Plan

- General Principles
- CP vs MIP
- Integration of OR algorithms
- A MIPLIB example: sports scheduling
- The maximum clique problem
- Strength of CP
- Weakness of CP
- New research areas
- Conclusion
Constraint Programming

- 3 notions:
  - constraint network: variables, domains, constraints + filtering (domain reduction)
  - propagation
  - search procedure (assignments + backtrack or B&B)
Problem = conjunction of sub-problems

- In CP a problem can be viewed as a conjunction of sub-problems that we are able to solve
- A sub-problem can be trivial: $x < y$ or complex: search for a feasible flow
- A sub-problem = a constraint
Filtering

• We are able to solve a sub-problem: a method is available
• CP uses this method to remove values from domain that do not belong to a solution of this sub-problem: **filtering or domain-reduction**
• E.g: \( x < y \) and \( D(x) = [10, 20], \ D(y) = [5, 15] \) => \( D(x) = [10, 14], \ D(y) = [11, 15] \)
Arc consistency

- All the values which do not belong to any solution of the constraint are deleted.
- Example: Alldiff({x,y,z}) with 
  D(x)=D(y)={0,1}, D(z)={0,1,2} 
  the two variables x and y take the values 0 and 1, thus z cannot take these values. 
  FA by AC => 0 and 1 are removed from D(z)
Propagation

- Domain Reduction due to one constraint can lead to new domain reduction of other variables
- When a domain is modified all the constraints involving this variable are studied and so on...
Why Propagation?

• Idea: problem = conjunction of easy sub-problems.

• Sub-problems: local point of view. Problem: global point of view. Propagation tries to obtain a global point of view from independent local point of view

• The conjunction is stronger than the union of independent resolution
Search

• Backtrack (or Branch and Bound) algorithm with strategies

• Strategy: define which variable and which value will be chosen.

• After each domain reduction (i.e. assignment included) filtering and propagation are triggered
Plan

- General Principles
- CP vs MIP
- Integration of OR algorithms
- A MIPLIB example: sports scheduling
- The maximum clique problem
- Strength of CP
- Weakness of CP
- New research areas
- Conclusion
CP vs MIP

MIP approach

CP approach
CP vs MIP

MIP approach

• Relax the problem: floats instead of integers
• 1) Use the Simplex algorithm ("polynomial")
• 2) Set float to integer value. Go to 1) and backtrack if necessary

CP approach
**CP vs MIP**

**MIP approach**
- Relax the problem: floats instead of integers
- 1) Use the Simplex algorithm (“polynomial”)
- 2) Set float to integer value. Go to 1) and backtrack if necessary

**CP approach**
- Identify sub-problems that are easy (called constraints)
- 1) Use specific algorithm for solving these sub-problems and for performing domain-reduction
- 2) Instantiate variable. Go to 1) and backtrack if necessary
CP vs MIP

MIP approach

- Relax the problem: floats instead of integers
- 1) Use the Simplex algorithm (“polynomial”)
- 2) Set float to integer value. Go to 1) and backtrack if necessary

- Global point of view on a relaxation of the problem

CP approach

- Identify sub-problems that are easy (called constraints)
- 1) Use specific algorithm for solving these sub-problems and for performing domain-reduction
- 2) Instantiate variable. Go to 1) and backtrack if necessary
- Local point of view on sub-problems. “Global” point of view by propagation of domain reductions
CP vs MIP

- In CP constraints can be non-linear
- Structure of the problem is used in CP
- Structure of the constraints is used
- First solution given by CP is generally good (better than MIP)
Plan

- General Principles
- CP vs MIP
- Integration of OR algorithms
- A MIPLIB example: sports scheduling
- The maximum clique problem
- Strength of CP
- Weakness of CP
- New research areas
- Conclusion
Alldiff constraint

The value graph:

\[ D(x_1) = \{1,2\} \]
\[ D(x_2) = \{2,3\} \]
\[ D(x_3) = \{1,3\} \]
\[ D(x_4) = \{3,4\} \]
\[ D(x_5) = \{2,4,5,6\} \]
\[ D(x_6) = \{5,6,7\} \]
Matching
Arc consistency

Berge’s theorem:
An edge belong to some but not all maximum matching, iff, for an arbitrary matching it belongs to either an even alternating path which begins at a free vertex, or an even alternating cycle.
Alternating path

x1
x2
x3
x4
x5
x6

1
2
3
4
5
6
7

Free vertex
Alternating cycle
Alldiff constraint
Arc consistency

The value graph:

D(x1)={1,2}
D(x2)={2,3}
D(x3)={1,3}
D(x4)={4}
D(x5)={5,6}
D(x6)={5,6,7}
Alldiff constraint

