Simple solutions for complex problems

ACP Research Excellence Award talk

Jean-Charles RÉGIN

Université Nice-Sophia Antipolis,
CNRS, I3S, UMR 7271, France
jcregin@gmail.com

Constraint Programming

- 3 notions:
  - constraint network: variables, domains constraints
  + filtering (domain reduction)
  - propagation
  - search procedure (assignments + backtrack)

- If there is no filtering then this is not CP.

General considerations

- **When solving a problem in CP:**
- Potential performance gain:
  - data structure optimization (code): x 10
  - search strategies: x 1 000
  - model : x 1 000 000
- Chance of success
  - data structure optimization (code): 95 %
  - search strategies: 1 %
  - model: 0,001 %
- In this talk, I will mainly speak of modeling

- « With Distribute and Table constraints I can prototype any problem » said an ILOG consultant

- Distribute = flow based constraint

Plan

- A simple solution: Flow based constraints
  - Definition
  - Filtering algorithms
  - Their incredible modeling power
- What is missing?
- What could be the evolution?
- Conclusion

Solvers in the 90s

- Only simple constraints were used (given in extension, arithmetic (>, >=, +, …)
- When people tried to solve some real world problems they discovered that
  - It was not easy to define some problems and they repeatedly used the same code **(lack of expressiveness)**
  - Some deductions was not made **(lack of filtering)**

Solvers in the 90s

- **Solution:** They proposed (Beldiceanu et al., Puget …) to introduce more global constraints and to define some filtering rules.

- Flow based constraints are the most popular
- **Thanks to them we have simple solutions to complex problems**

Flow based constraints

- These are constraints that may be expressed by a flow.
- The best examples are the alldiff and the global cardinality constraints (JC Régin AAAI-94, JC Régin AAAI-96, JC Régin CP'99 and JC Régin Constraints 02, JC Régin and C. Gomes CP'04, JC Régin CPAIOR'05)
- We will see that a huge number of constraints may be reformulated as a flow based constraints

Some references

- K. Stergiou and T. Walsh. **The difference all-difference makes.** In IJCAI-99.
- Willem Jan van Hoeve: its phD thesis, **The alldifferent Constraint: A Survey.**
- I. Gent, I. Miguel and P. Nightingale, **Generalised Arc Consistency for the AllDifferent Constraint: An Empirical Survey**, Artificial Intelligence, 2008.
- Peter Nightingale, **The Extended Global Cardinality Constraint: An Empirical Survey**, Artificial Intelligence, 2011.

Some references

- C. Bessiere, N. Narodytska, C-G. Quimper, T. Walsh:
  - **The AllDifferent Constraint with Precedences.** CPAIOR'11
  - **Propagating Conjunctions of AllDifferent Constraints.** AAAI-10

Flow based constraints: filtering

- Filtering of 0-1 arcs
- Introduction of card variables
- Filtering of 0-1 arcs with costs
- Identification of constant flow value arcs
- Particular case of costs on cardinality variables only
- Convex graphs (graph having the 0-1 property)

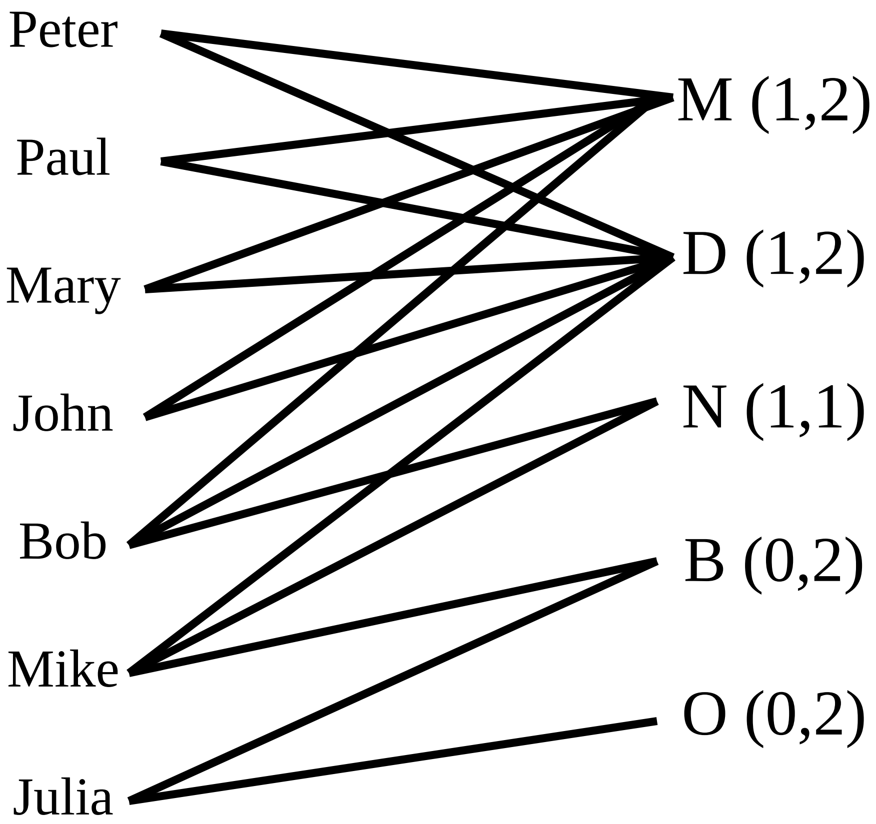
Flows

- Let G be a graph in which every arc (i,j) is associated with 2 integers:
  - l(i,j) the lower bound capacity of the arc
  - u(i,j) the upper bound capacity of the arc
- A flow is a function f satisfying:
  - For any arc (i,j), f(i,j) represents the amount of some commodity that can ``flow'' through the arc.
    Such a flow is permitted only in the indicated direction of the arc, i.e., from i to j.
    For convenience, we assume f(i,j)=0 if (i,j) is not an arc.
  - A **conservation law** is observed at each node:
    for every node j: $\sum f(i,j) = \sum f(j,k)$.

Flows

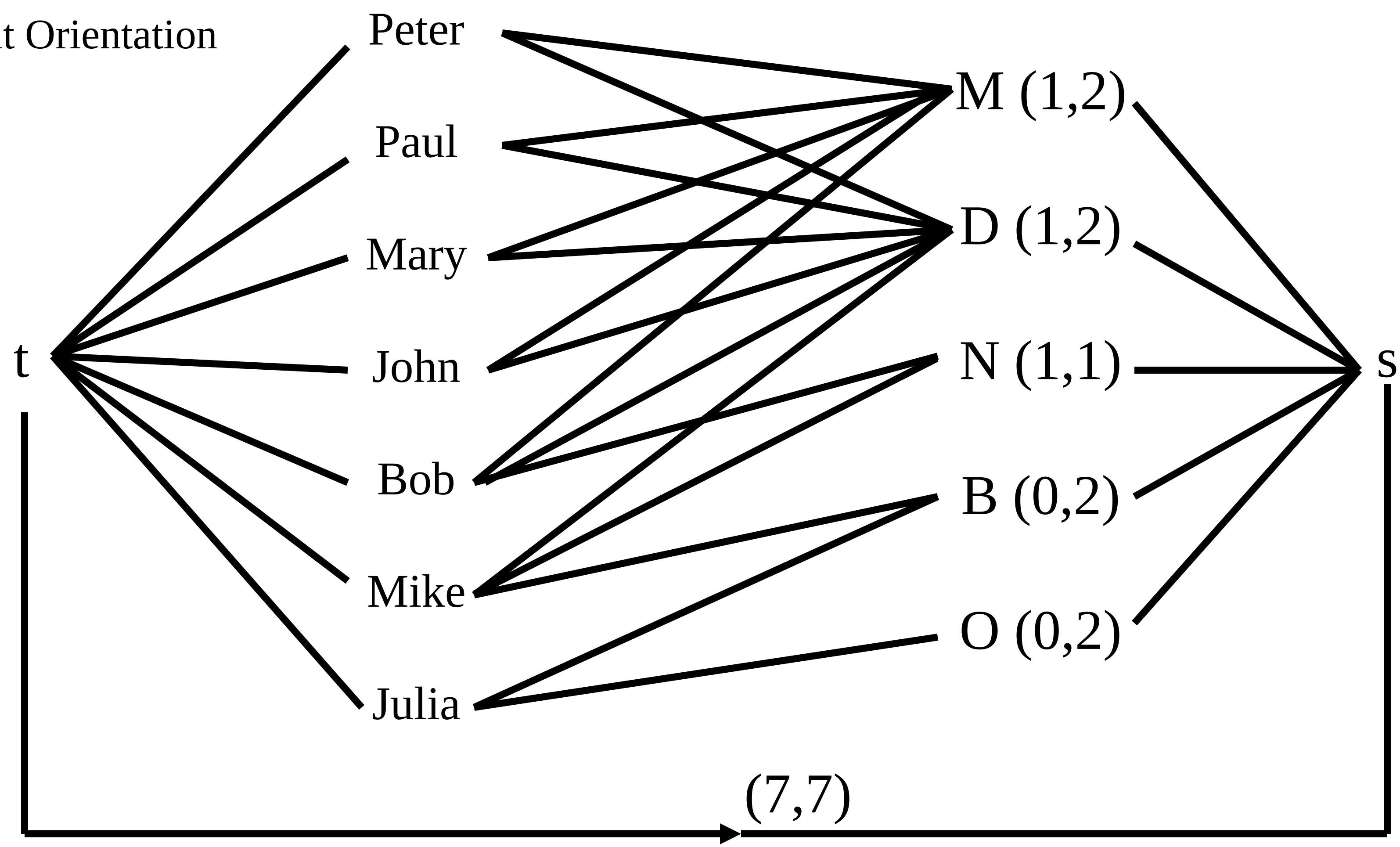
- The feasible flow problem:
  - Does there exist a flow in G that satisfies the capacity constraints?
    That is find f such that
    for every arc (i,j) in $U(G)$: $l(i,j) \leq f(i,j) \leq u(i,j)$.
- The problem of the maximum flow for an arc (i,j):
  - Find a feasible flow in G for which the value of $f(i,j)$ is maximum.

GCC



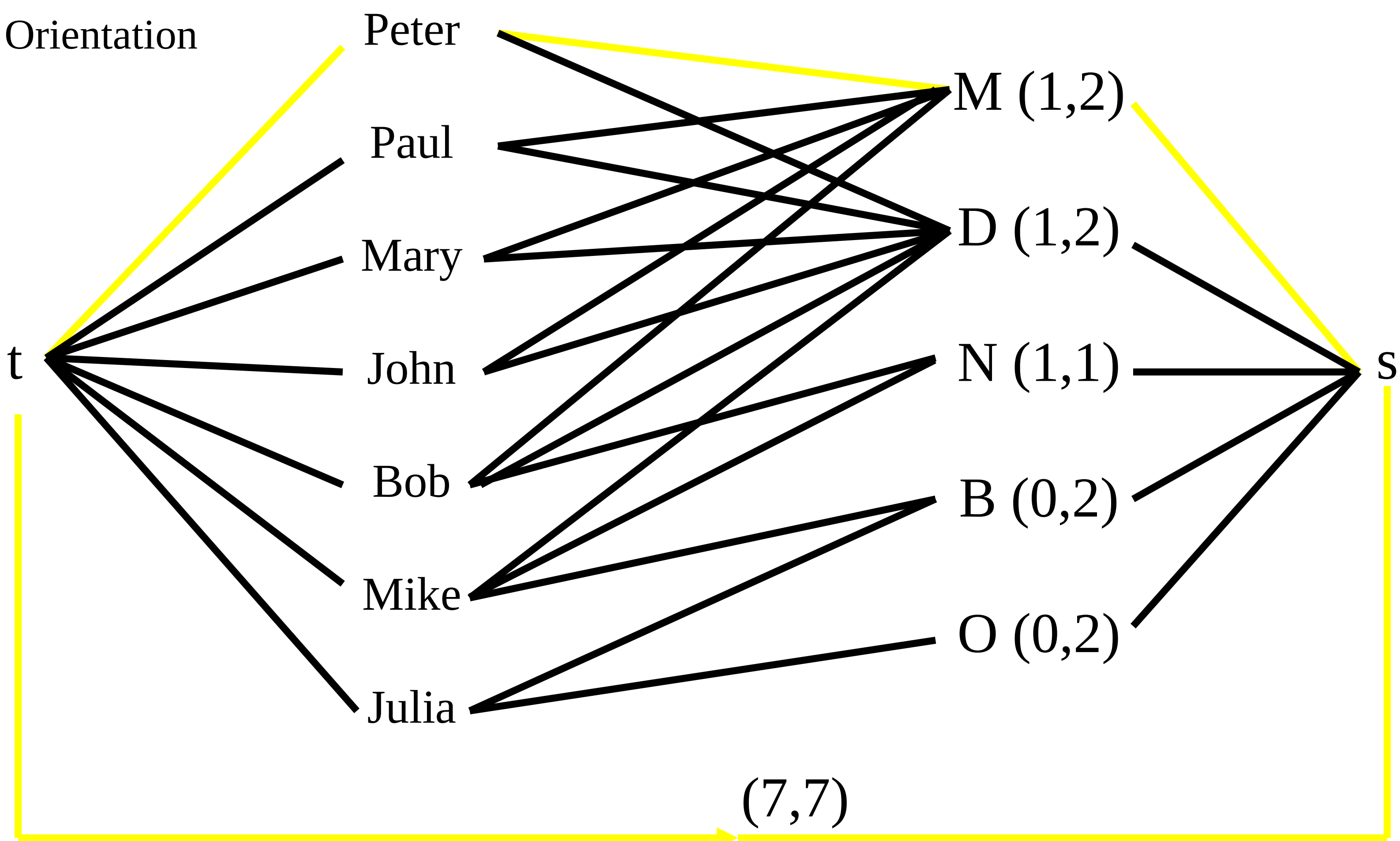
Value Network

Default Orientation



Feasible Flow

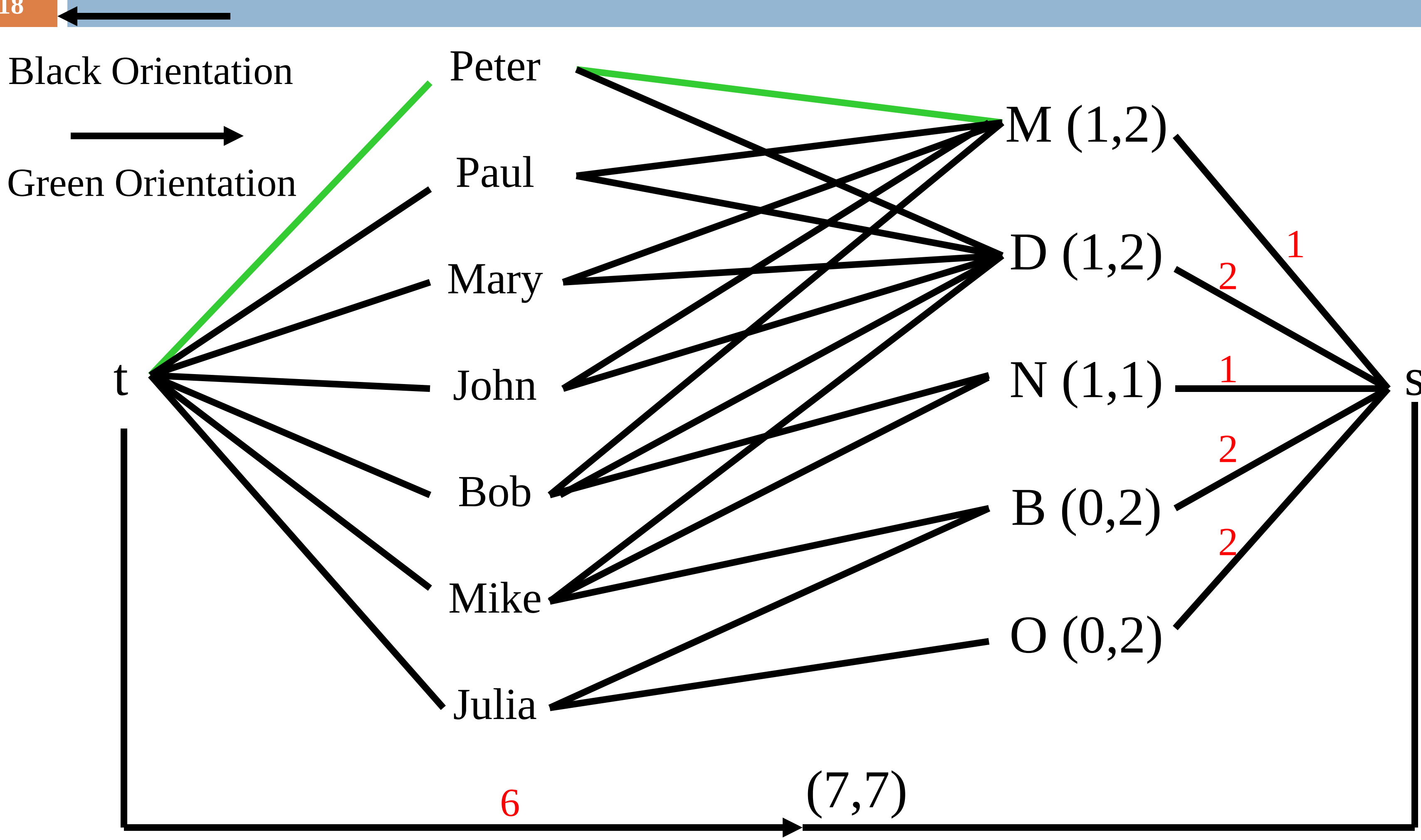
Black Orientation



Successive augmentation

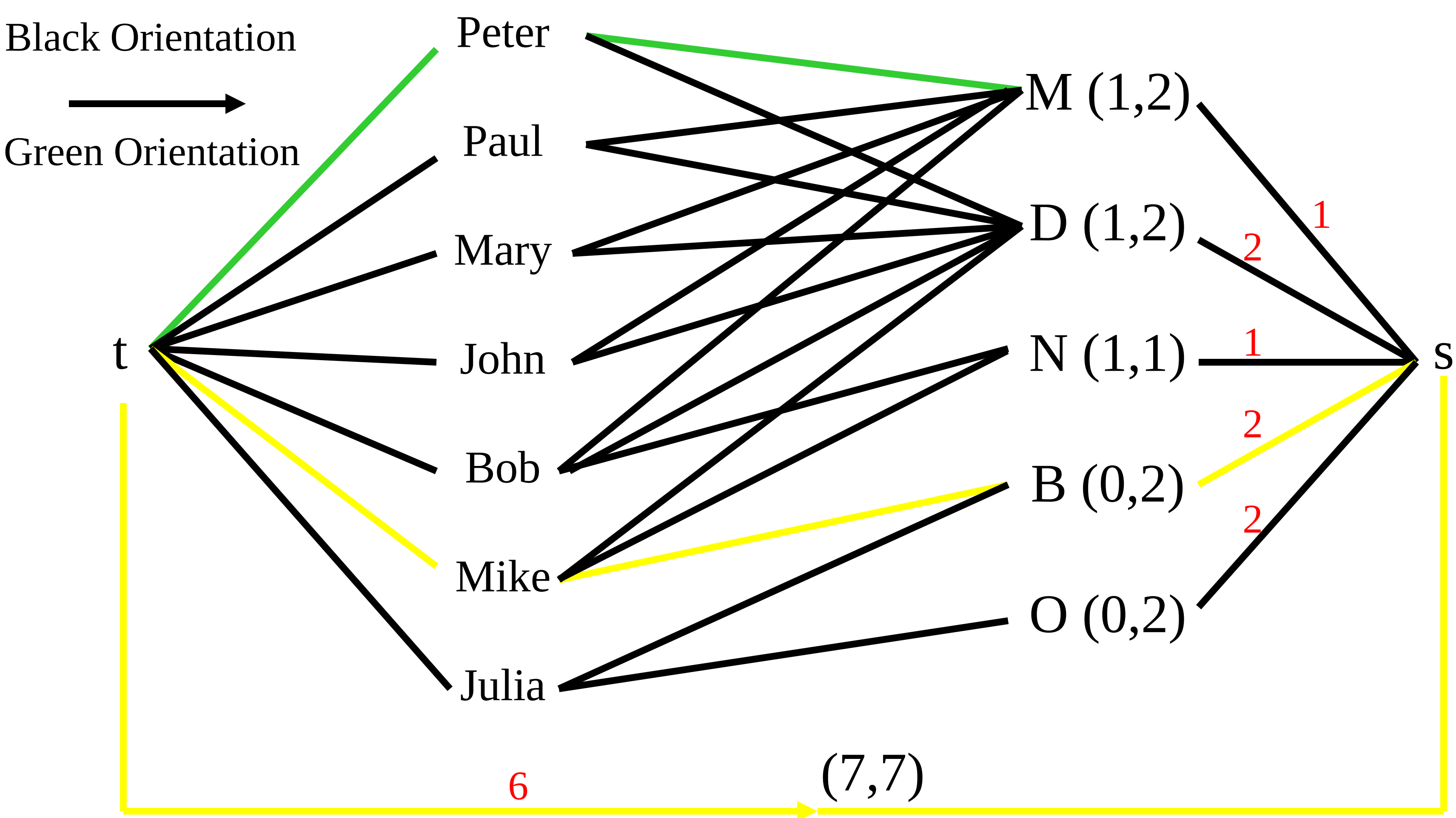
- Successive augmentation are computed in a particular graph:
  The **residual graph**
- The residual graph has **no lower bounds**
- In our case this algorithm is equivalent to the best ones.

Residual Graph



If xij < uij then (i,j) and rij= uij - xij If xij > lij then (j,i) and rij= xij - lij

