Topic 10: Propagation for a DISTINCT Constraint[1]
(Version of 30th October 2017)

Pierre Flener

ASTRA Research Group
on Combinatorial Optimisation
Uppsala University
Sweden

Course 1DL441:
Combinatorial Optimisation using
Constraint Programming

[1] Based also on some material by Christian Schulte

UPPSALA UNIVERSITET

Outline

1. Prelude
2. The DISTINCT Predicate
3. Naïve DC Propagation
4. Efficient DC Propagation
5. Efficient BC Propagation
6. Bibliography

UPPSALA UNIVERSITET

Outline

1 Prelude

2 The DISTINCT Predicate

3 Naïve DC Propagation

4 Efficient DC Propagation

5 Efficient BC Propagation

6 Bibliography

Domain Consistency Revisited

All solutions to the constraint need to be considered:

- A naïve propagator, computing first all solutions, takes too much time and space. Recall Topic 3: the store $\{x \mapsto \{2,\ldots,7\}, y \mapsto \{0,1,2\}, z \mapsto \{-1,\ldots,2\}\}$ has solutions $\langle 3,1,0\rangle$, $\langle 5,0,1\rangle$, $\langle 6,2,0\rangle$ to $x = 3 \cdot y + 5 \cdot z$; hence $\{x \mapsto \{3,5,6\}, y \mapsto \{0,1,2\}, z \mapsto \{0,1\}\}$ is DC.
- An efficient DC propagator does in general not exist (for LINEAR$(A, X, =, d)$ it takes time exponential in $|X|$), but there exist polynomial-time DC propagators for some predicates, such as ELEMENT (☞ see Topic 6).
- Are there efficient DC propagators for more complex predicates than ELEMENT, such as DISTINCT? ☞ Yes!

UPPSALA UNIVERSITET

Outline

1 Prelude

2 The DISTINCT Predicate

3 Naïve DC Propagation

4 Efficient DC Propagation

5 Efficient BC Propagation

6 Bibliography

Definition

The DISTINCT($\{x_1, \ldots, x_n\}$) constraint, also known as the ALLDIFFERENT($\{x_1, \ldots, x_n\}$) constraint, has the semantics of a conjunction of $\frac{n \cdot (n-1)}{2}$ disequality constraints:

$$\textbf{forall}(i, j \textbf{ in } 1..n \textbf{ where } i < j)(x_i \neq x_j)$$

Example (DISTINCT($\{u, v, w, x, y, z\}$))

From the store

$$\left\{ \begin{array}{c} u \mapsto \{0,1\},\ v \mapsto \{1,2\},\ w \mapsto \{0,2\}, \\ x \mapsto \{1,3\},\ y \mapsto \{2,3,4,5\},\ z \mapsto \{5,6\} \end{array} \right\}$$

the pink values are pruned upon DC.

Is DC Needed for DISTINCT?

Example (Golomb Rulers)

Design a ruler with n ticks such that:

- The distances between any 2 distinct ticks are distinct.
- The length of the ruler is minimal.

For $n = 6$, an optimal ruler is $[0, 1, 4, 10, 12, 17]$.
This very hard problem has applications in crystallography.

n	value consistency	domain consistency
7	950 nodes	474 nodes
8	7,622 nodes	3,076 nodes
9	55,930 nodes	16,608 nodes
10	413,922 nodes	97,782 nodes
11	6,330,568 nodes	1,448,666 nodes

Good search-tree reduction: worth looking for a propagator!

UPPSALA UNIVERSITET

Outline

Prelude

The DISTINCT Predicate

Naïve DC Propagation

Efficient DC Propagation

Efficient BC Propagation

Bibliography

1 Prelude

2 The DISTINCT Predicate

3 Naïve DC Propagation

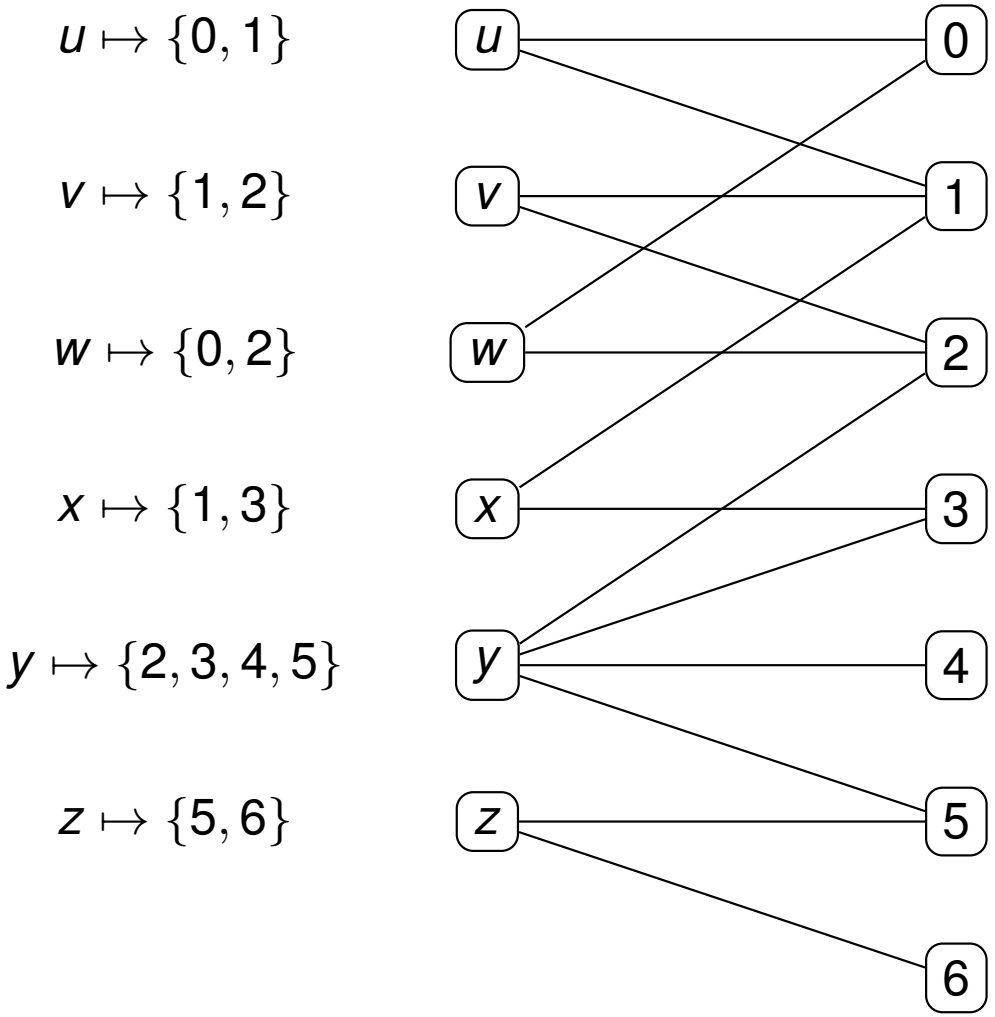
4 Efficient DC Propagation

5 Efficient BC Propagation

6 Bibliography

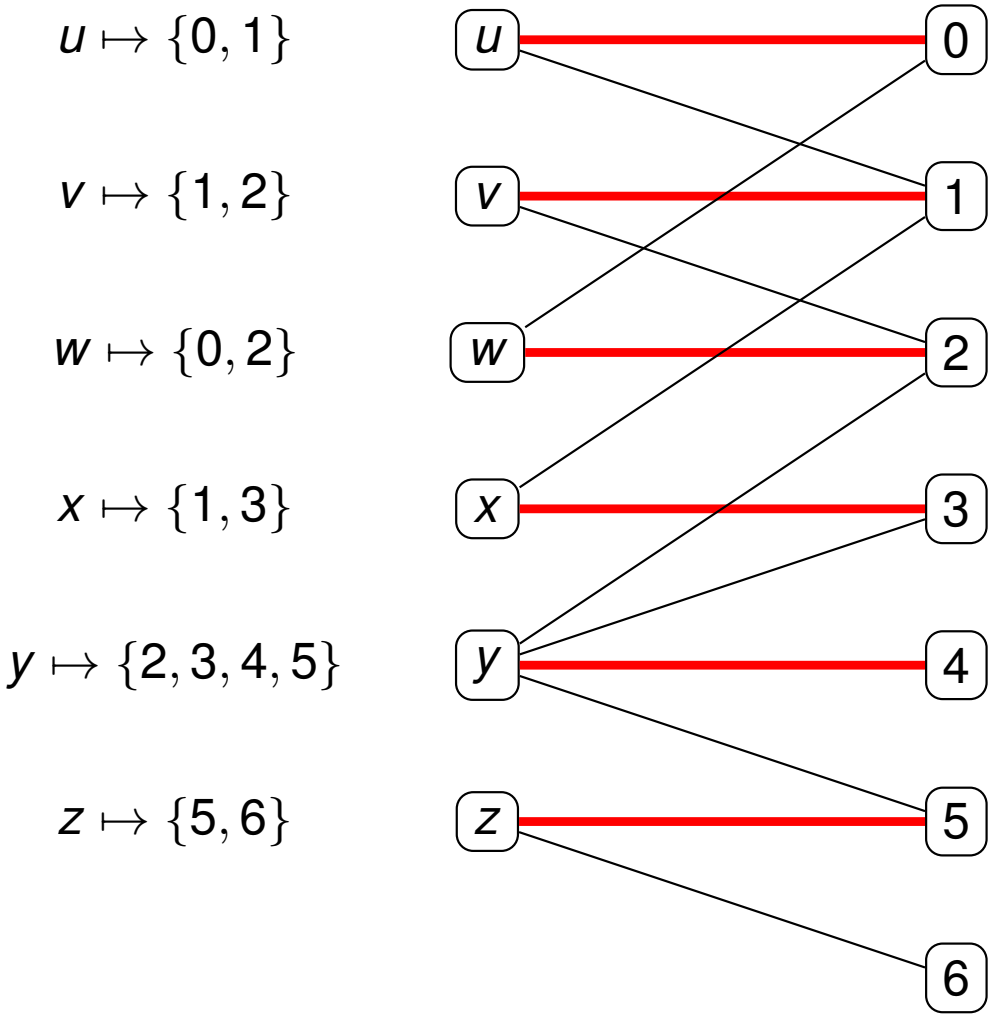
Variable-Value Graph:

