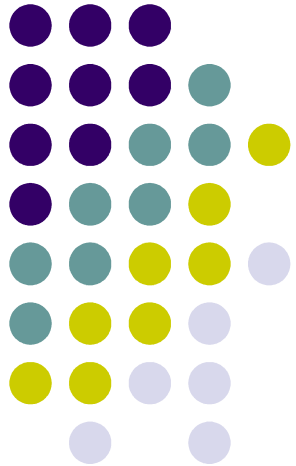
Constraint Programming and Operations Research

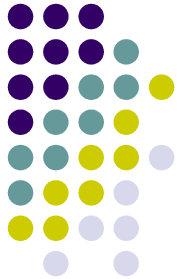


Jean-Charles REGIN
Cornell University

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

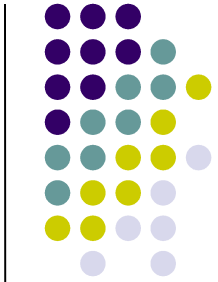
Filtering

- 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

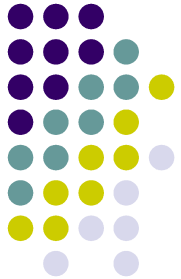
- All the values which do not belong to any solution of the constraint are deleted.
- Example: Alldiff({x,y,z}) with
  D(x)=D(y)={0,1}, D(z)={0,1,2}
  the two variables x and y take the values 0 and 1, thus z cannot take these values.
  FA by AC => 0 and 1 are removed from D(z)

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?

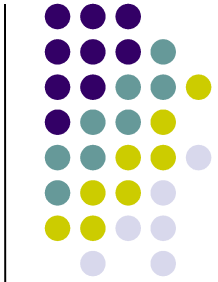


- Idea: problem = conjunction of easy sub-problems.
- Sub-problems: local point of view. Problem: global point of view. Propagation tries to obtain a global point of view from independent local point of view
- The conjunction is stronger that the union of independent resolution

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

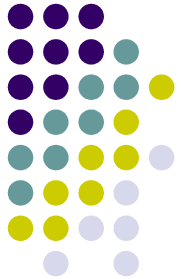


- General Principles
- **CP vs MIP**
- Integration of OR algorithms
- A MIPLIB example: sports scheduling
- The maximum clique problem
- Strength of CP
- Weakness of CP
- New research areas
- Conclusion

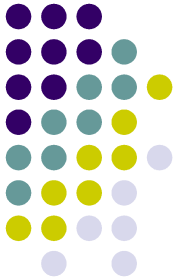
CP vs MIP

MIP approach

CP approach



CP vs MIP

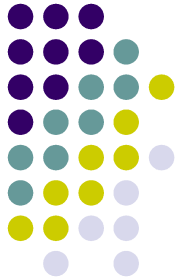


MIP approach

- 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

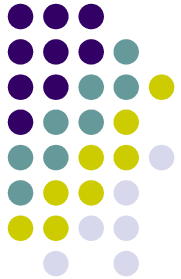
- Relax the problem: floats instead of integers
- 1) Use the Simplex algorithm (“polynomial”)
- 2) Set float to integer value. Go to 1) and backtrack if necessary

- **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)

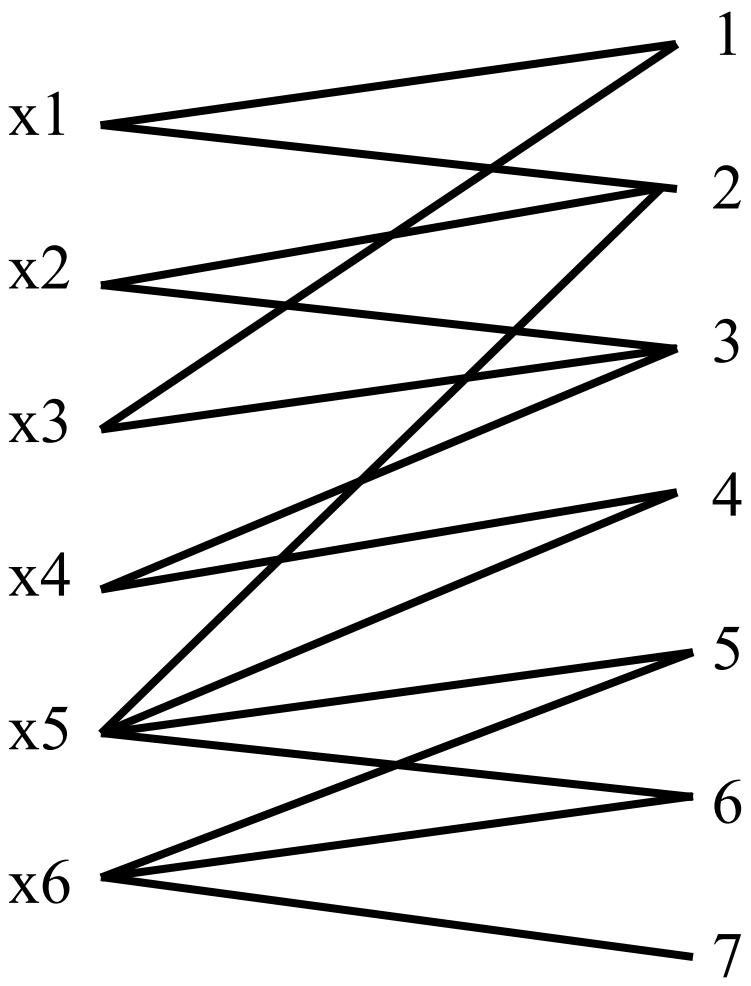
Plan

- 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

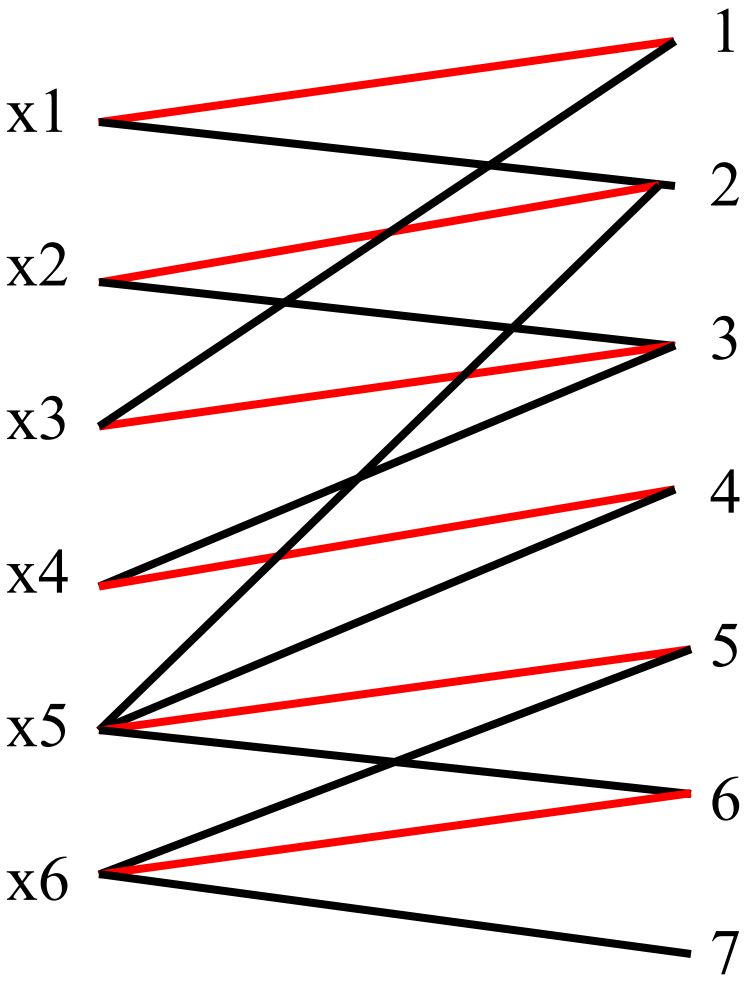
Alldiff constraint

The value graph:

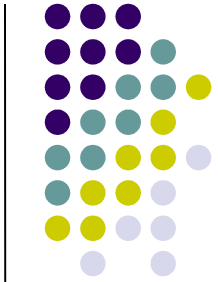
D(x1)={1,2}
D(x2)={2,3}
D(x3)={1,3}
D(x4)={3,4}
D(x5)={2,4,5,6}
D(x6)={5,6,7}