• Compute a matching which covers \( X \)
• Compute the strongly connected components
• Remove every unmatched arc for which the ends belong to two different components
• **Linear algorithm** establishing arc consistency \( O(m) = O(nd) \)
Global Cardinality Constraint

- GCC(X,\{l_i\},\{u_i\})
- Defined on a set X of variables, the number of times each value v_i can be taken must be in a given interval [l_i, u_i]
- Example: D(x1)=\{a,b,c,d\}, D(x2)=\{a,b,c,d\}, D(x3)=\{b,c\}, D(x4)=\{c,d\}. Values b and c must be taken at most 2, values a and d must be taken at least 1.
GCC

Peter
Paul
Mary
John
Bob
Mike
Julia

M (1,2)
D (1,2)
N (1,1)
B (0,2)
O (0,2)
Value Network

Default Orientation

- Peter
- Paul
- Mary
- John
- Bob
- Mike
- Julia
- M (1,2)
- D (1,2)
- N (1,1)
- B (0,2)
- O (0,2)

(t, s)
A Solution

Default Orientation

- Peter
- Paul
- Mary
- John
- Bob
- Mike
- Julia

- M (1,2)
- D (1,2)
- N (1,1)
- B (0,2)
- O (0,2)

7 flow value
Residual Graph

Black Orientation

Green Orientation

Peter
Paul
Mary
John
Bob
Mike
Julia

M (1,2)
D (1,2)
N (1,1)
B (0,2)
O (0,2)

s

2

29
Properties

- The flow value $x_{ij}$ of $(i,j)$ can be increased iff there is a path from $j$ to $i$ in $R - \{(j,i)\}$

- The flow value $x_{ij}$ of $(i,j)$ can be decreased iff there is a path from $i$ to $j$ in $R - \{(i,j)\}$
Arc consistency

- The flow value of an arc is constant iff the arc does not belong to a directed cycle of the residual graph
- Definition of strongly connected components
Filtering algorithm for GCC

- Compute a feasible flow
- Compute the strongly connected components
- Remove every arc with a zero flow value for which the ends belong to two different components
- Linear algorithm achieving arc consistency
- Work well due to (0,1) arcs
GCC

Peter
Paul
Mary
John
Bob
Mike
Julia

M (1,2)
D (1,2)
N (1,1)
B (0,2)
O (0,2)
GCC after AC

Peter -> M (1,2)
Paul -> D (1,2)
Mary -> N (1,1)
John -> B (0,2)
Bob -> O (0,2)
Mike
Julia
GCC with costs

Sum < 12
Arc consistency

Peter → M (1,2)
Paul → M (1,2)
Mary → D (1,2)
John → N (1,1)
Bob → B (0,2)
Mike → O (0,2)
Julia → O (0,2)

Sum < 12
GCC with costs

- Consistency can be computed by searching for a minimum cost flow.
- Arc consistency can be computed by searching for shortest paths in the residual graph. The length of an arc is its reduced cost.
- Complexity $O(n S(n,m,\chi))$.
- Can be improved by searching for shortest path from the values that are assigned.
Plan

- General Principles
- CP vs MIP
- Integration of OR algorithms
- **A MIPLIB example: sports scheduling**
- The maximum clique problem
- Strength of CP
- Weakness of CP
- New research areas
- Conclusion
The problem

- n teams and n-1 weeks and n/2 periods
- every two teams play each other exactly once
- every team plays one game in each week
- no team plays more than twice in the same period

<table>
<thead>
<tr>
<th></th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>0 vs 1</td>
<td>0 vs 2</td>
<td>4 vs 7</td>
<td>3 vs 6</td>
<td>3 vs 7</td>
<td>1 vs 5</td>
<td>2 vs 4</td>
</tr>
<tr>
<td>Period 2</td>
<td>2 vs 3</td>
<td>1 vs 7</td>
<td>0 vs 3</td>
<td>5 vs 7</td>
<td>1 vs 4</td>
<td>0 vs 6</td>
<td>5 vs 6</td>
</tr>
<tr>
<td>Period 3</td>
<td>4 vs 5</td>
<td>3 vs 5</td>
<td>1 vs 6</td>
<td>0 vs 4</td>
<td>2 vs 6</td>
<td>2 vs 7</td>
<td>0 vs 7</td>
</tr>
<tr>
<td>Period 4</td>
<td>6 vs 7</td>
<td>4 vs 6</td>
<td>2 vs 5</td>
<td>1 vs 2</td>
<td>0 vs 5</td>
<td>3 vs 4</td>
<td>1 vs 3</td>
</tr>
</tbody>
</table>

- Problem 10 teams of the MIPLIB
  (n=10 and the objective function is dummy)
- MIP is not able to find a solution for n=14
- CP finds a solution for n=10 in 0.06s, n=14 in 0.2, n=40 in 6h
**CP model: variables**

For each slot: 2 variables represent the teams and 1 variable represents the match are defined.