Construct from the current domains a bipartite graph:



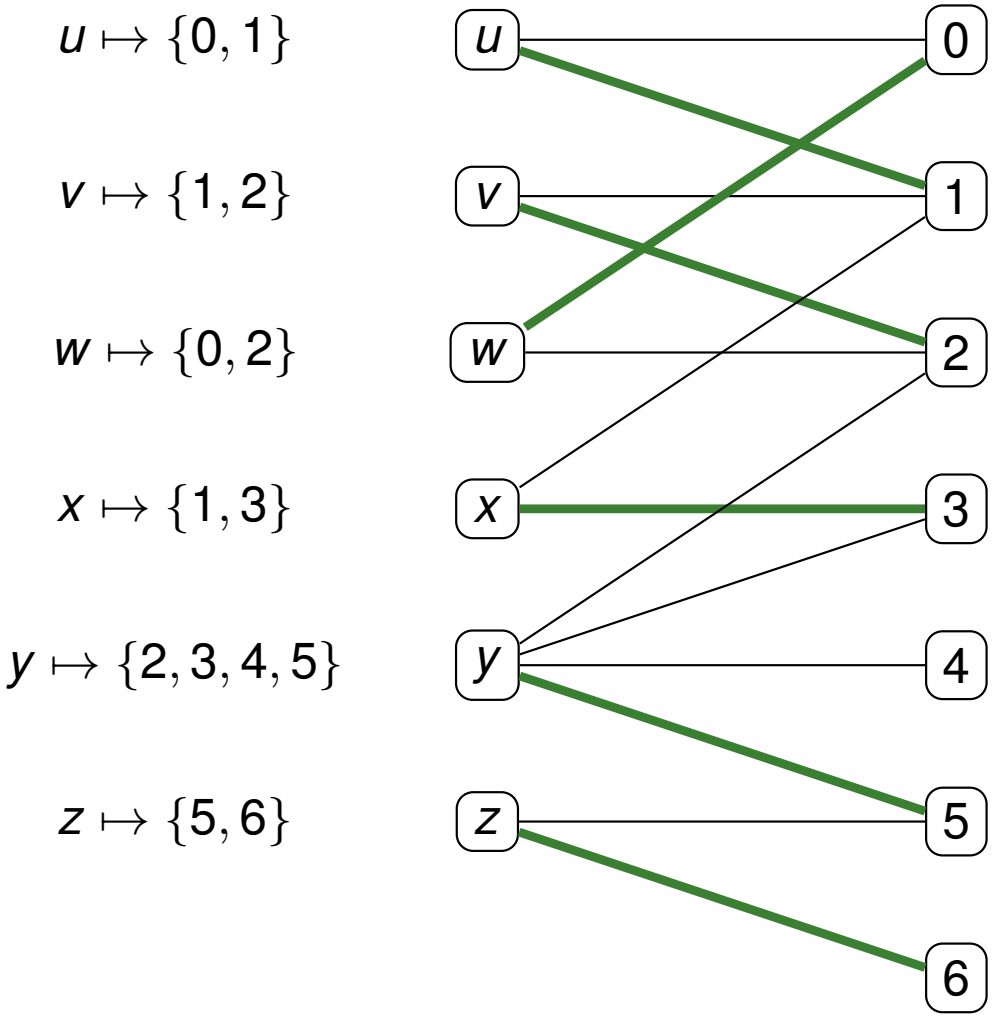
Variable-Value Graph: Matchings Are Solutions:

A (maximum) matching is a (maximum) subset of edges so that no vertex is incident to two of its edges. Example 1:



Variable-Value Graph: Matchings Are Solutions:

A (maximum) matching is a (maximum) subset of edges so that no vertex is incident to two of its edges. Example 2:



UPPSALA UNIVERSITET

Prelude
The DISTINCT Predicate
Naïve DC Propagation
Efficient DC Propagation
Efficient BC Propagation
Bibliography

Naïve DC propagator:

1. If no maximum matching exists, then fail.
2. Compute all maximum matchings & mark their edges.
3. For every unmarked edge between a decision variable α and a value d: prune value d from dom(α).

But there are as many maximum matchings as solutions!

☞ We have solved the space issue, but not the time issue.

Matching theory to the rescue!

There is a relationship between the edges in a maximum matching and the edges in all other maximum matchings!

☞ Hence we need only compute one maximum matching!

UPPSALA UNIVERSITET

Outline

Prelude

The DISTINCT Predicate

Naïve DC Propagation

Efficient DC Propagation

Efficient BC Propagation

Bibliography

1 Prelude

2 The DISTINCT Predicate

3 Naïve DC Propagation

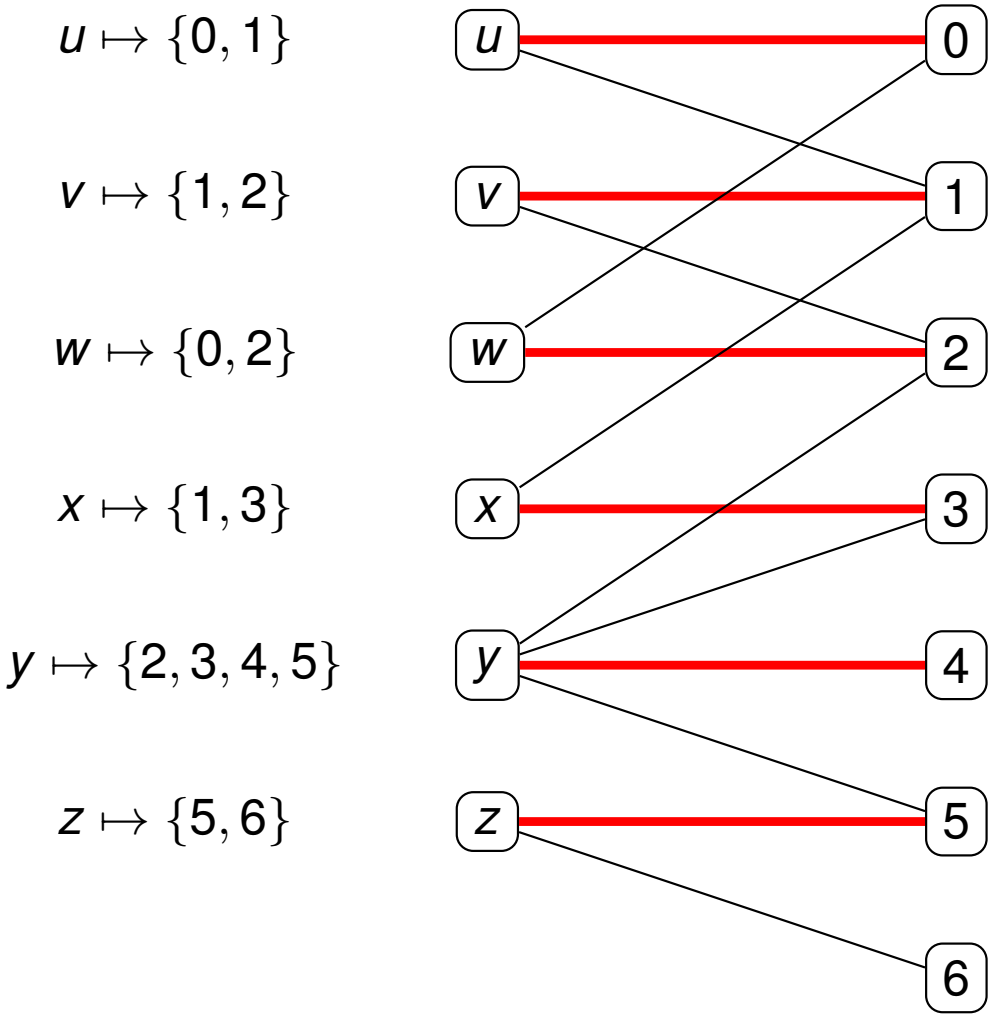
4 **Efficient DC Propagation**

5 Efficient BC Propagation

6 Bibliography

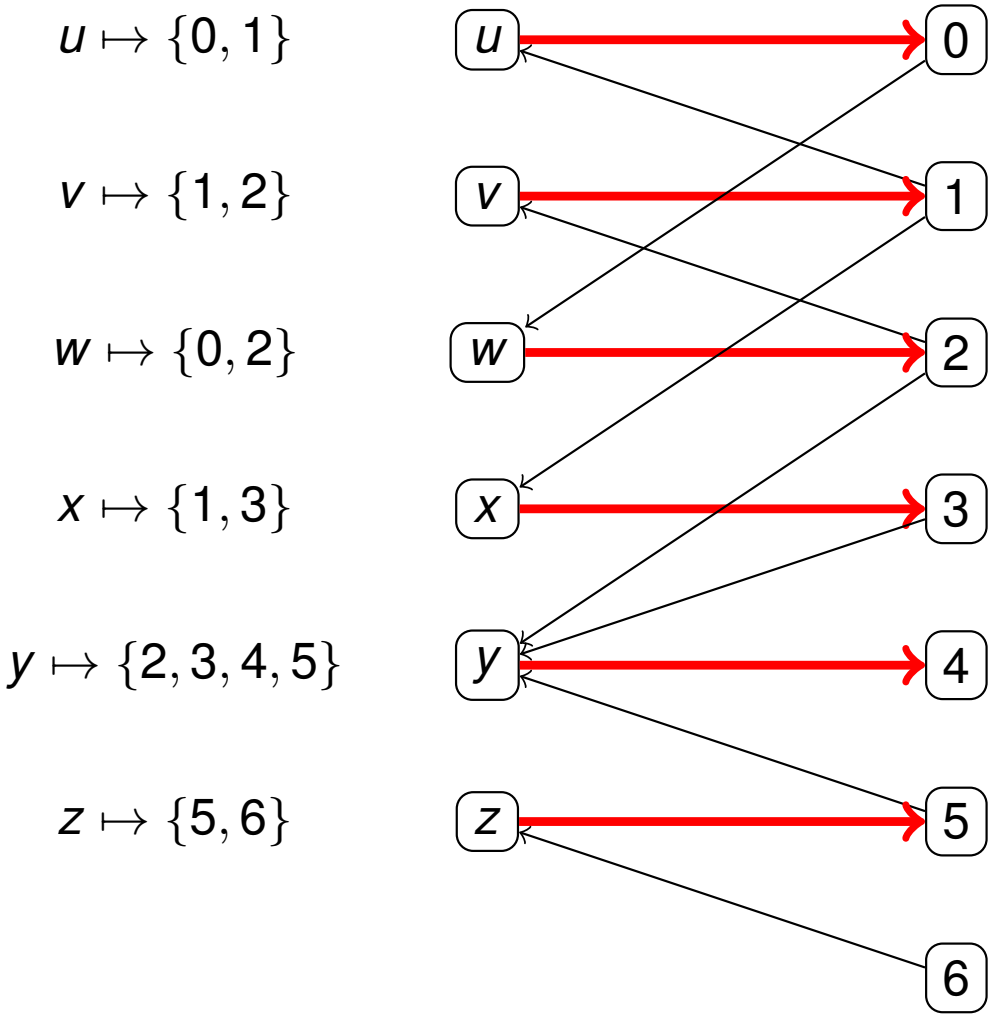
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Start from a maximum matching, and orient all edges: if in matching, then from variable to value, else the other way.