Matching



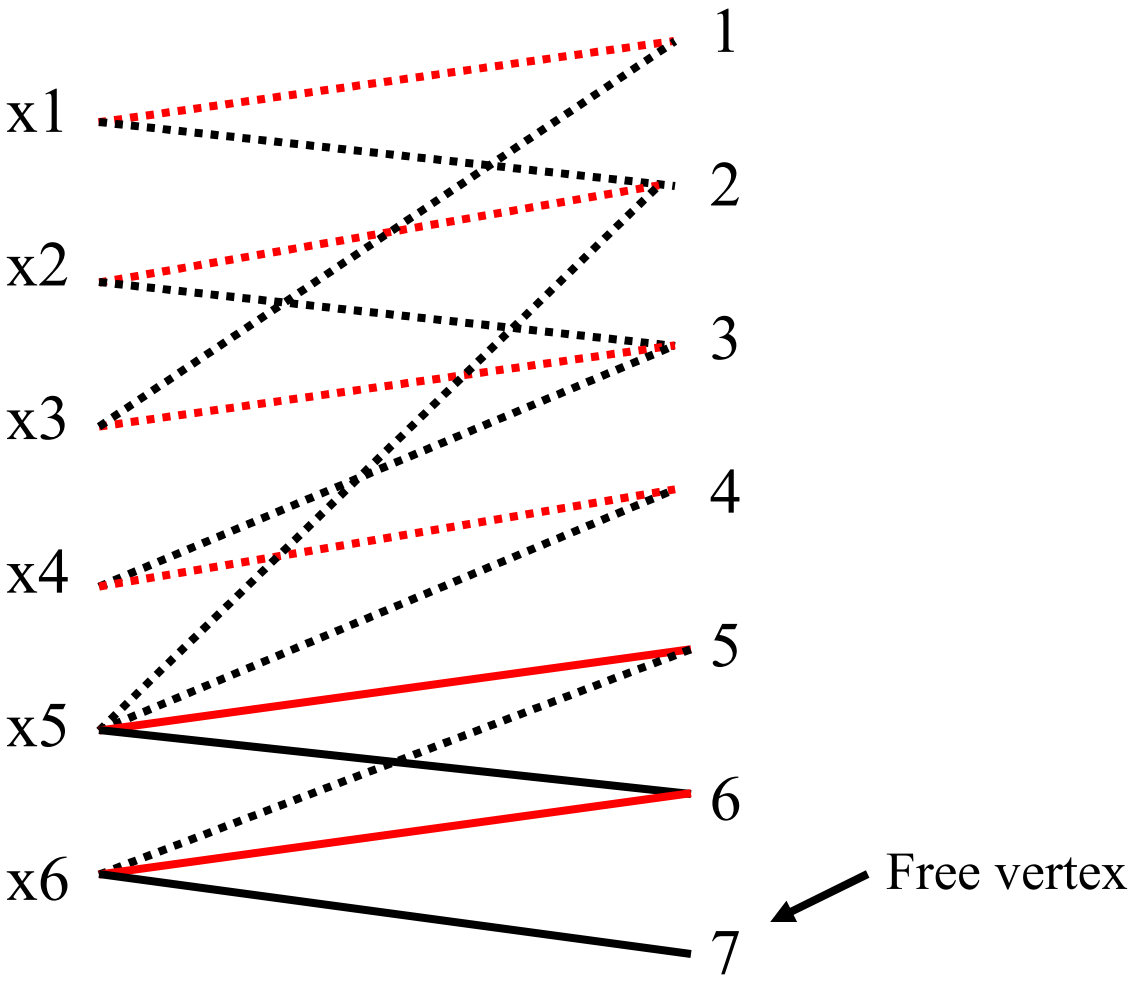
Arc consistency



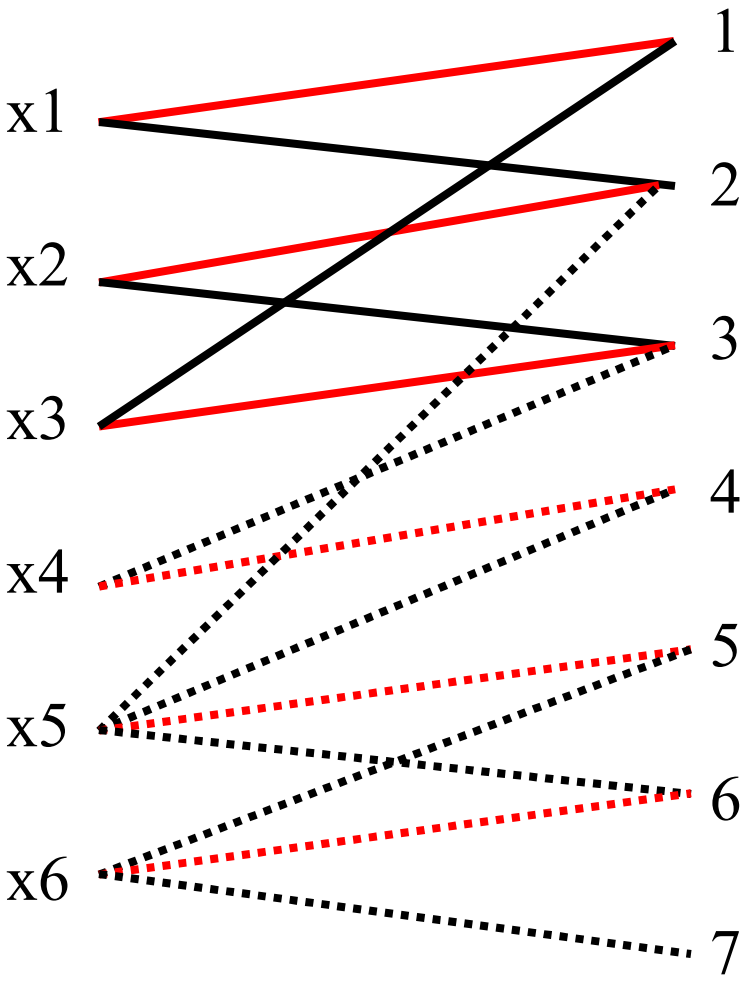
Berge's theorem:

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.

