Constraint-based Round Robin Tournament Planning

Martin Henz

National University
of Singapore

Round Robin Tournament Planning Problems

- n teams, each playing a fixed number of times r against every other team
- r = 1: single, r = 2: double round robin.
- each match is home match for one and away match for the other
- dense round robin:
  - At each date, each team plays at most once.
  - The number of dates is minimal.

Example: The ACC 1997/98 Problem

Nemhauser, Trick; Scheduling a Major College Basketball Conference; Operations Research, 46(1), 1998.

- 9 teams participate in tournament
- dense double round robin:
  - there are 2 * 9 dates
  - at each date, each team plays either home, away or has a “bye”
- alternating weekday and weekend matches

The ACC 1997/98 Problem (cont'd)

- No team can play away on both last dates
- No team may have more than two away matches in a row.
- No team may have more than two home matches in a row.
- No team may have more than three away matches or byes in a row.
- No team may have more than four home matches or byes in a row.

The ACC 1997/98 Problem (cont'd)

- Of the weekends, each team plays four at home, four away, and one bye.
- Each team must have home matches or byes at least on two of the first five weekends.
- Every team except FSU has a traditional rival. The rival pairs are Clem-GT, Duke-UNC, UMD-UVA and NCSt-Wake. In the last date, every team except FSU plays against its rival, unless it plays against FSU or has a bye.

The ACC 1997/98 Problem (cont'd)

- The following pairings must occur at least once in dates 11 to 18: Duke-GT, Duke-Wake, GT-UNC, UNC-Wake.
- No team plays in two consecutive dates away against Duke and UNC. No team plays in three consecutive dates against Duke UNC and Wake.

- UNC plays Duke in last date and date 11.
- UNC plays Clem in the second date.
- Duke has bye in the first date 16.

The ACC 1997/98 Problem (cont'd)

- Wake does not play home in date 17.
- Wake has a bye in the first date.
- Clem, Duke, UMD and Wake do not play away in the last date.
- Clem, FSU, GT and Wake do not play away in the fist date.
- Neither FSU nor NCSt have a bye in the last date.
- UNC does not have a bye in the first date.

Preferences

- Nemhauser and Trick give a number of additional preferences.
- However, it turns out that there are only 179 solutions to the problem.
- If you find all 179 solutions, you can easily single out the most preferred ones.

Nemhauser/Trick Solution

- enumerate home/away/bye patterns
  - explicit enumeration (very fast)
- compute pattern sets
  - integer programming (below 1 minute)
- compute abstract schedules
  - integer programming (several minutes)
- compute concrete schedules
  - explicit enumeration (approx. 24 hours)

Schreuder, Combinatorial Aspects of Construction of Competition Dutch Football Leagues, Discr. Appl. Math, 35:301-312, 1992.

Modeling ACC 97/98 as Constraint Satisfaction Problem

Variables

- 9 * 18 variables taking values from {0,1} that express which team plays home when. Example: $\mathcal{H}_{UNC,\,5}$=1 means UNC plays home on date 5.
- away, bye similar, e.g. $\mathcal{A}_{UNC,\,5}$ or $\mathcal{B}_{UNC,\,5}$
- 9 * 18 variables taking values from {0,1,...,9} that express against which team which other team plays. Example: $\alpha_{UNC,\,5}$ =1 means UNC plays team 1 (Clem) on date 5

Modeling ACC 97/97 as Constraint Satisfaction Problem (cont'd)

Constraints

Example: No team plays away on both last dates.

$A_{Clem,17} + A_{Clem,18} < 2$, $A_{Duke,17} + A_{Duke,18} < 2$, ...

All constraints can be easily formalized in this manner.

Constraint Programming Approach for Solving CSP

"Propagate and Distribute"

- store possible values of variables in constraint store
- encode constraints as computational agents that strengthen the constraint store whenever possible
- compute fixpoint over propagators
- distribute: divide and conquer, explore search tree

Search Strategies: Naive Approach

Enumerate variables in one way or another.

Any direct enumeration approach fails miserably.

Reason: not enough propagation

$\rightarrow$ too large search space

First Step: Back to Nemhauser/Trick!

- constraint programming for generating all patterns.
  - CSP representation straightforward.
  - computing time below 1 second (Pentium II, 233MHz)
- constraint programming for generating all pattern sets.
  - CSP representation straightforward.
  - computing time 3.1 seconds

Back to Schreuder

- constraint programming for abstract schedules
  - Introduce variable matrix for “abstract opponents” similar to α in naïve model
  - there are many abstract schedules
  - runtime over 20 minutes
- constraint programming for concrete schedules
  - model somewhat complicated, using two levels of the “element” constraint
  - runtime several minutes