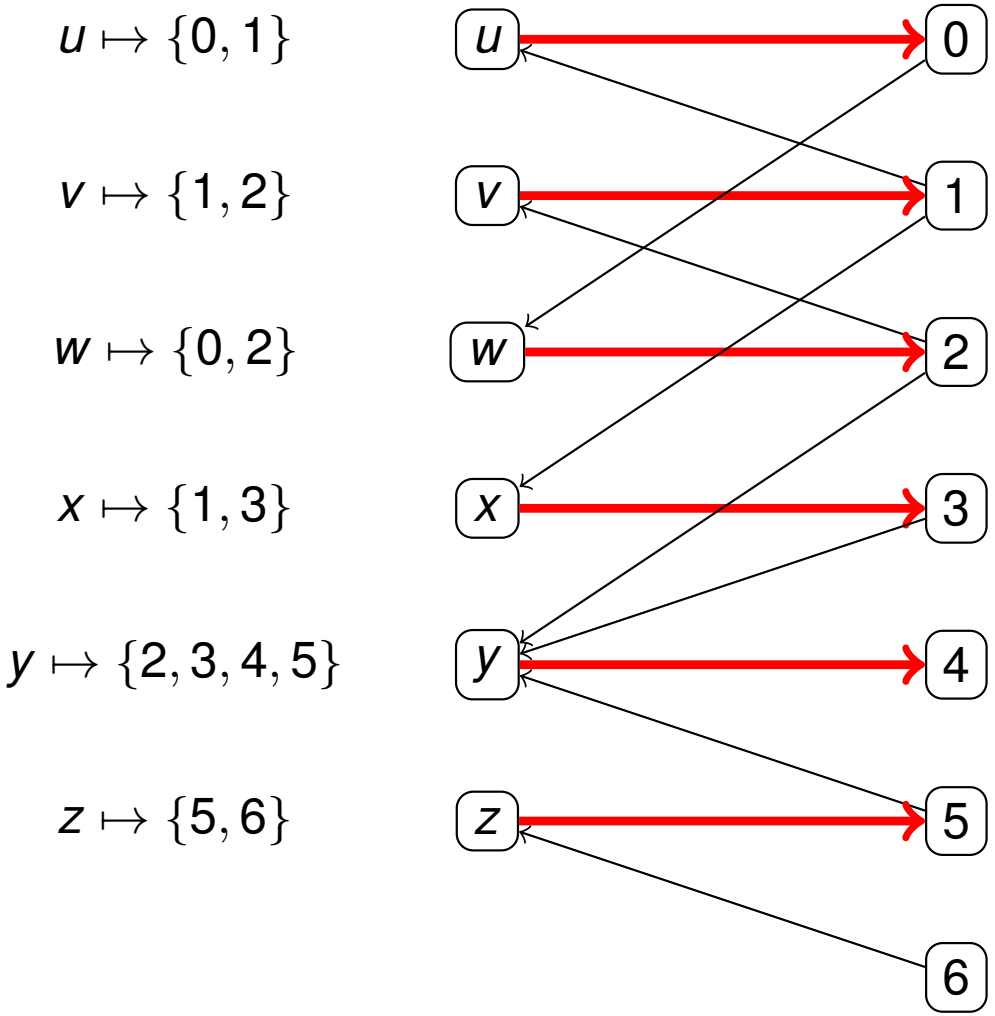
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Start from a maximum matching, and orient all edges: if in matching, then from variable to value, else the other way.



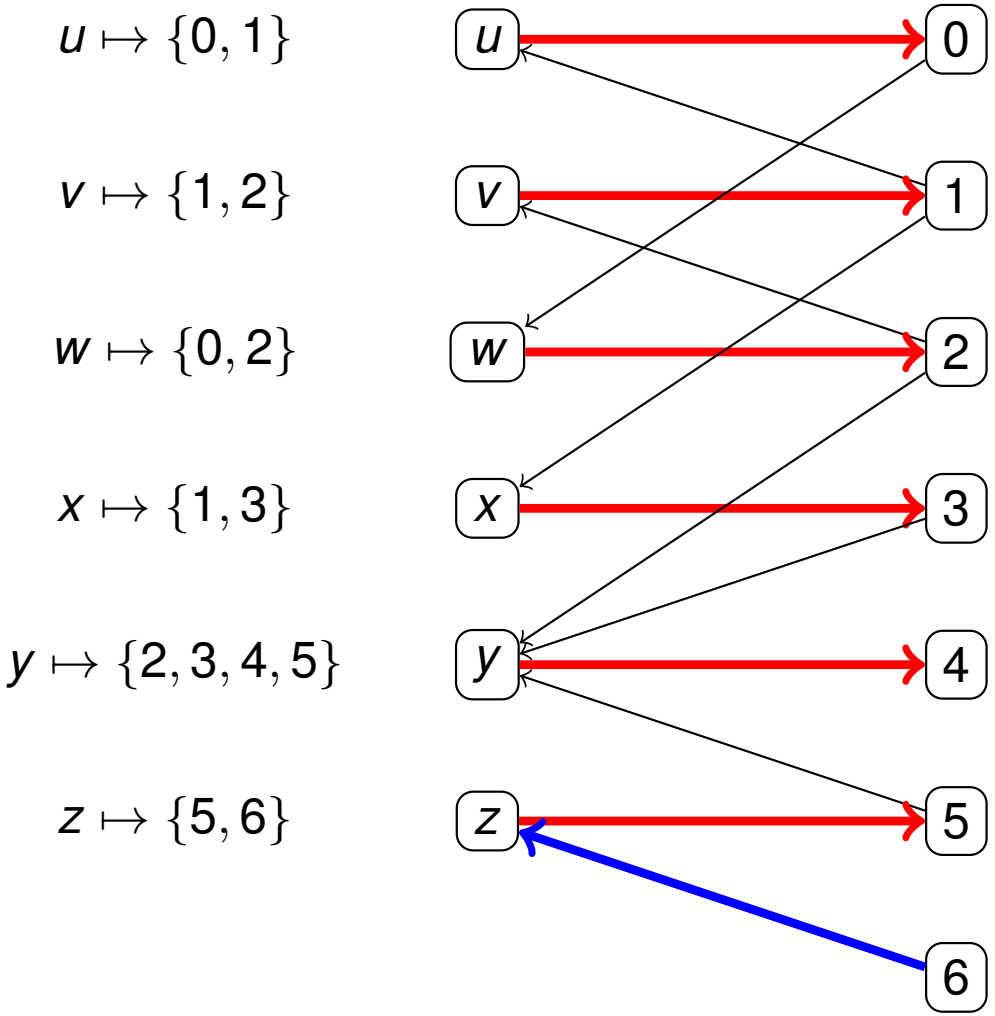
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Start from all unmatched vertices (necessarily values here) and mark all arcs on all simple paths: arcs can be flipped.



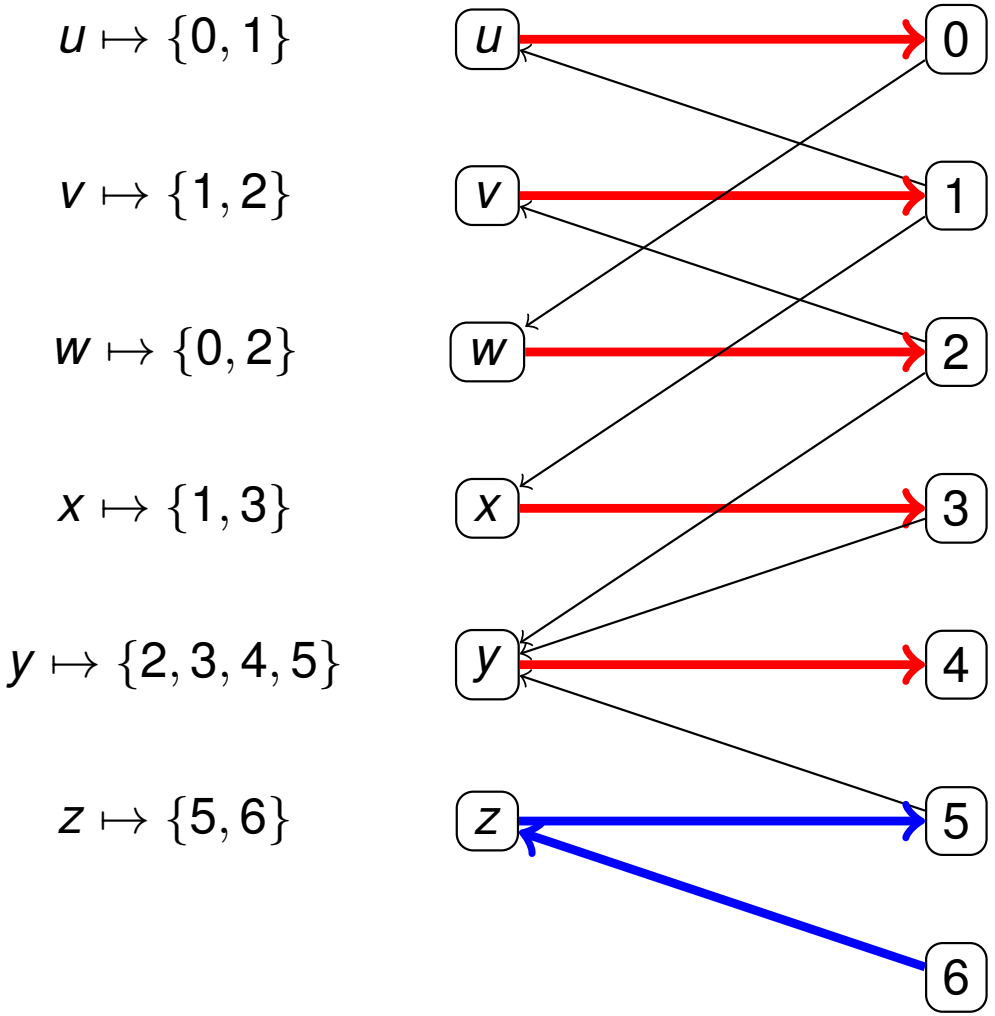
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Start from all unmatched vertices (necessarily values here) and mark all arcs on all simple paths: arcs can be flipped.