<table>
<thead>
<tr>
<th></th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>0 vs 1</td>
<td>0 vs 2</td>
<td>4 vs 7</td>
<td>3 vs 6</td>
<td>3 vs 7</td>
<td>1 vs 5</td>
<td>2 vs 4</td>
</tr>
<tr>
<td>Period 2</td>
<td>2 vs 3</td>
<td>1 vs 7</td>
<td>0 vs 3</td>
<td>5 vs 7</td>
<td>1 vs 4</td>
<td>0 vs 6</td>
<td>5 vs 6</td>
</tr>
<tr>
<td>Period 3</td>
<td>4 vs 5</td>
<td>3 vs 5</td>
<td>1 vs 6</td>
<td>0 vs 4</td>
<td>2 vs 6</td>
<td>2 vs 7</td>
<td>0 vs 7</td>
</tr>
<tr>
<td>Period 4</td>
<td>6 vs 7</td>
<td>4 vs 6</td>
<td>2 vs 5</td>
<td>1 vs 2</td>
<td>0 vs 5</td>
<td>3 vs 4</td>
<td>1 vs 3</td>
</tr>
</tbody>
</table>

**M33 variable (M33=12)**

1 vs 6

**T33a variable (T33a=6)**

**T33h variable (T33h=1)**

<table>
<thead>
<tr>
<th></th>
<th>Mij=1 &lt;=&gt;&lt;= 0 vs 1 or 1 vs 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mij=12 &lt;=&gt;&lt;= 1 vs 6 or 6 vs 1</td>
<td></td>
</tr>
</tbody>
</table>
CP model: T variables

<table>
<thead>
<tr>
<th></th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td><strong>T11h</strong> vs <strong>T11a</strong></td>
<td><strong>T12h</strong> vs <strong>T12a</strong></td>
<td><strong>T13h</strong> vs <strong>T13a</strong></td>
<td><strong>T14h</strong> vs <strong>T14a</strong></td>
<td><strong>T15h</strong> vs <strong>T15a</strong></td>
<td><strong>T16h</strong> vs <strong>T16a</strong></td>
<td><strong>T17h</strong> vs <strong>T17a</strong></td>
</tr>
<tr>
<td>Period 2</td>
<td><strong>T21h</strong> vs <strong>T21a</strong></td>
<td><strong>T22h</strong> vs <strong>T22a</strong></td>
<td><strong>T23h</strong> vs <strong>T23a</strong></td>
<td><strong>T24h</strong> vs <strong>T24a</strong></td>
<td><strong>T25h</strong> vs <strong>T25a</strong></td>
<td><strong>T26h</strong> vs <strong>T26a</strong></td>
<td><strong>T27h</strong> vs <strong>T27a</strong></td>
</tr>
<tr>
<td>Period 3</td>
<td><strong>T31h</strong> vs <strong>T31a</strong></td>
<td><strong>T32h</strong> vs <strong>T32a</strong></td>
<td><strong>T33h</strong> vs <strong>T33a</strong></td>
<td><strong>T34h</strong> vs <strong>T34a</strong></td>
<td><strong>T35h</strong> vs <strong>T35a</strong></td>
<td><strong>T36h</strong> vs <strong>T36a</strong></td>
<td><strong>T37h</strong> vs <strong>T37a</strong></td>
</tr>
<tr>
<td>Period 4</td>
<td><strong>T41h</strong> vs <strong>T41a</strong></td>
<td><strong>T42h</strong> vs <strong>T42a</strong></td>
<td><strong>T43h</strong> vs <strong>T43a</strong></td>
<td><strong>T44h</strong> vs <strong>T44a</strong></td>
<td><strong>T45h</strong> vs <strong>T45a</strong></td>
<td><strong>T46h</strong> vs <strong>T46a</strong></td>
<td><strong>T47h</strong> vs <strong>T47a</strong></td>
</tr>
</tbody>
</table>

\[D(Tija) = [1, n-1]\]

\[D(Tijh) = [0, n-2]\]