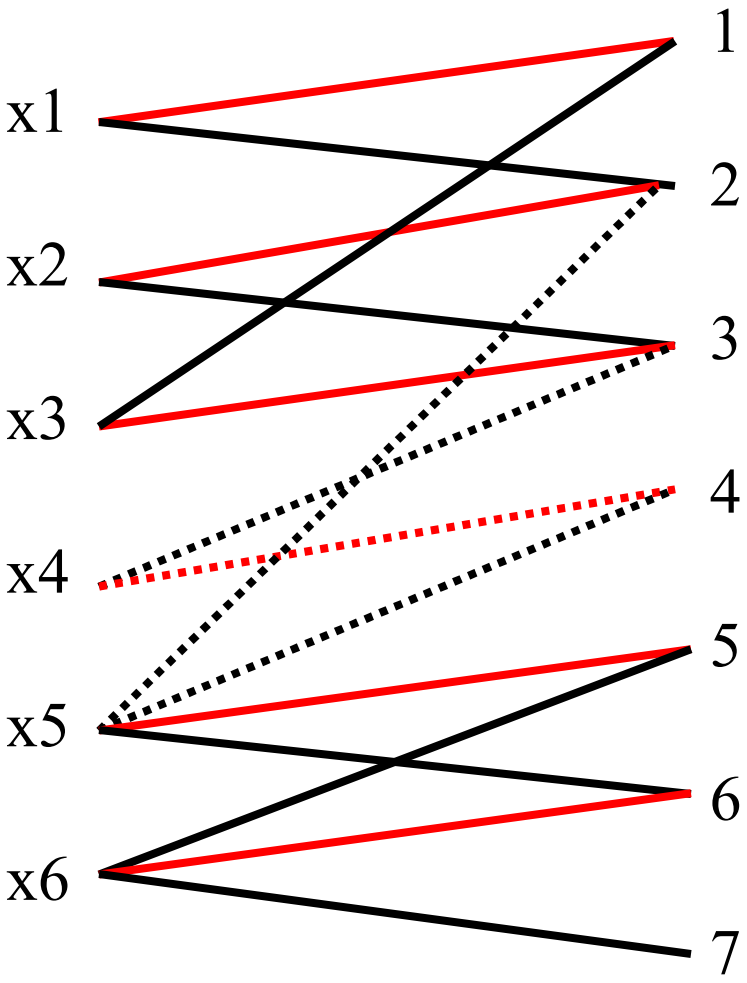
Alternating path



Alternating cycle



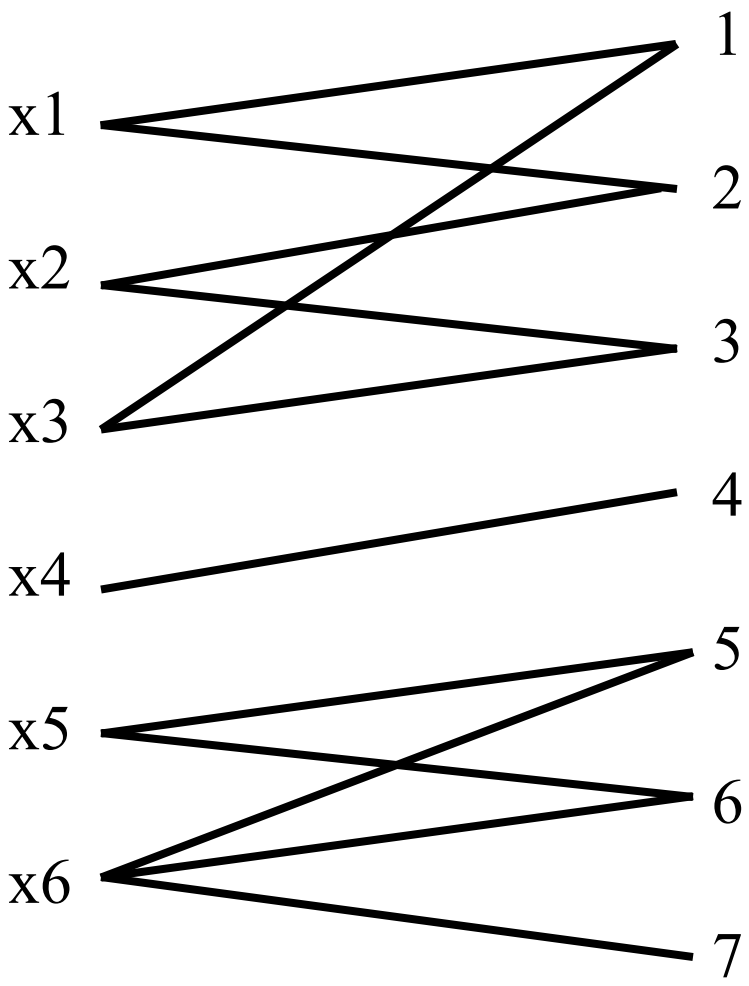
Alldiff constraint



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}



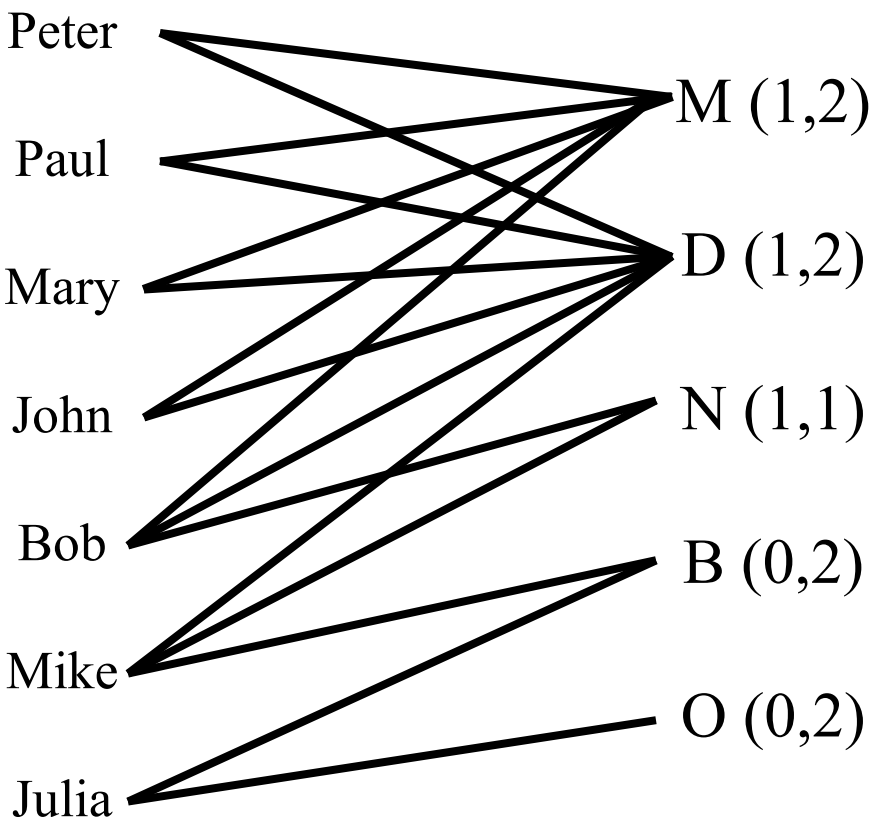
Alldiff constraint

- 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)

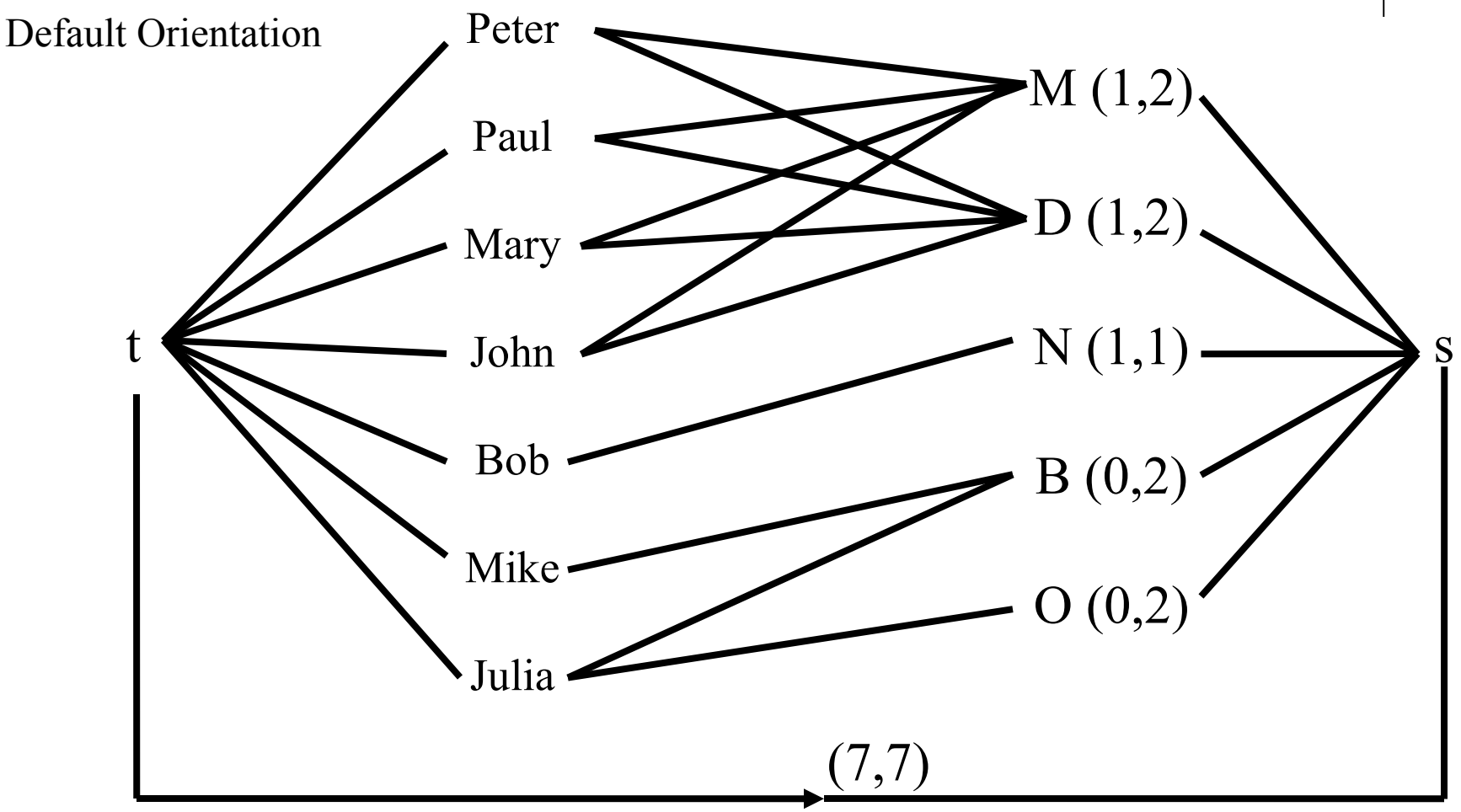
Global Cardinality Constraint

- 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.