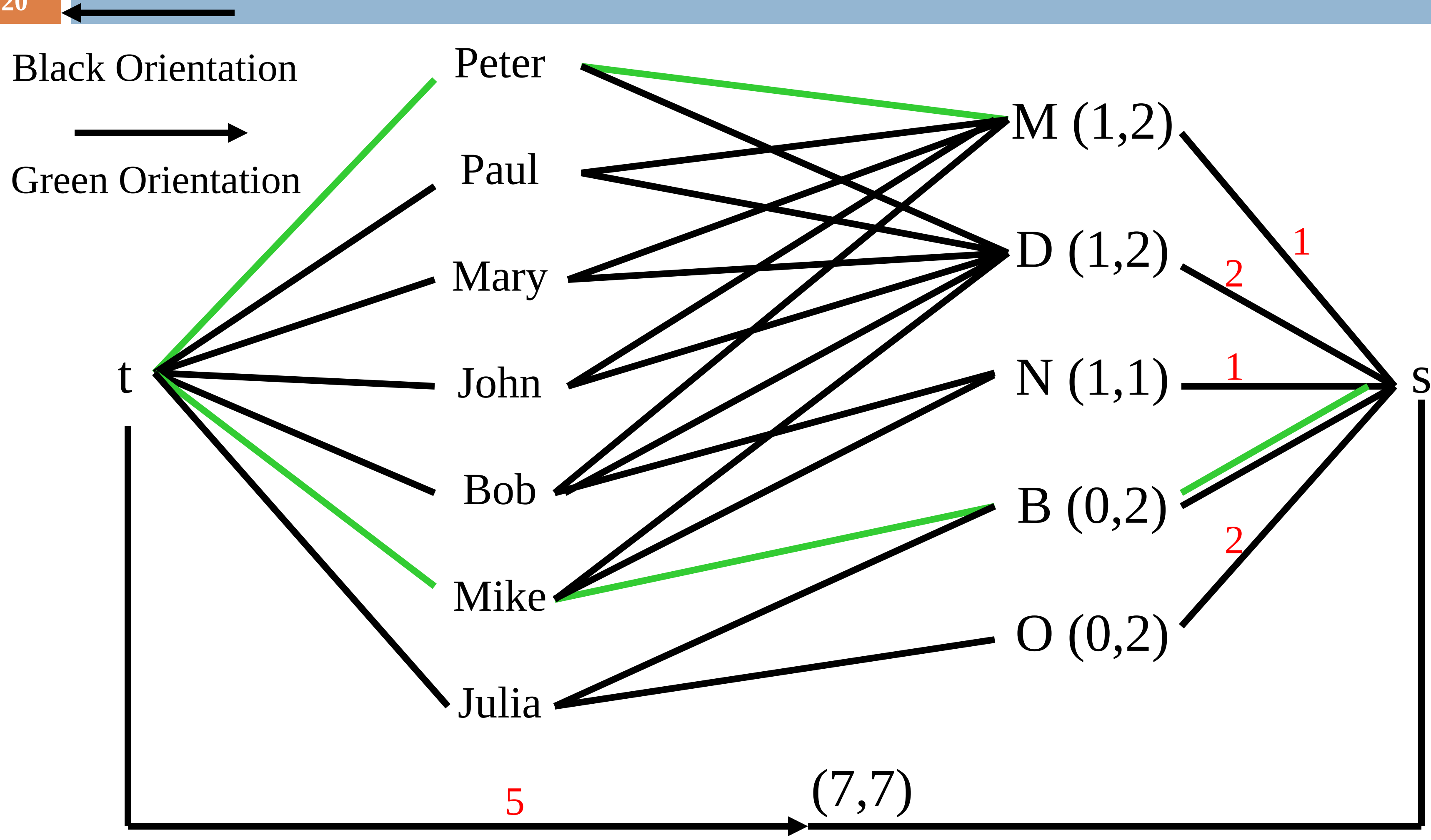
Residual Graph



If xij < uij then (i,j) and rij= uij - xij

If xij > lij then (j,i) and rij= xij - lij

Residual Graph

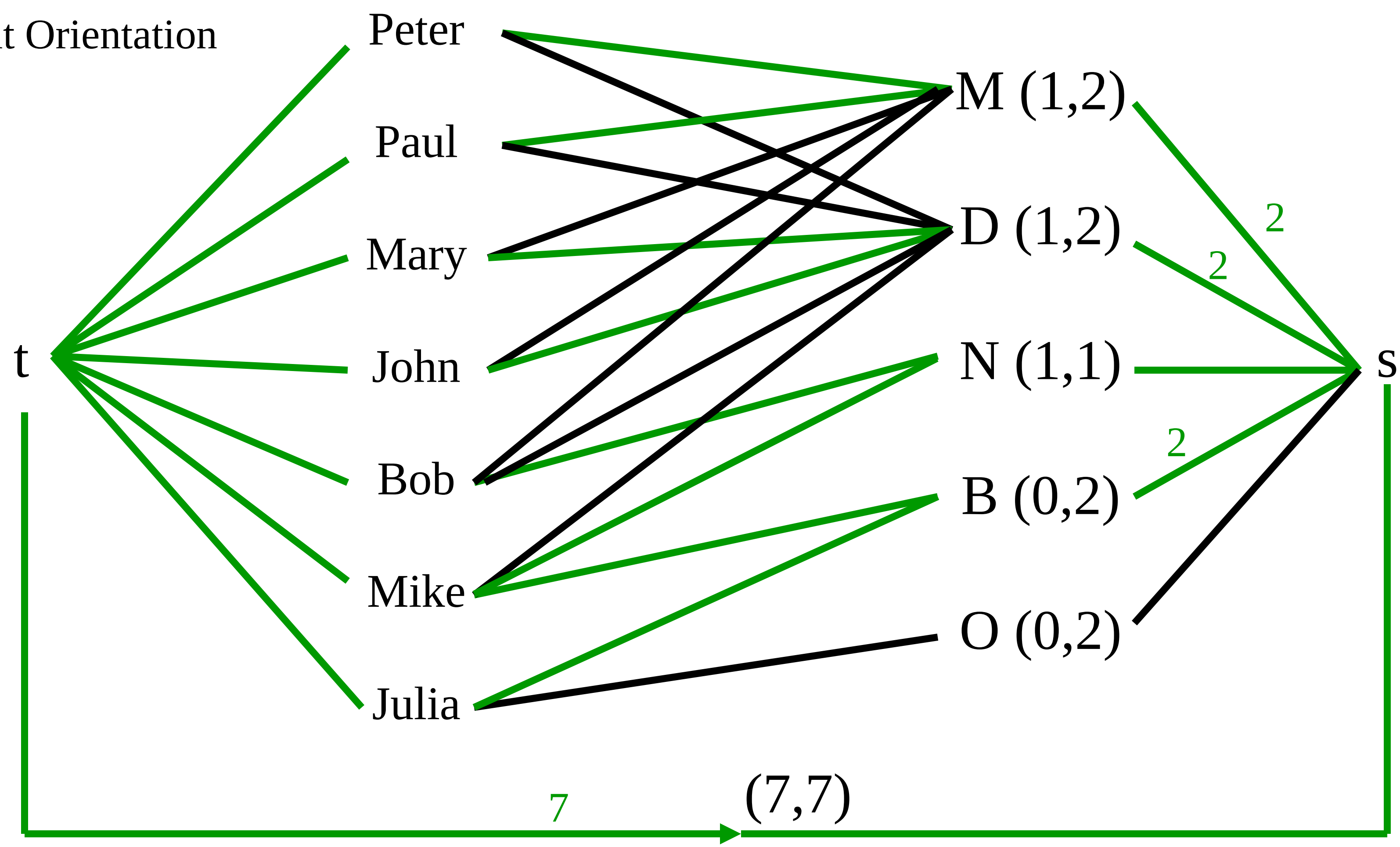


If xij < uij then (i,j) and rij= uij - xij

If xij > lij then (j,i) and rij= xij - lij

A Solution

Default Orientation

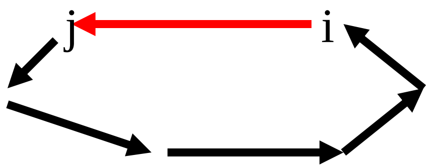


7 flow value

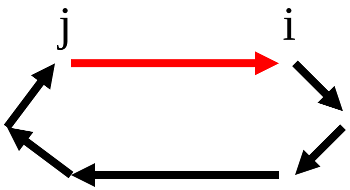
Sum = 7

Properties

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



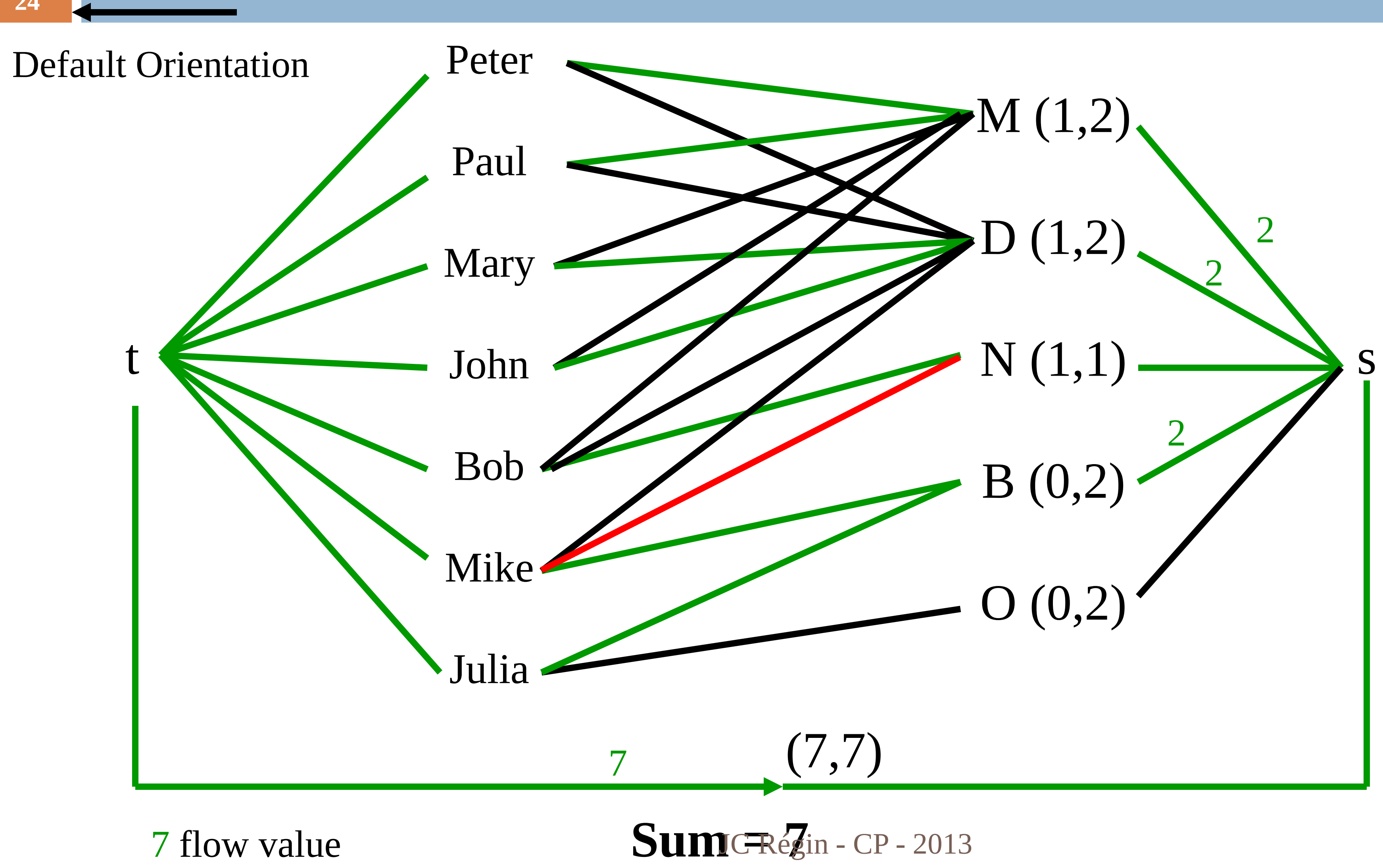
- The flow value xij of (i,j) can be decreased iff there is a path from i to j in R - {(i,j)}



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

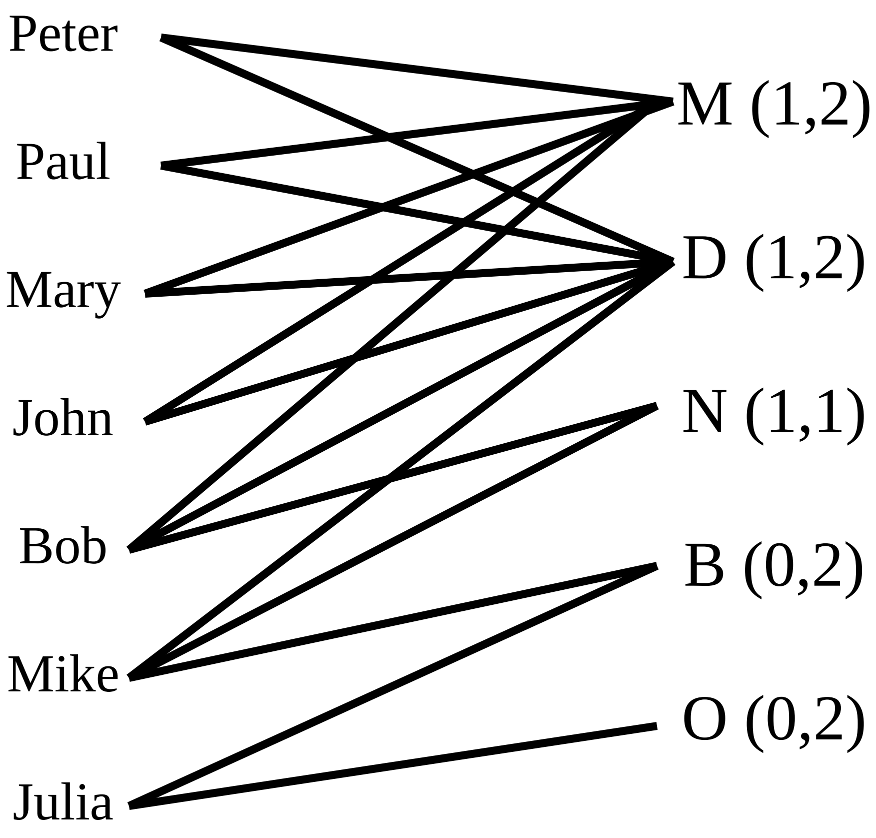
A Solution


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
Sum = 7

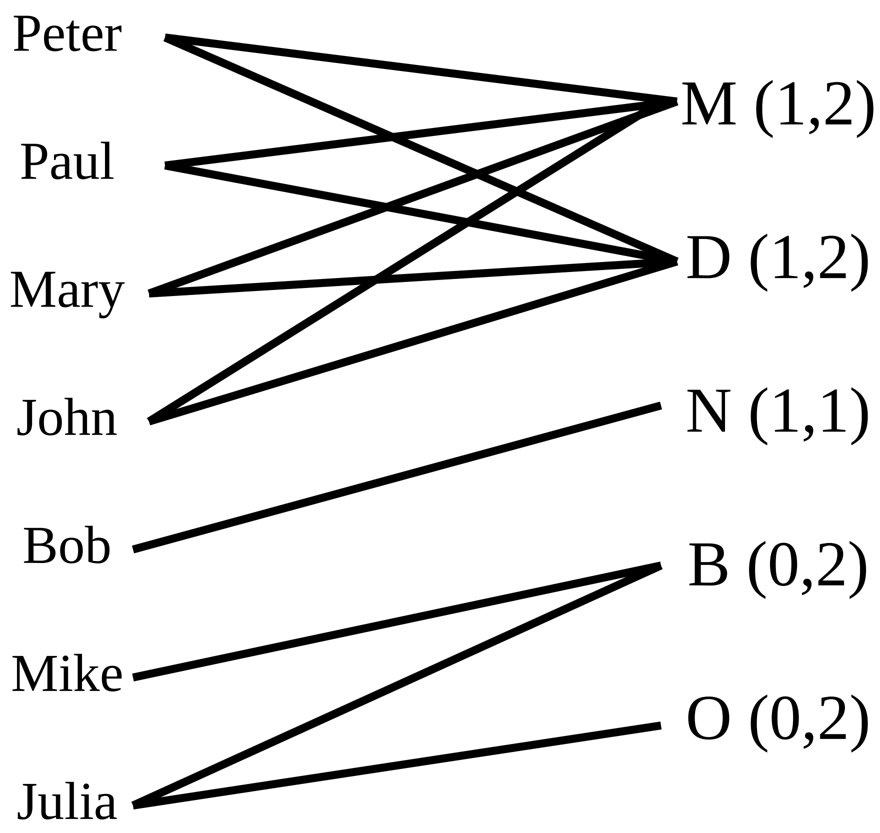
Filtering algorithm for GCC

- 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 establishing arc consistency: O(nd)**
- work well due to (0,1) arcs

GCC



GCC after AC



Flow based constraints: filtering

- Filtering of 0-1 arcs
- **Introduction of card variables**
- Filtering of 0-1 arcs with costs
- Identification of constant flow value arcs
- Particular case of costs on cardinality variables only
- Convex graphs (graph having the 0-1 property)

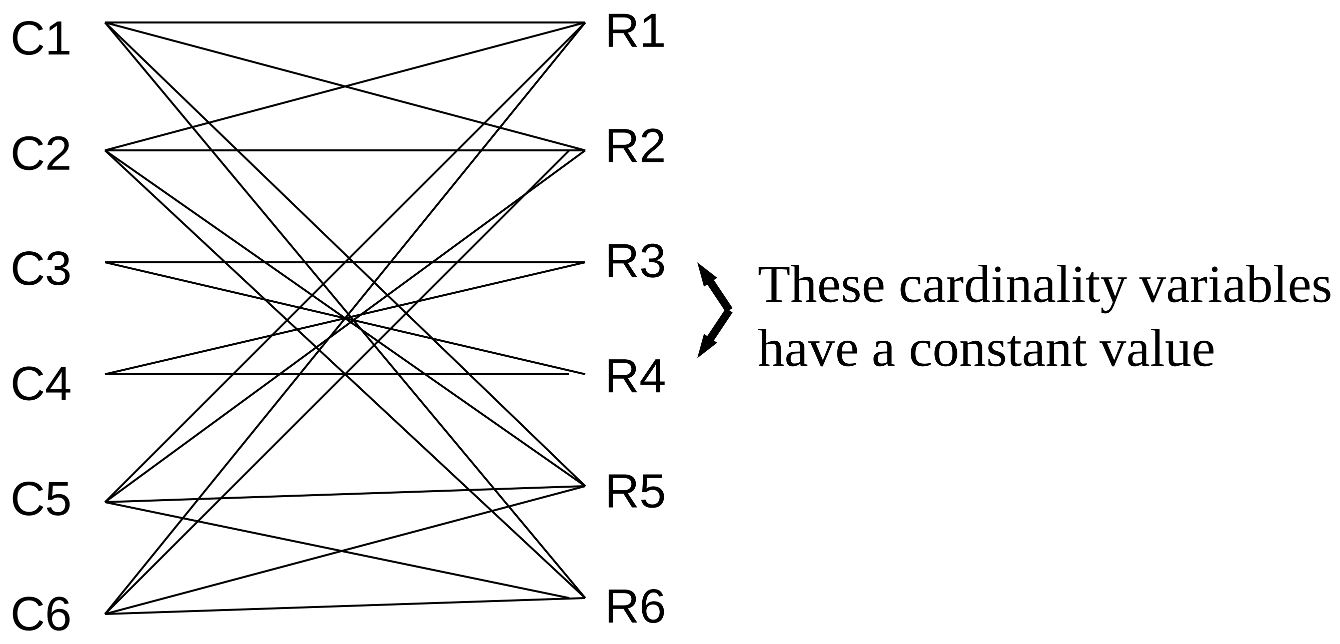
GCC with card variables

- In the original version, the boundaries of the gcc are integers
- They can be defined from variables and we may expect to filter the range of these variables
- This version is called a Gcc with cardinality variables

GCC with card variables

- Filtering algorithms are detailed in the paper: C-G Quimper, A. López-Ortiz, P. van Beek, and A. Golynski. « Improved algorithms for the global cardinality constraint », CP'04
  - Filtering all lower bounds cost n searches for an augmenting path (or reducing path)
    - O(nm)
  - Filtering all upper bounds is in $O(n^{2,66})$
- The extended GCC (cardinality variables are no long ranges and may have holes in their domain) is an NP-Complete (and so filtering is NP-Hard)

GCC with card variables



After the filtering algorithm we have two disjoint GCC
We can prove that for the strongly connected component which does not contain the sink, the cardinality variables have a constant value
(see Régin, Gomes, The Cardinality Matrix Constraint, CP 04)

Flow based constraints: filtering

- Filtering of 0-1 arcs
- Introduction of card variables
- **Filtering of 0-1 arcs with costs**
- Identification of constant flow value arcs
- Particular case of costs on cardinality variables only
- Convex graphs (graph having the 0-1 property)

Weighted GCC

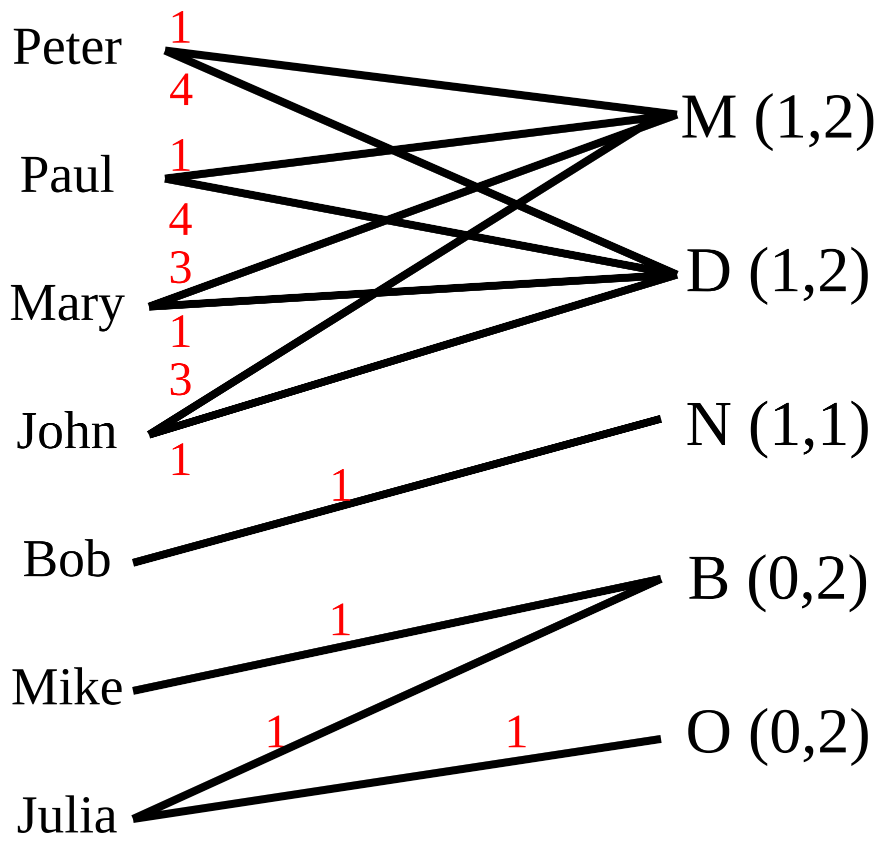
- We can add costs on the arcs
- We will have to solve a min cost flow problem

GCC with costs

- GCC with costs =
  Global cardinality constraint
  \+ Sum constraint on the assignment costs

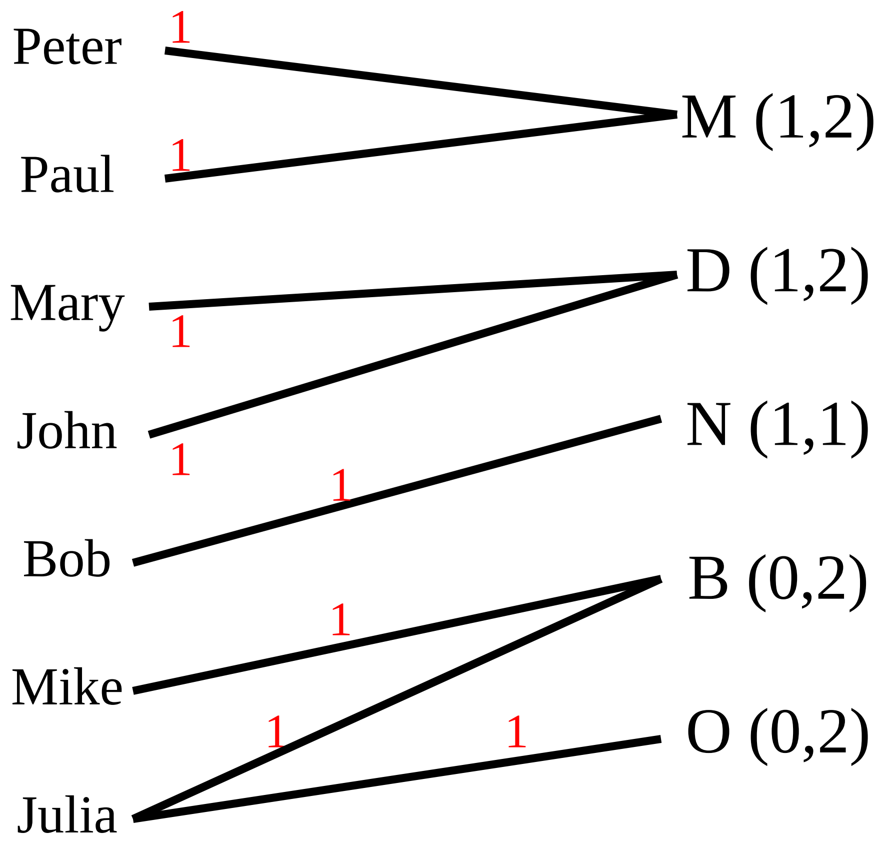
- This constraint should be named weighted constraints

GCC with costs



Sum < 12

Arc consistency



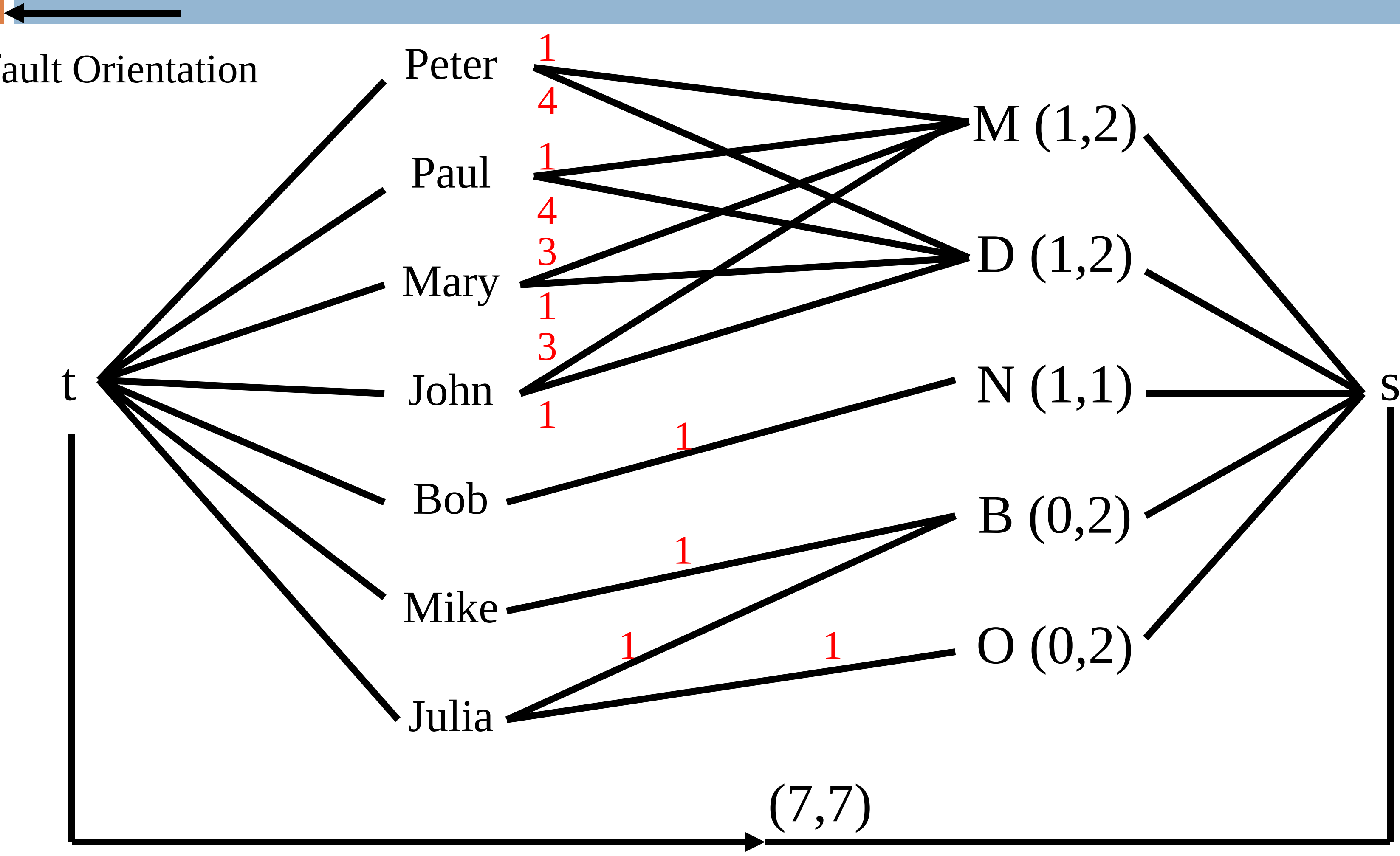
Sum < 12

GCC with costs

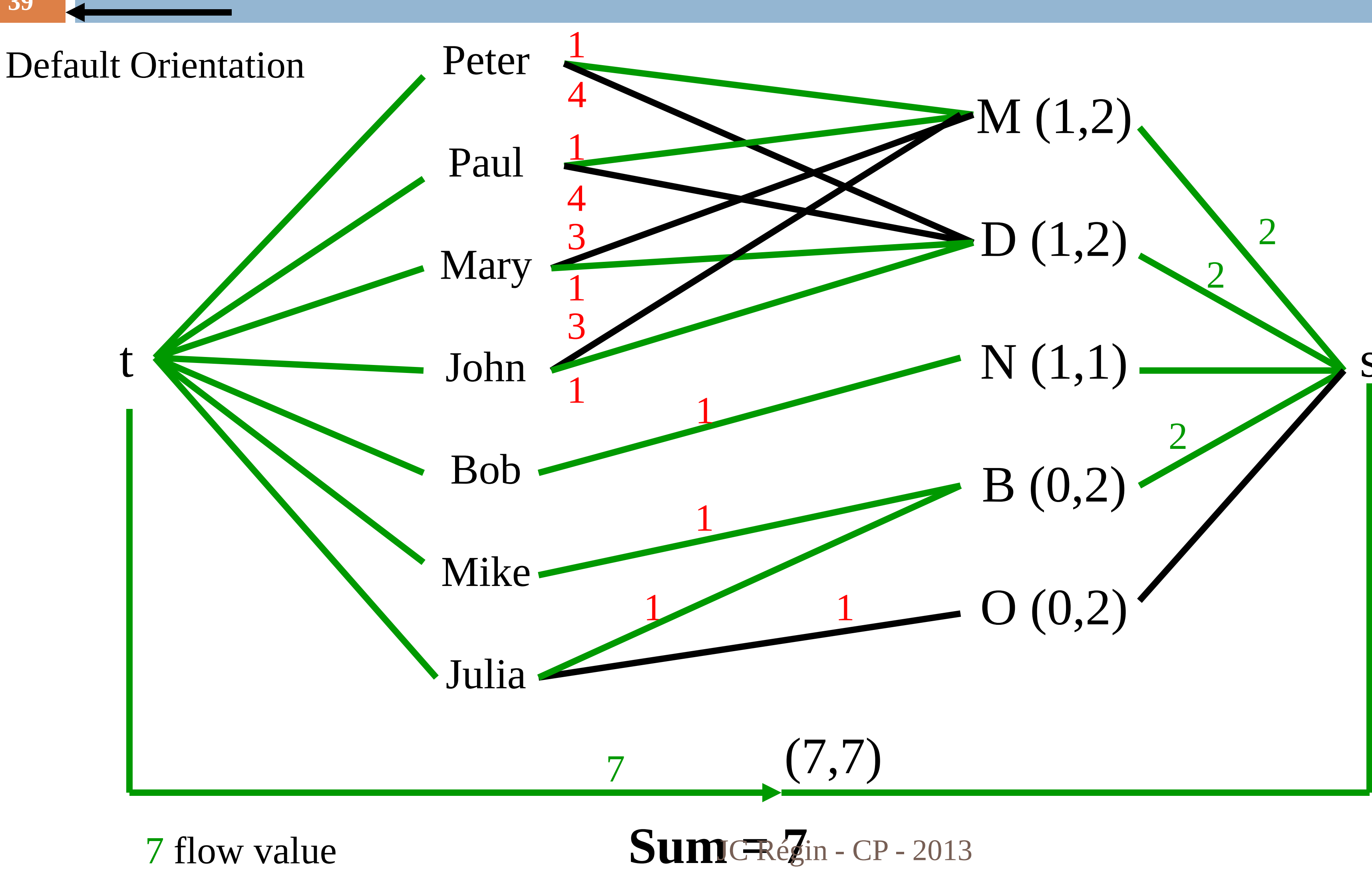
- Consistency can be computed by searching for a minimum cost flow
- Arc consistency can be computed by searching for shortest paths in a special graph.