**Tijh < Tija**
CP model: M variables

<table>
<thead>
<tr>
<th></th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>M11</td>
<td>M12</td>
<td>M13</td>
<td>M14</td>
<td>M15</td>
<td>M16</td>
<td>M17</td>
</tr>
<tr>
<td>Period 2</td>
<td>M21</td>
<td>M22</td>
<td>M23</td>
<td>M24</td>
<td>M25</td>
<td>M26</td>
<td>M27</td>
</tr>
<tr>
<td>Period 3</td>
<td>M31</td>
<td>M32</td>
<td>M33</td>
<td>M34</td>
<td>M35</td>
<td>M36</td>
<td>M37</td>
</tr>
<tr>
<td>Period 4</td>
<td>M41</td>
<td>M42</td>
<td>M43</td>
<td>M44</td>
<td>M45</td>
<td>M46</td>
<td>M47</td>
</tr>
</tbody>
</table>

\[ D(M_{ij}) = [1, n(n-1)/2] \]
CP model: constraints

- n teams and n-1 weeks and n/2 periods
- every two teams play each other exactly once
- every team plays one game in each week
- no team plays more than twice in the same period

<table>
<thead>
<tr>
<th></th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>M11</td>
<td>M12</td>
<td>M13</td>
<td>M14</td>
<td>M15</td>
<td>M16</td>
<td>M17</td>
</tr>
<tr>
<td>Period 2</td>
<td>M21</td>
<td>M22</td>
<td>M23</td>
<td>M24</td>
<td>M25</td>
<td>M26</td>
<td>M27</td>
</tr>
<tr>
<td>Period 3</td>
<td>M31</td>
<td>M32</td>
<td>M33</td>
<td>M34</td>
<td>M35</td>
<td>M36</td>
<td>M37</td>
</tr>
<tr>
<td>Period 4</td>
<td>M41</td>
<td>M42</td>
<td>M43</td>
<td>M44</td>
<td>M45</td>
<td>M46</td>
<td>M47</td>
</tr>
</tbody>
</table>

Alldiff constraints defined on M variables
CP model: constraints

- n teams and n-1 weeks and n/2 periods
- every two teams play each other exactly once
- every team plays one game in each week
- no team plays more than twice in the same period

<table>
<thead>
<tr>
<th></th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>T11h vs T11a</td>
<td>T12h vs T12a</td>
<td>T13h vs T13a</td>
<td>T14h vs T14a</td>
<td>T15h vs T15a</td>
<td>T16h vs T16a</td>
<td>T17h vs T17a</td>
</tr>
<tr>
<td>Period 2</td>
<td>T21h vs T21a</td>
<td>T22h vs T22a</td>
<td>T23h vs T23a</td>
<td>T24h vs T24a</td>
<td>T25h vs T25a</td>
<td>T26h vs T26a</td>
<td>T27h vs T27a</td>
</tr>
<tr>
<td>Period 3</td>
<td>T31h vs T31a</td>
<td>T32h vs T32a</td>
<td>T33h vs T33a</td>
<td>T34h vs T34a</td>
<td>T35h vs T35a</td>
<td>T36h vs T36a</td>
<td>T37h vs T37a</td>
</tr>
<tr>
<td>Period 4</td>
<td>T41h vs T41a</td>
<td>T42h vs T42a</td>
<td>T43h vs T43a</td>
<td>T44h vs T44a</td>
<td>T45h vs T45a</td>
<td>T46h vs T46a</td>
<td>T47h vs T47a</td>
</tr>
</tbody>
</table>

For each week w:
Alldiff constraint defined
on \{T_{pwh}, p=1..4\} U \{T_{pwa}, p=1..4\}
CP model: constraints

- n teams and n-1 weeks and n/2 periods
- every two teams play each other exactly once
- every team plays one game in each week
- no team plays more than twice in the same period

For each period p:
Global cardinality constraint defined on
\{T_p^{wh}, w=1..7\} \cup \{T_p^{wa}, w=1..7\}
every team t is taken at most 2
CP model: constraints

- For each slot the two T variables and the M variable must be linked together; example:
  \[ M_{12} = \text{game } T_{12h} \text{ vs } T_{12a} \]
- For each slot we add \( C_{ij} \) a ternary constraint defined on the two T variables and the M variable; example:
  \( C_{12} \) defined on \( \{T_{12h}, T_{12a}, M_{12}\} \)
- \( C_{ij} \) are defined by the list of allowed tuples:
  for \( n=4 \): \( \{(0,1,1),(0,2,2),(0,3,3),(1,2,4),(1,3,5),(2,3,6)\} \)
  \( (1,2,4) \) means game 1 vs 2 is the game number 4
- All these constraints have the same list of allowed tuples
- Efficient arc consistency algorithm for this kind of constraint is known
First model