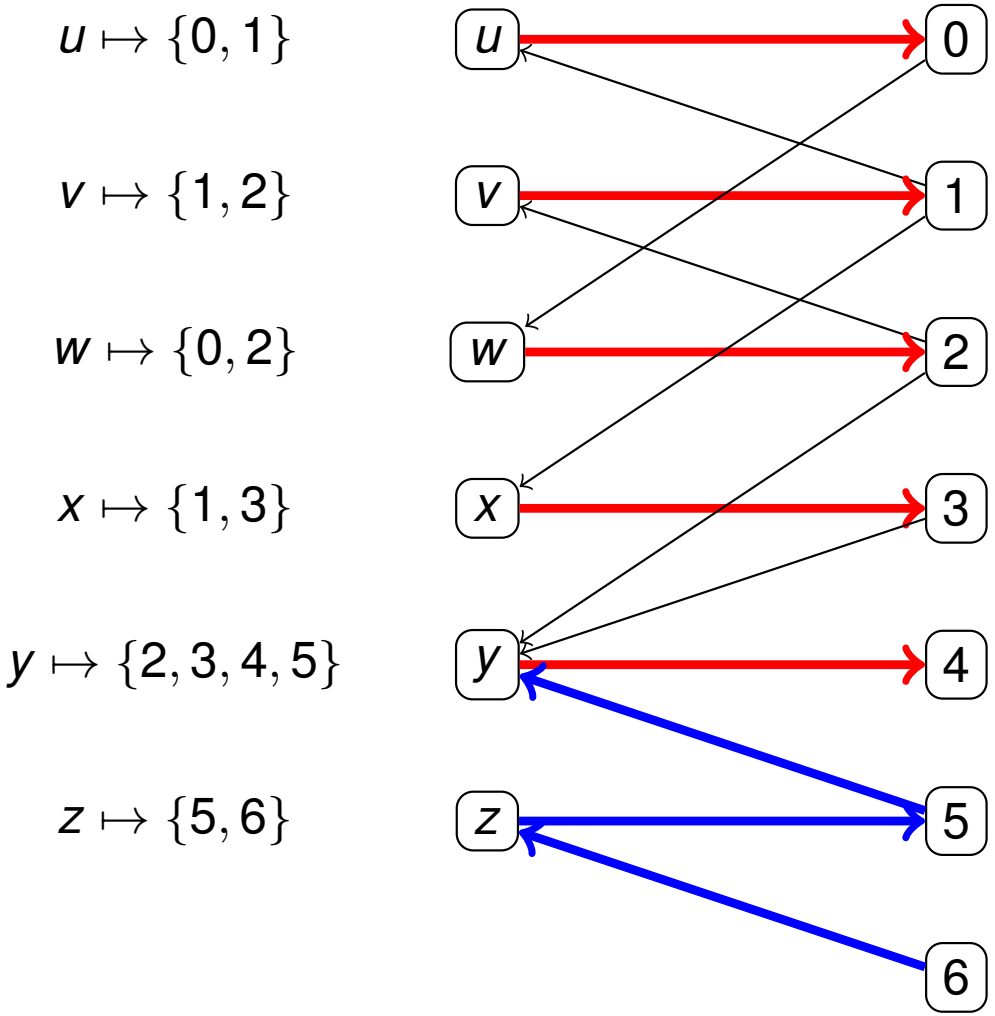
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Start from all unmatched vertices (necessarily values here) and mark all arcs on all simple paths: arcs can be flipped.



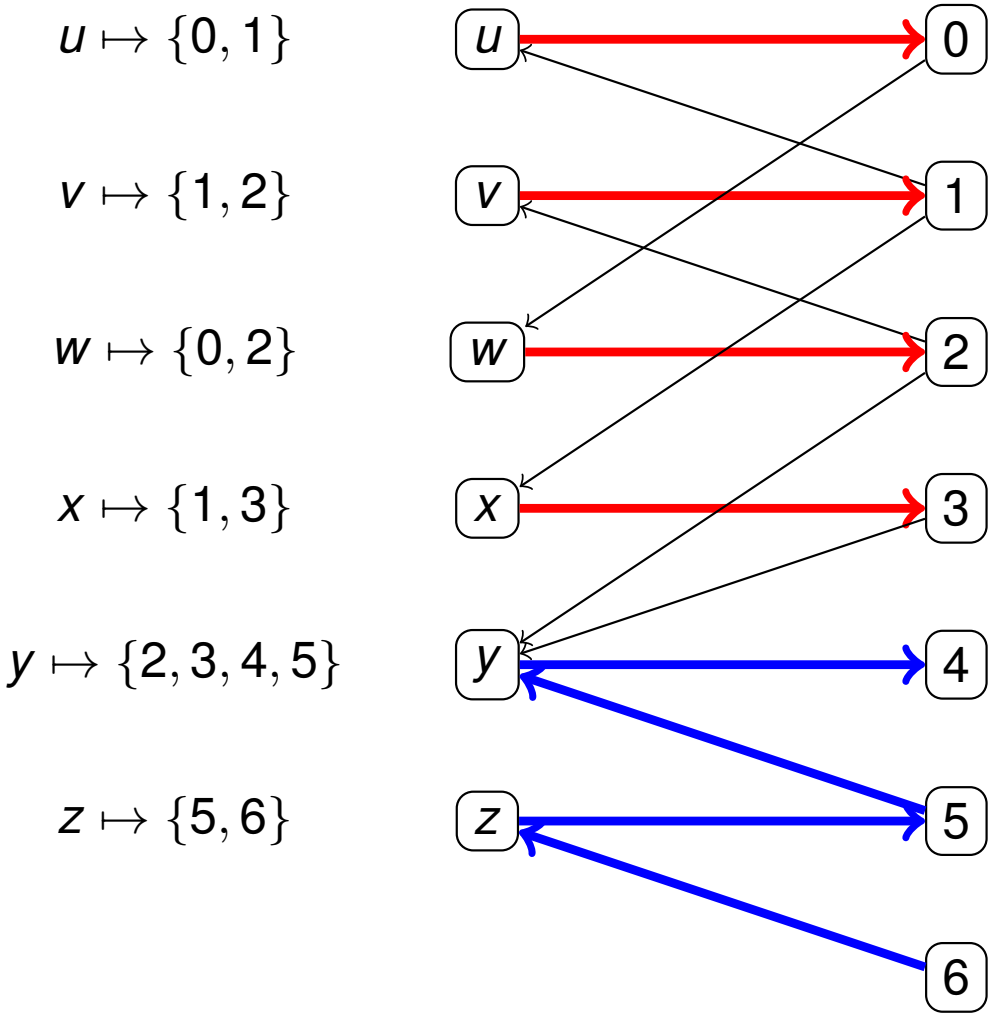
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Start from all unmatched vertices (necessarily values here) and mark all arcs on all simple paths: arcs can be flipped.



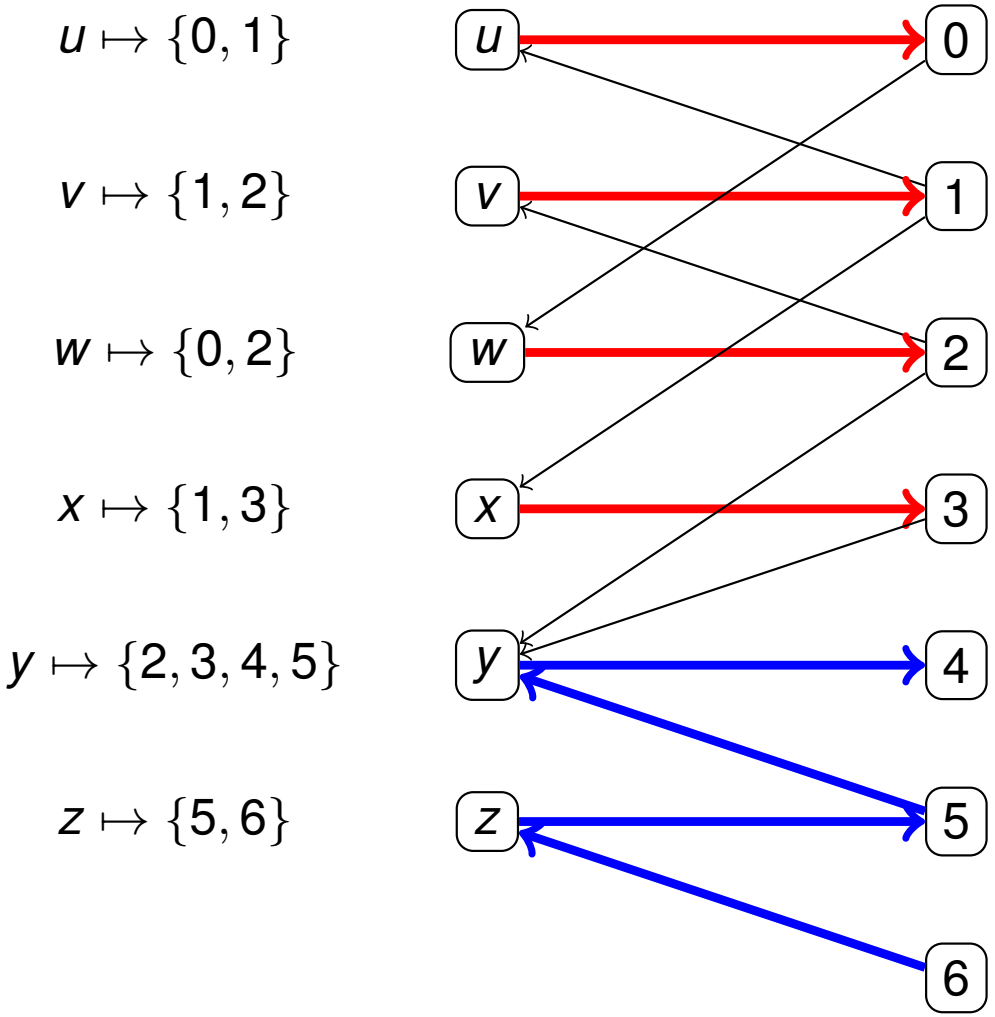
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Start from all unmatched vertices (necessarily values here) and mark all arcs on all simple paths: arcs can be flipped.