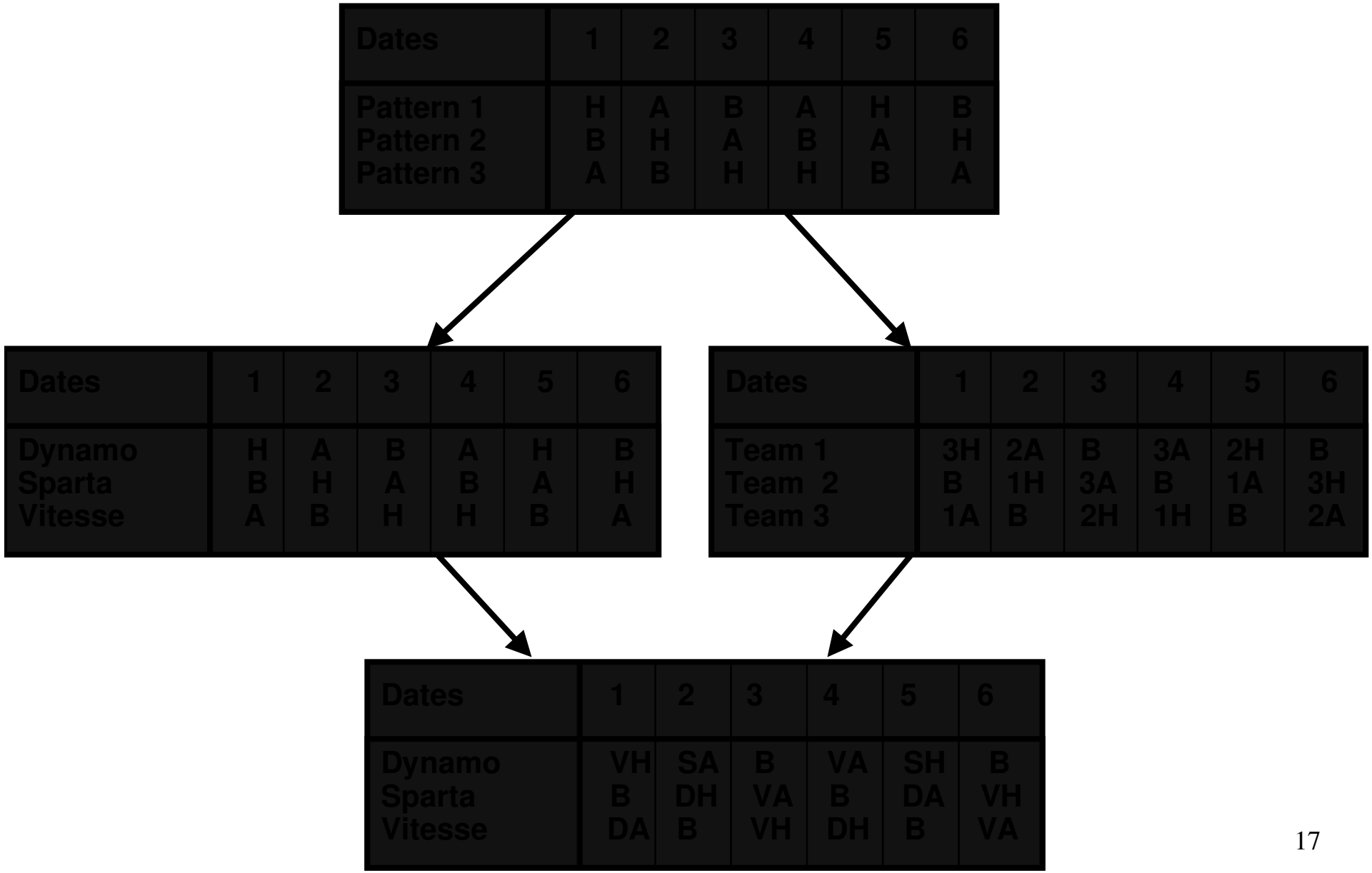
Cain's Model

- Alternative to last two phases of Nemhauser/Trick
- assign teams to patterns in a given pattern set.
- assign opponent teams for each team and date.

W.O. Cain, Jr, The computer-assisted heuristic approach used to schedule the major league baseball clubs, Optimal Strategies in Sports, North-Holland, 1977

Cain 1977

Schreuder 1992



Cain's Model in CP: main idea

- remember: matrices $\mathcal{H}$, $\mathcal{A}$, $\mathcal{B}$ represent patterns and matrix α represents opponents
- given: matrices H, A, B of 0/1 variables representing given pattern set.
- vector p of variables ranging from 1 to n; p_{GT} identifies the pattern for team GT.
- team GT plays according to pattern indicated by p on date 2: $H_{p_{GT},2}=\mathcal{H}_{GT,2}$
- implemented: $\text{element}(p_{GT},H_2 , \mathcal{H}_{GT,2})$

Using Cain's Model in CP

- model for Cain simpler than model for Schreuder
- runtimes:
  - patterns to teams: 33 seconds
  - opponent team assignment: 20.7 seconds
  - overall runtime for all 179 solutions: 57.1 seconds

Details in: Martin Henz, Scheduling a Major College Basketball Conference - Revisited, Operations Research, 2000, to appear

Friar Tuck

- constraint programming tool for sport scheduling, ACC 97/98 just one instance
- convenient entry of constraints through GUI
- Friar Tuck is distributed under GPL
- Friar Tuck 1.1 available for Unix and Windows 95/98
- implementation language: Oz using Mozart (see www.mozart-oz.org)

Implementation of Friar Tuck

- special purpose editors for constraint entry
- access to all phases of the solution process
- choice between Schreuder's and Cain's methods
- computes schedules for over 30 teams
- some new results on number of schedules

Some Users of Friar Tuck

- West England Club Cricket Championship
- Wisconsin Intercollegiate Athletic Conference
- Colonial Athletic Conference
- Bill George Youth Football League
- Euchre tournaments

Future Work

- better constraints for pattern set generation needed
- using local search for sport scheduling
- application specific modeling language for round robin tournaments

Conclusion

- finite domain constraint programming well suited for solving round robin tournament planning
- competing models have natural implementation using CP(FD)
- Friar Tuck: a tool for round robin tournament planning

www.comp.nus.edu.sg/~henz/projects/FriarTuck/

Truth or Consequences, NM

(a few miles north of Las Cruces)

Truth or Consequences, NM

(a few miles north of Las Cruces)

ICLP 2001

Truth or Consequences, NM