Introduction of a dummy column

<table>
<thead>
<tr>
<th></th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
<th>Dummy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>0 vs 1</td>
<td>0 vs 2</td>
<td>4 vs 7</td>
<td>3 vs 6</td>
<td>3 vs 7</td>
<td>1 vs 5</td>
<td>2 vs 4</td>
<td>. vs .</td>
</tr>
<tr>
<td>Period 2</td>
<td>2 vs 3</td>
<td>1 vs 7</td>
<td>0 vs 3</td>
<td>5 vs 7</td>
<td>1 vs 4</td>
<td>0 vs 6</td>
<td>5 vs 6</td>
<td>. vs .</td>
</tr>
<tr>
<td>Period 3</td>
<td>4 vs 5</td>
<td>3 vs 5</td>
<td>1 vs 6</td>
<td>0 vs 4</td>
<td>2 vs 6</td>
<td>2 vs 7</td>
<td>0 vs 7</td>
<td>. vs .</td>
</tr>
<tr>
<td>Period 4</td>
<td>6 vs 7</td>
<td>4 vs 6</td>
<td>2 vs 5</td>
<td>1 vs 2</td>
<td>0 vs 5</td>
<td>3 vs 4</td>
<td>1 vs 3</td>
<td>. vs .</td>
</tr>
</tbody>
</table>
First model

Introduction of a dummy column

<table>
<thead>
<tr>
<th></th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
<th>Dummy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>0 vs 1</td>
<td>0 vs 2</td>
<td>4 vs 7</td>
<td>3 vs 6</td>
<td>3 vs 7</td>
<td>1 vs 5</td>
<td>2 vs 4</td>
<td>5 vs 6</td>
</tr>
<tr>
<td>Period 2</td>
<td>2 vs 3</td>
<td>1 vs 7</td>
<td>0 vs 3</td>
<td>5 vs 7</td>
<td>1 vs 4</td>
<td>0 vs 6</td>
<td>5 vs 6</td>
<td>. vs .</td>
</tr>
<tr>
<td>Period 3</td>
<td>4 vs 5</td>
<td>3 vs 5</td>
<td>1 vs 6</td>
<td>0 vs 4</td>
<td>2 vs 6</td>
<td>2 vs 7</td>
<td>0 vs 7</td>
<td>. vs .</td>
</tr>
<tr>
<td>Period 4</td>
<td>6 vs 7</td>
<td>4 vs 6</td>
<td>2 vs 5</td>
<td>1 vs 2</td>
<td>0 vs 5</td>
<td>3 vs 4</td>
<td>1 vs 3</td>
<td>. vs .</td>
</tr>
</tbody>
</table>

We can prove that:
• each team occurs exactly twice for each period
First model

Introduction of a dummy column

<table>
<thead>
<tr>
<th></th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
<th>Dummy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>0 vs 1</td>
<td>0 vs 2</td>
<td>4 vs 7</td>
<td>3 vs 6</td>
<td>3 vs 7</td>
<td>1 vs 5</td>
<td>2 vs 4</td>
<td>5 vs 6</td>
</tr>
<tr>
<td>Period 2</td>
<td>2 vs 3</td>
<td>1 vs 7</td>
<td>0 vs 3</td>
<td>5 vs 7</td>
<td>1 vs 4</td>
<td>0 vs 6</td>
<td>5 vs 6</td>
<td>2 vs 4</td>
</tr>
<tr>
<td>Period 3</td>
<td>4 vs 5</td>
<td>3 vs 5</td>
<td>1 vs 6</td>
<td>0 vs 4</td>
<td>2 vs 6</td>
<td>2 vs 7</td>
<td>0 vs 7</td>
<td>1 vs 3</td>
</tr>
<tr>
<td>Period 4</td>
<td>6 vs 7</td>
<td>4 vs 6</td>
<td>2 vs 5</td>
<td>1 vs 2</td>
<td>0 vs 5</td>
<td>3 vs 4</td>
<td>1 vs 3</td>
<td>0 vs 7</td>
</tr>
</tbody>
</table>

We can prove that:
• each team occurs exactly twice for each period
• each team occurs exactly once in the dummy column
First model