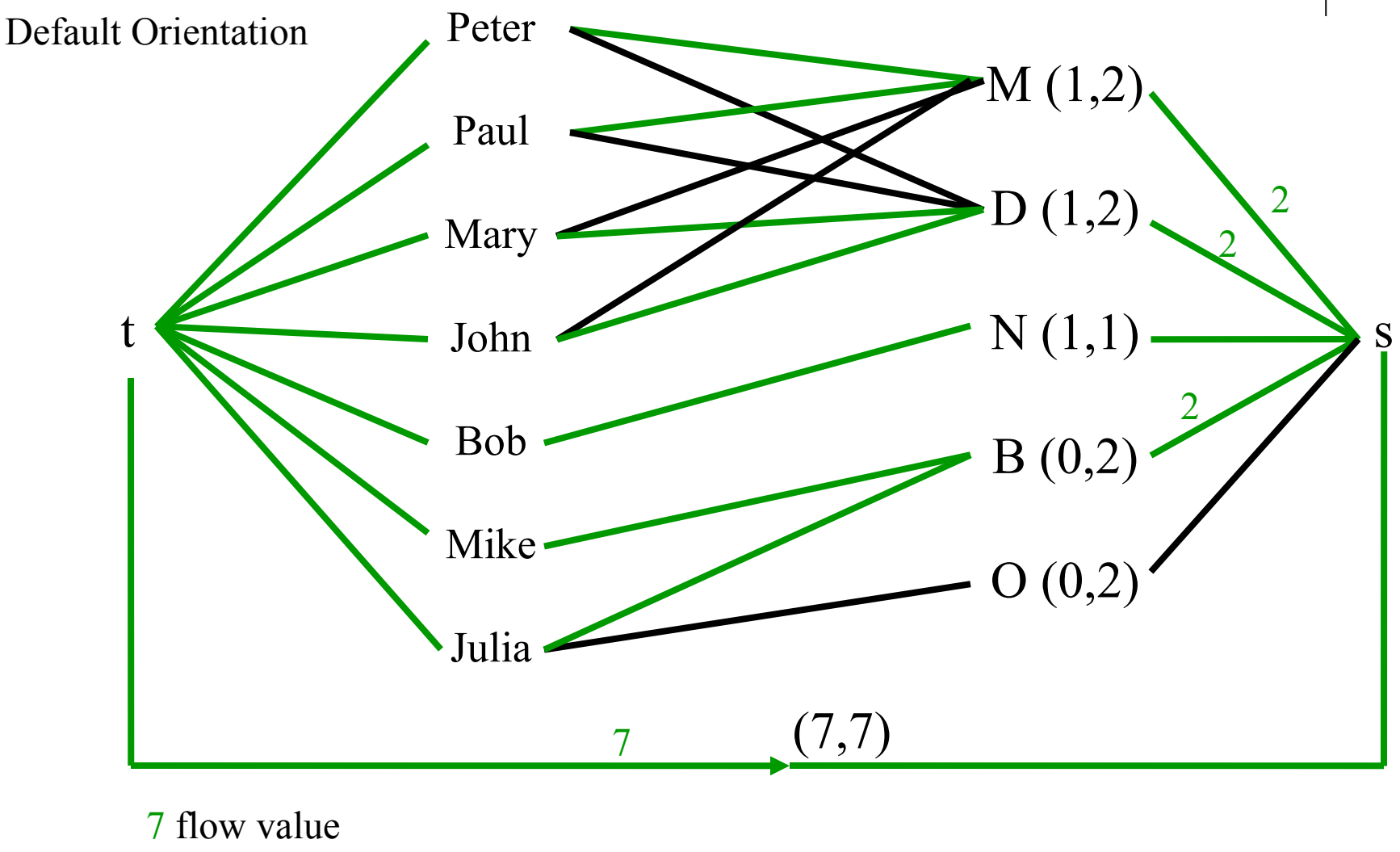
GCC



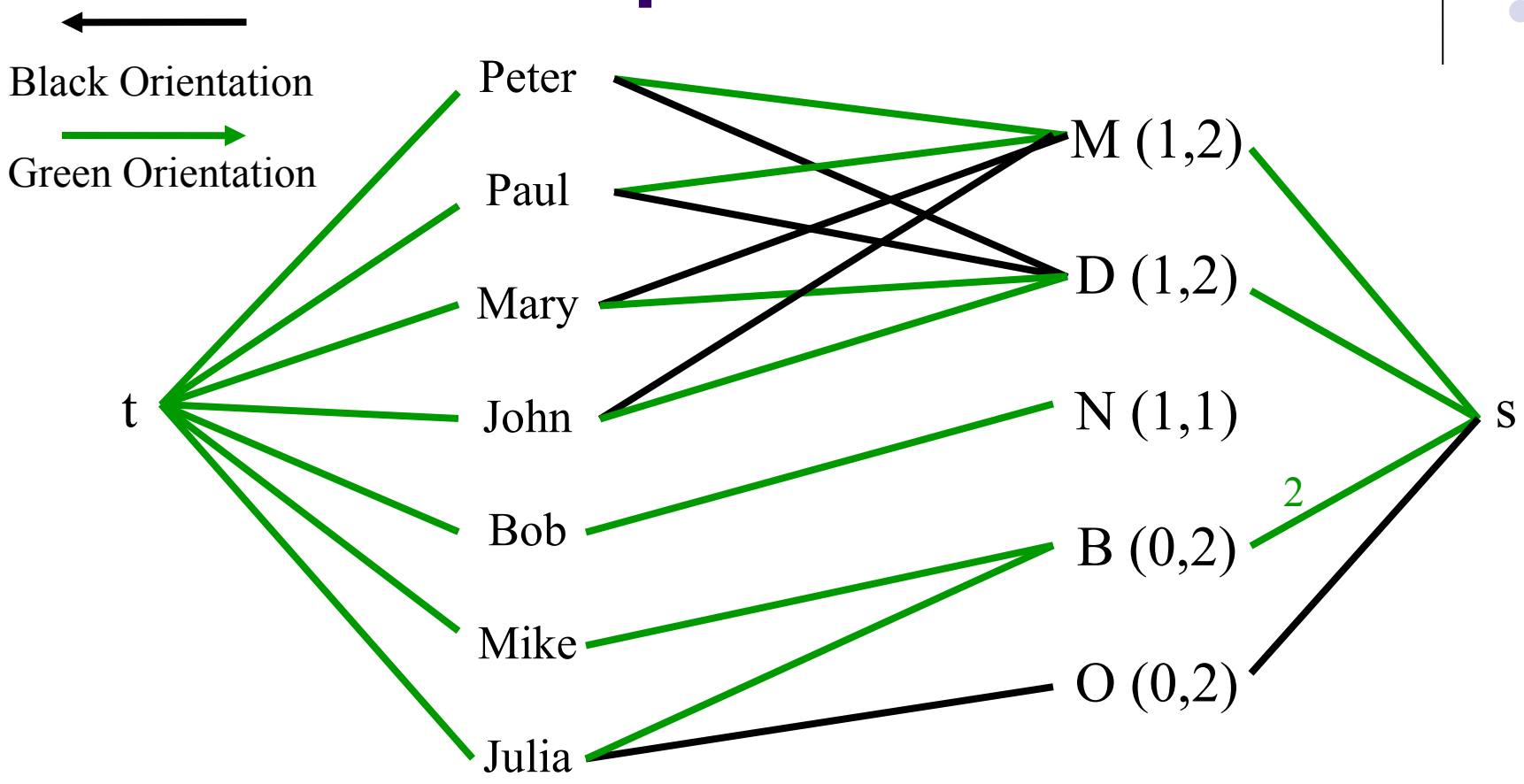
Value Network



A Solution

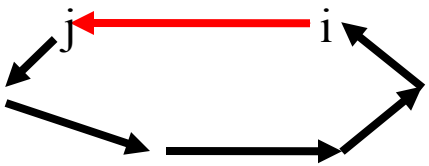

Default Orientation
Peter
Paul
Mary
John
Bob
Mike
Julia
t
M (1,2)
D (1,2)
N (1,1)
B (0,2)
O (0,2)
s
2
2
2
7
(7,7)
7 flow value

Residual Graph

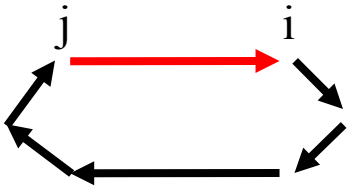


Properties

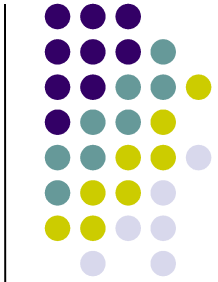
- The flow value xij of (i,j) can be increased iff there is a path from j to i in R - {(j,i)}



- The flow value xij of (i,j) can be decreased iff there is a path from i to j in R - {(i,j)}



Arc consistency

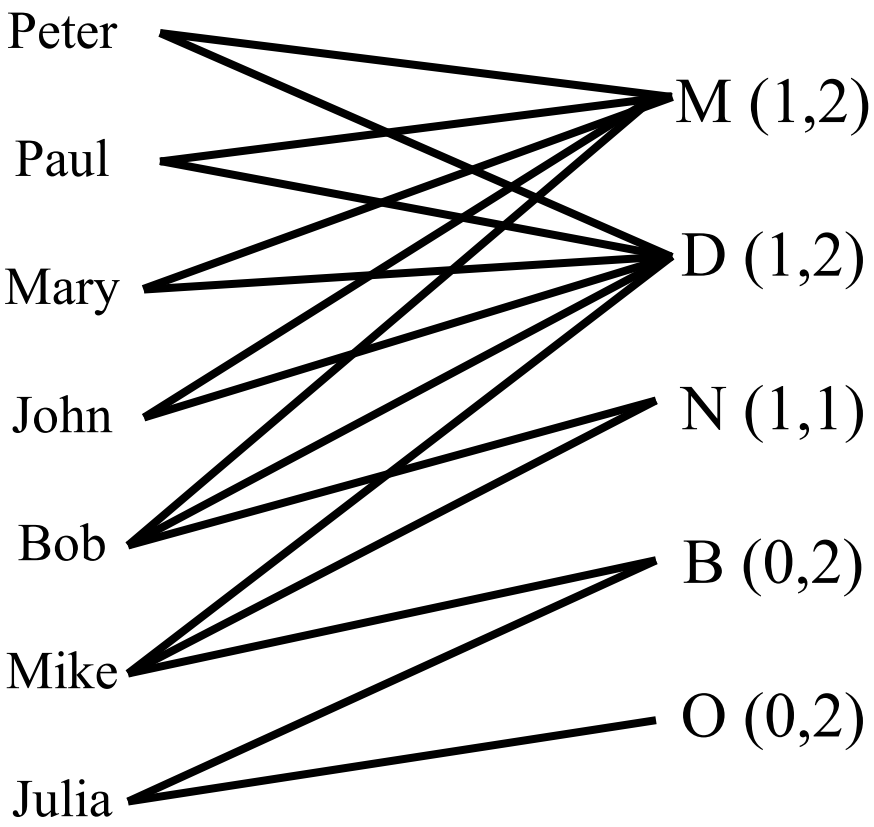


- The flow value of an arc is constant iff the arc does not belong to a directed cycle of the residual graph
- Definition of strongly connected components