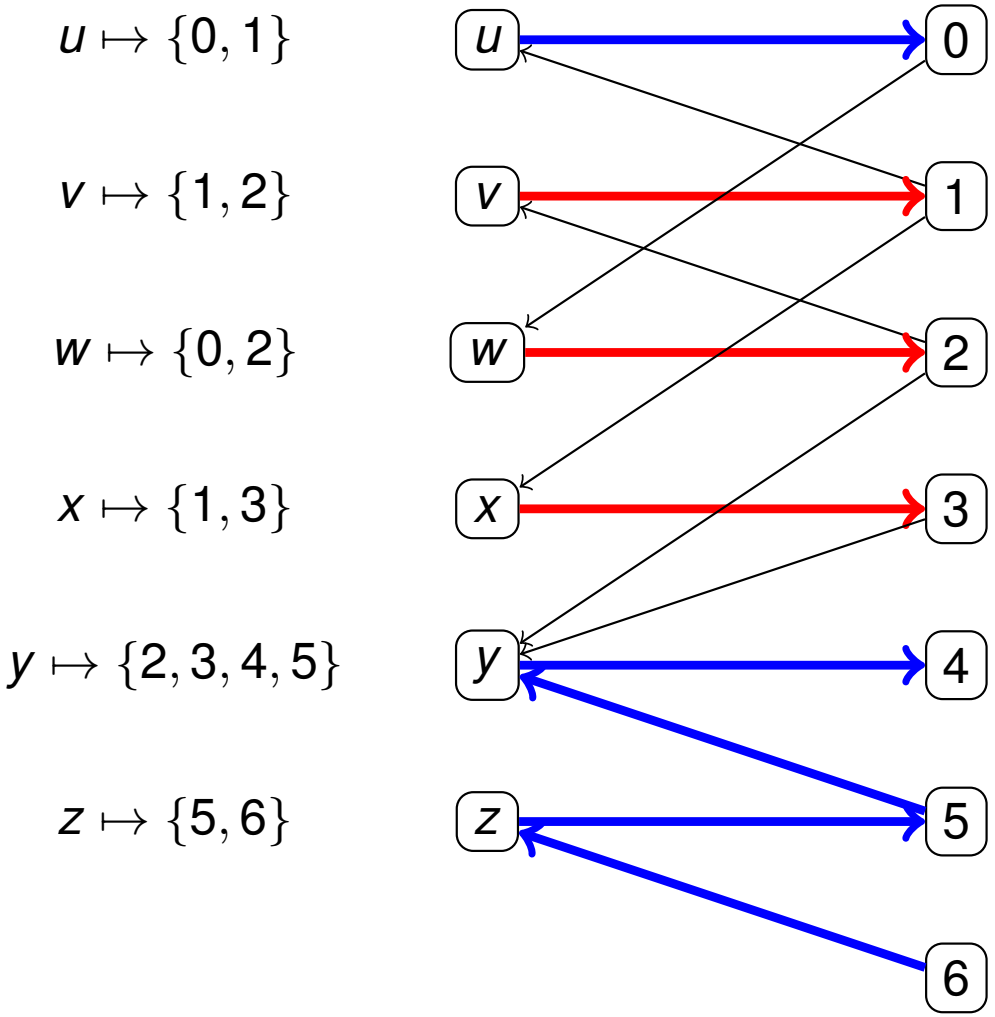
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Mark all arcs in all strongly connected components (SCCs): the variables of an SCC take all the values of that SCC.



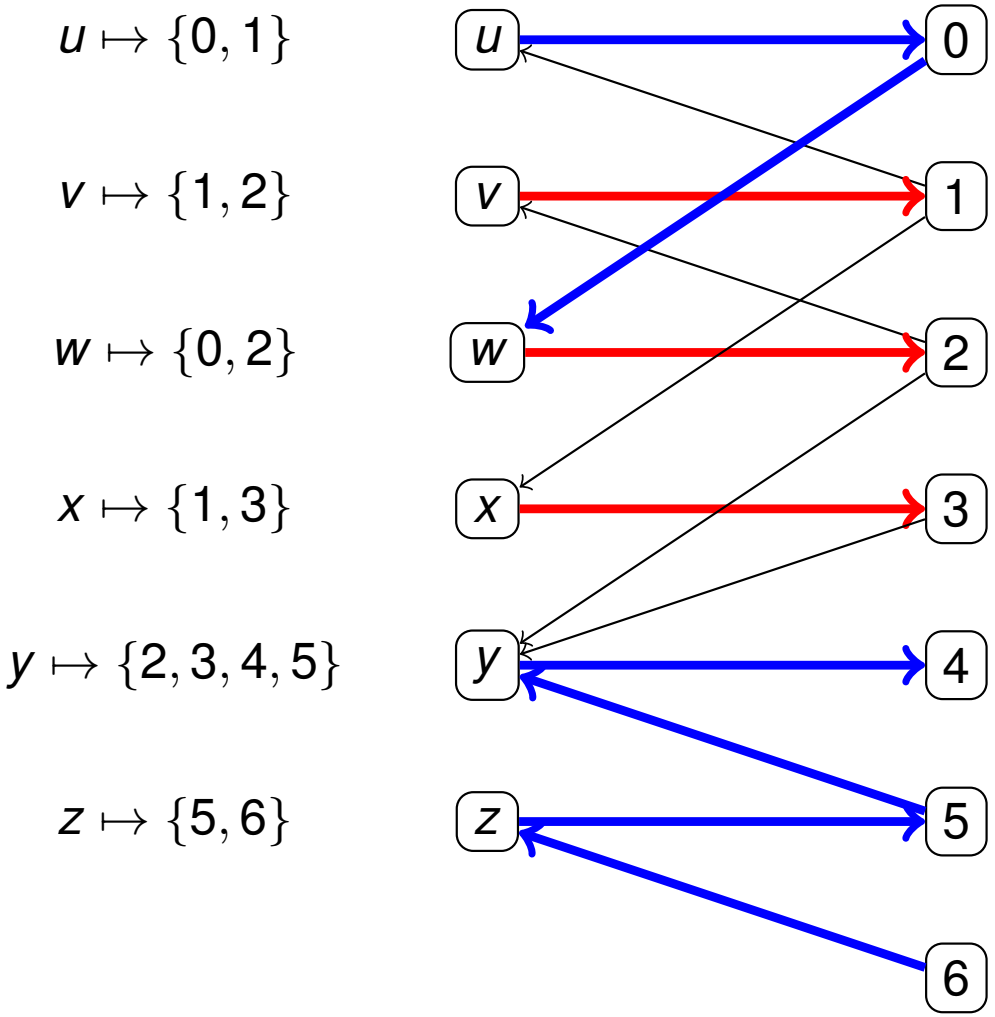
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Mark all arcs in all strongly connected components (SCCs): the variables of an SCC take all the values of that SCC.



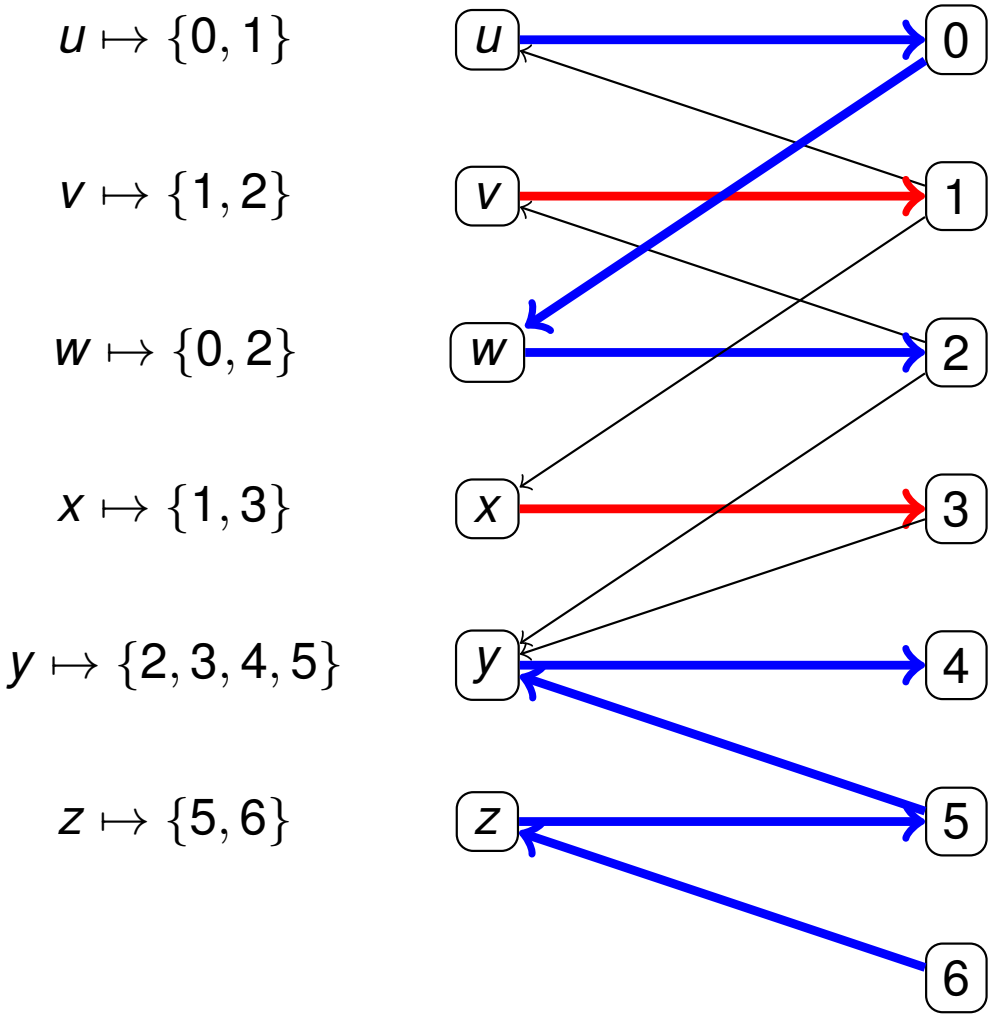
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Mark all arcs in all strongly connected components (SCCs): the variables of an SCC take all the values of that SCC.