Introduction of a dummy column

<table>
<thead>
<tr>
<th></th>
<th>Week 1</th>
<th>Week 2</th>
<th>Week 3</th>
<th>Week 4</th>
<th>Week 5</th>
<th>Week 6</th>
<th>Week 7</th>
<th>Dummy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>0 vs 1</td>
<td>0 vs 2</td>
<td>4 vs 7</td>
<td>3 vs 6</td>
<td>3 vs 7</td>
<td>1 vs 5</td>
<td>2 vs 4</td>
<td>5 vs 6</td>
</tr>
<tr>
<td>Period 2</td>
<td>2 vs 3</td>
<td>1 vs 7</td>
<td>0 vs 3</td>
<td>5 vs 7</td>
<td>1 vs 4</td>
<td>0 vs 6</td>
<td>5 vs 6</td>
<td>2 vs 4</td>
</tr>
<tr>
<td>Period 3</td>
<td>4 vs 5</td>
<td>3 vs 5</td>
<td>1 vs 6</td>
<td>0 vs 4</td>
<td>2 vs 6</td>
<td>2 vs 7</td>
<td>0 vs 7</td>
<td>1 vs 3</td>
</tr>
<tr>
<td>Period 4</td>
<td>6 vs 7</td>
<td>4 vs 6</td>
<td>2 vs 5</td>
<td>1 vs 2</td>
<td>0 vs 5</td>
<td>3 vs 4</td>
<td>1 vs 3</td>
<td>0 vs 7</td>
</tr>
</tbody>
</table>

• The problem is exactly the same
• The solver is helped by such constraint. It can deduce some inconsistencies more quickly
First model: strategies

- Break symmetries: 0 vs w appears in week w
- Teams are instantiated:
  - the most instantiated team is chosen
  - the slots that has the less remaining possibilities (Tijh or Tija is minimal) is instantiated with that team
# First model: results

<table>
<thead>
<tr>
<th># teams</th>
<th># fails</th>
<th>Time (in s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>0.01</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>0.03</td>
</tr>
<tr>
<td>8</td>
<td>32</td>
<td>0.08</td>
</tr>
<tr>
<td>10</td>
<td>417</td>
<td>0.8</td>
</tr>
<tr>
<td>12</td>
<td>41</td>
<td>0.2</td>
</tr>
<tr>
<td>14</td>
<td>3,514</td>
<td>9.2</td>
</tr>
<tr>
<td>16</td>
<td>1,112</td>
<td>4.2</td>
</tr>
<tr>
<td>18</td>
<td>8,756</td>
<td>36</td>
</tr>
<tr>
<td>20</td>
<td>72,095</td>
<td>338</td>
</tr>
<tr>
<td>22</td>
<td>6,172,672</td>
<td>10h</td>
</tr>
<tr>
<td>24</td>
<td>6,391,470</td>
<td>12h</td>
</tr>
</tbody>
</table>
Second model

- Break symmetry: 0 vs 1 is the first game of the dummy column
- 1) Find a round-robin. Define all the games for each column (except for the dummy)
  - Alldiff constraint on M is satisfied
  - Alldiff constraint for each week is satisfied
- 2) set the games in order to satisfy constraints on periods. If no solution go to 1)
## Second model: results

<table>
<thead>
<tr>
<th># teams</th>
<th># fails</th>
<th>Time (in s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
<td>0.01</td>
</tr>
<tr>
<td>10</td>
<td>24</td>
<td>0.06</td>
</tr>
<tr>
<td>12</td>
<td>58</td>
<td>0.2</td>
</tr>
<tr>
<td>14</td>
<td>21</td>
<td>0.2</td>
</tr>
<tr>
<td>16</td>
<td>182</td>
<td>0.6</td>
</tr>
<tr>
<td>18</td>
<td>263</td>
<td>0.9</td>
</tr>
<tr>
<td>20</td>
<td>226</td>
<td>1.2</td>
</tr>
<tr>
<td>24</td>
<td>2702</td>
<td>10.5</td>
</tr>
<tr>
<td>26</td>
<td>5,683</td>
<td>26.4</td>
</tr>
<tr>
<td>30</td>
<td>11,895</td>
<td>138</td>
</tr>
<tr>
<td>40</td>
<td>2,834,754</td>
<td>6h</td>
</tr>
</tbody>
</table>
Why does CP perform well?

- Pure discrete problem. You can give any number to the teams
- This is a feasibility problem (no objective function).
- No arithmetic symbol: +, -, = is used
- A global point of view on the global cardinality constraints (i.e. group these constraints into only one) does not help
Plan

- General Principles
- CP vs MIP
- Integration of OR algorithms
- A MIPLIB example: sports scheduling
- The maximum clique problem
- Strength of CP
- Weakness of CP
- New research areas
- Conclusion
Motivations

• Contradict some conventional wisdoms about CP: not good for combinatorial optimization problem, need time to find the optimal solution.