Minimum Cost Flow

Default Orientation

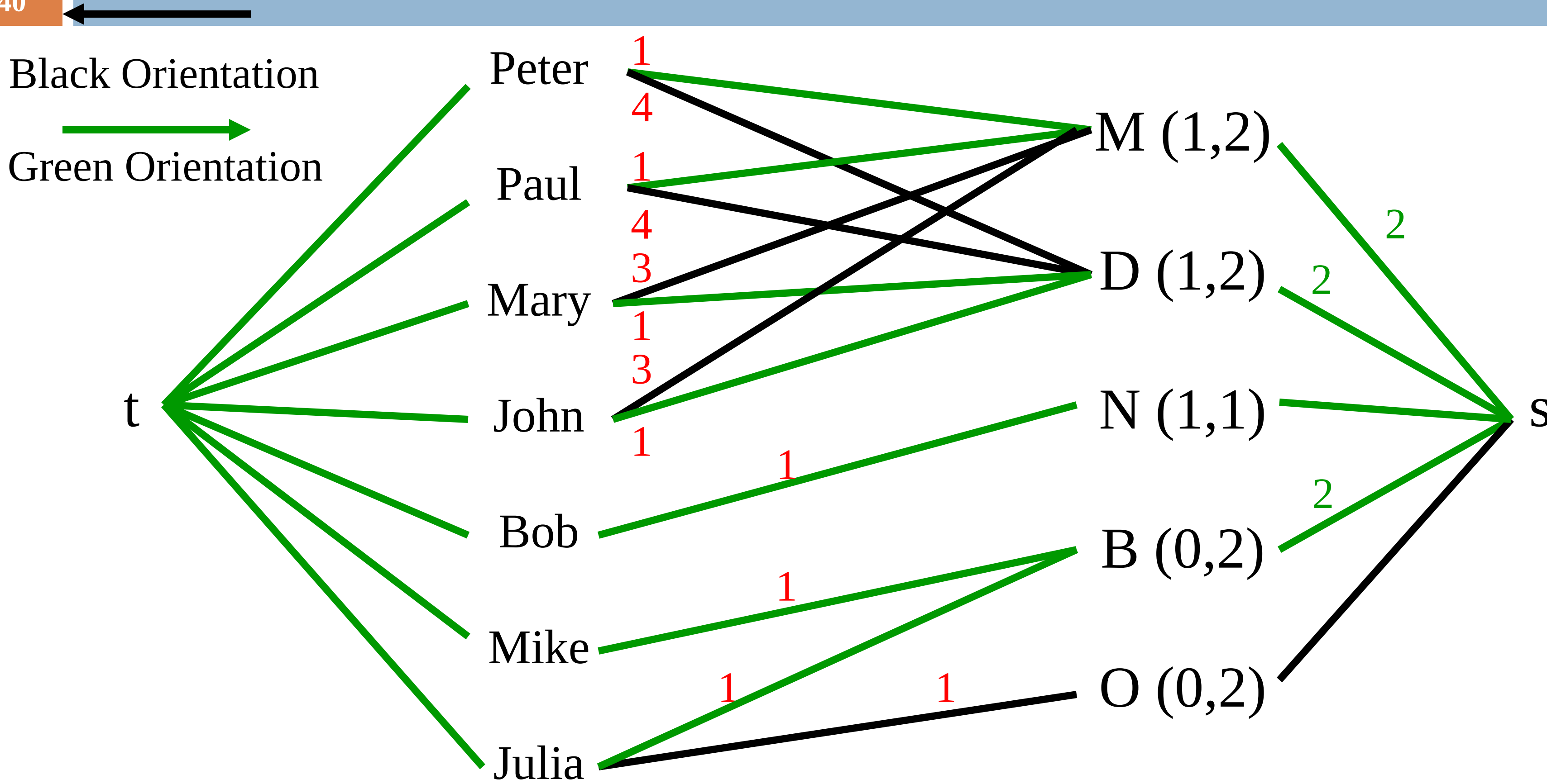


A Solution

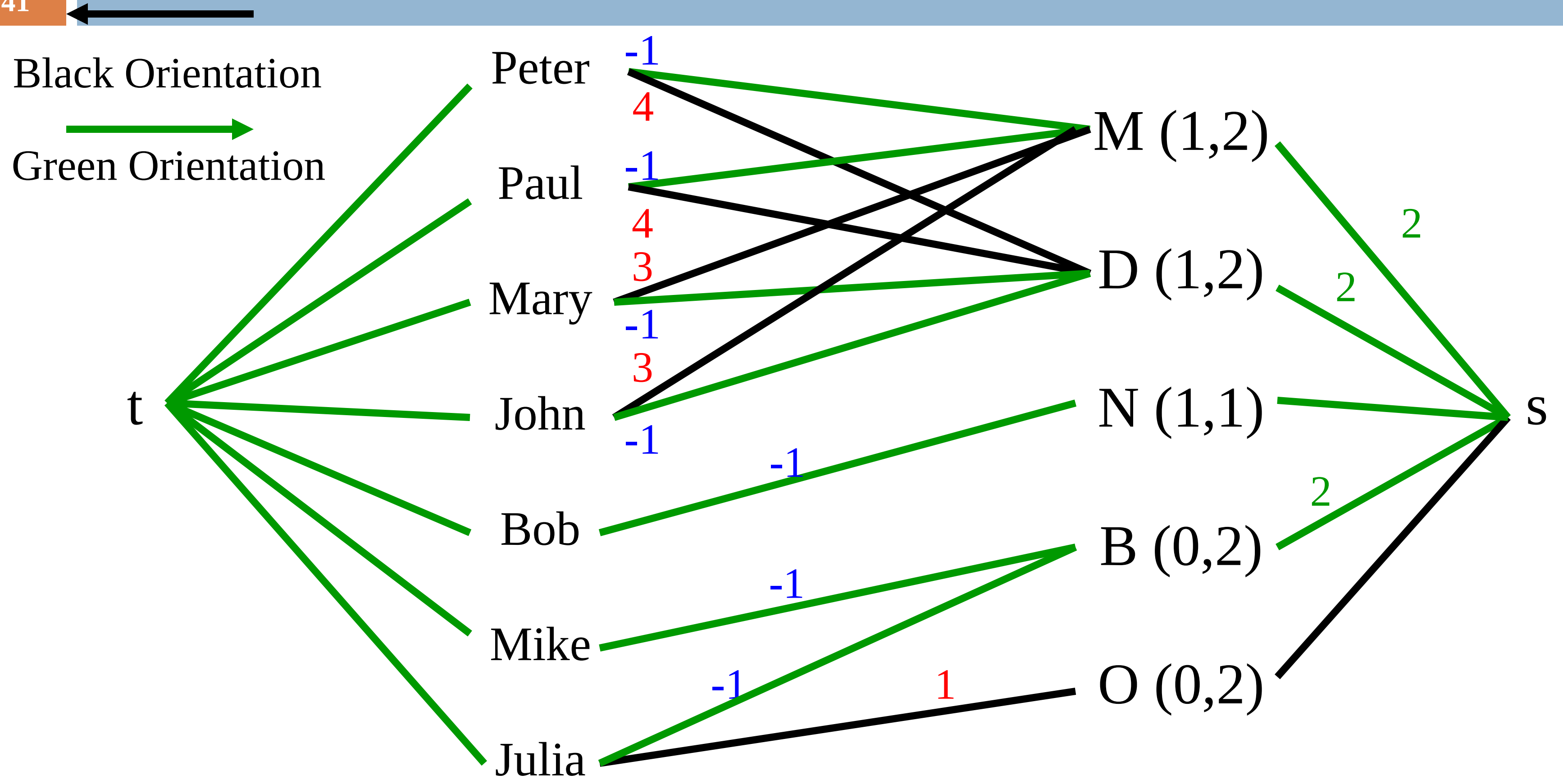


Default Orientation
Peter
Paul
Mary
John
Bob
Mike
Julia
t
s
M (1,2)
D (1,2)
N (1,1)
B (0,2)
O (0,2)
1
4
1
4
3
1
3
1
1
1
1
1
2
2
2
7
(7,7)
7 flow value
Sum = 7

Residual Graph

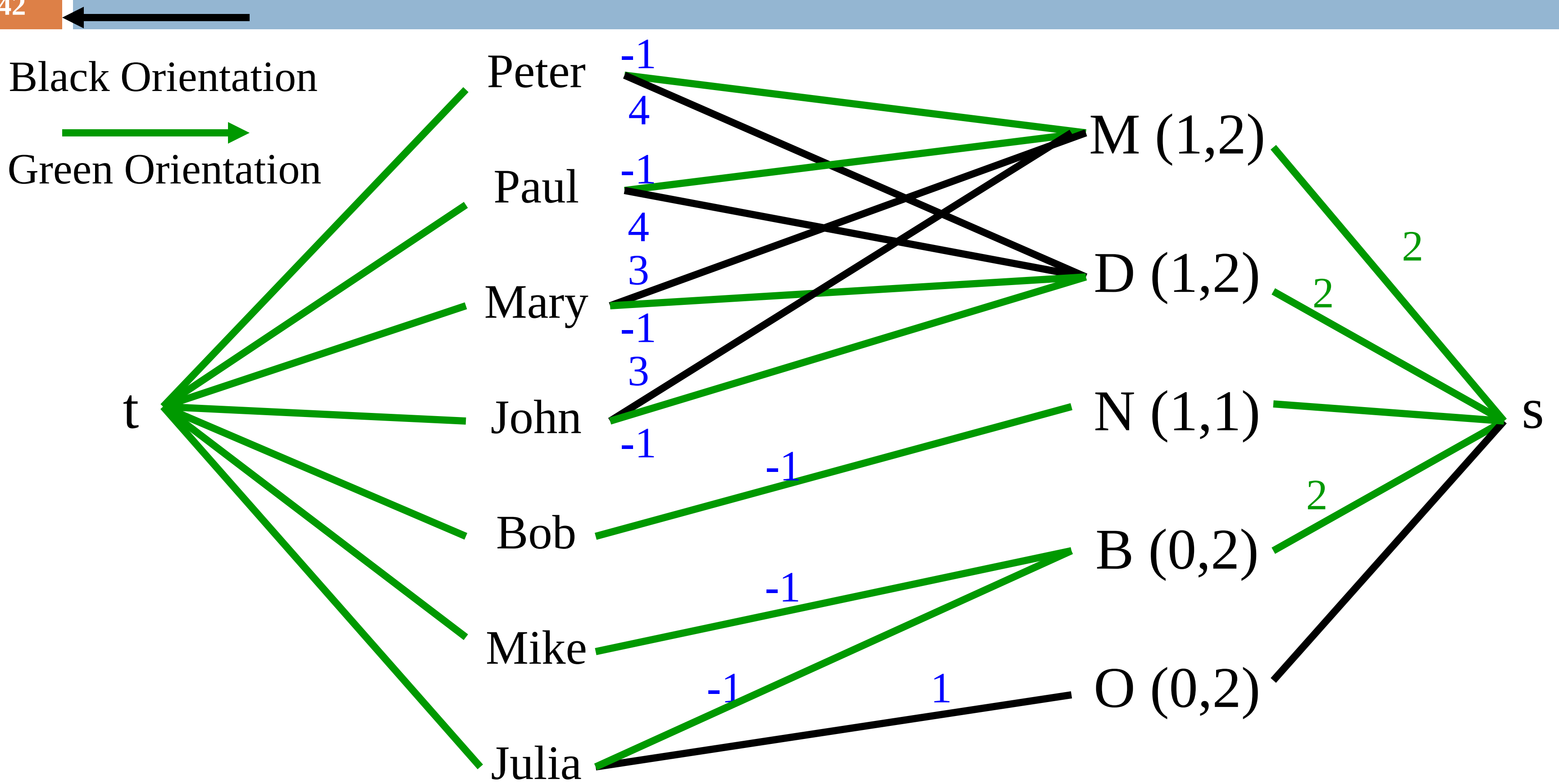


Residual Costs



-1 residual cost = - cost if opposite arc

Residual Costs



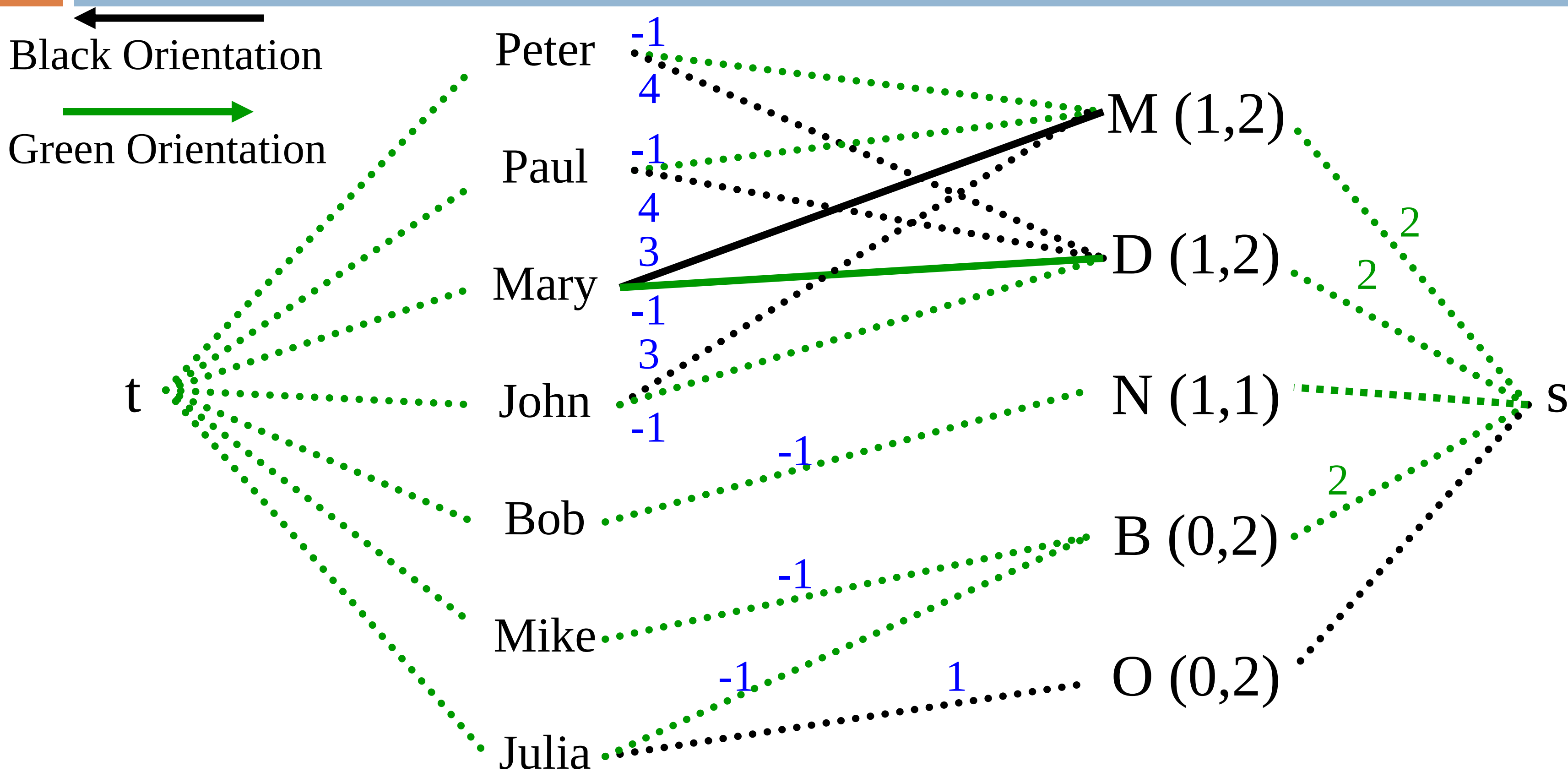
-1 residual cost = - cost if opposite arc

1 residual cost = cost if arc

Shortest path

- d(i,j) = length of the shortest path which does not use (i,j) in the residual graph. The length is the sum of the residual costs of the arc contained in the path.

Residual Costs



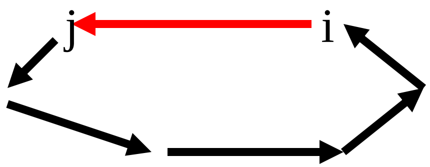
d(M,D)=3 + (-1) = 2

Minimum cost flow

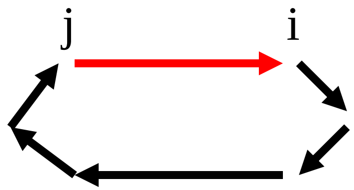
- If the feasible flow is computed by augmenting the flow along shortest paths then the solution is optimal.
- Complexity $O(n\ S(n,m,\chi))$ where χ is the maximum cost value.

Arc consistency

- 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)}



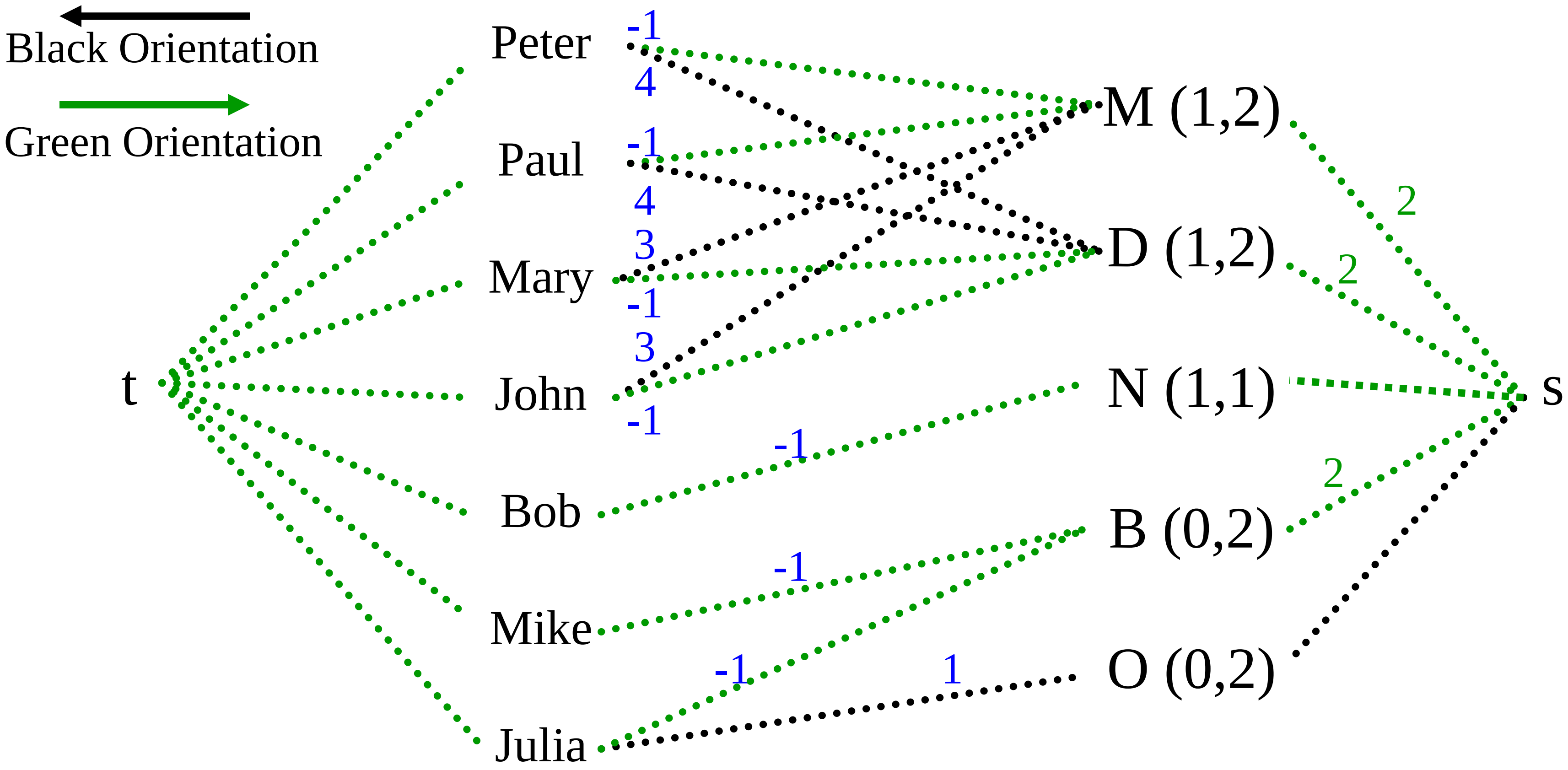
Arc consistency

- Let optcost be the value of the minimum cost feasible flow, and H be the maximum value of the assignments.
- The flow value of an arc (i,j) can be increased if and only if:

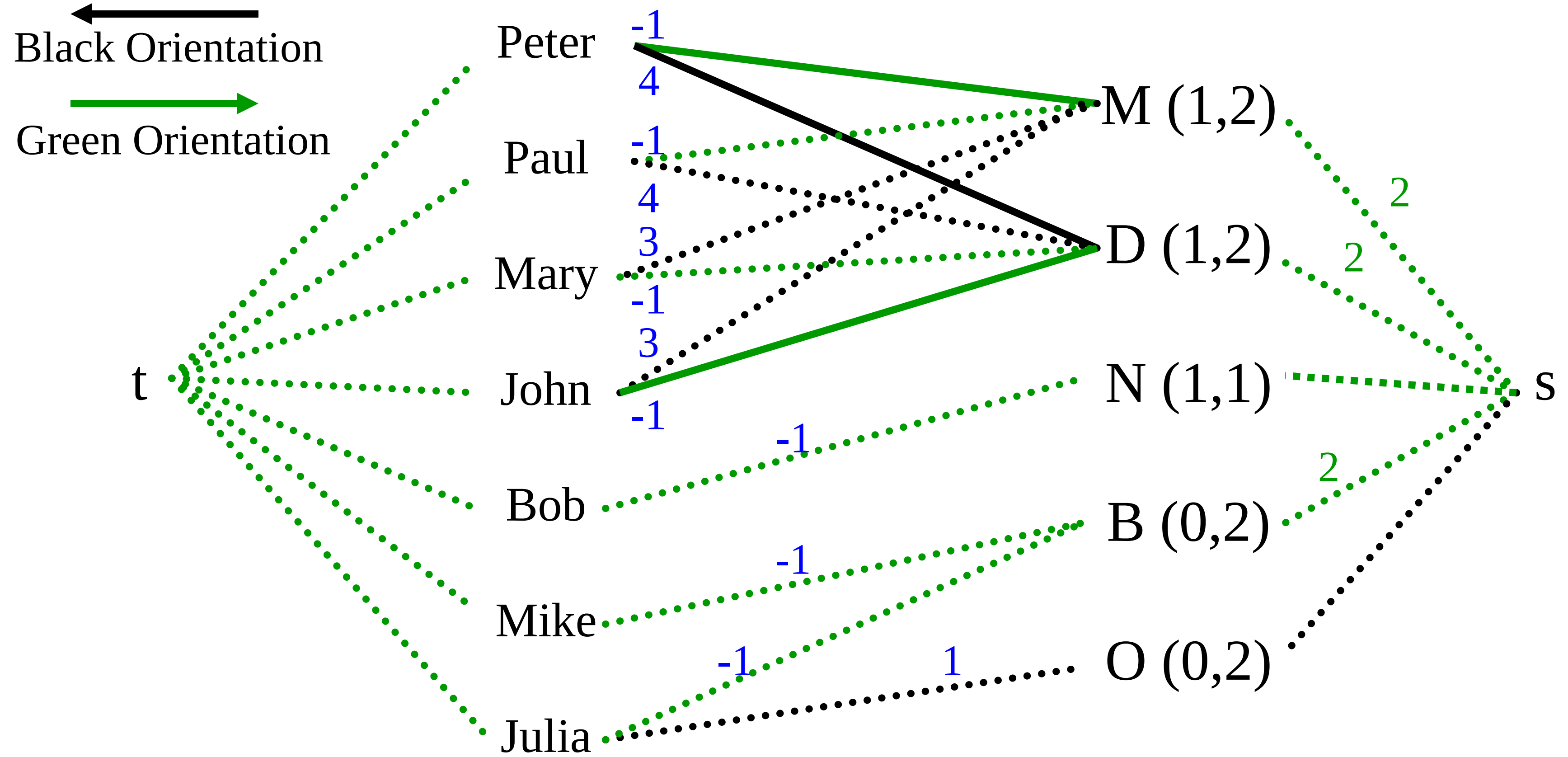
$$rcij + d(j,i) + optcost < H$$

The cost of the directed cycle is computed, that is the cost of rerouting the flow.

Can (M,John) be increased?



Can (M,John) be increased?



rc(M,John) + d(John,M) + optcost = 3 + (-1+4+(-1)) + 7 = 12 > 11: NO

Arc consistency

- Similar reasoning for decreasing the flow value.
- Complexity O(m S(n,m,χ))
- **can be improved!**

AC Improvement

- Problem: shortest paths from j to i cannot contain (j,i).
- How the computations can be grouped, since the graph changes for each computation?

AC Improvement

- Problem: shortest paths from j to i cannot contain (j,i).
- How the computations can be grouped, since the graph changes for each computation?
- **The graph does not change for (0,1) arcs!**

AC Improvement

- Between variables and values there are only (0,1) arcs.
- If we search for increasing the flow value of (i,j) then xij=0 and (j,i) does not exist in R
- If we search for decreasing the flow value of (i,j) then xij=1 and (i,j) does not exist in R

AC Improvement

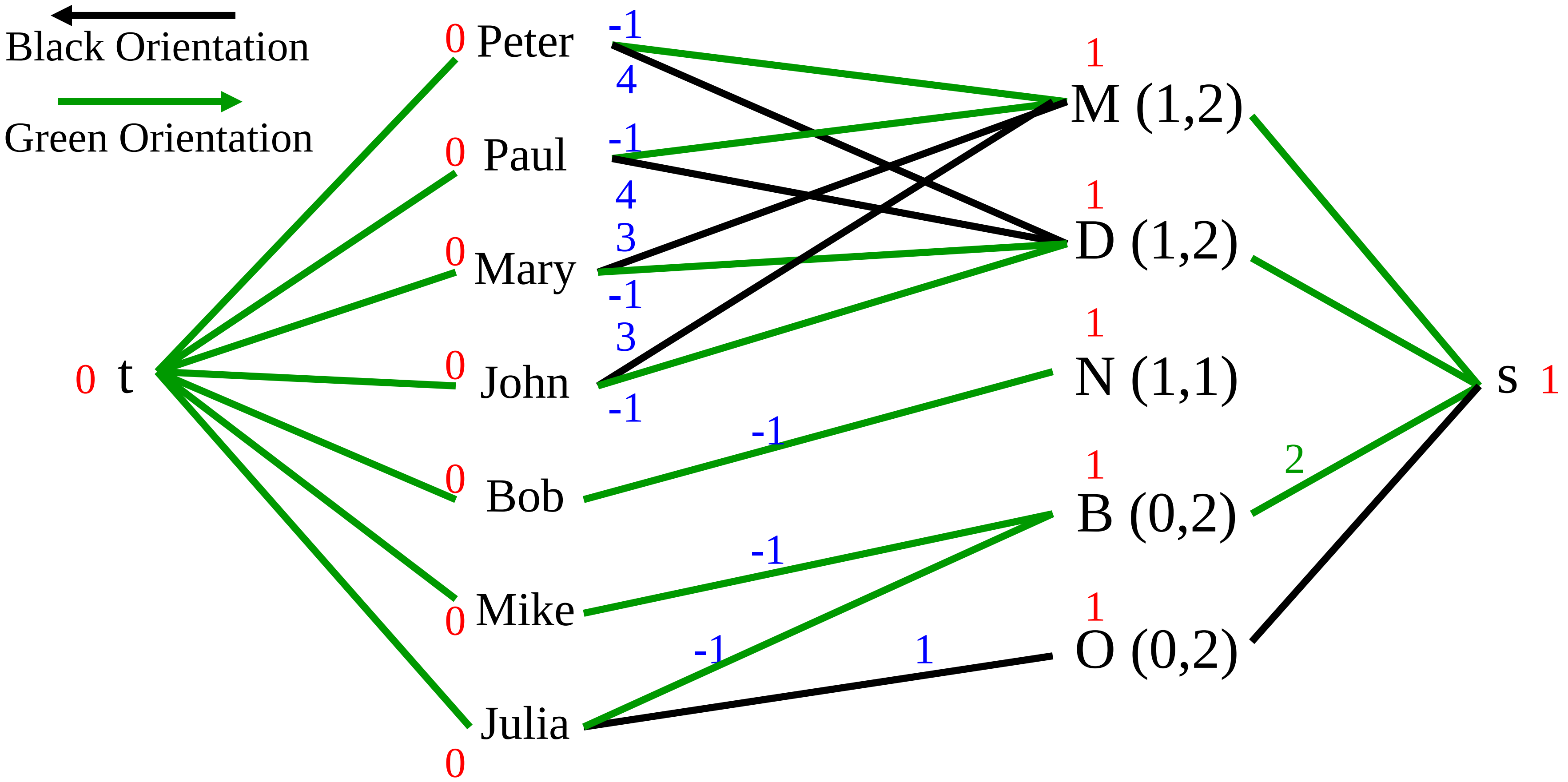
- The computation can be grouped:
  For each variable, the shortest paths to all the values are computed
- Complexity $O(n\ S(n,m,\chi))$.
- Drawback: “repeated” algorithm (n times something…)
- Can be improved by searching for shortest path from the values that are assigned.
- Reduced costs can be used instead of residual cost to have only nonnegative costs and to improve the search for shortest paths.