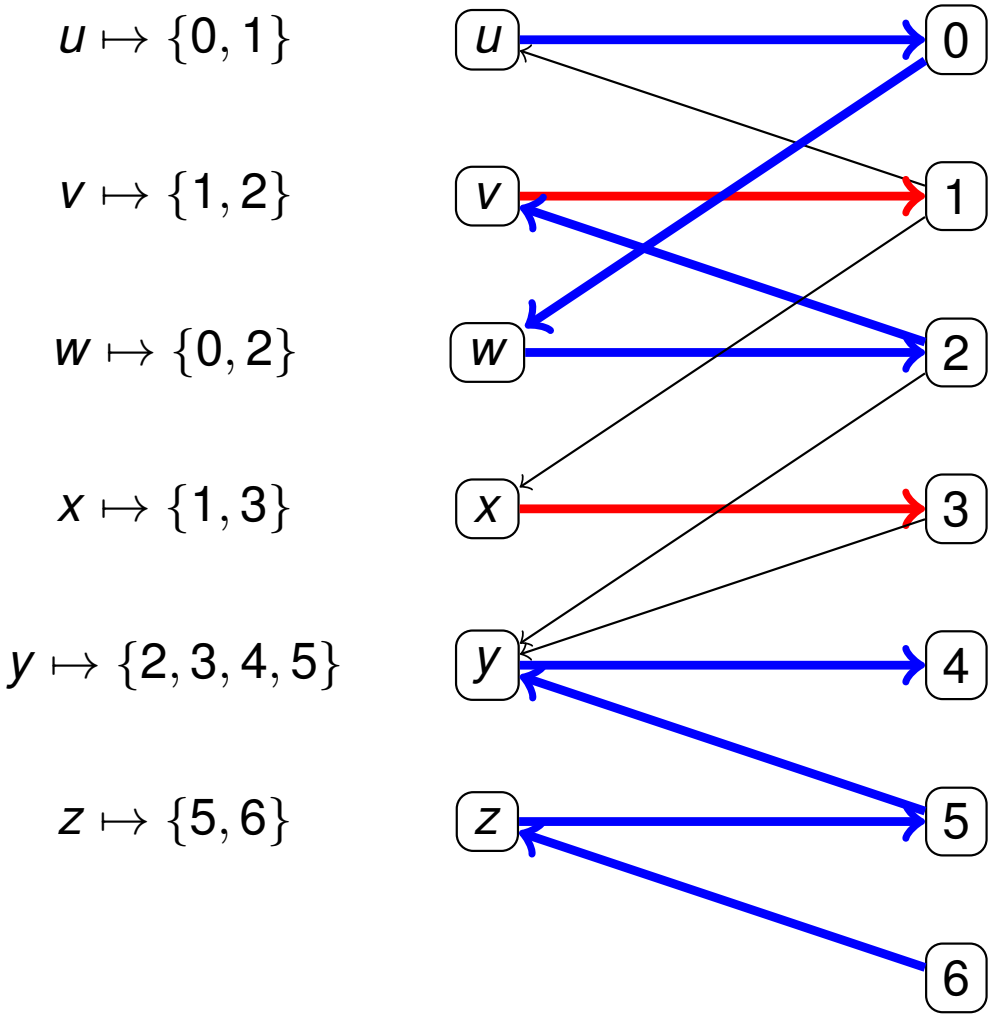
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Mark all arcs in all strongly connected components (SCCs): the variables of an SCC take all the values of that SCC.



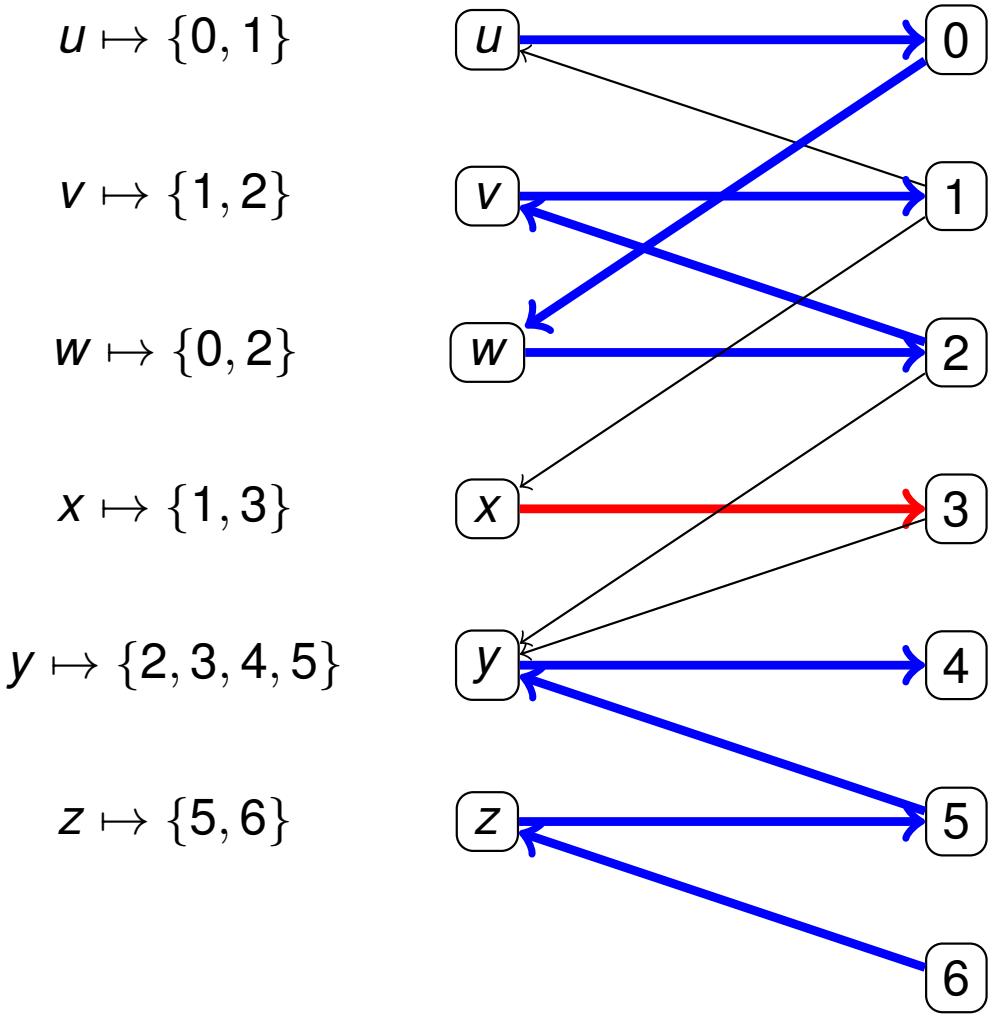
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Mark all arcs in all strongly connected components (SCCs): the variables of an SCC take all the values of that SCC.



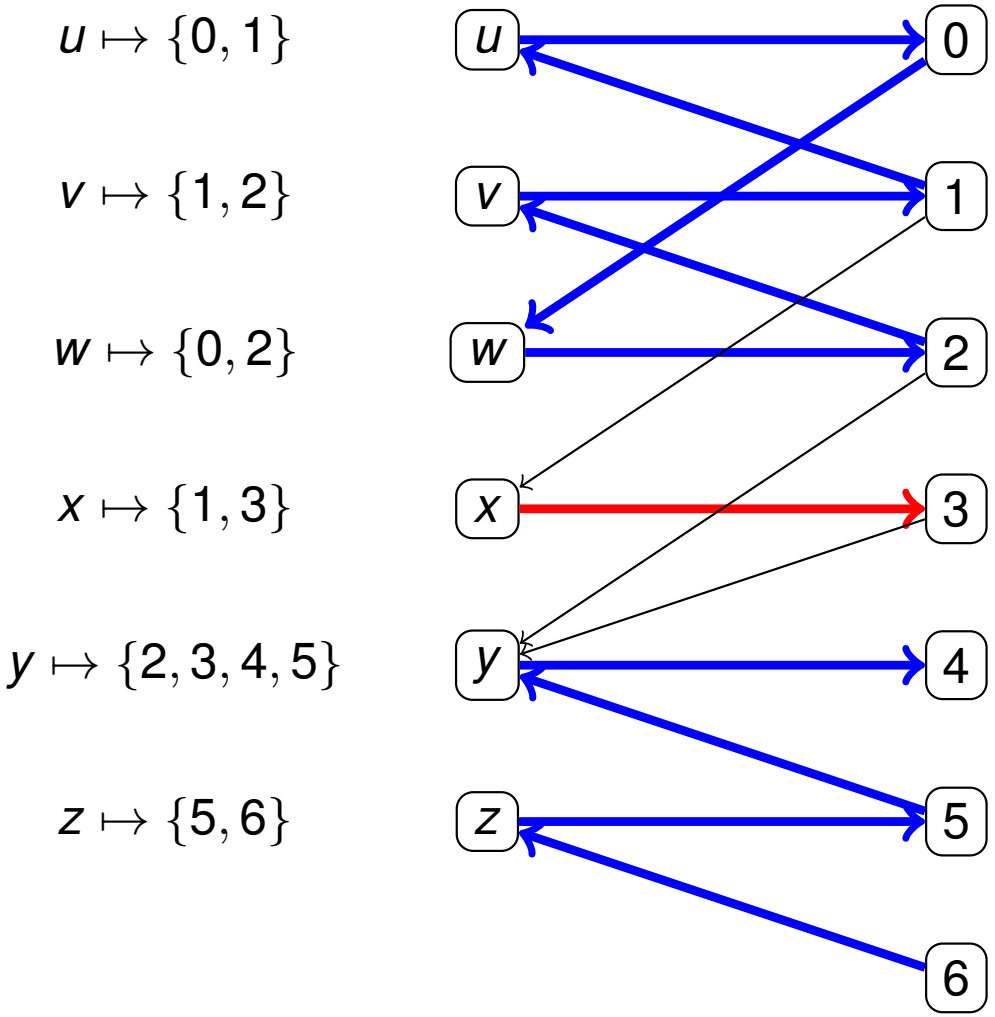
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Mark all arcs in all strongly connected components (SCCs): the variables of an SCC take all the values of that SCC.