• Consider a well known pure combinatorial optimization problem and show that CP
  • Is an efficient technique
  • Can give good results very quickly
Maximum clique
Maximum clique
The problem: studies

- Good and recent states of the art
- Advantages of this problem: 400 references in the states of the art.
- All existing techniques have been used (GA, Neural Network, Local Search, CP, MIP …)
- Active area: papers every year
Enumeration based Algo

- Try to successively augment a **Current** set of nodes by adding a new node to it.
- When a node is added: removes its non-neighborhood (nodes not linked).
- Possible set is called **Candidate** set.
- Branch-and-Bound algorithm is used
- Upper bounds of the max clique are used:
  - $|K|$ best solution found so far
  - If $(UB_{\text{maxClique}}(\text{candidate}) \leq |K| - |\text{Current}|)$ then fails
Efficient filtering algorithm is required

- Basic program performs more than 5 millions of backtracks per second.
- Some people consider that this is not possible to use FA + propagation for this kind of problem because it will be too long. Therefore, we need efficient properties that can be efficiently computed.
- To be worthwhile the FA must be powerful.
Max-Clique

- Other exact method: find the best possible upper bound and check with current. The check can be long.

- In CP: the UB + propagation must be considered and not only the UB.

- UB1 better than UB2 and better than UB3 but UB2 + UB3 + propagation can be better than UB1
Max-Clique

- **Classical method:**
  repeat:
  - select a node
  - remove nodes by applying the strongest property

- **In CP:**
  repeat:
  - select a node
  - while (a node is removed)
    - remove nodes by applying several properties
  end while
Max-Clique with CP

- Ideas:
  - Find a good ub which is easy and quick to compute
G

CG
Max clique
Max clique  Max Independent Set
Max clique
Max Independent Set
Min vertex Cover
Max clique \( \text{maxClique}(G) = n - \text{minCover}(CG) \)
UB of maxClique

- maxClique(G) = n – minCover(CG). So: maxClique(G) \leq n – LBminCover(CG)
- UBmaxClique(G) = n – LBminCover(CG)
- Idea: find LBminCover(CG)
UB of maxClique

- $\text{maxClique}(G) = n - \text{minCover}(CG)$. So:
  $\text{maxClique}(G) \leq n - \text{LBminCover}(CG)$
- $\text{UBmaxClique}(G) = n - \text{LBminCover}(CG)$
- Idea: find $\text{LBminCover}(CG)$
- Well known:
  $\text{minCover}(G) \geq \text{maxMatching}(G)$
  equality when $G$ is bipartite
UB of maxClique

- Matching: set of edges such that they have no node in common

Vertex Cover: all the edges must be covered, therefore any vertex cover contains at least one node of every edge of the matching
UB of maxClique

- Drawback: G can be non-bipartite and the matching algorithm is quite complex
- Goal: try to find an UB easier to implement and better.
UB of max clique

- maxClique(G) = n – minCover(CG)
- If we find a covering of CG with paths and cycles, we will have an UB of maxClique, because we can deduce an LB of minCover.
LB of minCover

Every vertex cover involves at least $\lceil k/2 \rceil$ nodes of a Cycle of length $k$

Every vertex cover involves at least $\lceil k/2 \rceil$ nodes of a Path of length $k$

$k = \text{number of edges}$

Matching gives 3,
Our formula gives $\lceil 3/2 \rceil + \lceil 3/2 \rceil = 2 + 2 = 4$
A cover by node disjoint paths and cycles can be found by searching for a matching in the “duplicated graph” and then by projecting this matching to the original graph.
A cover by node disjoint paths and cycles can be found by searching for a matching in the “duplicated graph” and then by projecting this matching to the original graph.
A cover by node disjoint paths and cycles can be found by searching for a matching in the “duplicated graph” and then by projecting this matching to the original graph.
LB of minCover

- Another LB of minCover:
  \[ \text{minCover}(G) \geq \lceil \text{maxMatching}(DG)/2 \rceil \]

  note: \( \lceil \text{maxMatching}(DG)/2 \rceil \geq \text{maxMatching}(G) \)

- UB for maxClique:
  \[ \text{maxClique}(G) \leq n - \lceil \text{maxMatching}(DCG)/2 \rceil \]
LB of minCover

- Advantages:
  - DCG is bipartite (means simple algorithm)
  - Very good pre-test:
    maxClique(G) \leq n - \left\lfloor \frac{n}{2} \right\rfloor

Only 5% of the nodes that satisfy this condition will not be removed
2nd Filtering Algorithm

- **Not** set of nodes: contain the nodes that have been tried and that are linked to all nodes of Current.
Bron & Kerbosh

If a node in NOT is linked to all candidate nodes then fail

Any clique containing x can be extended by added y to it
Filtering from B&K’s idea

Let \( x \) be a node in candidate. If there is a node \( y \) in NOT such that
- \( y \) is linked to \( x \) and,
- \( N(x) - \{ y \} \) is included in \( N(y) - \{ x \} \) then \( x \) can be removed