Comparison with MIP

- The filtering algorithm is stronger than the propagators based on the reduced costs.

- Let d be a distance. We have
  - $d(v) \leq d(u) + cost(u,v)$
  - $0 \leq d(u) + cost(u,v) – d(v)$
  - $ReducedCost\ (u,v) = d(u) + cost(u,v) – d(v)$
  - With ¶ = -d: ReducedCost (u,v) = ¶(v) + cost(u,v) -¶(u)

Node Potential



-1 residual cost = - cost if opposite arc

1 residual cost = cost if arc

1 node potential = - distance from t

Reduced Costs

Black Orientation

Green Orientation

0 Peter

-1

1

M (1,2)

Reduced Cost (Peter,M) = residual cost - π(Peter) + π(M)

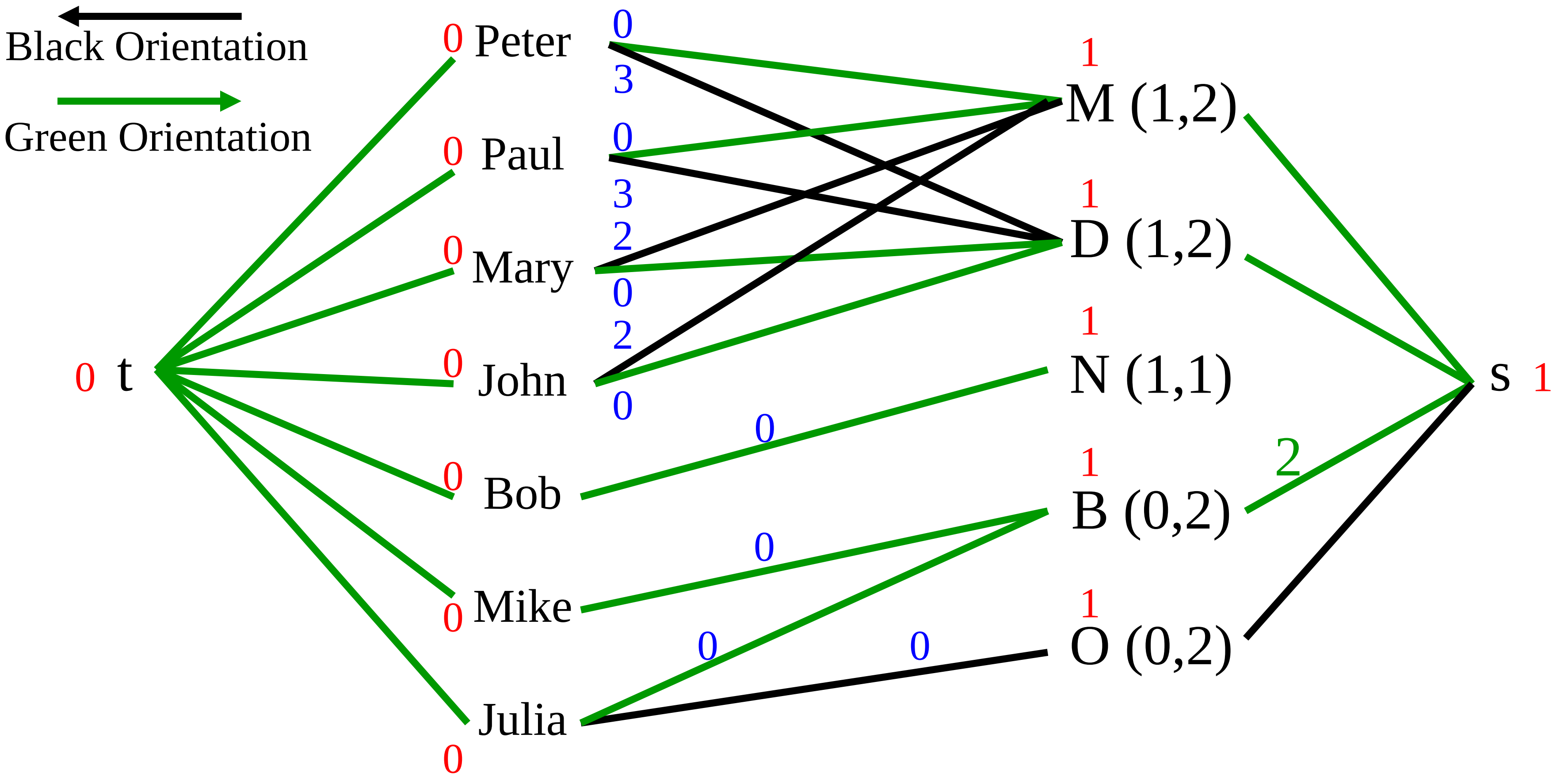
= -1 -0 +1 = 0

-1 residual cost = - cost if opposite arc

1 residual cost = cost if arc

1 **node potential = - distance from t**

Reduced Costs

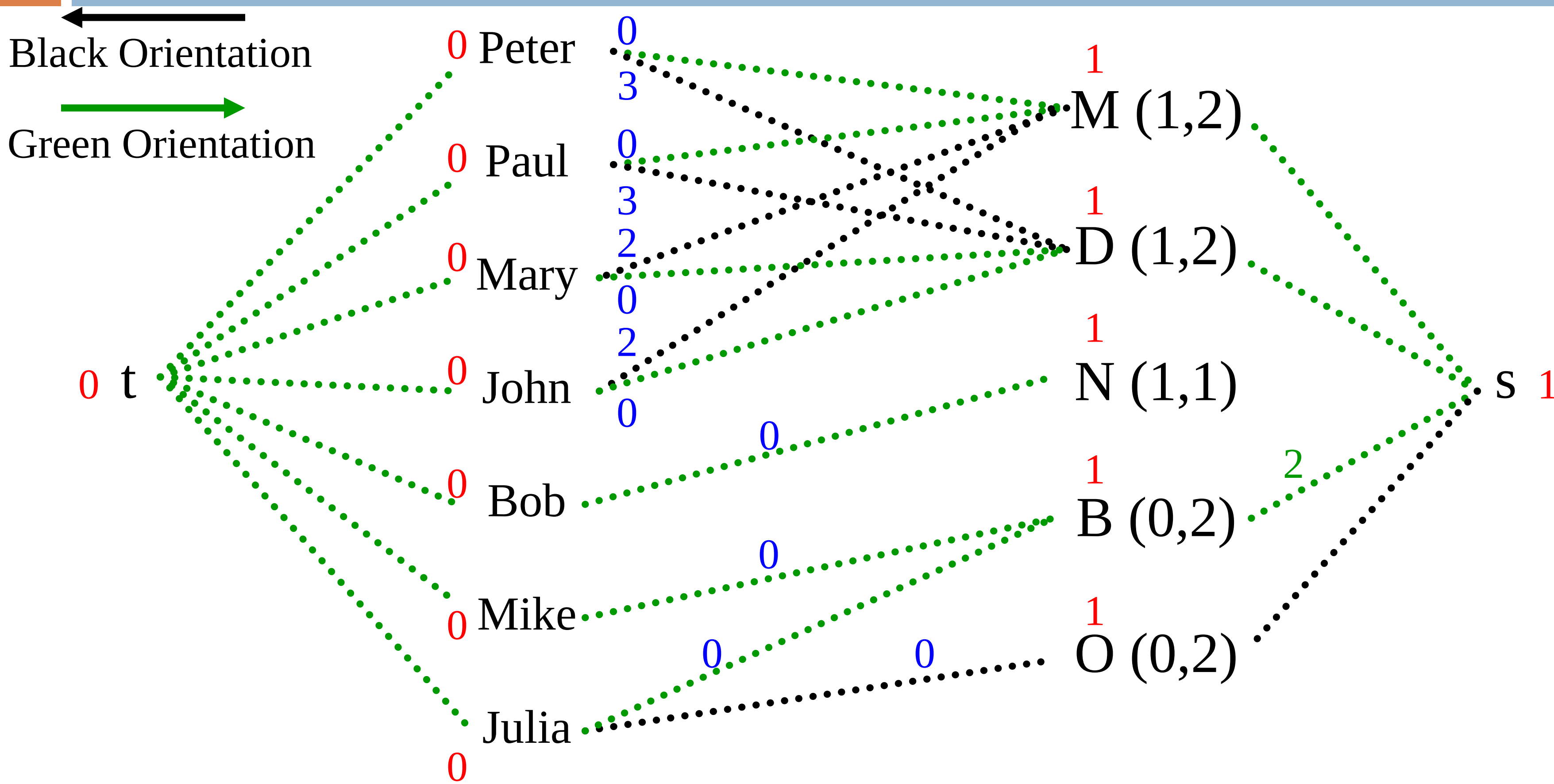


-1 reduced cost 1 node potential = - distance from t

Reduced Costs

- Intuitive Idea:
  Given (i,j) an arc:
  if a unit a flow is sending from i to j then
  cost flow is increased **at least** by rcij.

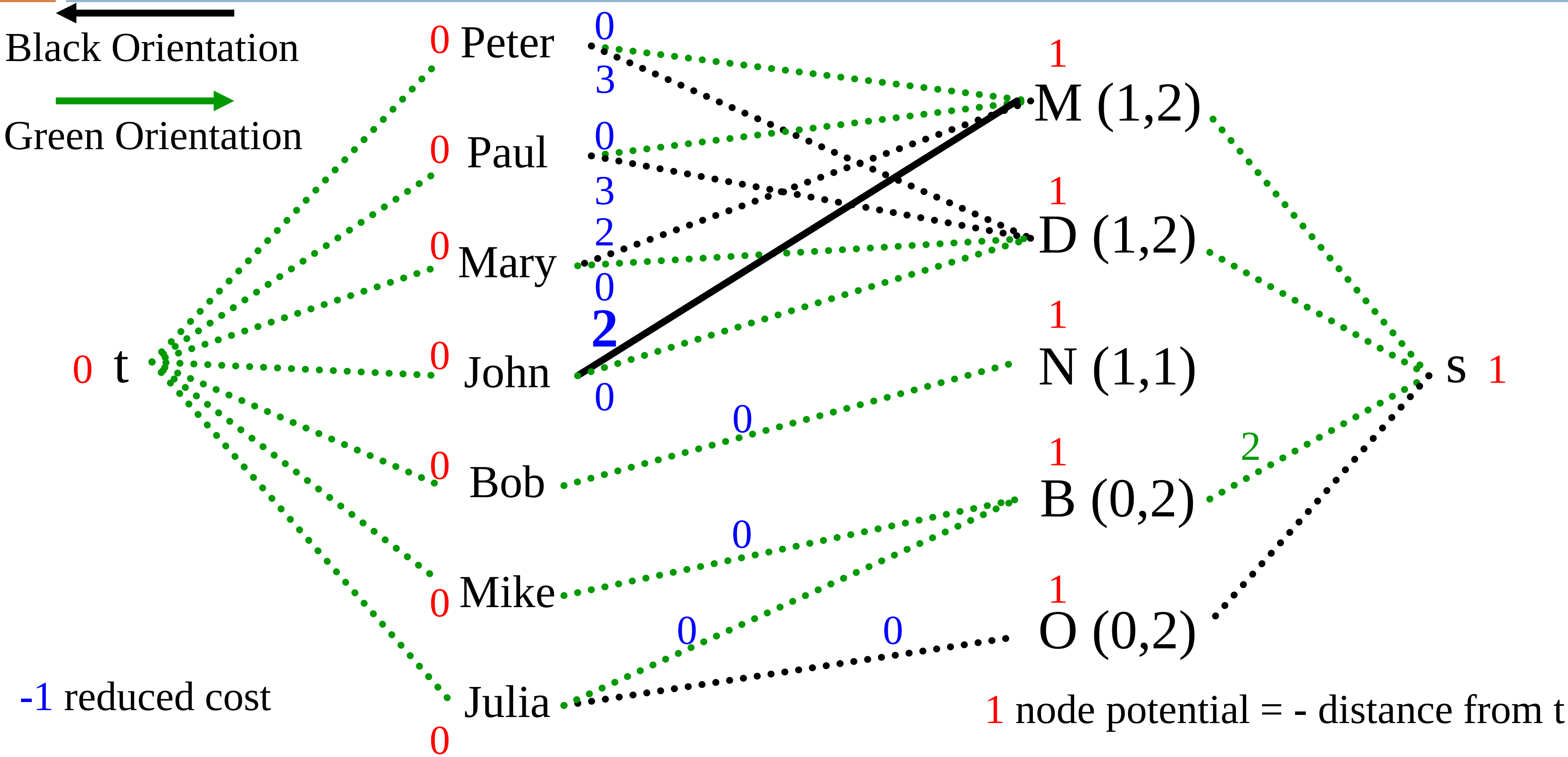
Can (M,John) be increased?



-1 reduced cost

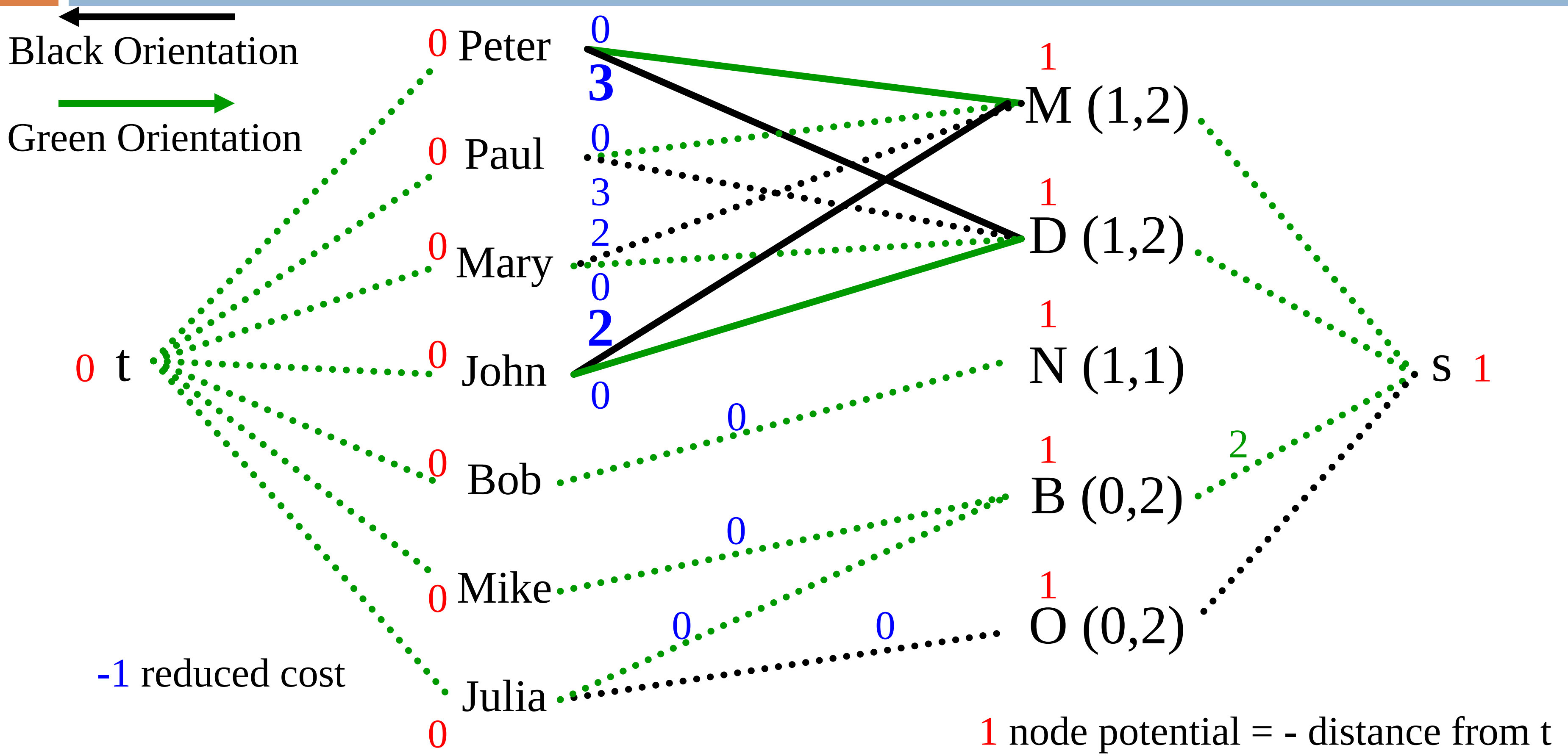
1 node potential = - distance from t

Can (M,John) be increased?



Reduced cost : it will cost at least 2

Can (M,John) be increased?



Reduced cost : it will cost at least 2

Exact computation : it will cost 2+3=5

Flow based constraints: filtering

- Filtering of 0-1 arcs
- Introduction of card variables
- Filtering of 0-1 arcs with costs
- **Identification of constant flow value arcs**
- Particular case of costs on cardinality variables only
- Convex graphs (graph having the 0-1 property)

Constant flow value

- If an arc (u,v) is not a 0-1 arc then the arcs (u,v) and (v,u) can both belong to the residual graph
- (u,v) and (v,u) define a strongly connected component!
- We can identify these arcs that carry a constant flow value by a simple algorithm (see the cardinality matrix paper)

Constant flow value

- Let f be any feasible flow. The flow value in an arc (u,v) is constant if and only if
  - (u,v) and (v,u) do not belong to R(f)
  - R(f) contain (u,v) or (v,u) but not both and u and v belong to 2 different strongly connected components
  - (u,v) and (v,u) belong to R(f) and {u,v} is a bridge of ud(scc(R(f),u)) where ud(scc(R(f),u)) is the undirected version of the strongly connected component of R(f) containing u.

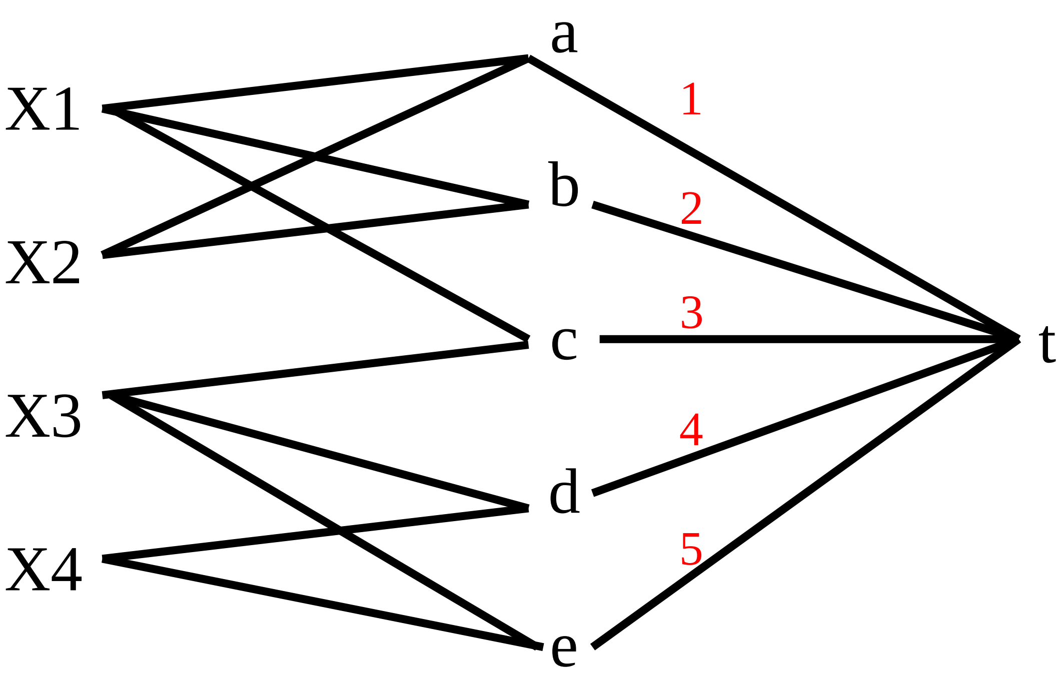
Flow based constraints: filtering

- Filtering of 0-1 arcs
- Introduction of card variables
- Filtering of 0-1 arcs with costs
- Identification of constant flow value arcs
- **Particular case of costs on cardinality variables only**
- Convex graphs (graph having the 0-1 property)

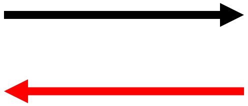
Costs only on cardinality variables

- There is a cost only for arcs going from nodes to the sink.
- We can speed up the min cost flow computation and the filtering algorithm (« On global warming: Flow-based Soft Global Constraints », W-J Van Hoeve, G. Pesant and L-M Rousseau, Constraints, 2006)
- If we have ordered the costs, then we can compute the min cost flow in O(nm) and the filtering in O(m) for a weighted gcc.

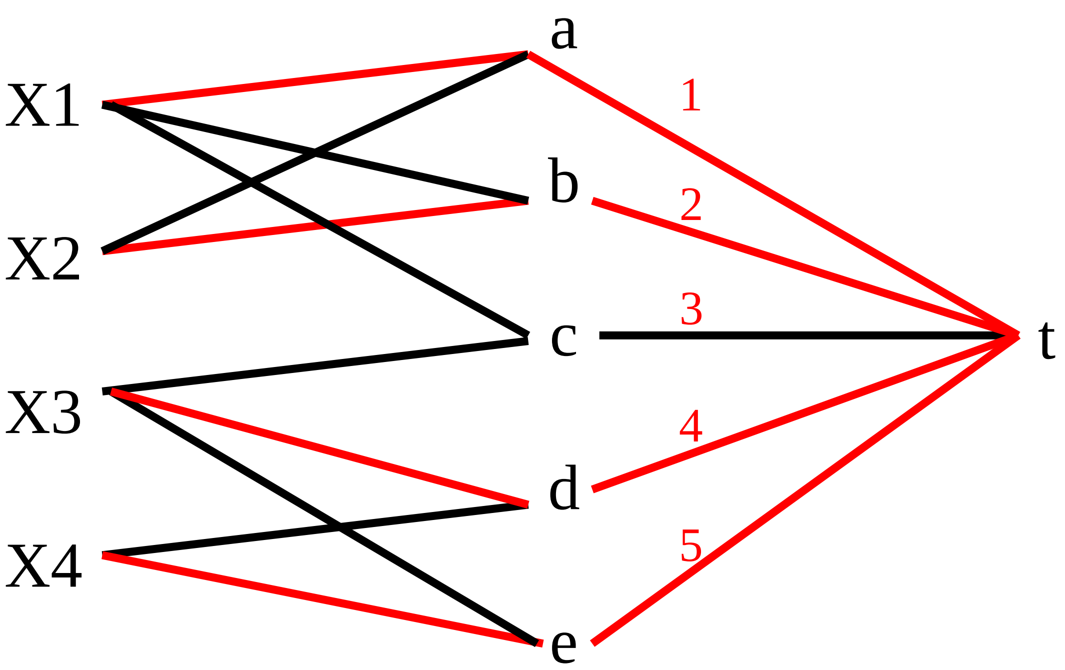
Costs only on arcs (a,t)



Costs only on arcs (a,t)



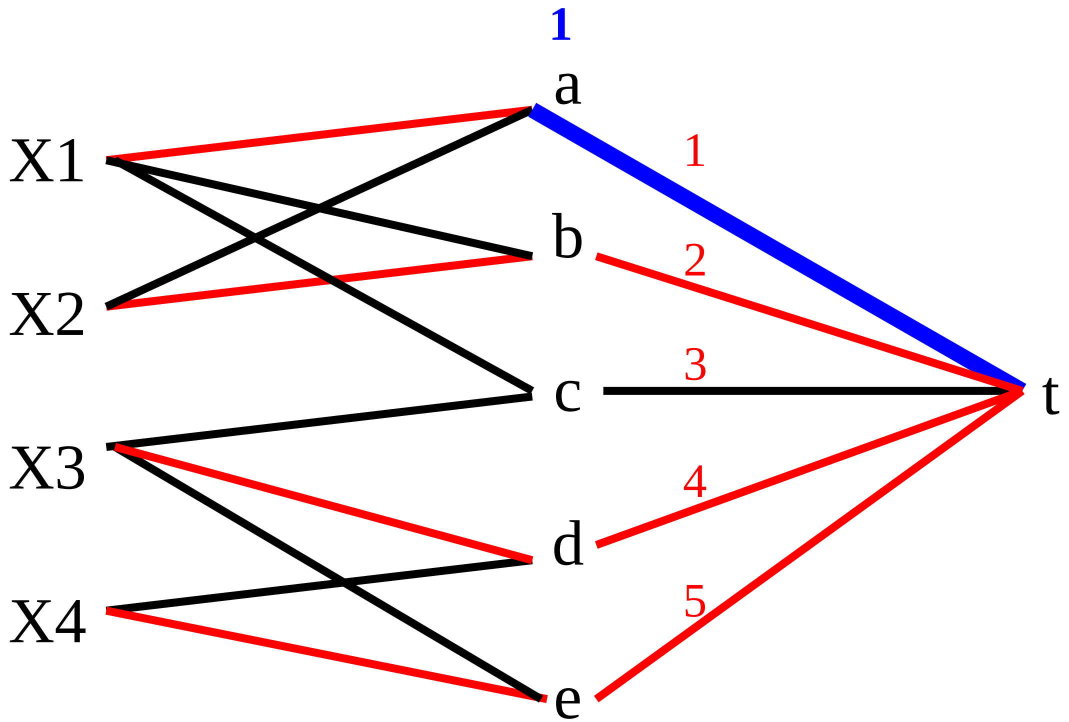
Orientation



From t to values

We take the values according to the min cost ordering

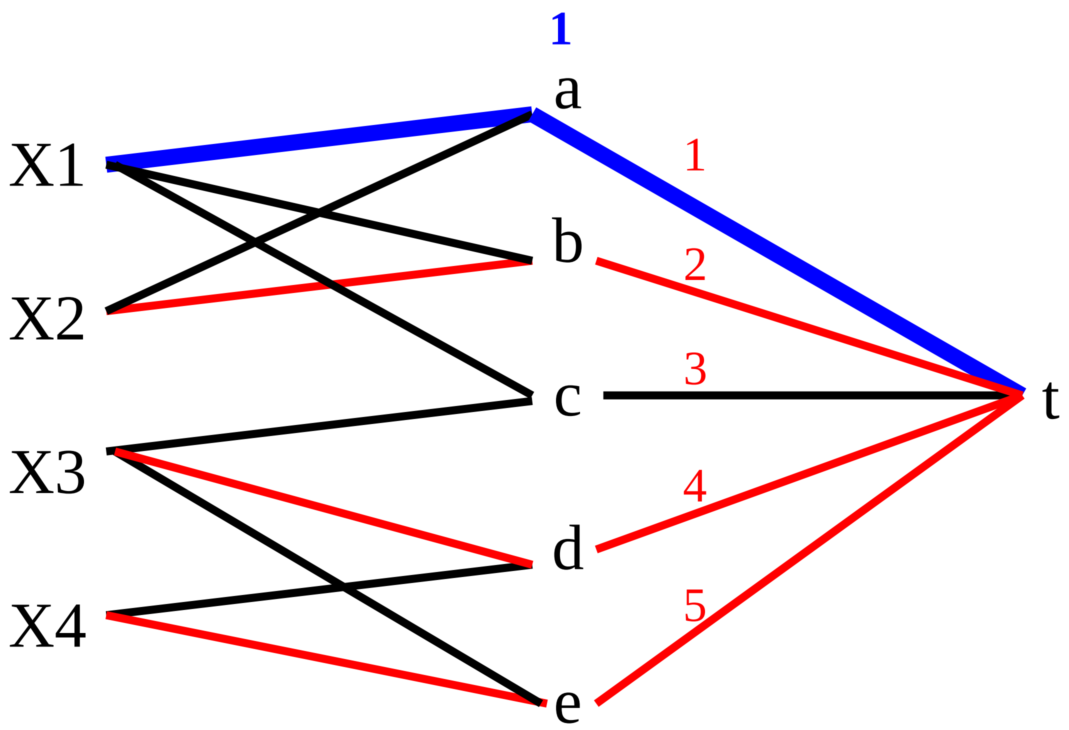
Costs only on arcs (a,t)