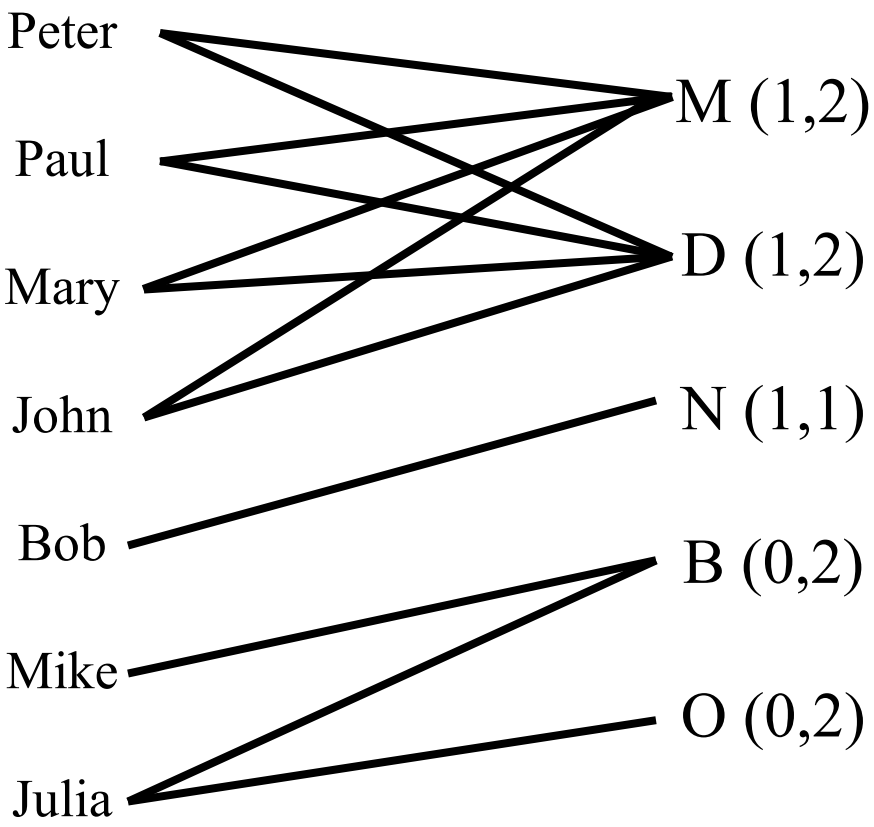
Filtering algorithm for GCC

- 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

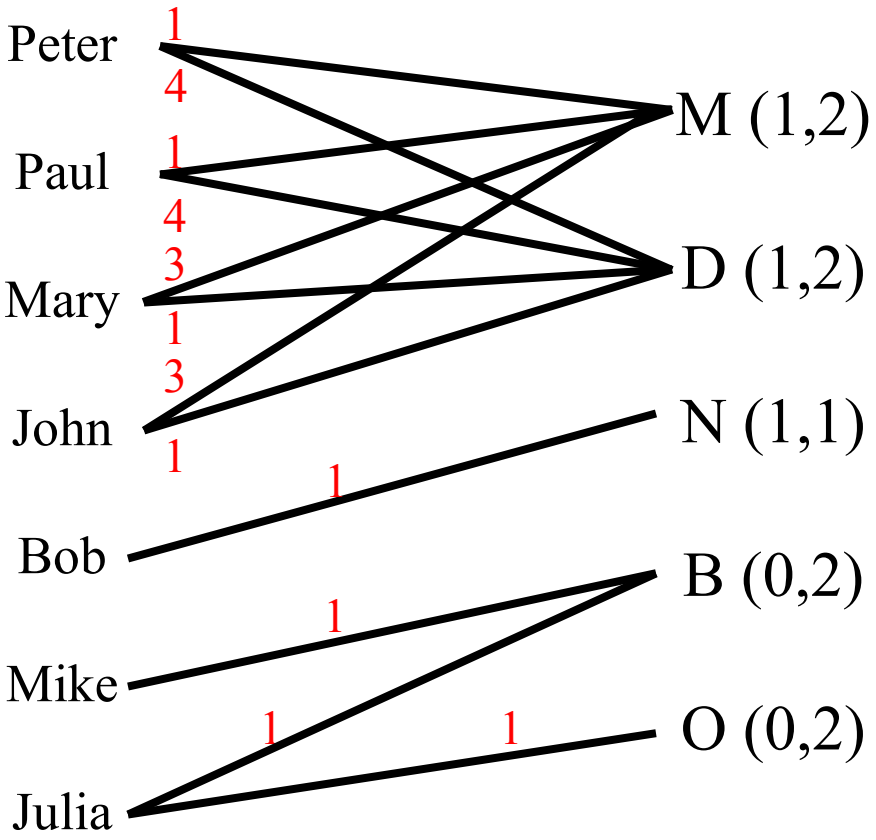
GCC



GCC after AC

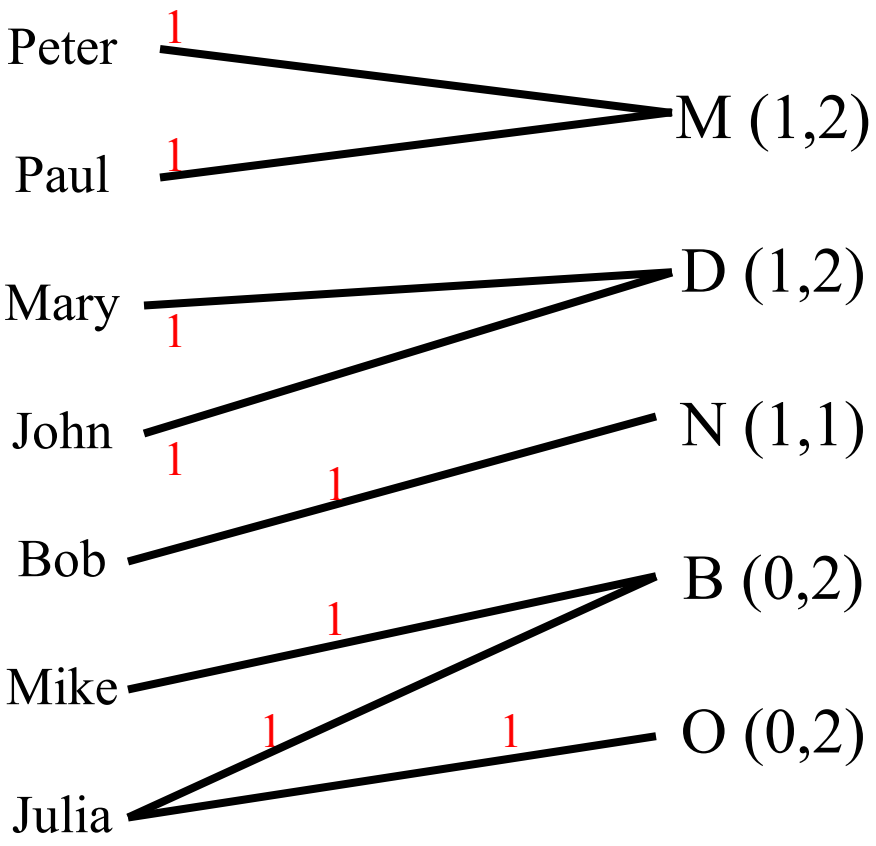


GCC with costs



Sum < 12

Arc consistency

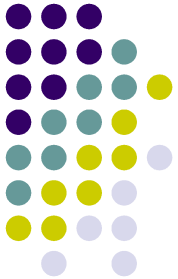


Sum < 12

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,χ)).
- 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

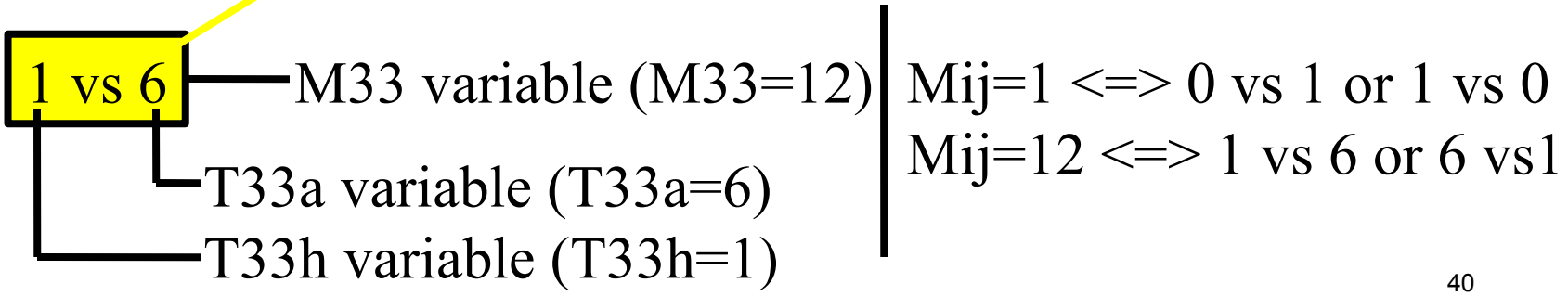
	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	0 vs 1	0 vs 2	4 vs 7	3 vs 6	3 vs 7	1 vs 5	2 vs 4
Period 2	2 vs 3	1 vs 7	0 vs 3	5 vs 7	1 vs 4	0 vs 6	5 vs 6
Period 3	4 vs 5	3 vs 5	1 vs 6	0 vs 4	2 vs 6	2 vs 7	0 vs 7
Period 4	6 vs 7	4 vs 6	2 vs 5	1 vs 2	0 vs 5	3 vs 4	1 vs 3

- Problem 10teams 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

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Period 1	0 vs 1	0 vs 2	4 vs 7	3 vs 6	3 vs 7	1 vs 5	2 vs 4
Period 2	2 vs 3	1 vs 7	0 vs 3	5 vs 7	1 vs 4	0 vs 6	5 vs 6
Period 3	4 vs 5	3 vs 5	1 vs 6	0 vs 4	2 vs 6	2 vs 7	0 vs 7
Period 4	6 vs 7	4 vs 6	2 vs 5	1 vs 2	0 vs 5	3 vs 4	1 vs 3



CP model: T variables

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

$$D(Tija)=[1,n-1]$$
$$D(Tijh)=[0,n-2]$$

$$Tijh < Tija$$

CP model: M variables

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

$$D(Mij)=[1,n(n-1)/2]$$

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

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

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

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

For each week w:
Alldiff constraint defined
on {Tpwh, p=1..4} U {Tpwa, 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

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

For each period p:
Global cardinality constraint defined on
{Tpwh, w=1..7} U {Tpwa, 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:
  M12 = game T12h vs T12a
- For each slot we add Cij a ternary constraint defined on the two T variables and the M variable; example:
  C12 defined on {T12h,T12a,M12}
- Cij 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

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Dummy
Period 1	0 vs 1	0 vs 2	4 vs 7	3 vs 6	3 vs 7	1 vs 5	2 vs 4	. vs .
Period 2	2 vs 3	1 vs 7	0 vs 3	5 vs 7	1 vs 4	0 vs 6	5 vs 6	. vs .
Period 3	4 vs 5	3 vs 5	1 vs 6	0 vs 4	2 vs 6	2 vs 7	0 vs 7	. vs .
Period 4	6 vs 7	4 vs 6	2 vs 5	1 vs 2	0 vs 5	3 vs 4	1 vs 3	. vs .