Some other refinements given in my paper.
Max-Clique with CP

- **Propagation:**
  
  while (a node is removed from Candidate) do
  
  call maxCliqueUBFilter
  
  call NotBasedFilter
  
  done
Results

- All problems with less than or equal to 400 nodes are solved for the first time (notably all brock400)
- Idem for 500 except for one
- $P_{\text{hat}300-1}$: 40s instead of 800s
- $P_{\text{hat}700-2}$: 250s instead of 2200s
- An open problem closed in 150s
Results:
CP vs complete methods

- Wood, Ostegard, Fahle, Regin

<table>
<thead>
<tr>
<th></th>
<th>#solved</th>
<th>&lt; 10 min</th>
<th>best time</th>
<th>best LB</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP</td>
<td>38</td>
<td>38</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>CP (option)</td>
<td>36</td>
<td>35</td>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>CP (option)</td>
<td>45</td>
<td>38</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>CP (option)</td>
<td>52</td>
<td>44</td>
<td>30</td>
<td>9</td>
</tr>
</tbody>
</table>

- Ostegard (Dynamic programming, RDS in CP) in less than 10 min: 350s CP: 285s
Results: CP vs heuristic methods

- Qualex: 50 best bounds
- St-Louis, Gendron, Ferland (Optimization days): 50 best bounds
- CP: 58 (52 proved)
- CP < 10 min: 49 (44 proved)
- CP < 1 min: 41 (37 proved)
Plan

- General Principles
- CP vs MIP
- Integration of OR algorithms
- A MIPLIB example: sports scheduling
- The maximum clique problem
- **Strength of CP**
- Weakness of CP
- New research areas
- Conclusion
Strength of CP

- Very flexible (easy to take into account new constraints)
- The system is open: you can define you own constraints, your own search mechanism.
- CP allows the use of sophisticated strategies, you can use the knowledge of the domain of application.
- You just have to respect a protocol given by a **solver**. The solver manages the propagation and provides you with a lot of predefined things
Strength of CP

- CP is **exact**: no solution is lost even for float variables
- Any existing algorithm can be integrated in CP as a filtering algorithm of a constraints
- Concepts are simple
- A first model can be defined and tested quickly
- For optimization problems: the first solution is a good one
- Easy to introduce your new “idea” in the system
- Cooperation is easy thanks to constraints
Plan

- General Principles
- CP vs MIP
- Integration of OR algorithms
- A MIPLIB example: sports scheduling
- The maximum clique problem
- Strength of CP
- **Weakness of CP**
- New research areas
- Conclusion
Weakness of CP

- Must be improved for optimization problems: spend too much time in proving sub-optimality
  - First step: integration of cost in the constraints
- Sometimes lack of global point of view
- Dark zones: press Enter key then ?
- Relaxation is not good for CP. We learn relaxation at school!
Plan

- General Principles
- CP vs MIP
- Integration of OR algorithms
- A MIPLIB example: sports scheduling
- The maximum clique problem
- Strength of CP
- Weakness of CP
- New research areas
- Conclusion
New research areas

• New point of view: CP is based on filtering algorithm, i.e. : Given a property P defining a necessary condition for an element to be in a solution
Find as quickly as possible ALL elements that do not satisfy P

• Ex: alldiff constraint and matching, cardinality constraint and flows etc…

• Close to sensitivity analysis, but also different (for instance we only have monotonic modifications).
New research areas

- Consider a Minimization problem, and OBJ the objective. Suppose that we found a “solution” with OBJ=25.
- We will reject any “solution” with OBJ > 24. So if x=a leads to an OBJ > 24 then value a must be removed from D(x).
- If we have a lower bound of OBJ then we can use it: if lb(OBJ,x=a) > 24 then remove a from D(x)
- Problem: literature mainly gives upper bound for minimization problems and lower bound for maximization problems.
- Not always easy to get lb of good quality
New research areas

- The very same algorithm is called thousand times (million sometimes)
- The incremental aspect of the algorithm becomes really important.
Plan

- General Principles
- CP vs MIP
- Integration of OR algorithms
- A MIPLIB example: sports scheduling
- The maximum clique problem
- Strength of CP
- Weakness of CP
- New research areas
- Conclusion
Conclusion

- CP is a general technique: can encapsulate a lot of work
- CP is an efficient method for solving some combinatorial problems: small or large
- Filtering algorithms are quite important
- CP allows the use of sophisticated strategies
- If you want to use CP: think CP (avoid Boolean (0-1) variables). CP allows the use of symbolic representation