games per team per regular season
- Could this be automated to lower the amount of work?



Aim

- Constraint programming to make scheduling easier
- Learning/acquiring constraints using tensor manipulation
- Make schedule problems easier
- Particular interest in sports!



Approach: Literature study

- Different kinds sport schedules? Terminology?
  - Single / double elimination(cfr knockout phases)
  - Multilevel (cfr Boxing/chess)
  - Round Robin variations
- Cost functions
  - Minimum breaks
  - Minimum carry-over
  - Minimum travel distance

KU LEUVEN

Approach: Literature study

- Sport scheduling constraints (Nurmi et al, 2010)
  - Hard constraints
  - Soft constraints
- Previous work on sport scheduling?
  - A Model Seeker
  - Travelling tournament problem

KU LEUVEN

Approach: Literature study

- A Model Seeker: Extracting Global Constraint Models from Positive Examples Beldiceanu N., Simonis H. (2012)
  - Searching conjunctions of identical constraints
  - Global constraints → Edge cases?

Approach: Literature study

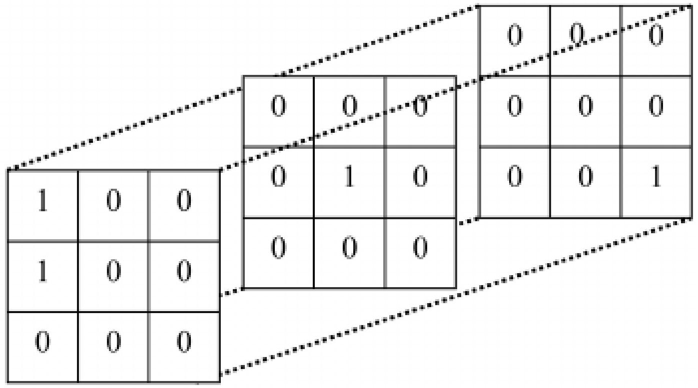
- Automating Personnel Rostering by Learning Constraints Using Tensors
  Kumar M., Teso S. De Causmaecker P., De Readt L. (2018)
- A Model Seeker: Extracting Global Constraint Models from Positive Examples
  Beldiceanu N., Simonis H. (2012)
  - Searching conjunctions of identical constraints
  - Global constraints → Edge cases?

Approach: Literature study

- The Traveling Tournament Problem: Description and Benchmarks
  Kelly Easton, George Nemhauser & Micheal Trick
  - Combining HAP patterns and Minimal distance travelling
  - Feasability & optimality combined
  - Constraint programming & integer programming
  - Unsolved if n>6

Approach: Research Questions

- To what extent could CountOR learn sportconstraints as is?
- Could we adapt/generify CountOR to CountSPORT to learn scheduling constraints?
    - To what extend can we acquire all constraints?
- Could we integrate minimizing/maximizing cost functions into CountSPORT?



Practical approach I: planning (until feb)

Generate schedules using own Hard Constraints

↓

Use CountOR to acquire constraints

↓

Generate new set using acquired constraints

↓

Check satisfiability

Practical approach: CountOR(benchmark)

- Generate set of schedules using own constraints
- Hard Constraints

1. A team never plays itself:
   (a) $\forall day, team : MD[day, team, team] = 0$
2. A team can only play maximum 1 game each matchday
   (a) $\forall day, team : \sum_{t'}(MD[day, team, t'] + MD[day, t', team]) \leqslant 1$ this if nt is odd
   (b) $\forall day, team : \sum_{t'}(MD[day, team, t'] + MD[day, t', team]) = 1$ this if nt is even

Practical approach: CountOR(benchmark)

3. End of cycle point: After the first cycle of your round robin tournament(the amount of cycles does not specifically matter, this constraint is mandatory after every cycle, but only modelled after the first because this is double round robin), you should have played every team once.

   (a) $\forall t \neq t' : \sum_{d=0}^{D/2} (MD[d, t, t'] + MD[d, t', t]) = 1$

4. Double round robin exclusive constraint: after the whole schedule, each team has played every other team twice, once home once away.

   (a) $\forall t \neq t' : \sum_{d=0}^{D} MD[d, t, t'] = 1$

Practical approach: CountOR(benchmark)

Learn with CountOR as is

- Generate new set schedules using acquired constraints
- Check satisfiability

Practical approach: planning II (mar-apr)

Generate schedules w. Hard Constraints → + Soft constraints

↓

CountSPORT to acquire constraints → And minimize cost penalty

↓

Generate set using acquired constraints

↓

Check satisfiability and match with benchmark

Practical approach: CountSPORT

- Adapt CountOR
    - Hard constraints
    - Soft constraints?

5. The Home and away pattern: for now we only model this for an odd number of teams, it consists of 2 parts: we never want to play consecutive games at home, nor away

    (a) $\forall t \wedge t' \neq t : MD[d, t, t'] + MD[d+1, t, t'] \leqslant 1$

    (b) $\forall t \wedge t' \neq t : MD[d, t', t] + MD[d+1, t', t] \leqslant 1$

Putting it to work: a practical use case

- Scheduling the Belgium football competition (format used since 2009)
    - 16 teams: 30 games in regular competition
    - Playoff 1: Division of 6 teams
    - Playoff 2: 2 divisions of 4 teams
    - Playoff 3: 1 division of 2 teams
- Lots of edge cases