First model

Introduction of a dummy column

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Dummy
Period 1	0 vs 1	0 vs 2	4 vs 7	3 vs 6	3 vs 7	1 vs 5	2 vs 4	5 vs 6
Period 2	2 vs 3	1 vs 7	0 vs 3	5 vs 7	1 vs 4	0 vs 6	5 vs 6	. vs .
Period 3	4 vs 5	3 vs 5	1 vs 6	0 vs 4	2 vs 6	2 vs 7	0 vs 7	. vs .
Period 4	6 vs 7	4 vs 6	2 vs 5	1 vs 2	0 vs 5	3 vs 4	1 vs 3	. vs .

We can prove that:

- each team occurs exactly twice for each period

First model

Introduction of a dummy column

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Dummy
Period 1	0 vs 1	0 vs 2	4 vs 7	3 vs 6	3 vs 7	1 vs 5	2 vs 4	5 vs 6
Period 2	2 vs 3	1 vs 7	0 vs 3	5 vs 7	1 vs 4	0 vs 6	5 vs 6	2 vs 4
Period 3	4 vs 5	3 vs 5	1 vs 6	0 vs 4	2 vs 6	2 vs 7	0 vs 7	1 vs 3
Period 4	6 vs 7	4 vs 6	2 vs 5	1 vs 2	0 vs 5	3 vs 4	1 vs 3	0 vs 7

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

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Dummy
Period 1	0 vs 1	0 vs 2	4 vs 7	3 vs 6	3 vs 7	1 vs 5	2 vs 4	5 vs 6
Period 2	2 vs 3	1 vs 7	0 vs 3	5 vs 7	1 vs 4	0 vs 6	5 vs 6	2 vs 4
Period 3	4 vs 5	3 vs 5	1 vs 6	0 vs 4	2 vs 6	2 vs 7	0 vs 7	1 vs 3
Period 4	6 vs 7	4 vs 6	2 vs 5	1 vs 2	0 vs 5	3 vs 4	1 vs 3	0 vs 7

- 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

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

MIPLIB

MIP solver limit

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

# teams	# fails	Time (in s)
8	10	0.01
10	24	0.06
12	58	0.2
14	21	0.2
16	182	0.6
18	263	0.9
20	226	1.2
24	2702	10.5
26	5,683	26.4
30	11,895	138
40	2,834,754	6h

MIPLIB

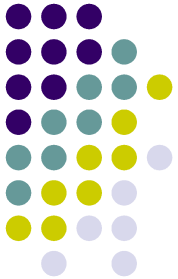
MIP limit

First model limit

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

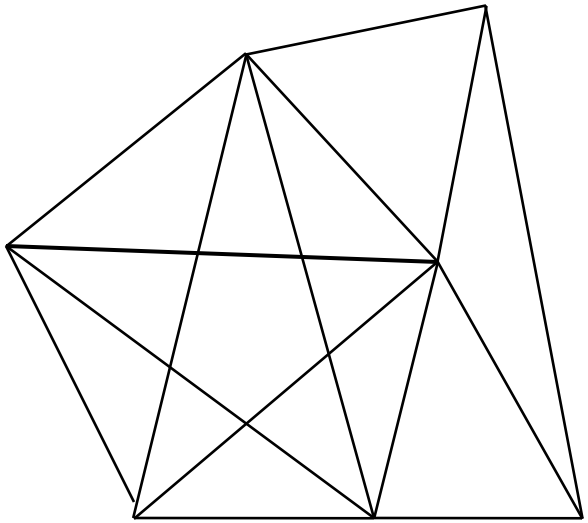


- General Principles
- CP vs MIP
- Integration of OR algorithms
- A MIPLIB example: sports scheduling
- **The maximum clique problem**
- Strength of CP
- Weakness of CP
- New research areas
- Conclusion

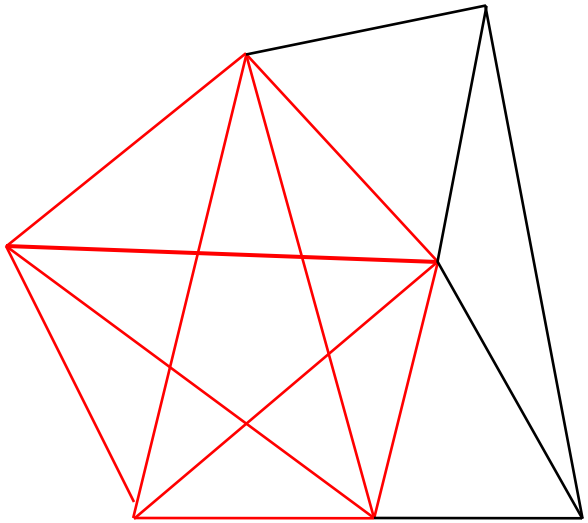
Motivations

- 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



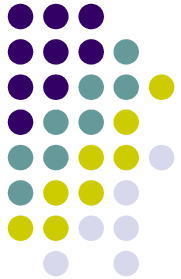
The problem: studies

- 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 …)
- DIMACS challenge in 1993.
- 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 (UBmaxClique(candidate) <= |K| - |Current|) then fails

Efficient filtering algorithm is required

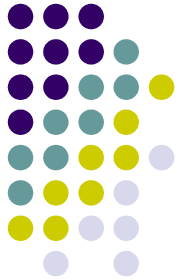


- Basic program performs more than 5 millions of backtracks per second.
- Some people consider that this is not possible to use FA + propagation for this kind of problem because it will be too long. Therefore, we need efficient properties that can be efficiently computed.
- To be worthwhile the FA must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

Max-Clique

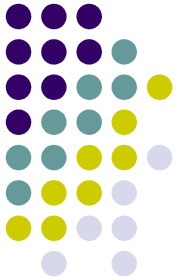
- Other exact method: find the best possible upper bound and check with current. The check can be long.
- **In CP**: the UB + propagation must be considered and not only the UB.
- **UB1 better than UB2 and better than UB3 but**
  UB2 + UB3 + propagation can be better than UB1

Max-Clique

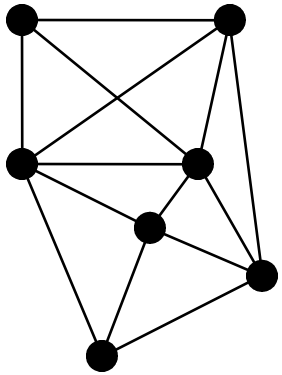


- **Classical method**:
  repeat:
  select a node
  remove nodes by applying the strongest property
- **In CP**:
  repeat:
  select a node
  while (a node is removed)
  remove nodes by applying **several** properties
  end while

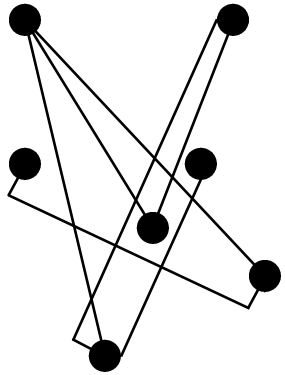
Max-Clique with CP



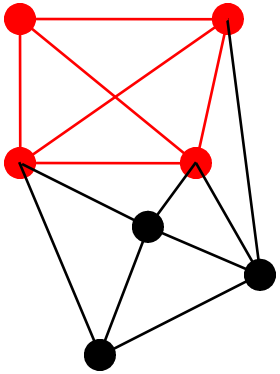
- Ideas:
  - Find a good ub which is easy and quick to compute