From t to values

We take the values according to the min cost ordering

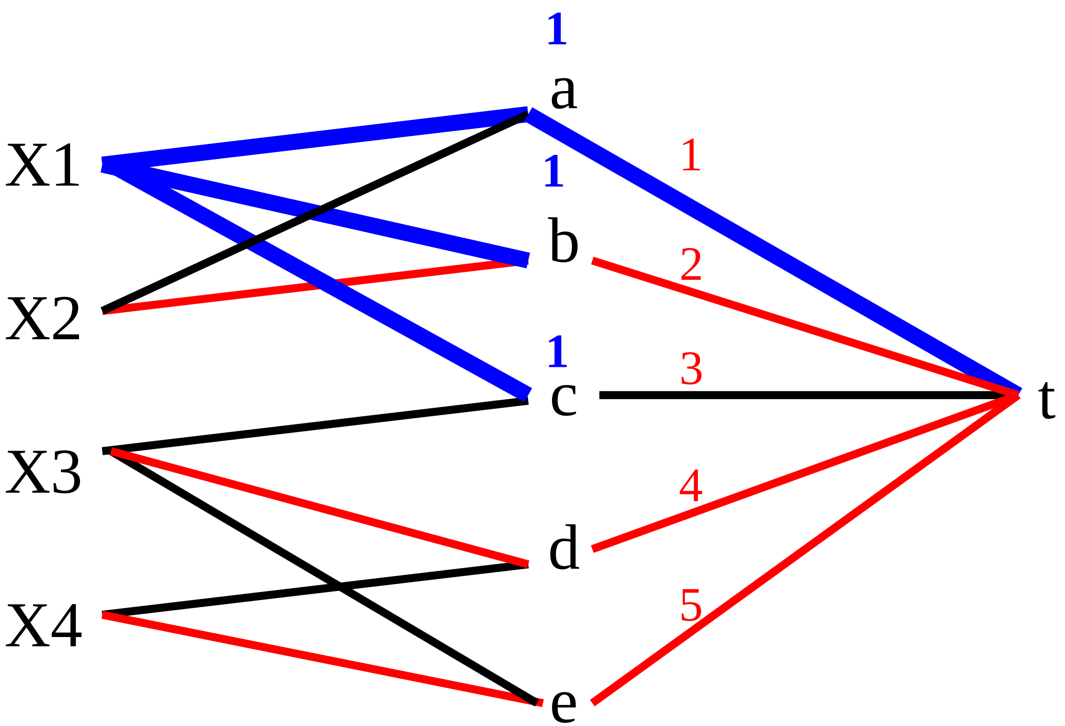
Costs only on arcs (a,t)



From t to values

We take the values according to the min cost ordering

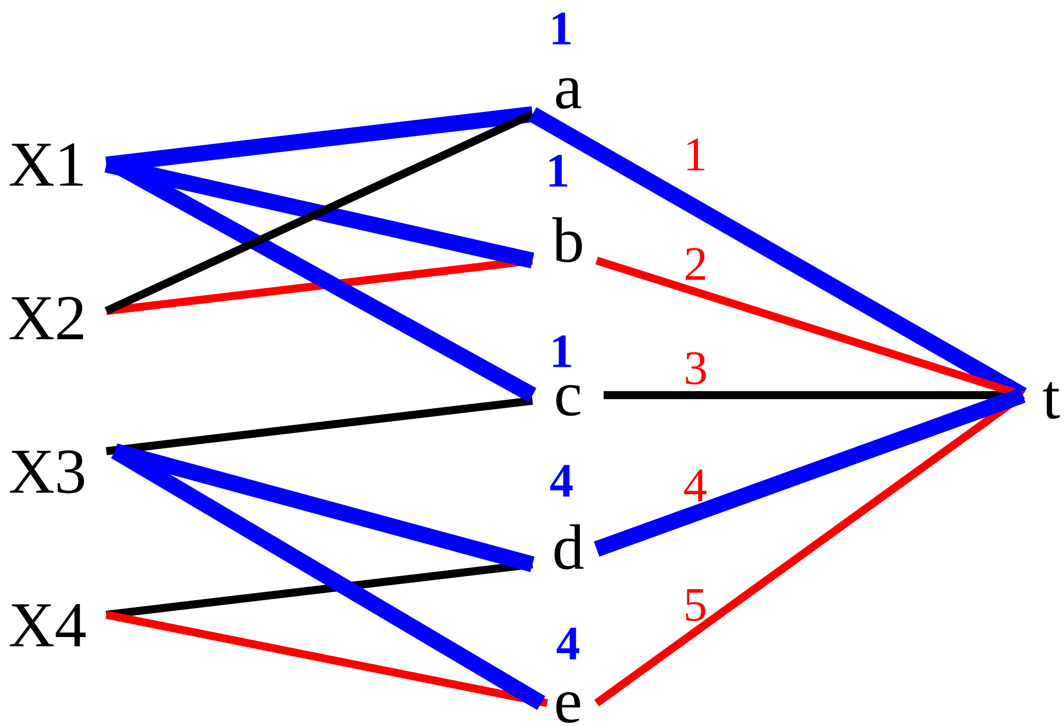
Costs only on arcs (a,t)



From t to values

We take the values according to the min cost ordering

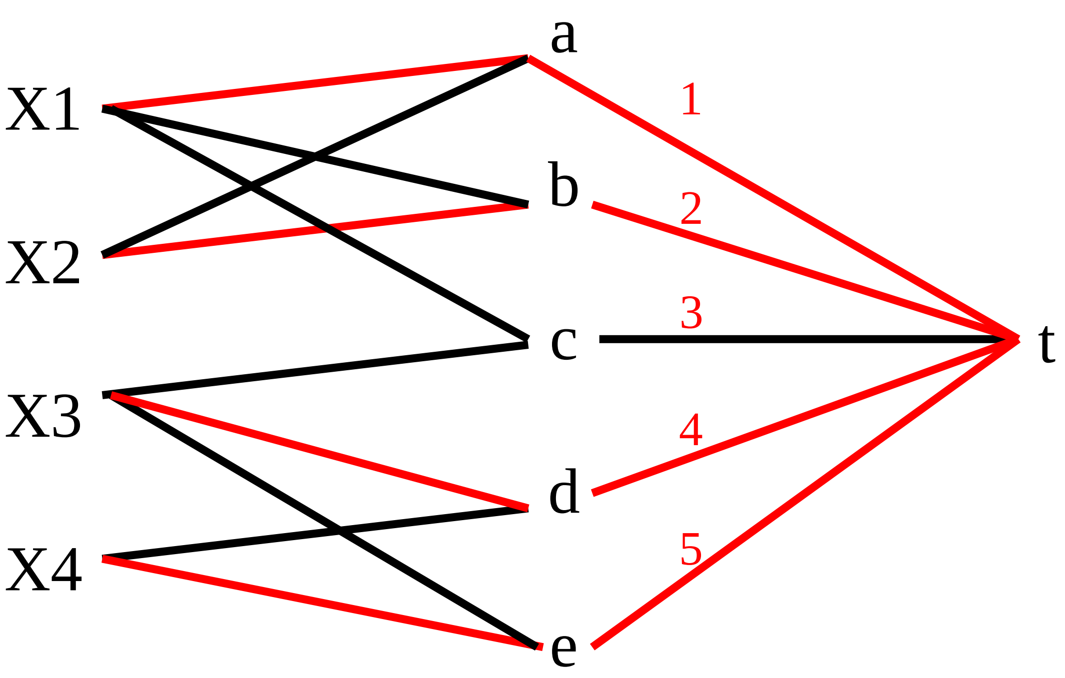
Costs only on arcs (a,t)



From t to values

We take the values according to the min cost ordering

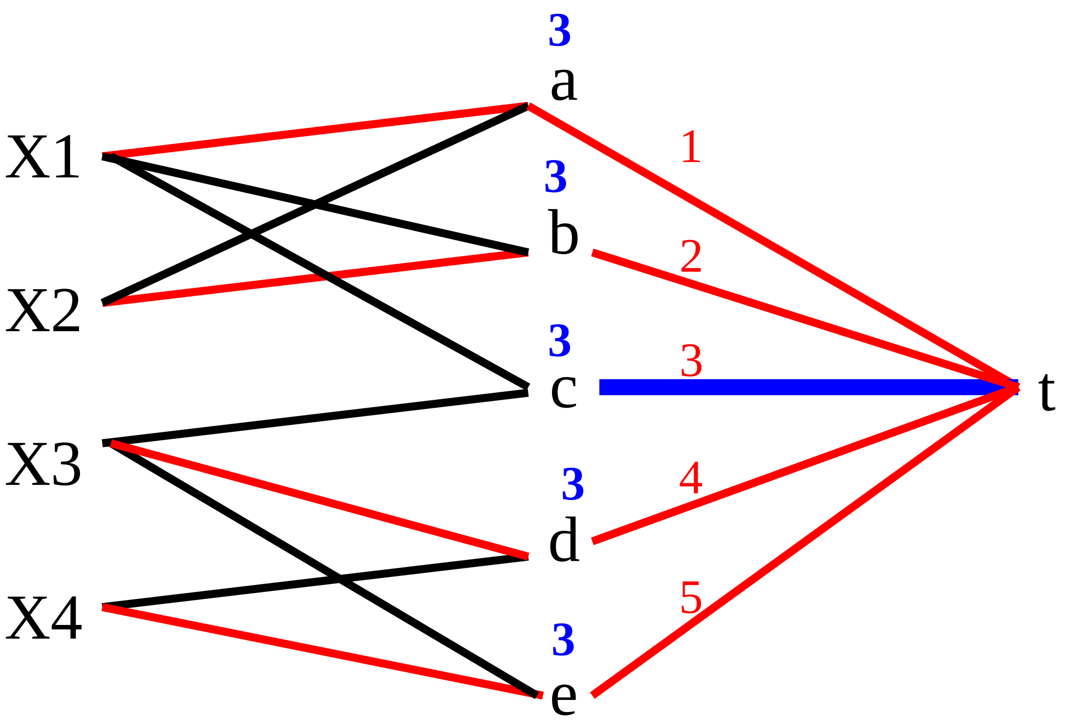
Costs only on arcs (a,t)



From values to t. We consider the transposed graph

We take the values according to the min cost ordering

Costs only on arcs (a,t)



From values to t. We consider the transposed graph
We take the values according to the min cost ordering

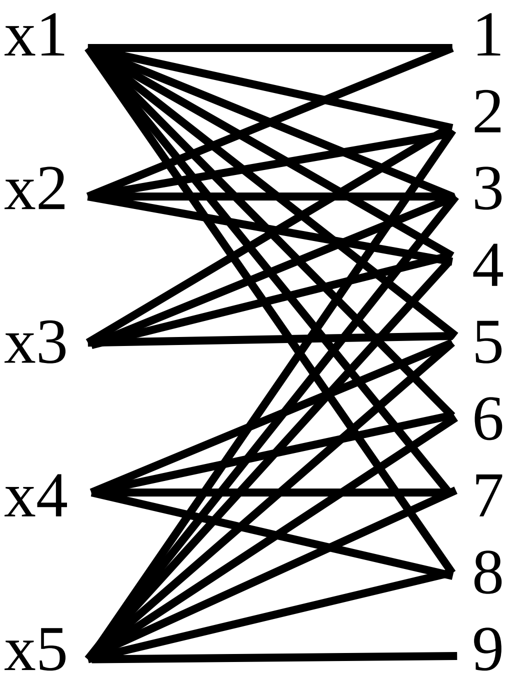
Flow based constraints: filtering

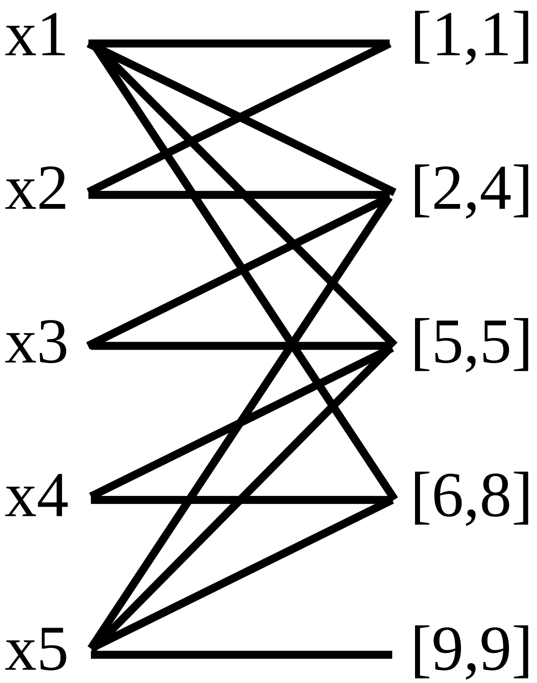
- Filtering of 0-1 arcs
- Introduction of card variables
- Filtering of 0-1 arcs with costs
- Identification of constant flow value arcs
- Particular case of costs on cardinality variables only
- **Convex graphs (graph having the 0-1 property)**

Flows in convex graph

- Convex graph (or graph having the 0-1 property)
  - For the value graph. For each variable x: if values u and v with v > u belong to D(x) then each value in the range [u,v] belongs to D(x)
- The value graph of range variables is a convex graph
- 0-1 property because we can rearrange the column of the adjacency matrix such that for each row we have consecutive 0s – consecutive 1s – consecutive 0s
- Flow computations can be speed-up for these graphs
- We can always consider that we have d=n by considering intervals of ranges instead of discrete values

Convex graph O(d) = O(n)





There are at most 2n intervals

- Gcc:
  - C-G Quimper, A. Golynski, A. López-Ortiz, P. van Beek, **An Efficient Bounds Consistency Algorithm for the Global Cardinality Constraint.** Constraints 2005; CP'03

- Alldiff:
  - A. López-Ortiz, C-G Quimper, J. Tromp, P. van Beek, **A Fast and Simple Algorithm for Bounds Consistency of the AllDifferent Constraint.** IJCAI'03
  - K. Mehlhorn and S. Thiel. Faster algorithms for bound-consistency of the sortedness and alldifferent constraint. In CP'00
  - J.-F. Puget. A fast algorithm for the bound consistency of alldiff constraints. AAAI-98.

Flows in convex graph

- If the values are sorted then consistency checking is close to O(sort + n) where sort is the time to sort the intervals of values, for the alldiff and the gcc
- The complexity can be reduced for the following reason (a simpler version than the one of Melhlorn and Thiel)
- Consider the alldiff constraint. Computation are based on a DFS

DFS and convex graph

- Visit(x)

```
previsit(x)
for each a in N(x) do
      if (a) is not marked
      then   mark(a)
             visit(match(a))
postvisit(x)
```

- Consider the set UM of unmarked value, then we can rewrite the algorithm

DFS and convex graph

- Visit(x)
  previsit(x)
  for each a in N(x) $\cap$ UM
      mark(a)
      visit(match(a))
  postvisit(x)

- N(x) is a range if UM is a range then picking an element in N(x) $\cap$ UM costs O(1) and the algorithm O(n)
- With the Lipsky-Preparata 's algorithm we can prove that node are opened such that UM is a range!

Flow based constraints: filtering

- Filtering of 0-1 arcs
  - Flot in O(nm), Filtering O(m)
- Introduction of card variables
  - Flot in O(nm), Filtering in O(m) (not AC)
- Filtering of 0-1 arcs with costs
  - Flot in $O(n\ S(n,m,\chi))$, Filtering in $O(n\ S(n,m,\chi))$
- Identification of constant flow value arcs
  - O(m)
- Particular case of costs on cardinality variables only
  - Flot in O(nm), Filtering O(m)
- Convex graph
  - Flot and Filtering almost in O(n)
  - Weighted version could be improved, costs only on card var: $O(n^2)$

Plan

- A simple solution: Flow based constraints
  - Definition
  - Filtering algorithms
  - **Their incredible modeling power**
- What is missing?
- What are the other techniques we could use in the future ?
- A wish and a question

The modeling power

- The flow algorithm is one of the strongest polynomial algorithm
- Therefore, it may be used to solve a lot of constraints, because it is more convenient to have polynomial filtering algorithms

- The counting constraints (among, gcc, alldiff,…) can be represented by flow based constraints

Flows in some other parts of CP

- Graph/Subgraph isomorphism
- Symmetries (permutation, automorphism)
- Ordering (sort constraint of A. Colmerauer and J. Zhou)
- …

The modeling power

- Flow based constraints are used for any category of global constraints
- Everyday we discover flow based constraints
- Some flow based constraints are really complex

Global Constraints Collection

- 5 categories of global constraints
  - Classical (alldiff, gcc, diff-n ...)
  - Weighted (cost gcc, knapsack, bin packing ...)
  - Soft constraints
  - Constraints on meta variables (set variables, graph variable ...)
  - Open constraints

Flow and Global Soft Constraint

- Global soft constraints introduced in T. Petit, J-C. Régin, C. Bessière, **Specific Filtering Algorithms for Over-Constrained Problems**. CP'01
- Global Soft Constraint = Global constraint + violation cost
- Several Global Soft Constraint exist:
  - Alldiff
  - Distribute
  - Sequence …
- Several general definition of violation cost exist
- Same problematic in Local Search: need to define a distance to a solution for a constraint

Variable based violation cost

- How many variables must be removed to satisfy the constraint?
- Alldiff({x1,x2,x3,x4,x5})
  (a,a,b,b, c) cost = 2
  (a,a,a,b,b) cost = 3
  (a,a,a,a,b) cost = 3

- We just need to search for a maximum flow for the alldiff and the gcc. We accept the flow value to be less than n.
- Note that if we ask for having at least q different value then if we have a flow whose value is q+1 then the constraint is arc consistent

Primal graph based partition cost

- For a global constraint corresponding to a conjunction of constraints. Number of the constraints in the conjunction that are violated
- Alldiff({x1,x2,x3,x4,x5})
  (a,a,a,b,b) cost = triangle(a,a,a) + pair (b,b)
  = 3 + 1 =4
  (a,a,a,a,b) cost = quadrangle (a,a,a,a)
  = 6

Soft Alldiff

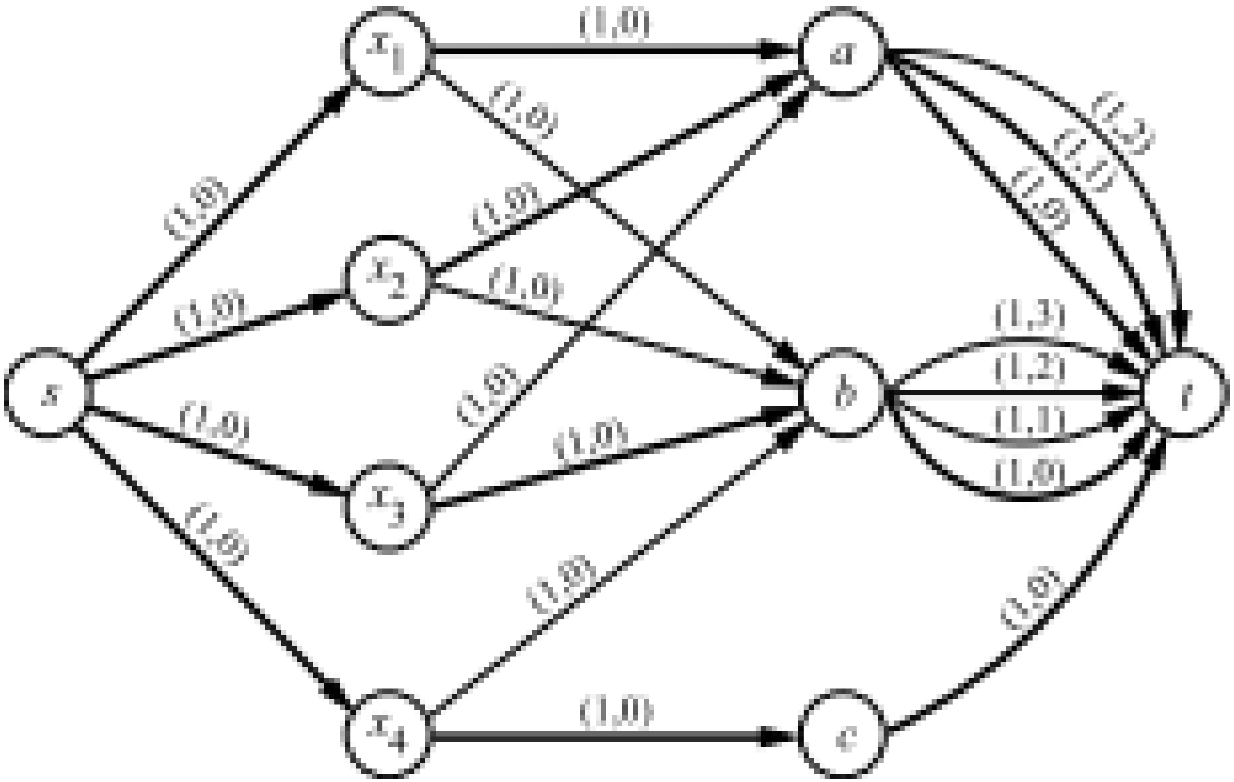
- Violation cost = primal based partition based cost
- A nice model has been given by WJ van Hoeve, **A Hyper-Arc Consistency Algorithm for the Soft Alldifferent Constraint.** See also WJ van Hoeve, G Pesant, LM Rousseau, **On global warming: Flow-based soft global constraints.** J. Heuristics 2006

- Willem modeled the problem with a convex cost flow.
  - usually when we have k units of flow traversing an arc whose cost is c, then it costs ck.
  - the algorithm remains the same if the cost function (here cost(k)=ck) is a convex function

Soft Alldiff

- Flow with cost (van Hoeve) for the arcs from value to sink
  - If flow value = 0 or 1 cost = 0
  - If flow value = 2 (2 var have the same value) cost =1
  - If flow value = 3, then cost = 3
  - If flow value = 4, then cost = 6
- Possible representation: for a value v, use several arcs from v to u
  - each with max capacity = 1
  - One with cost 0
  - One with cost 1
  - One with cost 2
  - One with cost 3 …
- Now
  - For a flow value = 2 (2 arcs), the 2 min cost values are 0 and 1, min cost=0+1=1
  - For a flow value = 3 (3 arcs), the 3 min cost values are 0, 1 and 2 , min cost=0+1+2=3

Soft Alldiff



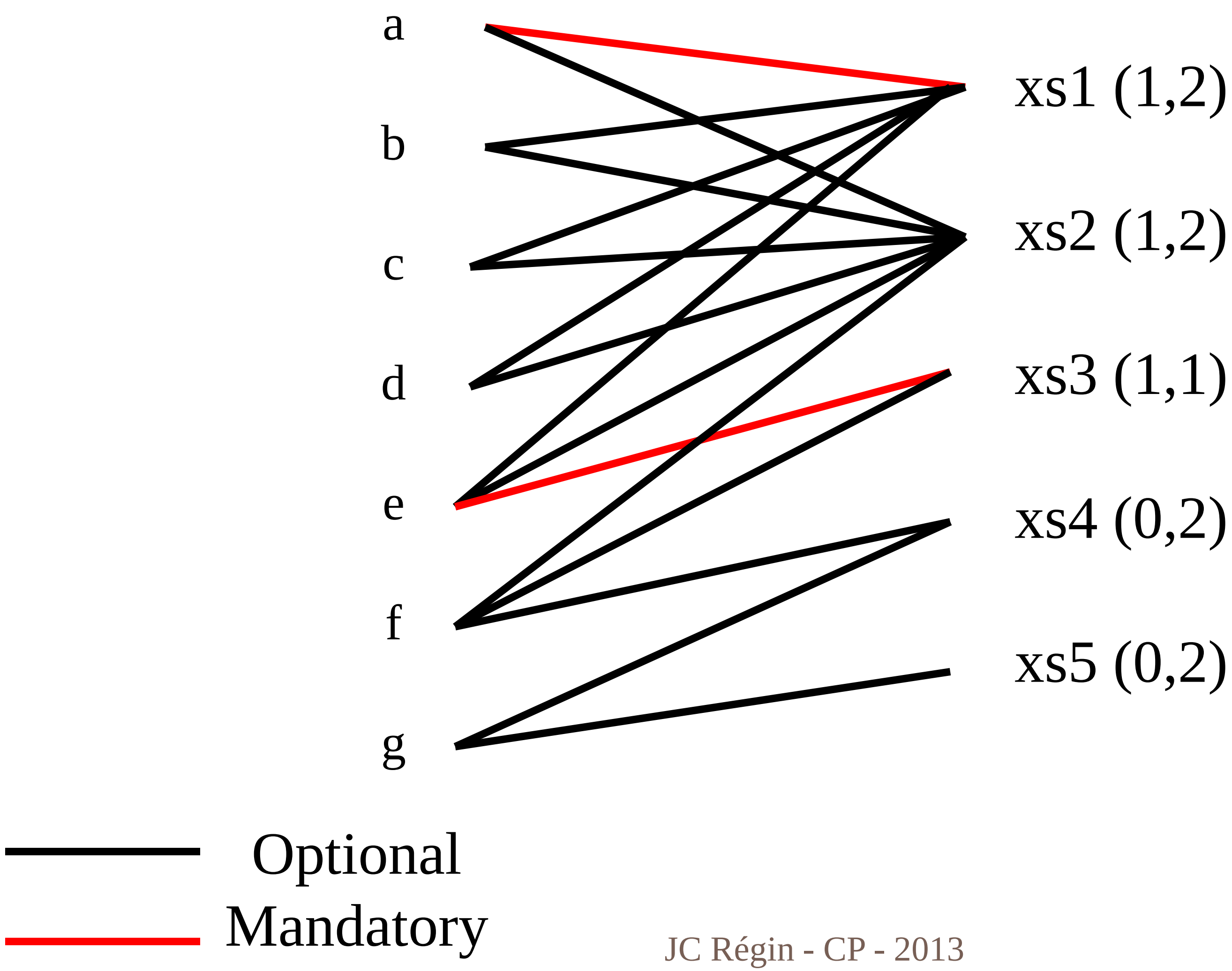
- For the GCC, there is nice algorithm in A. Zanarini, M. Milano, G. Pesant, **Improved Algorithm for the Soft Global Cardinality Constraint.** CPAIOR'06:

Flow and constraints on meta-variables

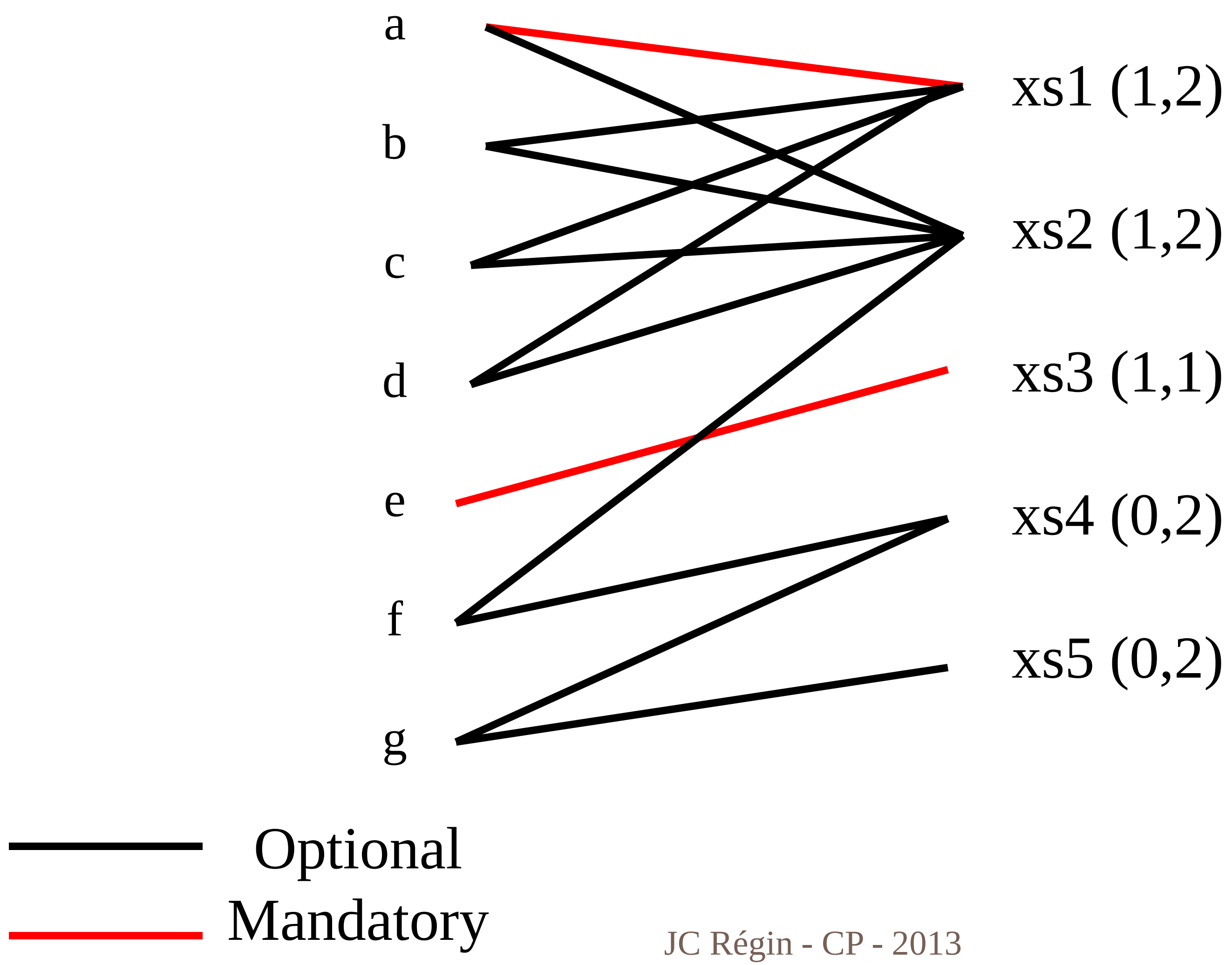
- Flow are also useful for meta-variables
  - SetVar
  - Path problems

- AllnullIntersect for Set Variables in ILOG Solver. See also in C. Bessière, E. Hebrard, B. Hnich, T. Walsh, **Disjoint, Partition and Intersection Constraints for Set and Multiset Variables.** CP'04

AllNullIntersect



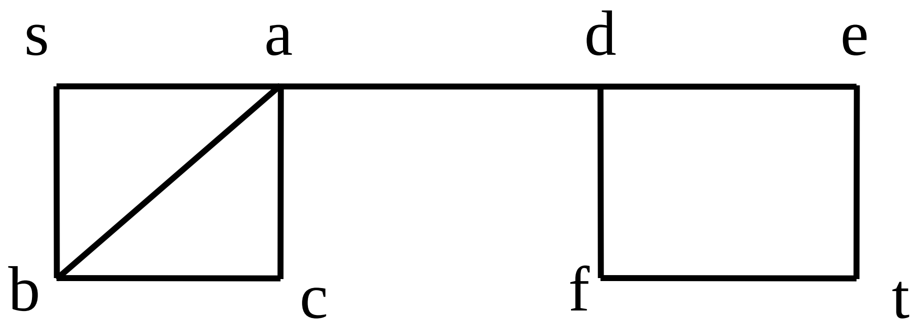
AllNullIntersect



Path representation in CP

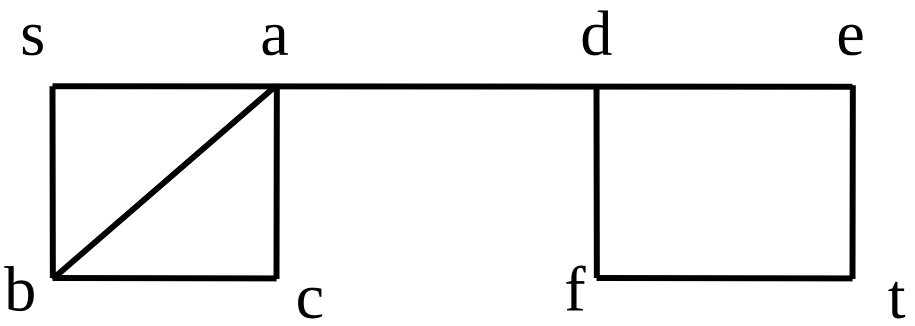
- “Classical” model:
  Graph represented by the nodes:
  One variable per node
  Value = possible neighboor
- Path from s to t: alldiff on nodes.

Path representation in CP



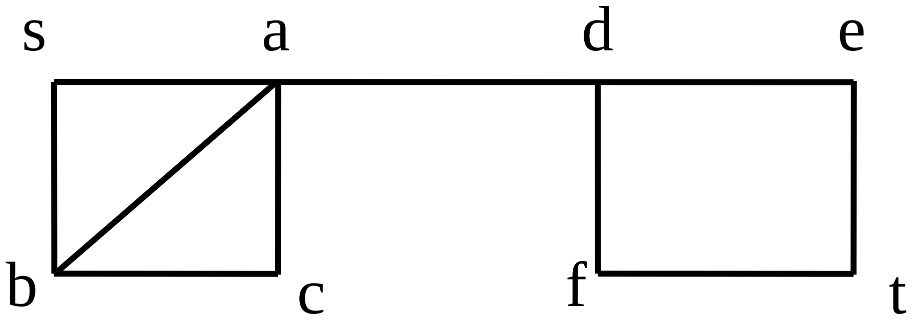
D(s)={a,b}, D(a)={s,b,c,d}, D(b)={s,a,c}, D(c)={a,b}
D(d)={a,e,f}, D(e)={d,t}, D(f)={d,t}, D(t)={s}

Path representation in CP



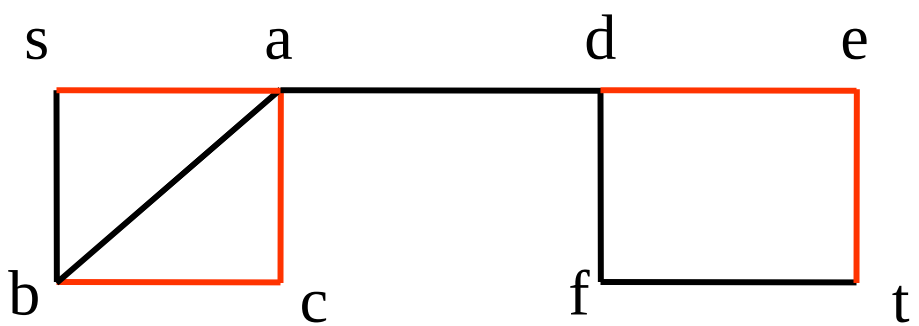
D(s)={a,b}, D(a)={s,b,c,d}, D(b)={s,a,c}, D(c)={a,b}
D(d)={a,e,f}, D(e)={d,t}, D(f)={d,t}, D(t)={s}

Path representation in CP



Problem if some variables do not belong to the path:
What is the value assigned to these variables?

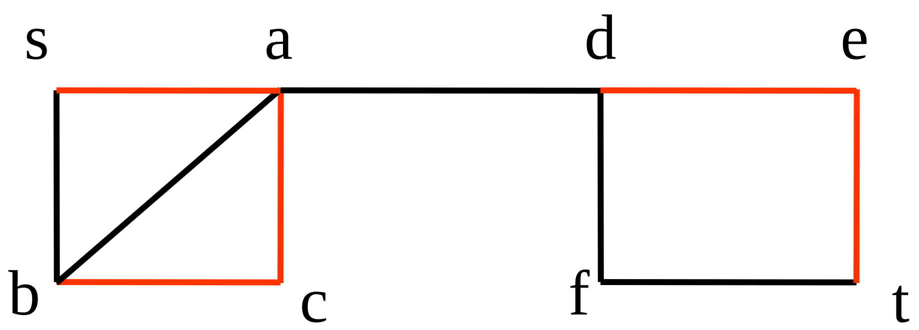
Path representation in CP



A dummy value is added to each domain: BAD IDEA

D(s)={a}, D(a)={c}, D(c)={b}, D(b)={dummyb}, D(d)={e}, D(e)={t}, D(f)={dummyf}, D(t)={s}

Path representation in CP



Loops are allowed (var links to itself): GOOD IDEA

D(s)={a}, D(a)={c}, D(c)={b}, D(b)={b},

not possible: b has been already taken by c

Path representation in CP

- “Classical” model:
  - One var per node
  - Alldiff constraint: cost for the matching: O(m) per modification

Open Global Constraints

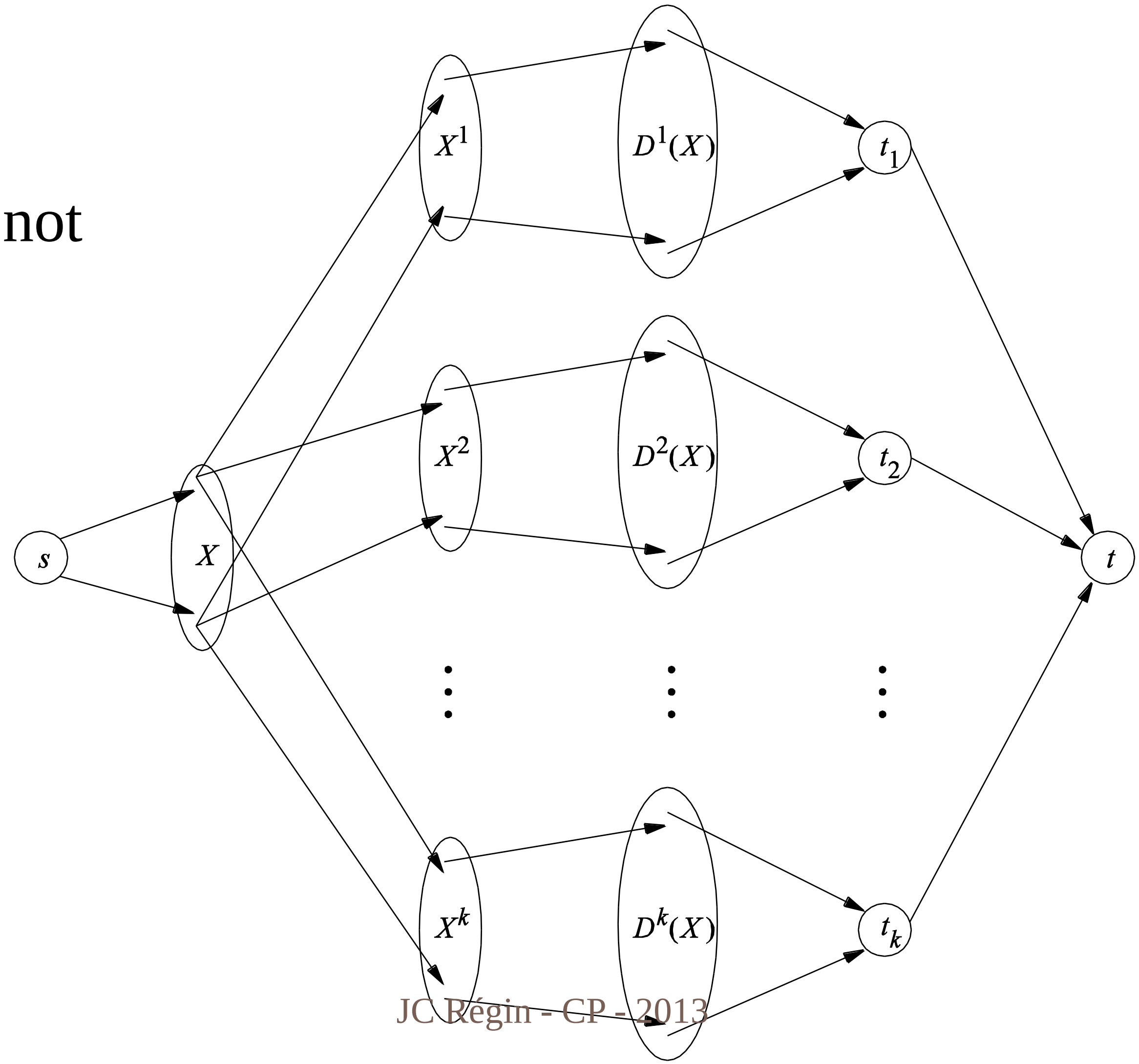
- Open constraint in a closed world:
  - The set of variables on which a constraint is defined is not exactly defined.
  - Instead of precisely knowing the set X, we know a superset of X
- Arise frequently in scheduling problems
  - Alternative: two possible branches
  - An object will be made by one branch but we don't know which one.
  - On every branch we have a sequence (an alldiff on the start variables)
- See WJ van Hoeve, JC Régin, **Open Constraints in a Closed World,** CPAIOR'06

Open global constraints

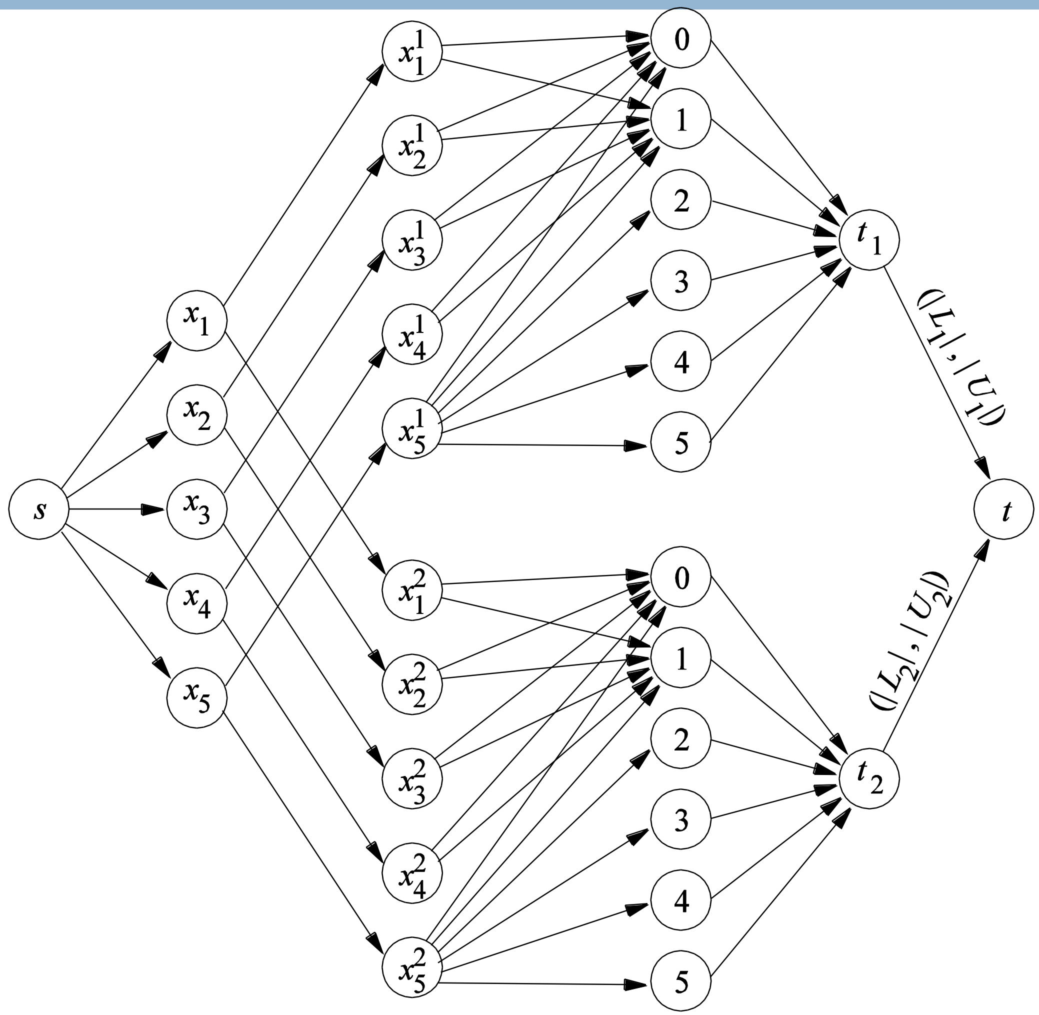
- Alternative: X = whole set of start variables of objects
- Model
  - Alldiff(X1 $\subseteq$ X)
  - Alldiff(X2 $\subseteq$ X)
  - X1 $\cup$ X2 = X
  - X1 $\cap$ X2 = $\varnothing$
- Even if we don't know exactly the variable set on which a constraint is defined we can deduce some things.
- Example: x1,x2,x3,x4 with domain = {a,b} and x5 with domain = {a,b,c,d}. We can deduce that x1,x2,x3,x4 can only take values a and b. Then, no other variable may take a and b and so D(x5)={c,d}

Open global constraints

Several alldiff
Maybe disjoint or not

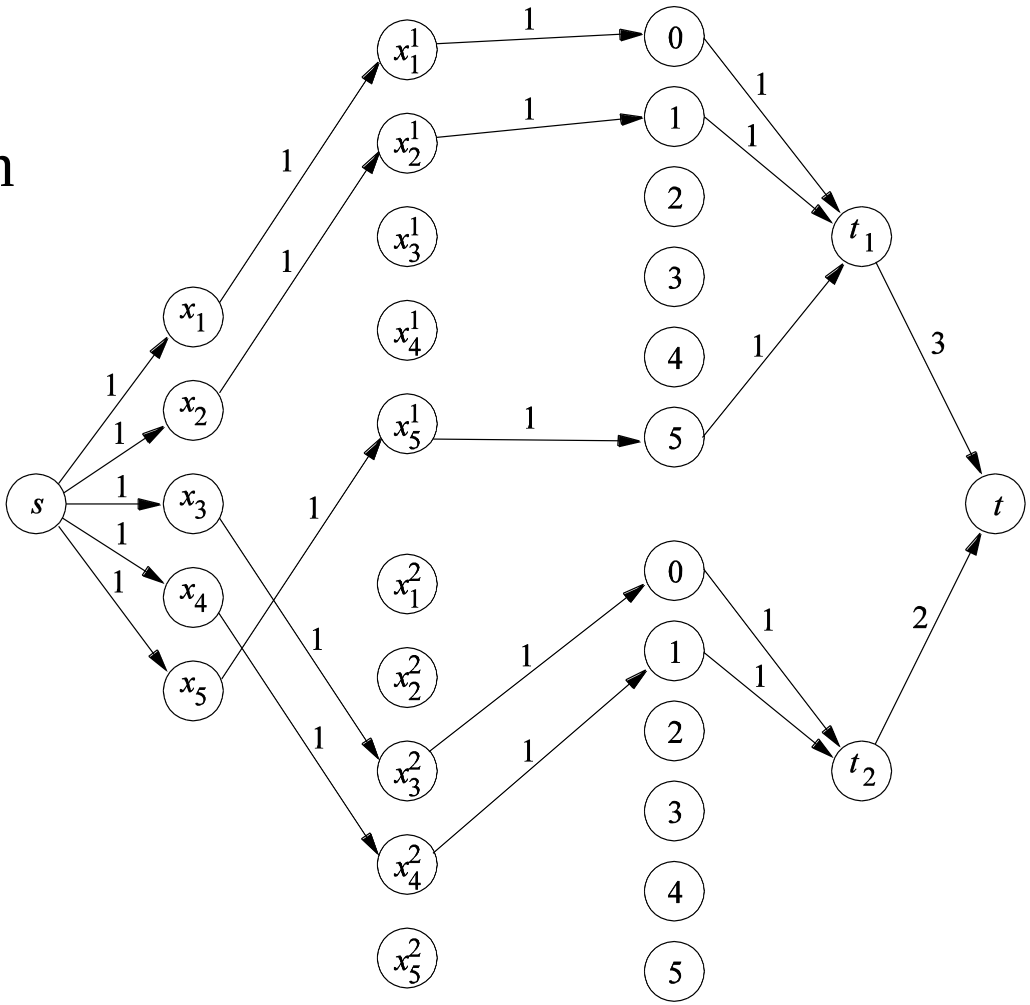


Combination of two open alldiff



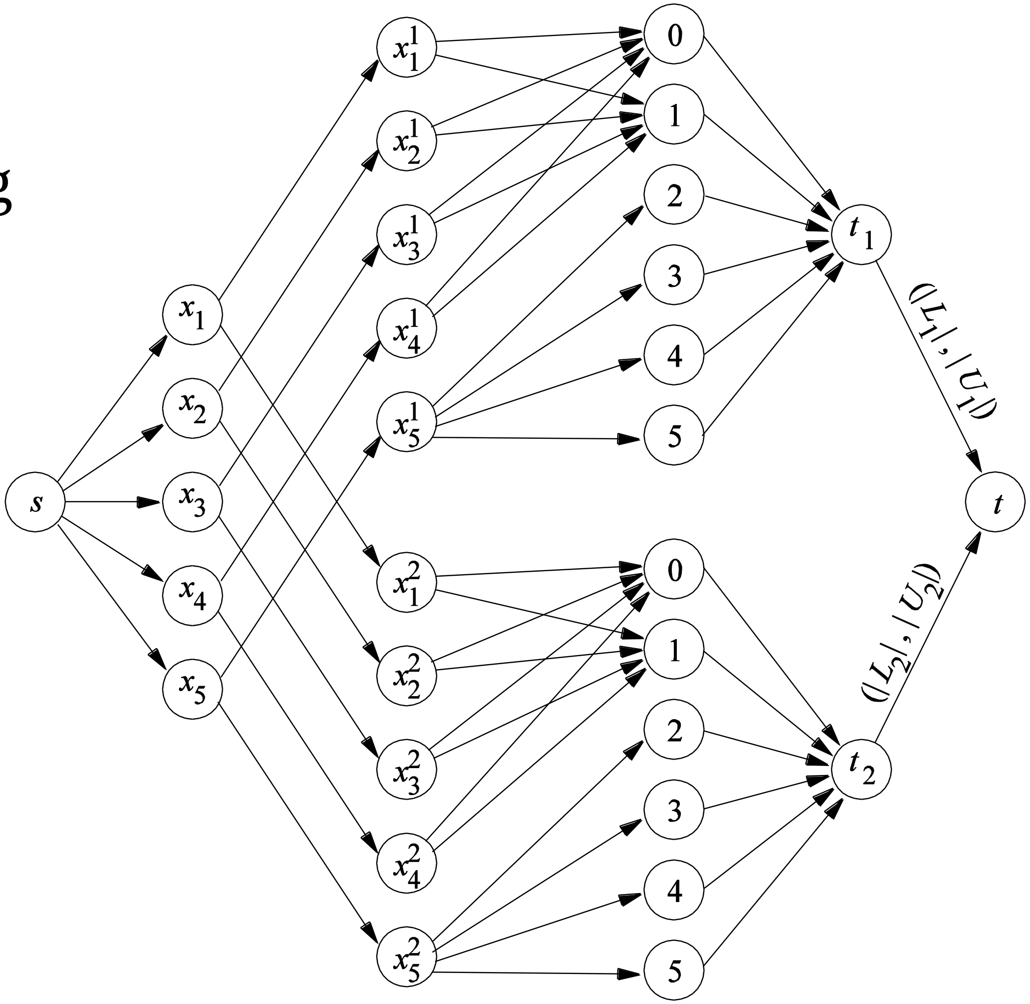
Combination of two open alldiff

A possible solution



Combination of two open alldiff

After AC Filtering



The modeling power

- Flow based constraints are used for any category of global constraints
- **Everyday we discover flow based constraints**
- Some flow based constraints are really complex

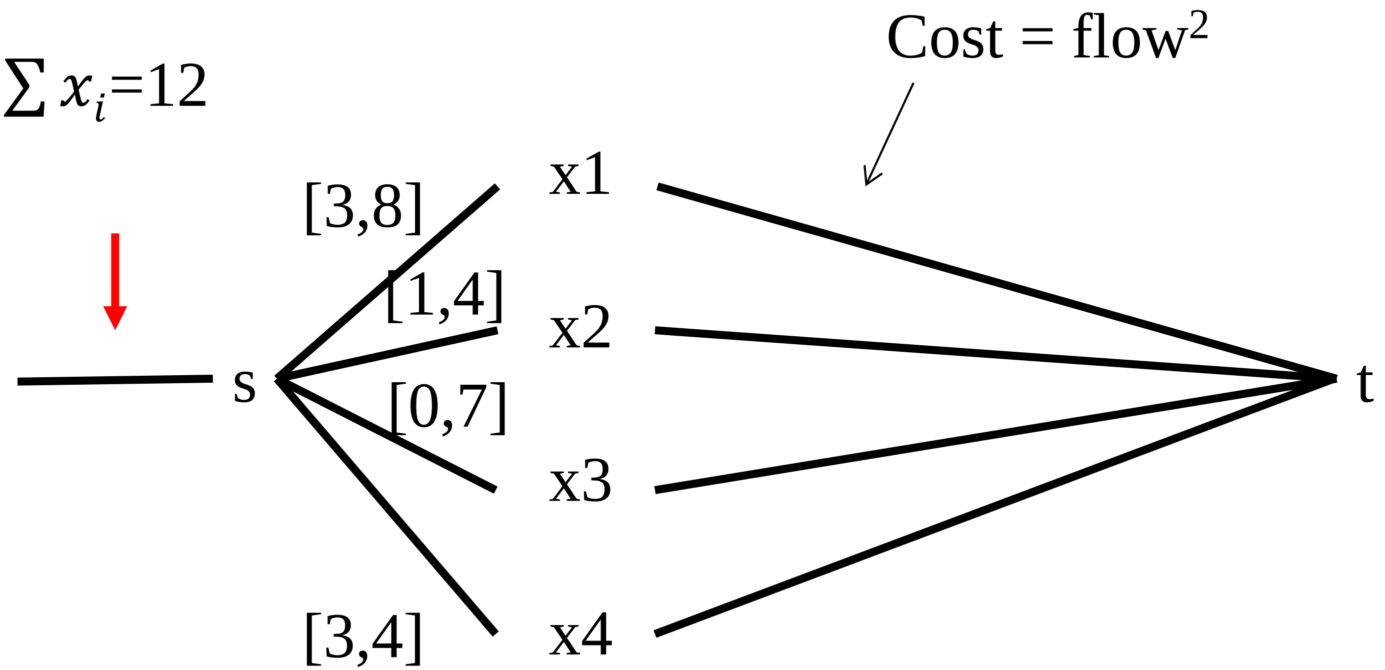
Everyday we discover flow based constraints!