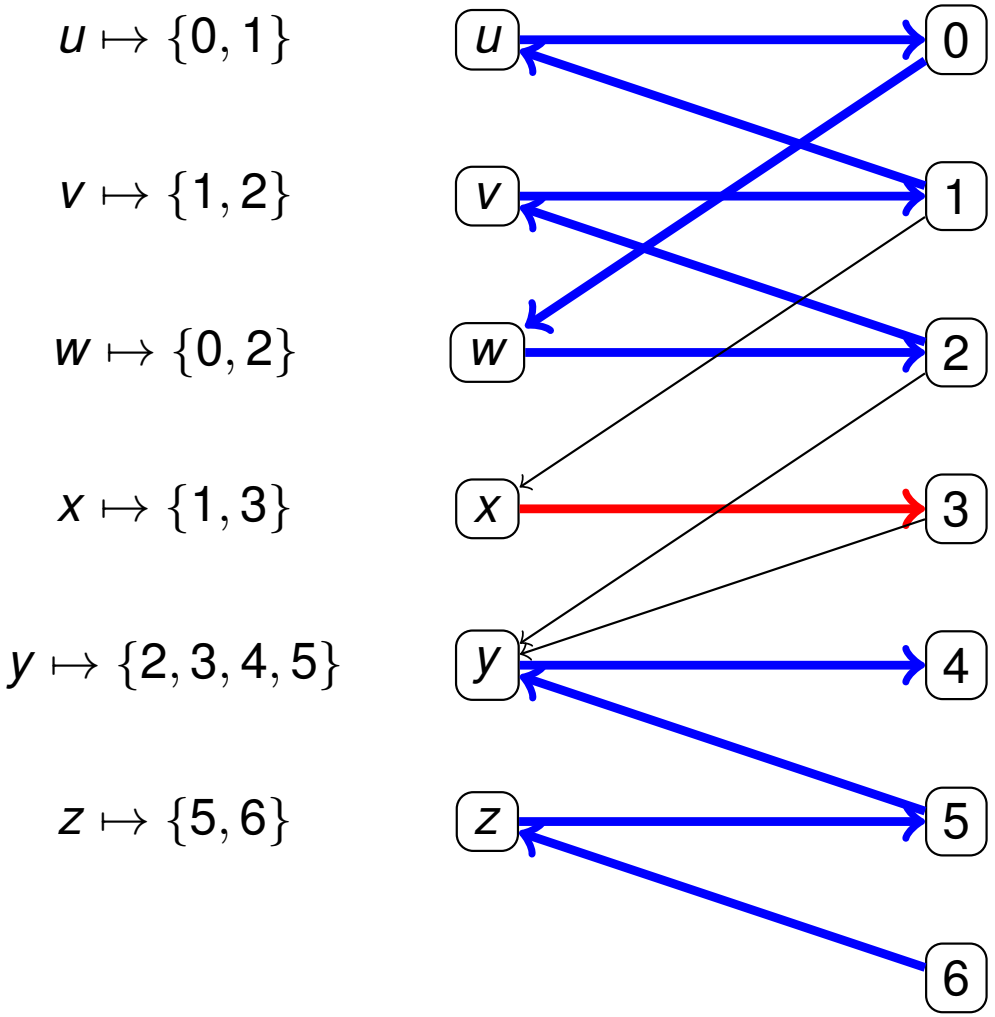
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Mark all arcs in all strongly connected components (SCCs): the variables of an SCC take all the values of that SCC.



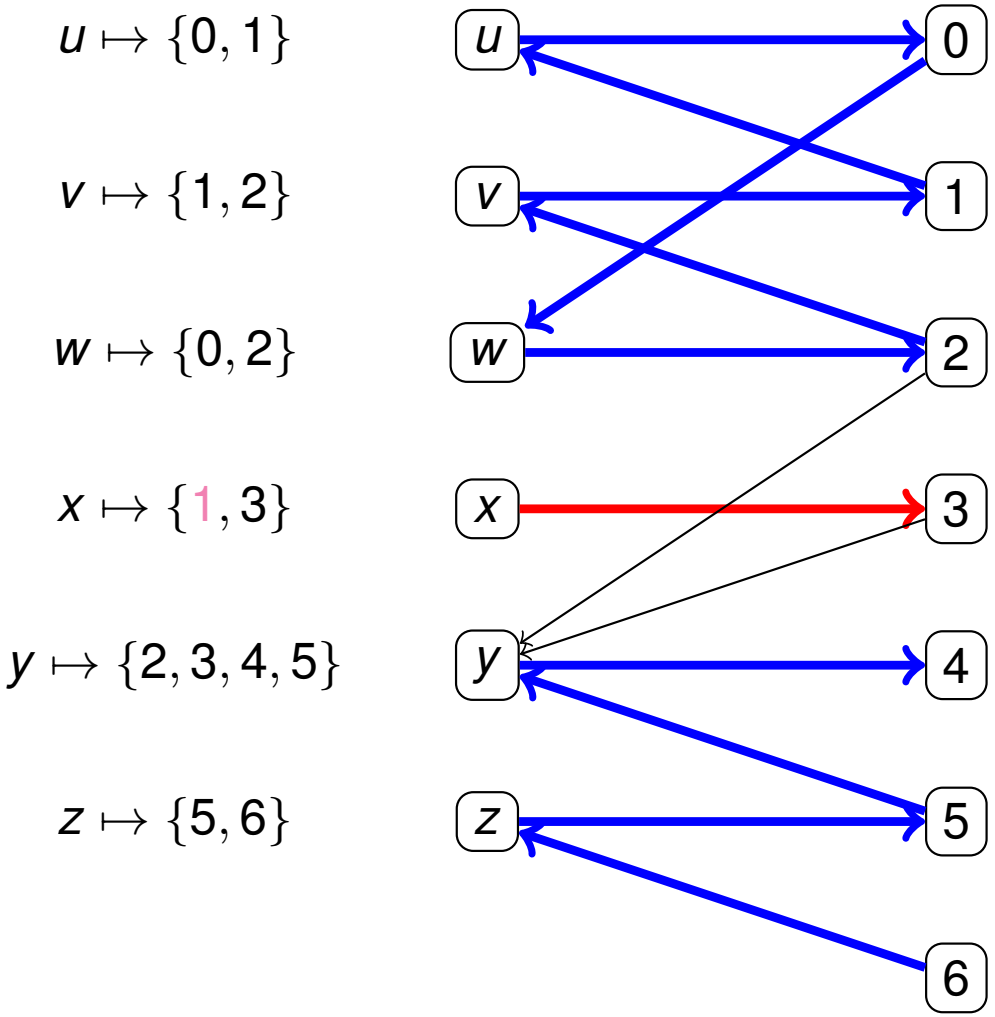
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Every arc that is neither in the chosen maximum matching nor marked is in *no* maximum matching: prune accordingly.



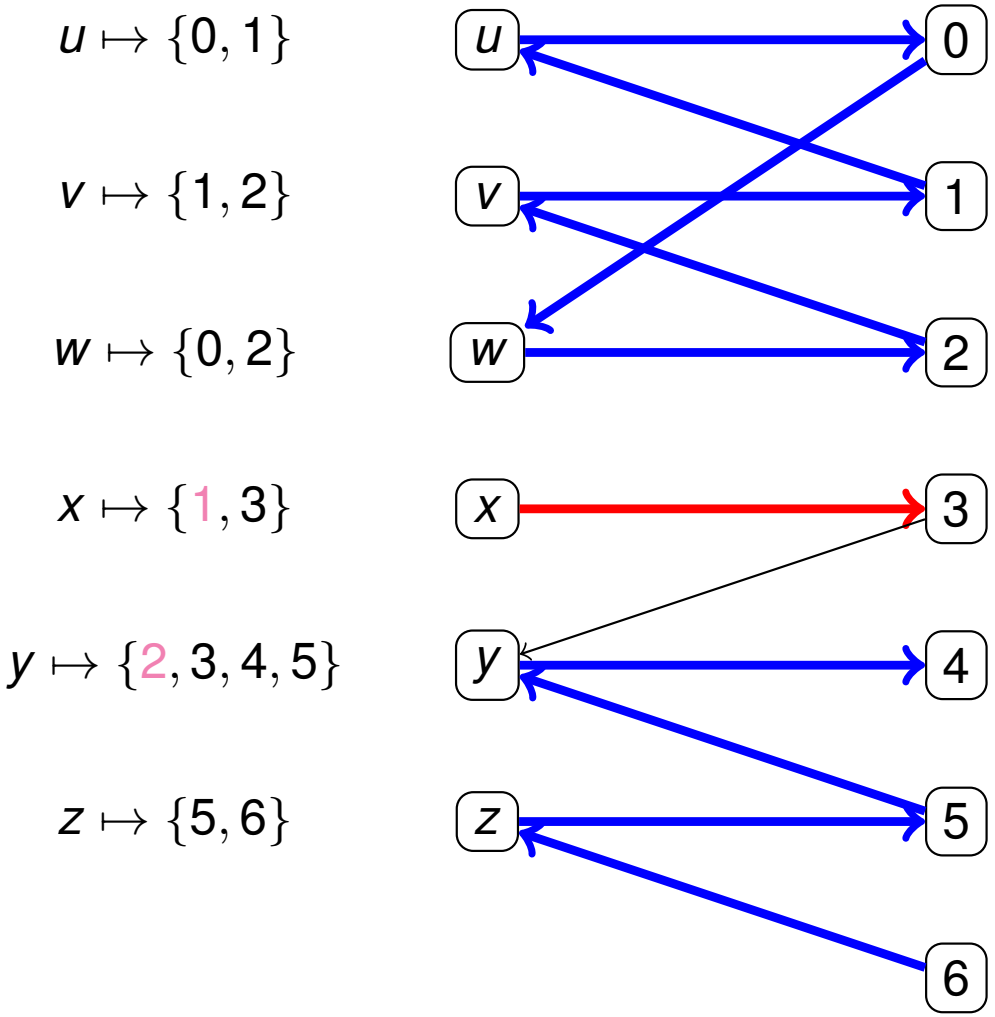
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Every arc that is neither in the chosen maximum matching nor marked is in *no* maximum matching: prune accordingly.