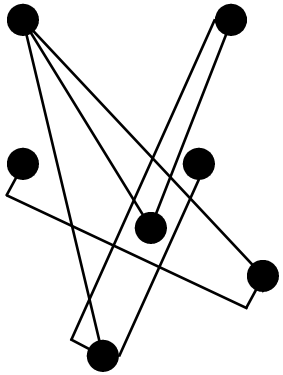
G



CG

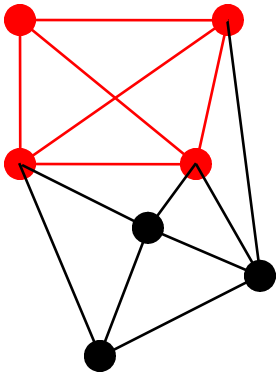


G



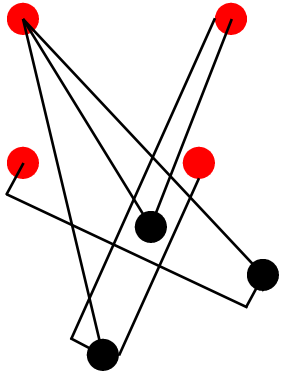
CG

Max clique



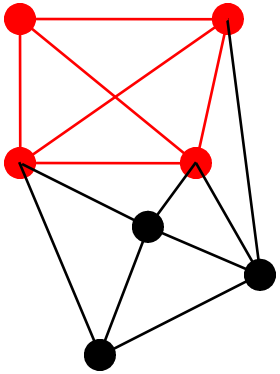
G

Max clique



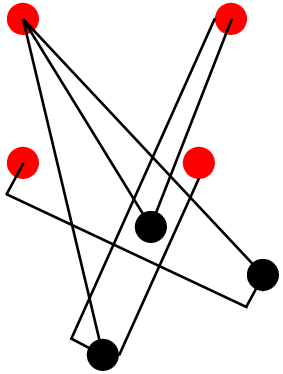
CG

Max Independent Set



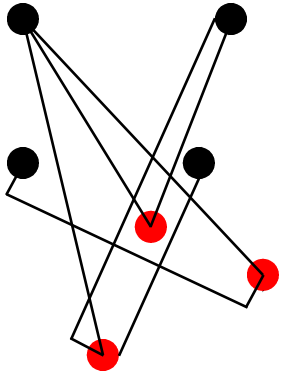
G

Max clique



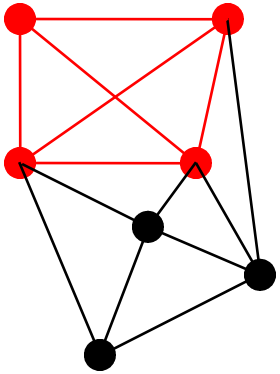
CG

Max Independent Set



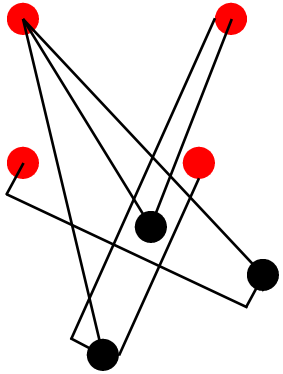
CG

Min vertex Cover



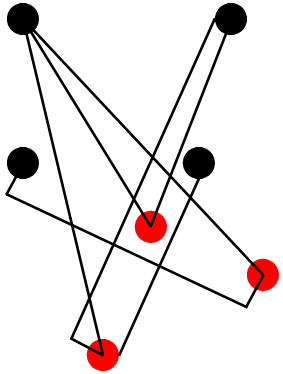
G

Max clique



CG

Max Independent Set

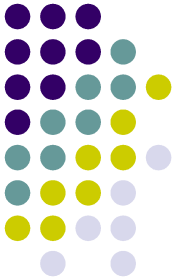


CG

Min vertex Cover

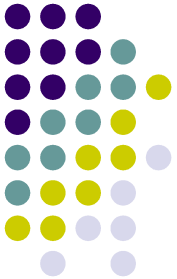
maxClique(G)=maxIndependent(CG)=n-minCover(CG)

UB of maxClique



- maxClique(G)=n – minCover(CG). So: maxClique(G) ≤ n – LBminCover(CG)
- UBmaxClique(G) = n – LBminCover(CG)
- Idea: find LBminCover(CG)

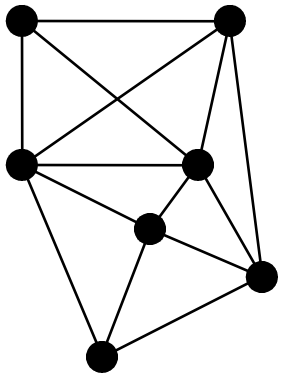
UB of maxClique

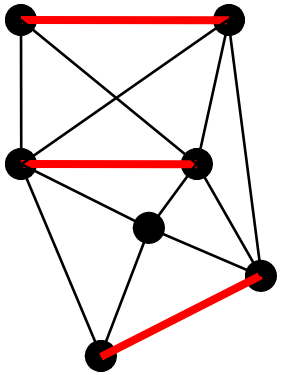


- maxClique(G)=n – minCover(CG). So:
  maxClique(G) ≤ n – LBminCover(CG)
- UBmaxClique(G) = n – LBminCover(CG)
- Idea: find LBminCover(CG)
- Well known:
  minCover(G) ≥ maxMatching(G)
  equality when G is bipartite

UB of maxClique

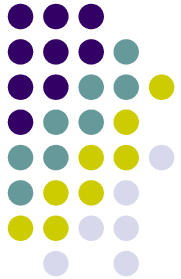
- Matching: set of edges such that they have no node in common





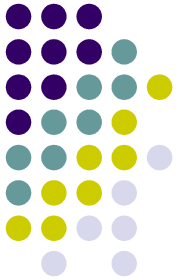
Vertex Cover: all the edges must be covered,
therefore any vertex cover contains
at least one node of every edge of the matching

UB of maxClique



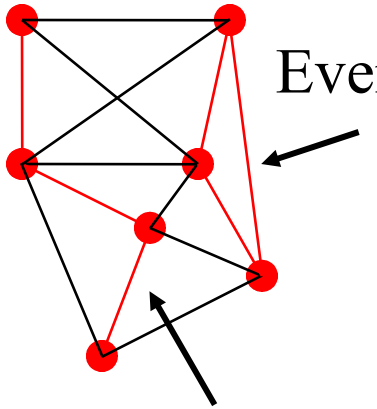
- Drawback: G can be non-bipartite and the matching algorithm is quite complex
- Goal: try to find an UB easier to implement and better.

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



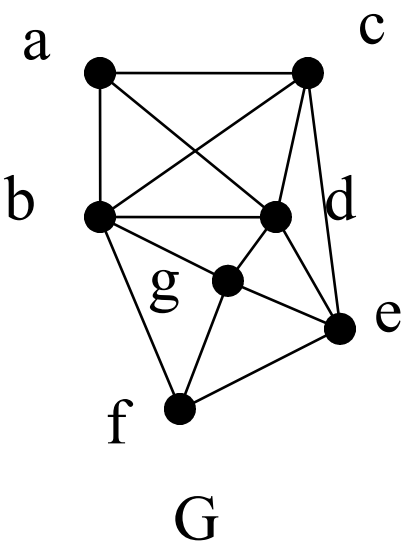
Every vertex cover involves at least $\lceil k/2 \rceil$ nodes of a Cycle of length k

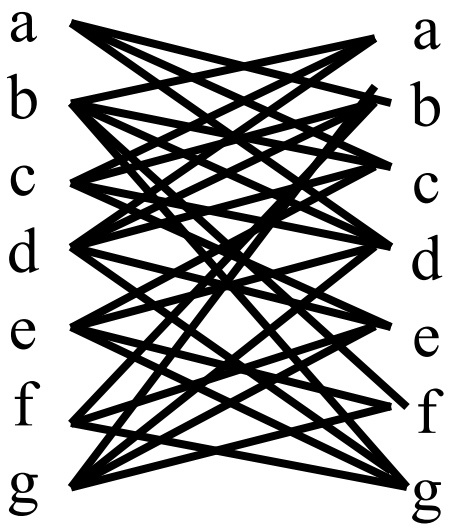
Every vertex cover involves at least $\lceil k/2 \rceil$ nodes of a Path of length k

k = number of edges

Matching gives 3,
Our formula gives $\lceil 3/2 \rceil + \lceil 3/2 \rceil = 2+2=4$

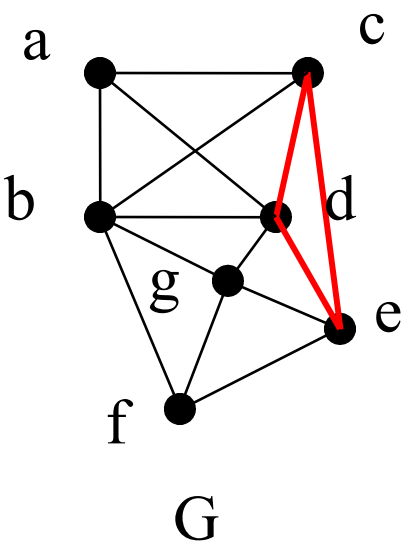
LB of minCover

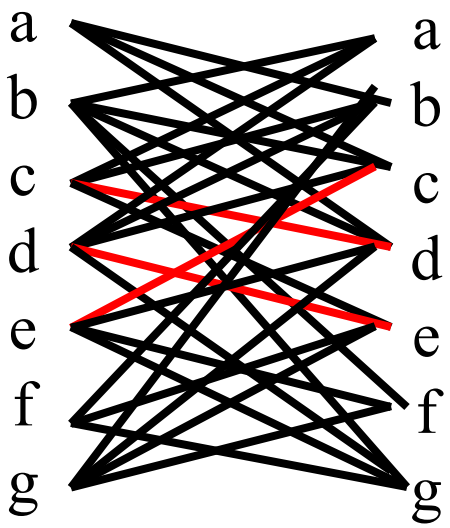




A cover by node disjoint paths and cycles can be found by searching for a matching in the “duplicated graph”
and then by projecting this matching to the original graph