- Consider the spread constraint
  - $\frac{1}{n}\sum x_i = \mathrm{m}$
  - Minimize $\sum(x_i - m)^2 = \sum x_i^2 - \mathrm{m}^2$

- We can model it by a simple flow
- I bet the complexity could be acceptable

The spread constraint

$\sum x_i = 12$



The modeling power

- Flow based constraints are used for any category of global constraints
- Everyday we discover flow based constraints
- **Some flow based constraints are really complex**

Complex flow based constraints

- Sequencing constraints
- Bin packing constraints

- Sequencing constraints
  - JC Régin and JF Puget, **A Filtering Algorithm for Global Sequencing Constraints**, CP'97
  - M. J. Maher, N. Narodytska, C-G Quimper, T. Walsh, **Flow-Based Propagators for the SEQUENCE and Related Global Constraints.** CP'08
- Inter-distance
  - JC Régin, **the allMinDistance constraint**, ILOG
  - C-G Quimper, A. López-Ortiz, G. Pesant, **A Quadratic Propagator for the Inter-Distance Constraint.** AAAI'06

Global Sequencing Constraint

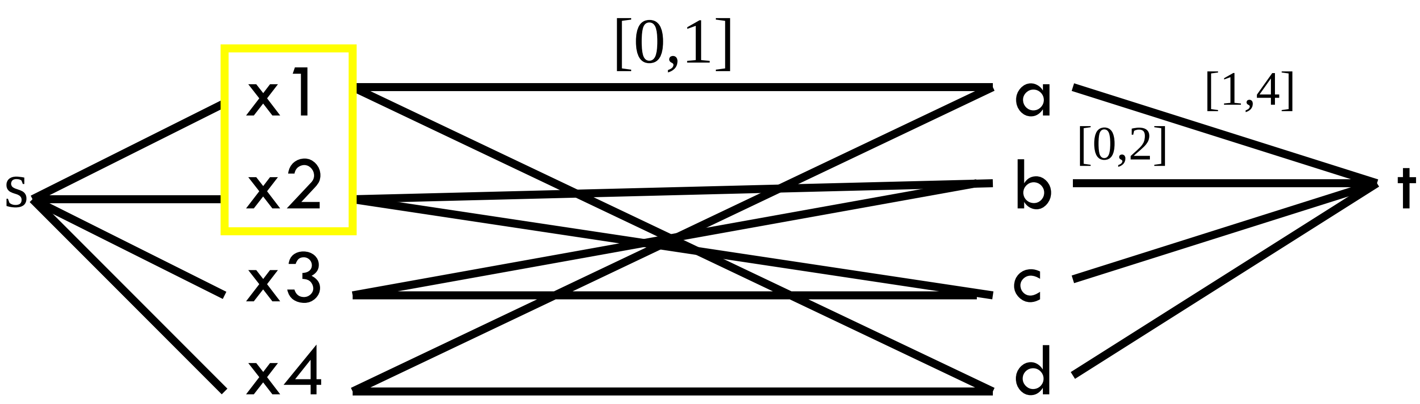
- GSC(X,V,min,max,q,{li},{ui})
- A GCC (Global cardinality constraint) for the values of V + constraint stating that for each sequence S of q consecutive variables, at least min and at most max variables of S takes their values in V.

Abstract Values

- GSC(X,V,…): the values of D(X) - V are not constrained individually. For each sequence S they can be replaced (inside the constraint) by e(S) an abstract value.

Abstract Value

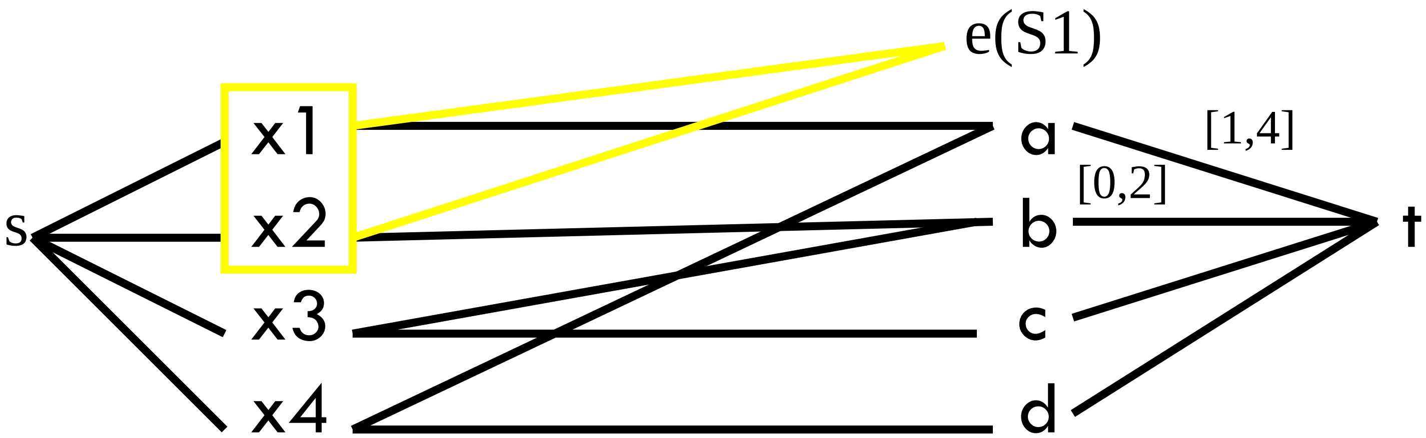
- GSC(X,V={a,b},min=0,max=1,q=2,…)



Constraints on values not in V are no longer considered

Abstract Value

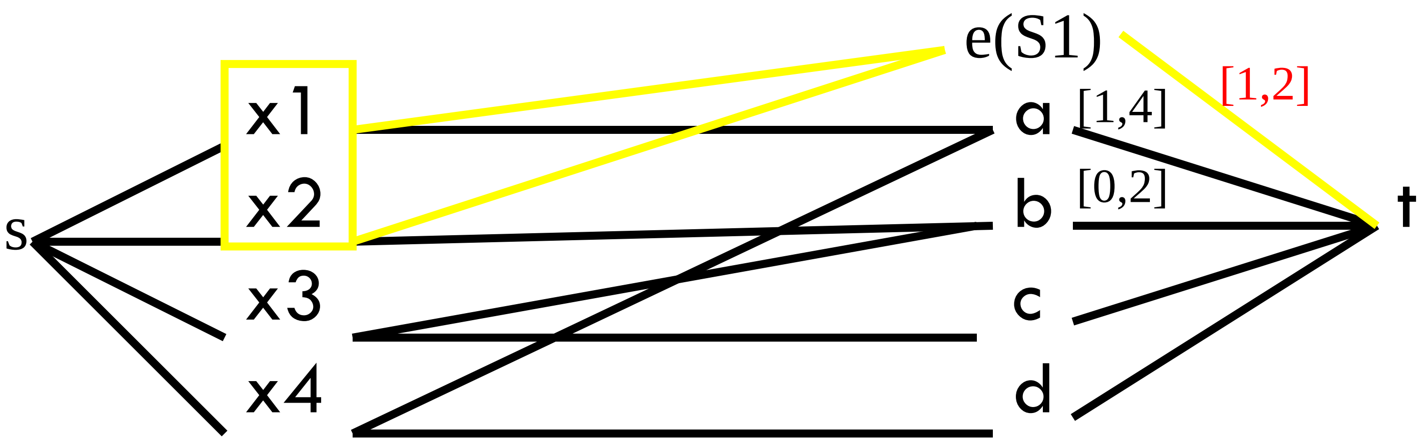
- GSC(X,V={a,b},min=0,max=1,q=2,…)



Values c and d does not belong to V, they are replaced by e(S1), for the sequence

Abstract Value

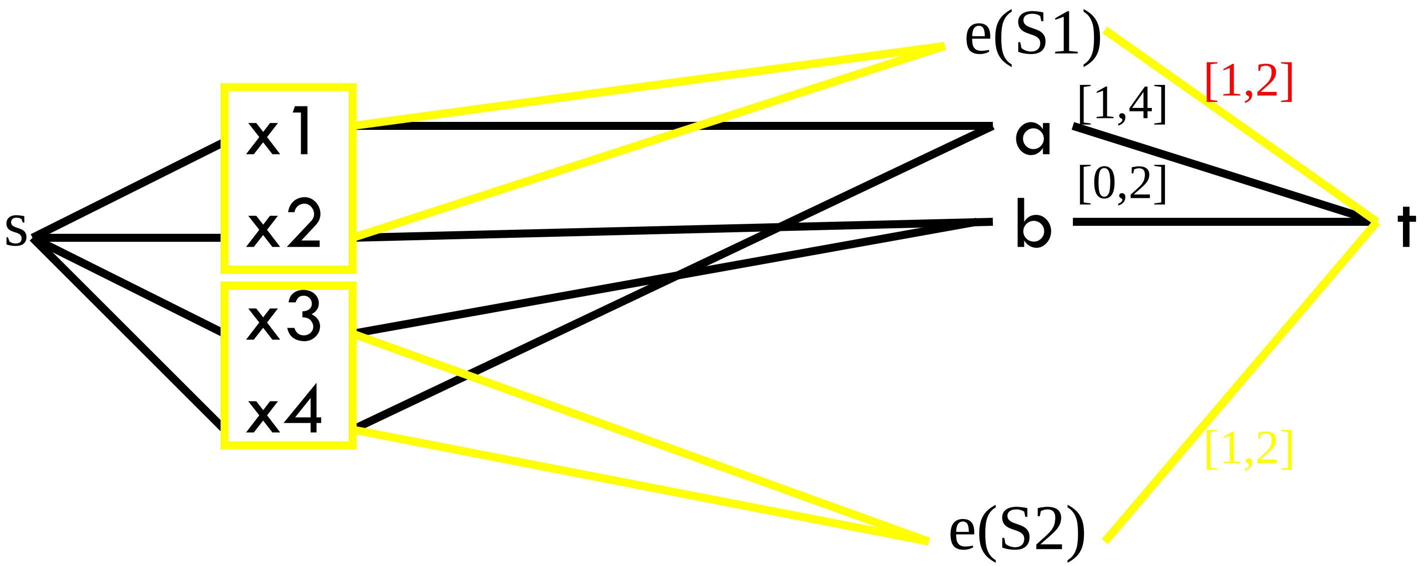
- GSC(X,V={a,b},min=0,max=1,q=2,…)



Value e(S1) must be taken at least |S|-max=2-1=1 and at most q -min = 2-0=2

Split of X into a partition of Sequence

- GSC(X,V={a,b},min=0,max=1,q=2,…)

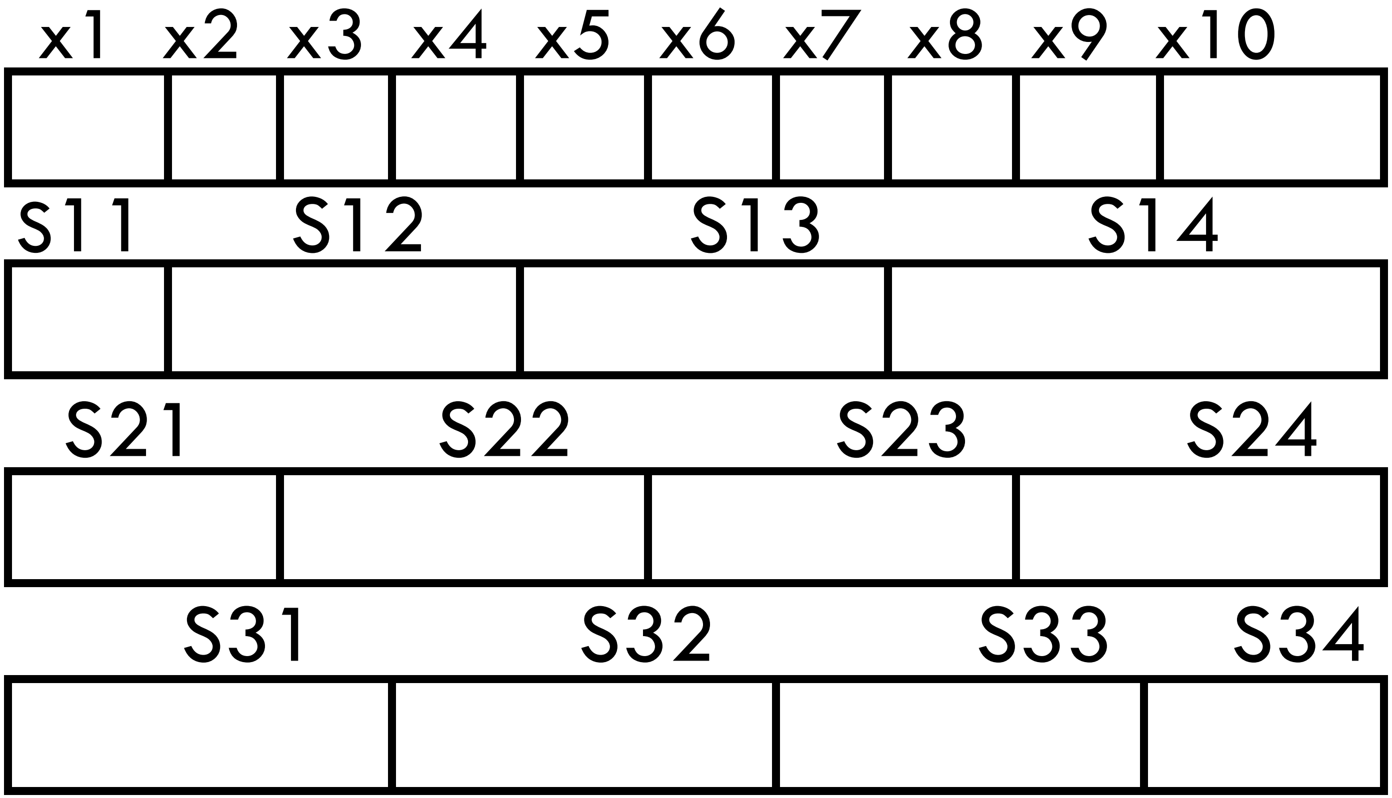


Red and Yellow arcs represent the constraints on sequences
Black arcs represent the global constraints on values of V

Split of X into partition of sequences

- Problem: an exponential number of partitions exist
- Solution: what is needed is just to have each sequence represented at least once. We propose to have $|X|$ partitions simultaneously.
- For our example: $P1=\{(x1,x2),(x3,x4)\}$ and $P2=\{(x1),(x2,x3),(x4)\}$

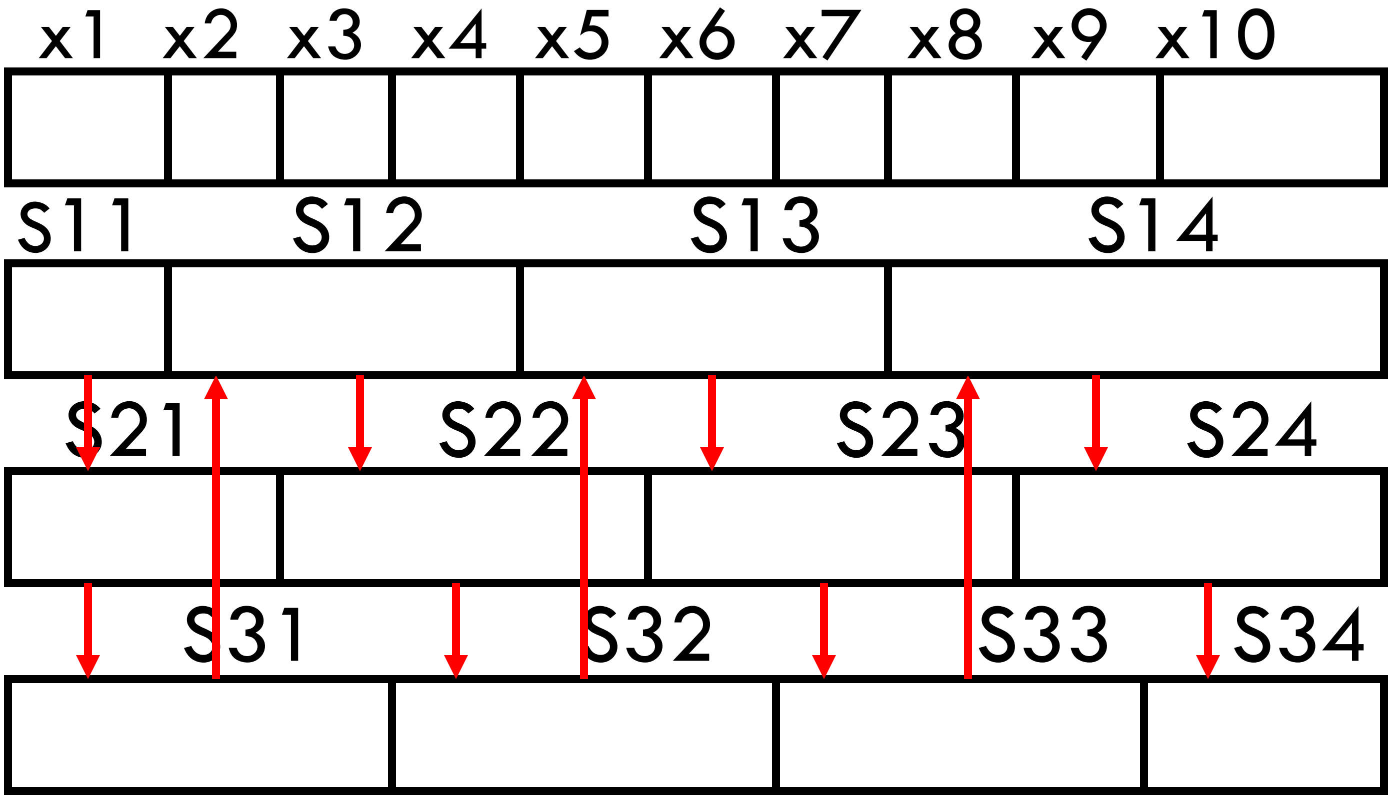
Partitions of sequences



Partitions of sequences

- Communication between sequences is necessary to solve the problem

Partitions of sequences

- 

Si → Sj means Sj is a successor of Si

Successor of a sequence

- The successor of a sequence is the same sequence translated by one variable.

  S1: x1 x2 x3
  S2:   x2 x3 x4

- #e(S1) and #e(S2) can be linked:
  $|\#e(S1) - \#e(S2)| \leq 1$

Constraints between sequences

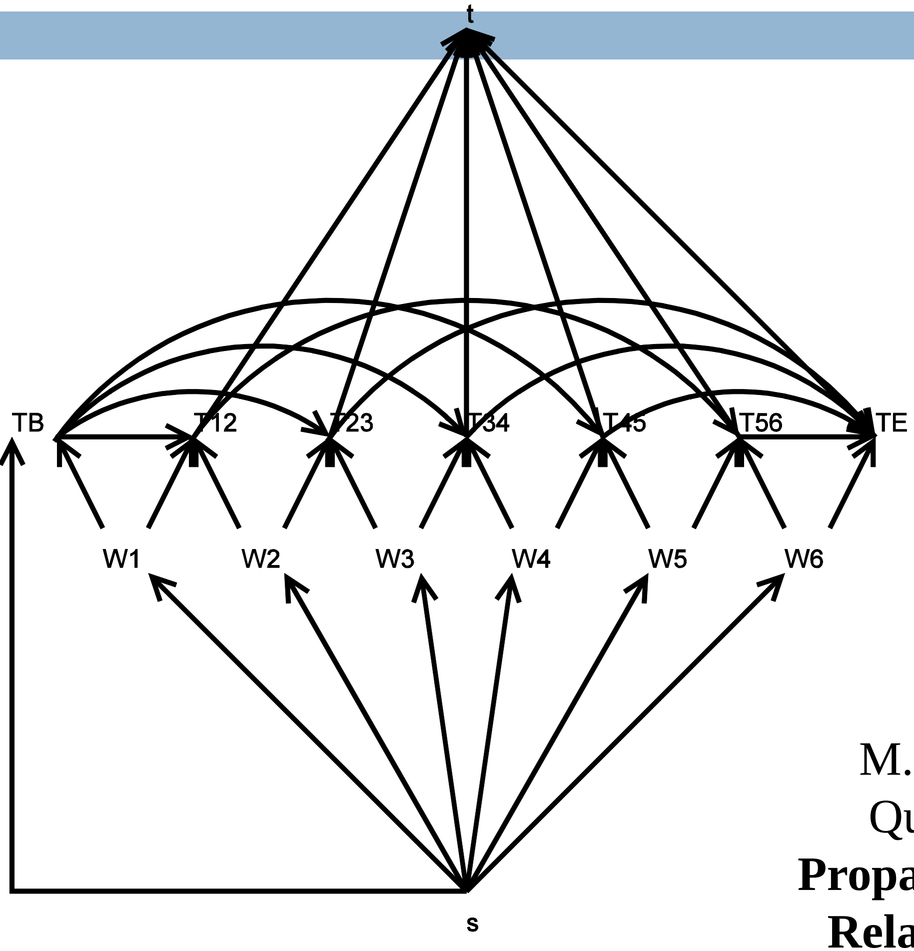
- $|\#e(S1) - \#e(S2)| \leq 1$ must be implemented carefully:
  S1: x1 x2 x3
  S2: x2 x3 x4

- $x1=e(S1)$ and $x4 \neq e(S2) \Leftrightarrow \#e(S1) = \#e(S2) +1$
- ($x1=e(S1)$ and $x4=e(S2)$) or ($x1 \neq e(S1)$ and $x4 \neq e(s2)$) $\Leftrightarrow \#e(S1) = \#e(S2)$
- $x1 \neq e(S1)$ and $x4=e(S2) \Leftrightarrow \#e(S1) = \#e(S2) - 1$

Sequencing constraint

- The constraints may be slow for easy instances.
- However, in 2009 this filtering algorithm was necessary for solving 12 car sequencing problems of the CSP Lib. (See W-J. van Hoeve, G. Pesant, L-M. Rousseau, and A. Sabharwal. **New filtering algorithms for combinations of among constraints.** Constraints, 2009)

Sequence constraint



M. J. Maher, N. Narodytska, C-G Quimper, T. Walsh, **Flow-Based Propagators for the SEQUENCE and Related Global Constraints**. CP'08

Bin packing

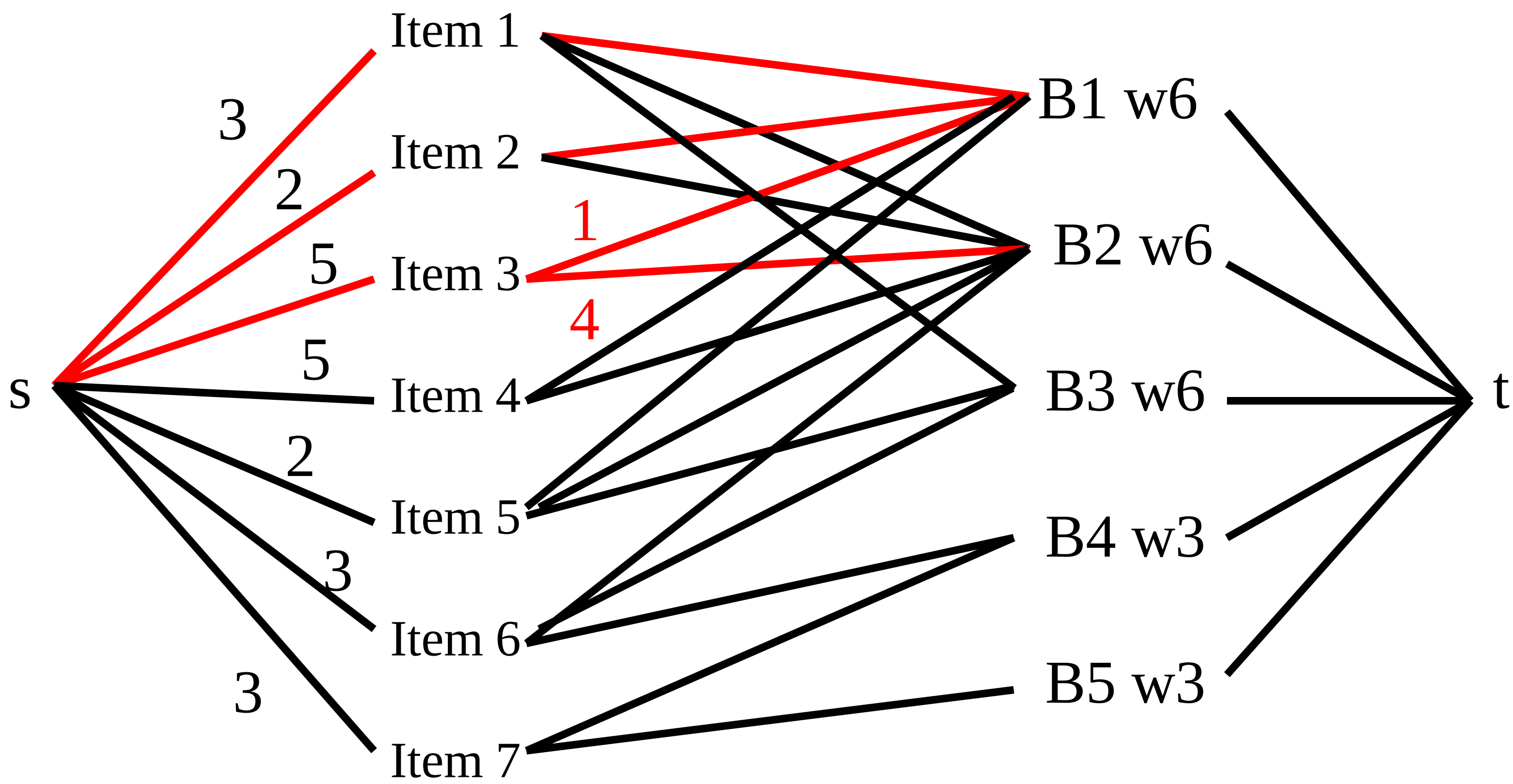
- DATA:
  - A set of items, each have item i has a weight w(i)
  - A set of bins, each bin b can accept some items and no more than a given weight W(b)
- QUESTION:
  - Put each item into a bin such that
    - The sum of weights of the items put in a bin is no more than W(b)
    - The bin accept the item assigned to it.