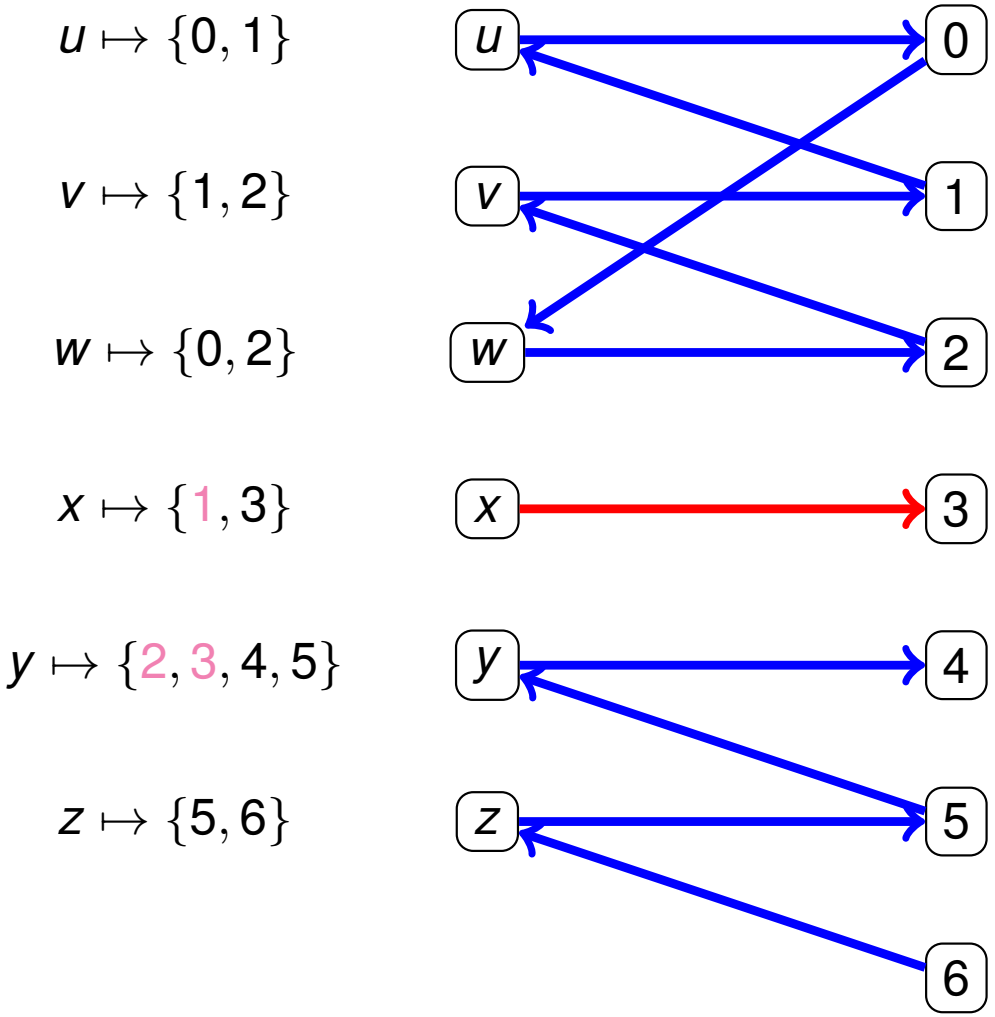
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Every arc that is neither in the chosen maximum matching nor marked is in *no* maximum matching: prune accordingly.



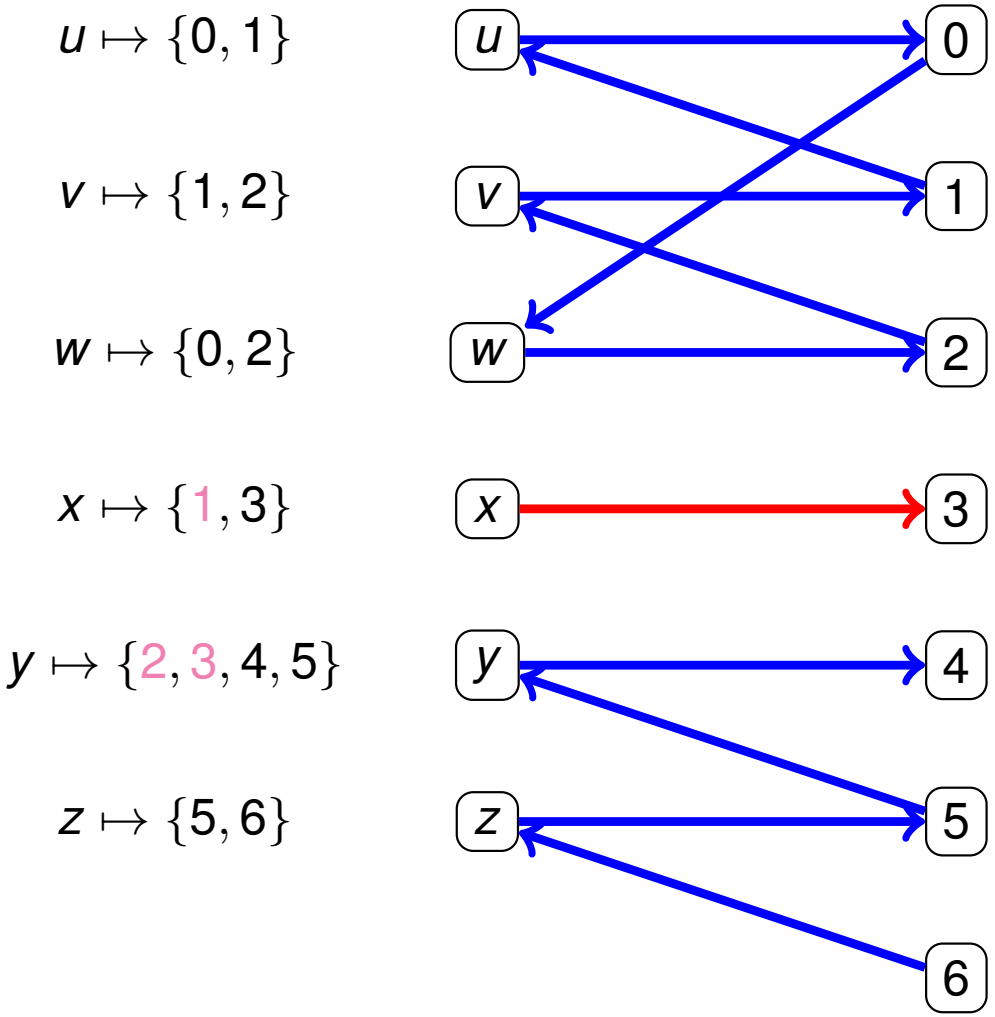
Efficient DC propagator (Régin, 1994) (Costa, 1994):

Every arc that is neither in the chosen maximum matching nor marked is in *no* maximum matching: prune accordingly.



Efficient DC propagator (Régin, 1994) (Costa, 1994):

Every arc that is in the chosen maximum matching but not marked is in *every* maximum matching: fixed variable.



Underlying Theorem from Matching Theory

Prelude
The DISTINCT Predicate
Naïve DC Propagation
Efficient DC Propagation
Efficient BC Propagation
Bibliography

Theorem (Berge, 1970) (Petersen, 1891)

Edge e belongs to some maximum matching if and only if, for an arbitrarily chosen maximum matching M:

- e belongs to a path of an even number of edges that starts at some node that is not incident to an edge of M and that alternates between edges in M and edges not in M.

or

- e belongs to a cycle of an even number of edges that alternates between edges in M and edges not in M (that is, the arc corresponding to e belongs to an SCC).

Complexity and Incrementality

Complexity:

The described DC propagator takes

$$\mathcal{O}\left(m \cdot \sqrt{n}\right) \text{ time and } \mathcal{O}\left(m \cdot n\right) \text{ space}$$

for n decision variables and $m \geq n$ values in their domains.

Incrementality via stateful propagator:

Keep the variable-value graph between invocations.
When the propagator is re-invoked:

1. Remove marks on arcs.
2. Remove arcs that no longer correspond to the store.
3. If an arc of the old maximum matching was removed, then first compute a new maximum matching.
4. Mark and prune.

UPPSALA UNIVERSITET

Outline

1 Prelude

2 The DISTINCT Predicate

3 Naïve DC Propagation

4 Efficient DC Propagation

5 **Efficient BC Propagation**

6 Bibliography

Is BC Needed for DISTINCT?

Propagation to BC often suffices for DISTINCT.

Example

Propagation to BC suffices to infer unsatisfiability for DISTINCT($\{x, y, z\}$) from the store $\{x, y, z \mapsto \{4, 5\}\}$.

Efficient BC propagators:

There are BC propagators that take $\mathcal{O}(n \cdot \lg n)$ time:

- Puget @ AAAI 1998
- Mehlhorn and Thiel @ CP 2000
- López-Ortiz, Quimper, Tromp, van Beek @ IJCAI 2003

The latter two run in $\mathcal{O}(n)$ time if sorting can be avoided, say when there are as many values as decision variables.

UPPSALA UNIVERSITET

Outline

1 Prelude

2 The DISTINCT Predicate

3 Naïve DC Propagation

4 Efficient DC Propagation

5 Efficient BC Propagation

6 **Bibliography**

UPPSALA UNIVERSITET

Prelude
The DISTINCT Predicate
Naïve DC Propagation
Efficient DC Propagation
Efficient BC Propagation
Bibliography


Régin, Jean-Charles.
A filtering algo. for constraints of difference in CSPs.
AAAI 1994, pages 362–367, 1994.


Costa, Marie-Christine.
Persistency in maximum cardinality bipartite matchings.
Operations Research Letters, 15(3):143–149, 1994.


Berge, Claude.
Graphes et Hypergraphes. Dunod, 1970.


Petersen, Julius.
Die Theorie der regulären graphs.
Acta Mathematica, 15(1):193–220, 1891.


van Hoeve, Willem-Jan.
The Alldifferent Constraint: A Survey.
Extended from the version in *ERCIM Workshop 2001*.