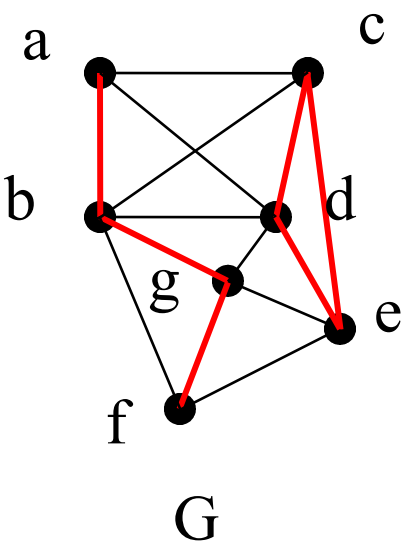
LB of minCover

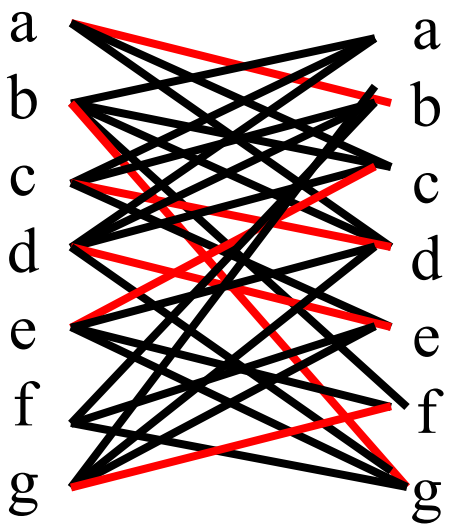




A cover by node disjoint paths and cycles can be found by searching for a matching in the “duplicated graph”
and then by projecting this matching to the original graph

LB of minCover





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

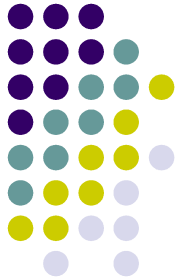
LB of minCover

- Another LB of minCover:
  minCover(G) ≥ ⌈maxMatching(DG)/2 ⌉

  note: ⌈maxMatching(DG)/2 ⌉ ≥ maxMatching(G)

- UB for maxClique:
  maxClique(G) ≤ n – ⌈maxMatching(DCG)/2 ⌉

LB of minCover



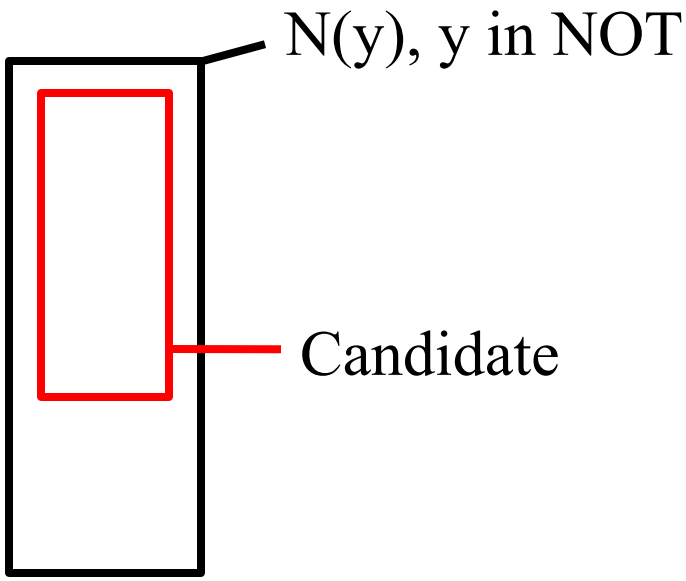
- Advantages:
  - DCG is bipartite (means simple algorithm)
  - Very good pre-test:
    maxClique(G) ≤ n – ⌈n/2 ⌉

    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.

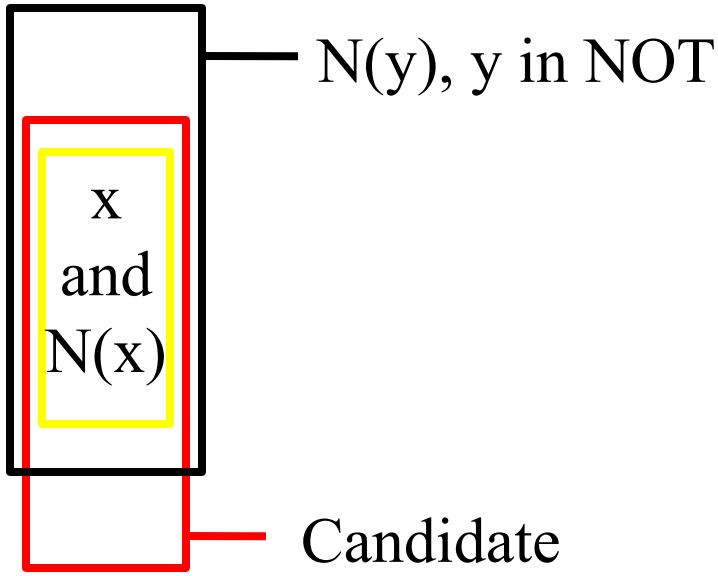
Bron & Kerbosh



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

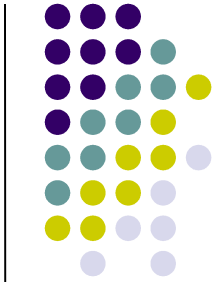
Filtering from B&K's idea



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.

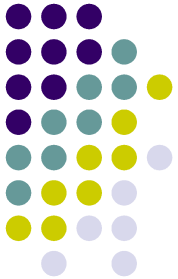
Max-Clique with CP



- **Propagation:**

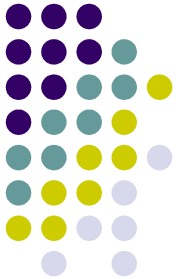
  while (a node is removed from Candidate) do
  call maxCliqueUBFilter
  call NotBasedFilter
  done

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_hat300-1: 40s instead of 800s
- P_hat700-2: 250s instead of 2200s
- An open problem closed in 150s

Results:
CP vs complete methods



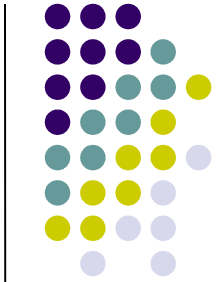
	Wood,	Ostegard,	Fahle,	Regin
#solved	38	36	45	52
< 10 min	38	35	38	44
best time	15	26	10	30
best LB	0	0	1	9

•Ostegard (Dynamic programming, RDS in CP)
in less than 10 min: 350s CP: 285s

Results:
CP vs heuristic methods

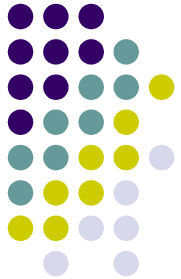
- Qualex: 50 best bounds
- St-Louis, Gendron, Ferland (Optimization days): 50 best bounds
- CP: 58 (52 proved)
- CP < 10 min: 49 (44 proved)
- CP < 1 min: 41 (37 proved)

Plan



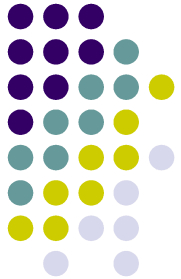
- General Principles
- CP vs MIP
- Integration of OR algorithms
- A MIPLIB example: sports scheduling
- The maximum clique problem
- **Strength of CP**
- Weakness of CP
- New research areas
- Conclusion

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

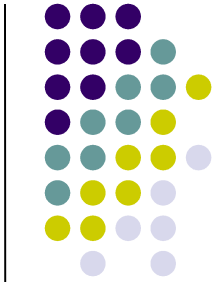
Plan

- 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

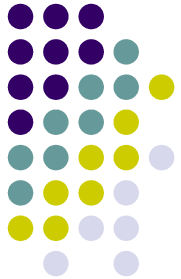
- Must be improved for optimization problems: spend too much time in proving sub-optimality
  - First step: integration of cost in the constraints
- Sometimes lack of global point of view
- Dark zones: press Enter key then ?
- Relaxation is not good for CP. We learn relaxation at school!

Plan



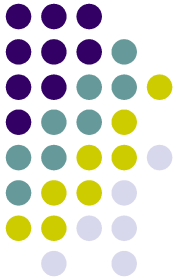
- General Principles
- CP vs MIP
- Integration of OR algorithms
- A MIPLIB example: sports scheduling
- The maximum clique problem
- Strength of CP
- Weakness of CP
- **New research areas**
- Conclusion

New research areas



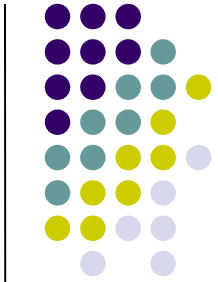
- New point of view: CP is based on filtering algorithm, i.e. : Given a property P defining a necessary condition for an element to be in a solution
  Find as quickly as possible ALL elements that do not satisfy P
- Ex: alldiff constraint and matching, cardinality constraint and flows etc…
- Close to sensitivity analysis, but also different (for instance we only have monotonic modifications).

New research areas



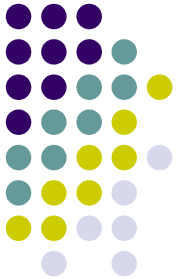
- Consider a Minimization problem, and OBJ the objective. Suppose that we found a “solution” with OBJ=25.
- We will reject any “solution” with OBJ > 24. So if x=a leads to an OBJ > 24 then value a must be removed from D(x).
- If we have a lower bound of OBJ then we can use it: if lb(OBJ,x=a) > 24 then 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



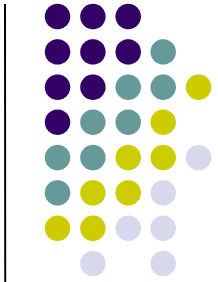
- The very same algorithm is called thousand times (million sometimes)
- The incremental aspect of the algorithm becomes really important.

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