Packing Constraints (bin packing):

- A lot of bin packing problems are dominated by the assignment part and not by the knapsack.
  - See the phD thesis of Pierre Schaus.
  - F. Pelsser, P. Schaus, J-C Régin, **Revisiting the Cardinality Reasoning for BinPacking Constraint.** CP'13
  - P. Schaus, J-C Régin, R. Van Schaeren, W. Dullaert, B. Raa, **Cardinality Reasoning for Bin-Packing Constraint: Application to a Tank Allocation Problem.** CP'12

- In the bin packing either an item goes to a bin or not. It cannot partially go to a bin. We cannot separate the items into some parts (this new formulation is polynomial)
- With the flow: when we use non 0-1 capacities then the flow may « split the quantity ». We cannot say I want either k units or 0 (this formulation is NP-Complete)

Value Network

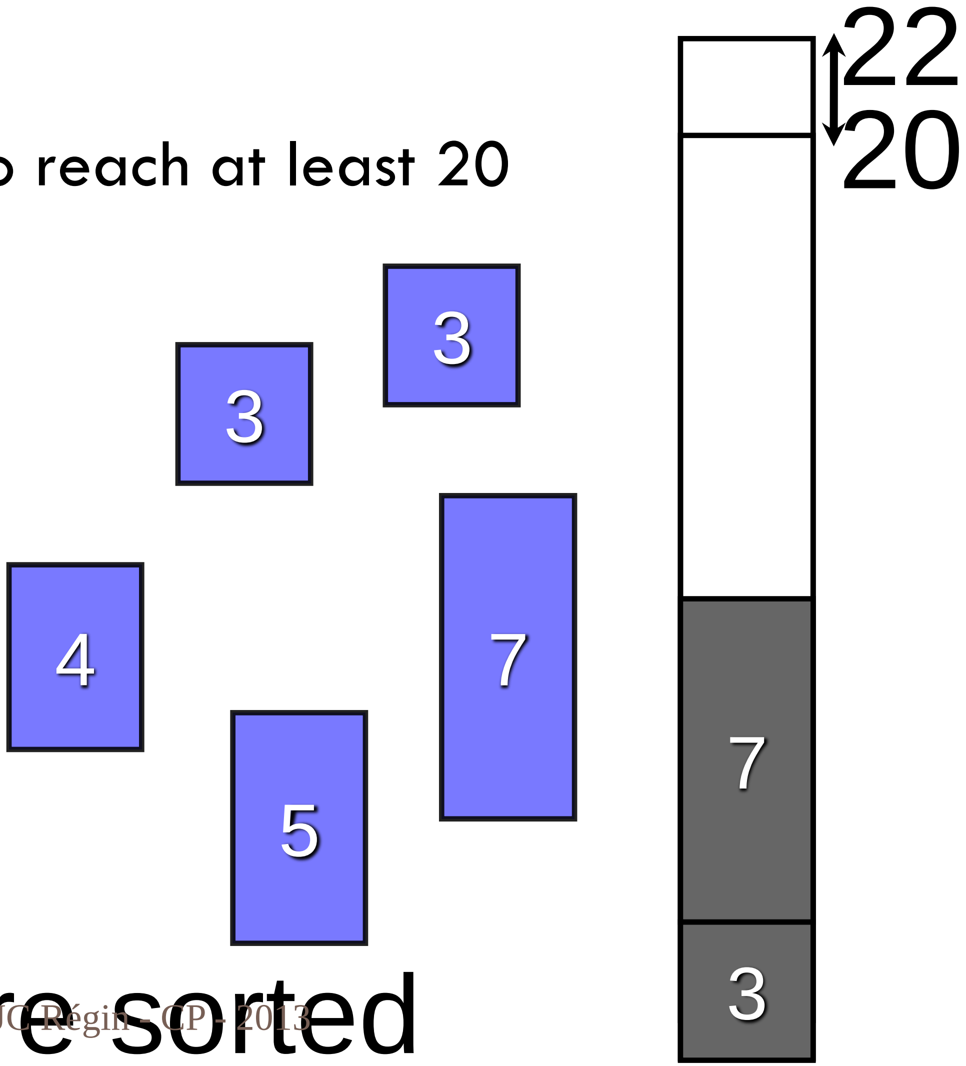


The idea: using an assignment problem

- Splitting the quantity of flow is not good
- We propose to deal with an assignment problem as a relaxation of the problem

Bounds on Cardinality

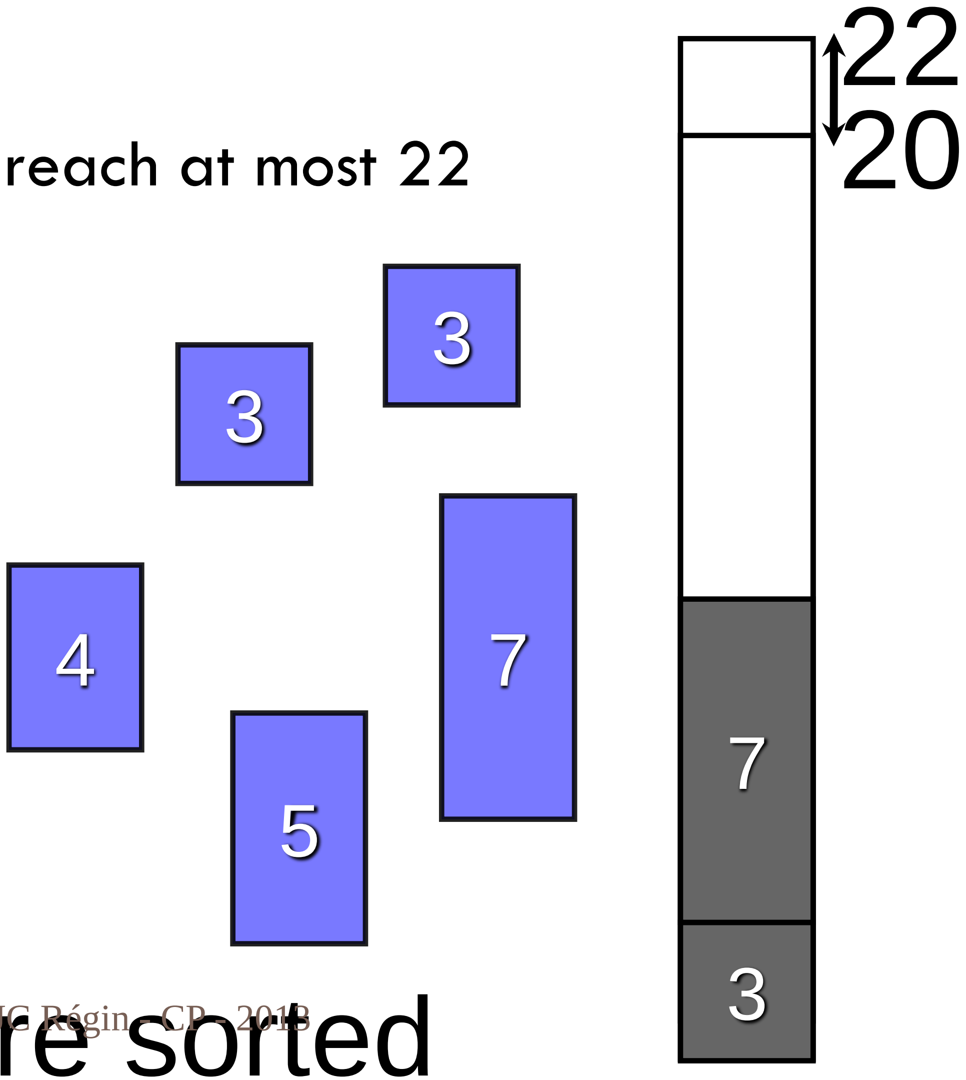
- Card Min =2+2
  - use heaviest ones first to reach at least 20



O(n) once items are sorted

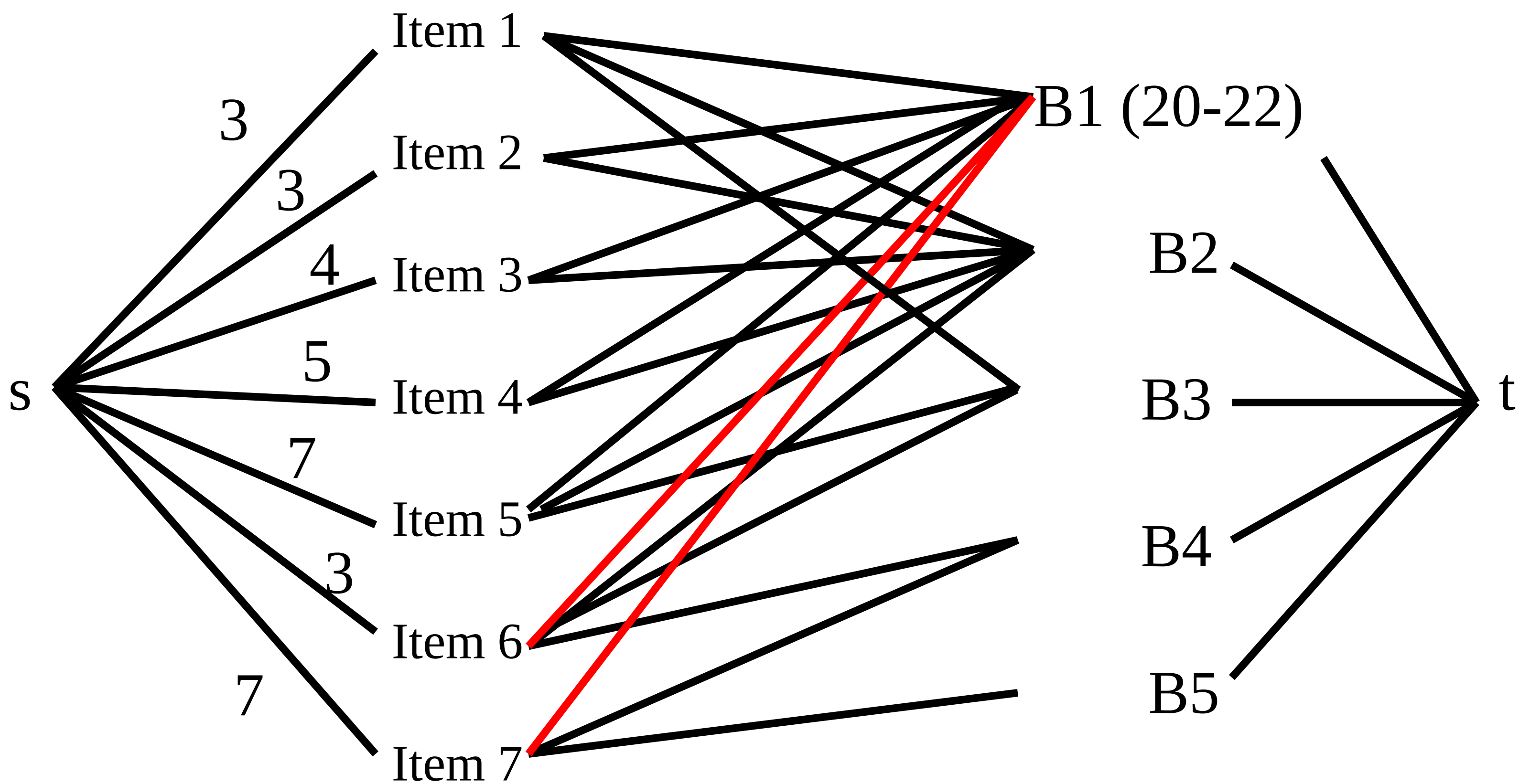
Bounds on Cardinality

- Card Max =2+3
  - use lightest ones first to reach at most 22

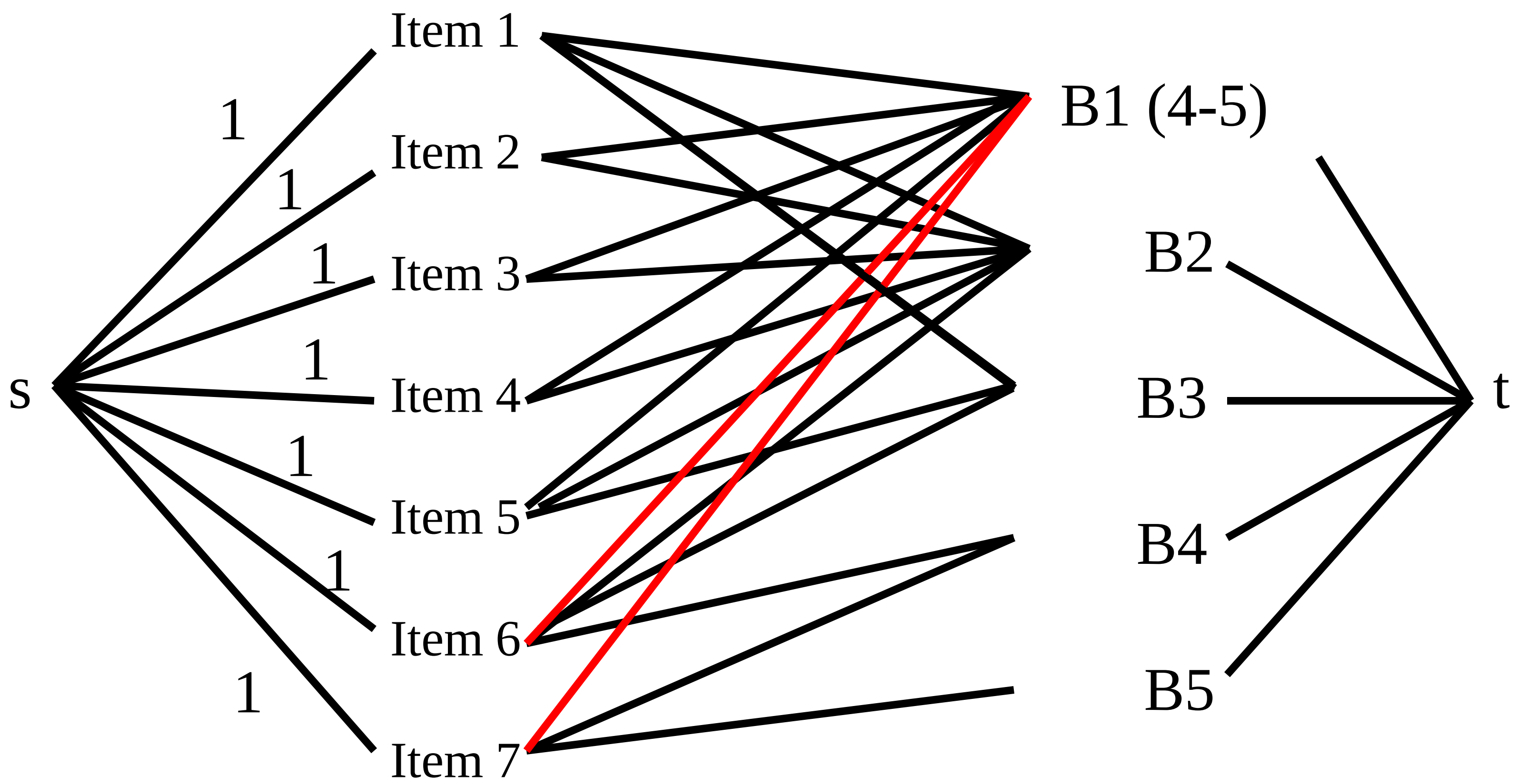


O(n) once items are sorted

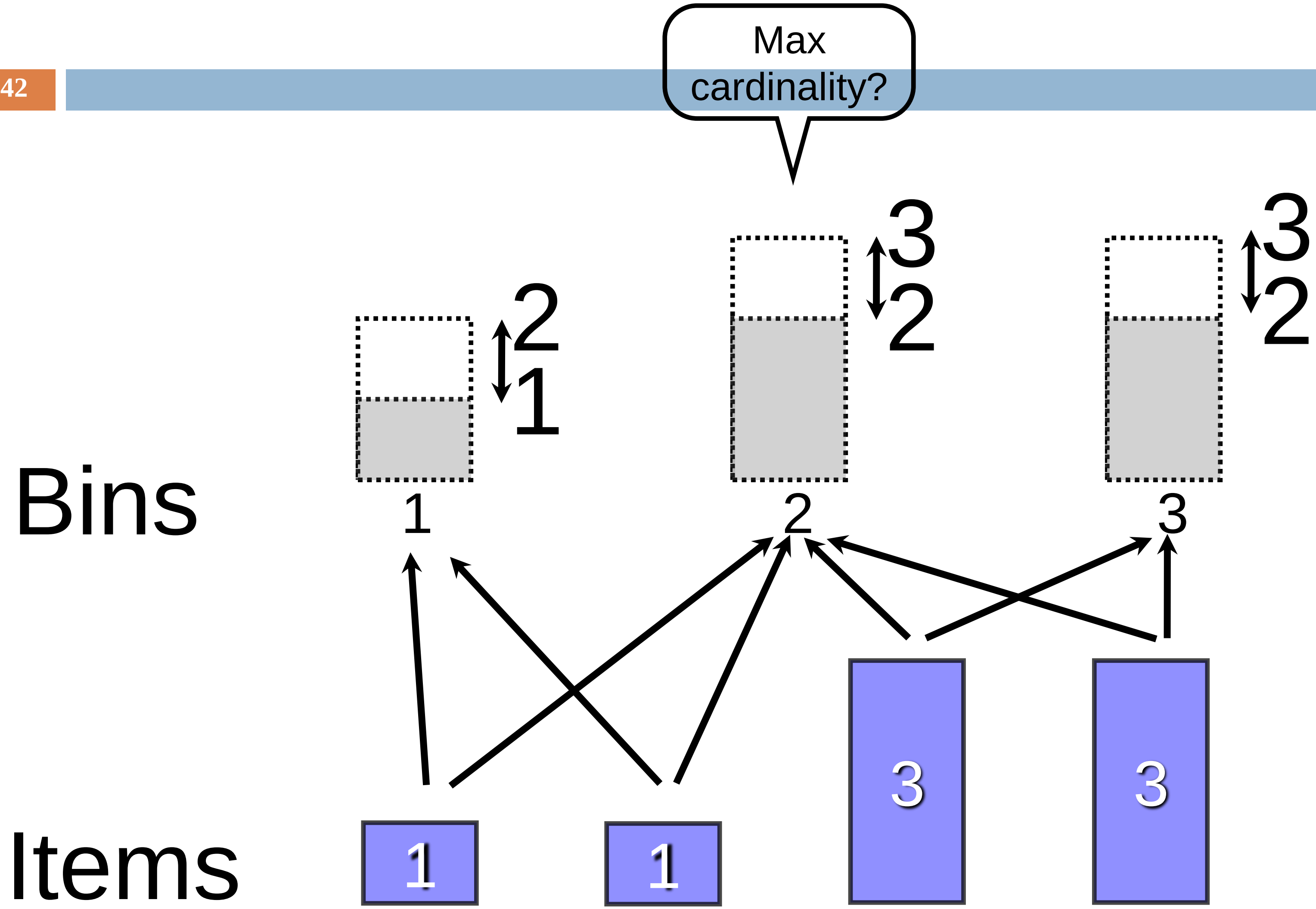
Value Network



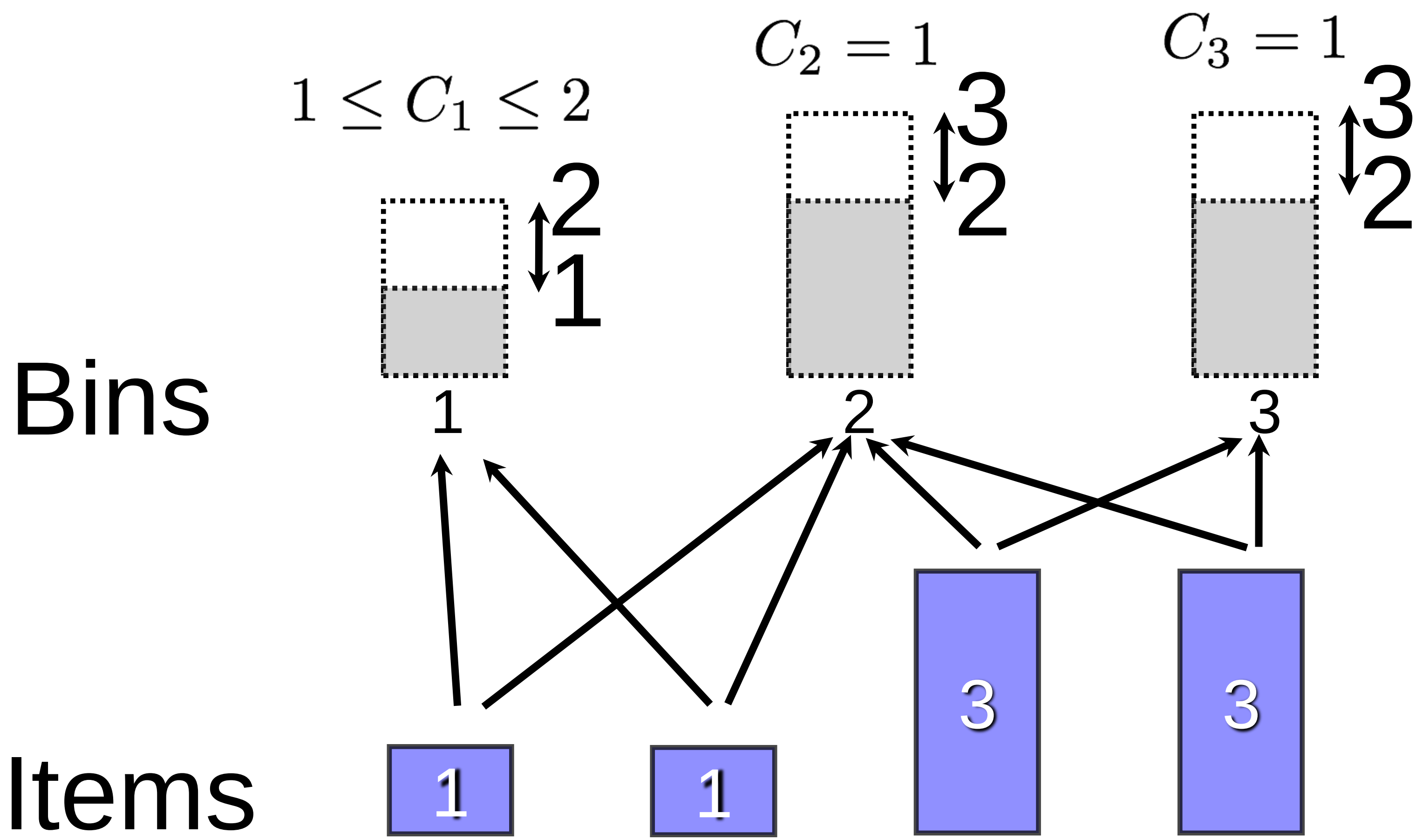
Value Network



- We can do better by considering several bins together …



Yes but Bin1 absolutely needs an item of size 1 ...


$1 \leq C_1 \leq 2$
$C_2 = 1$
$C_3 = 1$
2
1
3
2
3
2
Bins
1
2
3
Items
1
1
3
3

Plan

- The power of Flow based constraints
- **What is missing?**
- What could be the evolution?
- Conclusion

What is missing?

- Currently there is no general flow constraints with a good complexity. For any feasible flow:
  - **Minimum and maximum flow value in each arc**
  - **Arcs having always non zero flow value**

- Idem for min cost flow
- A tentative: R. Steiger, WJ van Hoeve, R. Szymanek, **An efficient generic network flow constraint.** SAC 2011

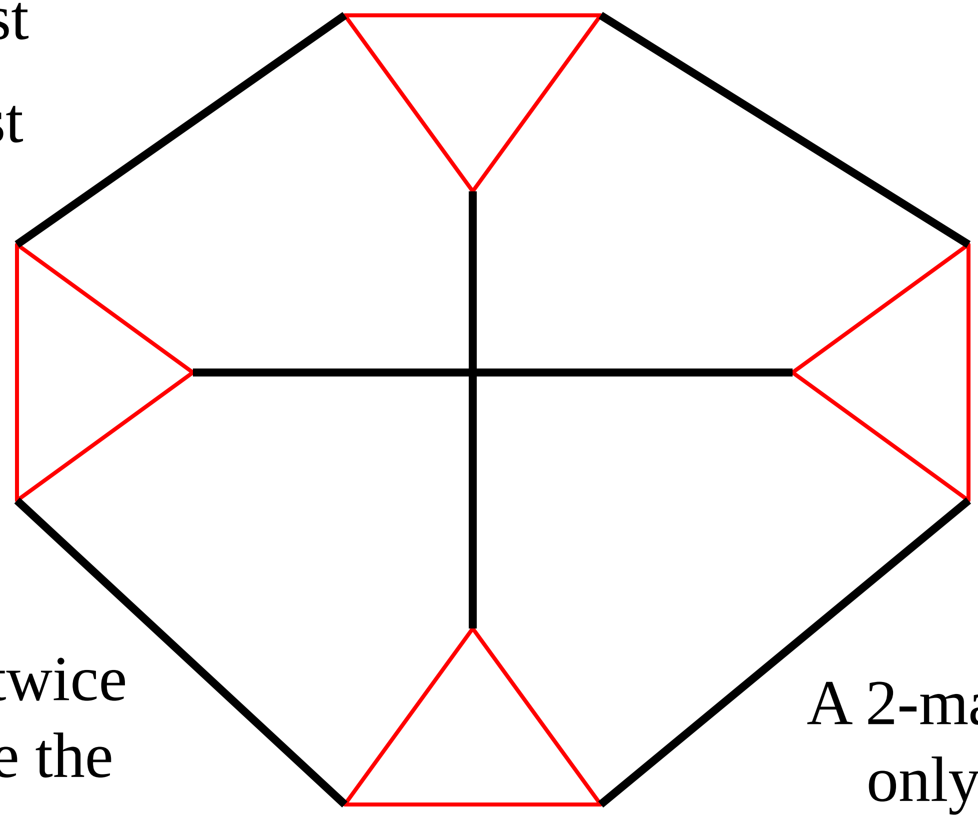
What is missing ?

- Be careful flows and matchings are not equivalent
- Bipartite matching are equivalent to flow problems
- General matching problems cannot be represented by flow problems
- Matching can also be solved in polynomial time but
  - The algorithms are quite complex (recognized as the most difficult ones in computer science)
  - The filtering algorithm are not really good
- Some work have been done
  - JC Régin, **The symmetric alldiff constraint**, IJCAI'99
  - M. Henz, T. Muller, and S. Thiel, **Global constraints for round robin tournament scheduling.** EJOR 2004

- Path problems are 2-matching problem and not 2 flows problems
- In TSP you need matchings instead of flows.

HIGH cost

LOW cost



A 2-flow saturates twice the blacks and once the reds

A 2-matching contains only the triangles

Plan

- The power of Flow based constraints
- What is missing?
- **What could be the evolution?**
- Conclusion

The evolution

- We have some other nice models using flows
  - See papers of A. Cambazard, B. O'Sullivan and H. Simonis about bin packing
- The models are quite complex. So the simple solutions are no longer simple…
- Should the user be capable of writing them?
  - If yes, can we still continue to say that CP is easy to use?
- I think we should stop asking the user for being able to write complex model using all the features of the flow algorithms and all their strengths while avoiding their weaknesses.

The evolution

- In the 90s we were using mainly primitive constraints
- We (the researchers) proposed (invented) more than 350 constraints.
- We got really good performance on a lot of problems
- However, is it good to propose so many constraints to the user?

The evolution

- We should stop to ask the user for selecting the right constraint
- I propose a new step: **we work on problems.**
- Users are able to understand problems (bin packing, tsp with time windows,...)
- They do not need to know which techniques is encapsulated
- Our job will be to automatically define what kind of constraints we should use in regards to the model

The evolution

- Be careful: working on problems and combinations of problems is not equivalent than having a black box solver
- This is just a natural generalization
  - Primitive to global constraints
  - Global constraints to Problems resolution

- Global constraints involve only the filtering part whereas the problem resolution may also integrate some search procedures or guides.

Conclusion and perspectives

- We should work more on difficult problems and on real world applications
- When I read the paper of P. Prosser about max clique I wanted to work again on this topic.
- We should be able to have more papers like that

Conclusion and perspectives

- Flow based constraints are widely used in CP
- We were able to define complex models using them
- Some progress could be made on
  - The weighted version especially when the graph is convex
  - The general version (non 0-1 arcs)
- We should also work for improving the « tools » we propose to the user.
- The evolution could be to deal directly with problem instead of global constraints

Open position

- There is an open position for being Maitre de Conférences at University Nice-Sophia Antipolis in the « Constraint and Proof » team.
  - Permanent position after one year
- Please contact me if you are interested

Thank you very much

- A Special thank to
  - Pierre Schaus, Michel Rueher, Laurent Perron, Michela Milano, Louis-Marin Rousseau, Gilles Pesant, Bernard Gendron, Laurent Michel, Willem-Jan van Hoeve, Barry O'Sullivan, Carla Gomes, Michael Trick, Narendra Jussien, Philippe Baptiste…