Optimal routing in deterministic delay-tolerant networks

Ronan Bocquillon, Antoine Jouglet

Sorbonne Universités, Université de Technologie de Compiègne,
CNRS, Heudiasyc UMR 7253

Journées Franciliennes de Recherche Opérationnelle
23/09/2015

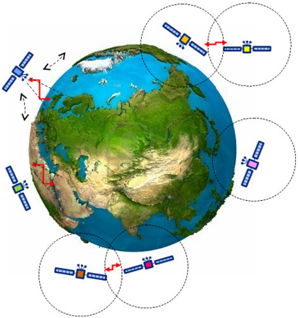
Guidelines

- Introduction
- Dominance rules
- Constraint programming
- Computational results
- Conclusion

Guidelines

- Introduction
  - Store-forward routing
  - Example
  - Formal description
  - Complexity
- Dominance rules
- Constraint programming
- Computational results
- Conclusion

Store-forward routing

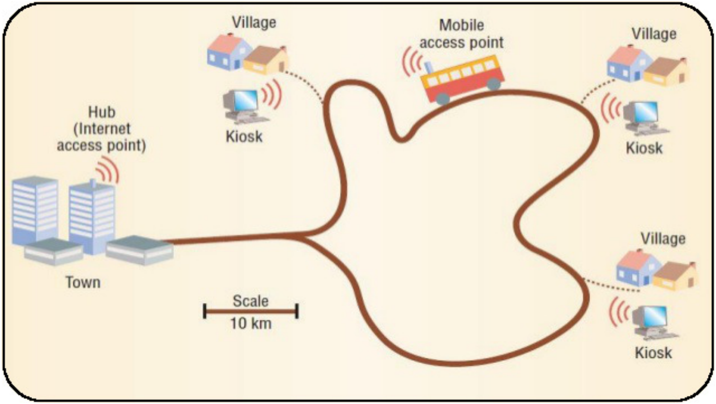


Routing through a deterministic delay-tolerant networks

Motivations

Making use of knowledge about node mobility (and possibilities of collaboration) to efficiently route information from the source nodes to a set of recipient nodes within a given time horizon.

DakNet project



DakNet – a Road To Universal Broadband Connectivity

Formal description

In this work, we consider only one datum (split into several identified datum units) to be delivered to all recipient nodes. A fixed amount of data can be transmitted during each contact. Formally, we consider:

- a set $\mathcal{N} = \{1, 2, \dots, n\}$ of n nodes,
- a datum $\mathcal{D} = \{1, 2, \dots, u\}$ of u datum units,
- each nodes $i \in \mathcal{N}$ stores a subset $\mathcal{O}_i \subseteq \mathcal{D}$ of datum units at the outset,
- a subset $\mathcal{R} \subseteq \mathcal{N}$ of recipients (these must recover all datum units),
- and a sequence $\sigma = \{\sigma_1, \sigma_2, \dots, \sigma_m\}$ of m contacts. During each contact $\sigma_c = (s_c, r_c) \in \mathcal{N}^2$, at most one datum unit can be transmitted from the sending node s_c to the receiving node r_c.

Objective

Finding a valid *transfer plan*, *i.e.* a function that indicates how to route the datum units from their sources to all recipient nodes.

Formal description

In this work, we consider only one datum (split into several identified datum units) to be delivered to all recipient nodes. A fixed amount of data can be transmitted during each contact. Formally, we consider:

- a set $\mathcal{N} = \{1, 2, \ldots, n\}$ of n nodes,
- a datum $\mathcal{D} = \{1, 2, \ldots, u\}$ of u datum units,
- each nodes $i \in \mathcal{N}$ stores a subset $\mathcal{O}_i \subseteq \mathcal{D}$ of datum units at the outset,
- a subset $\mathcal{R} \subseteq \mathcal{N}$ of recipients (these must recover all datum units),
- and a sequence $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ of m contacts. During each contact $\sigma_c = (s_c, r_c) \in \mathcal{N}^2$, at most one datum unit can be transmitted from the sending node s_c to the receiving node r_c.

Objective

Finding a valid *transfer plan*, *i.e.* a function that indicates how to route the datum units from their sources to all recipient nodes.

Formal description

In this work, we consider only one datum (split into several identified datum units) to be delivered to all recipient nodes. A fixed amount of data can be transmitted during each contact. Formally, we consider:

- a set $\mathcal{N} = \{1, 2, \ldots, n\}$ of n nodes,
- a datum $\mathcal{D} = \{1, 2, \ldots, u\}$ of u datum units,
- each nodes $i \in \mathcal{N}$ stores a subset $\mathcal{O}_i \subseteq \mathcal{D}$ of datum units at the outset,
- a subset $\mathcal{R} \subseteq \mathcal{N}$ of recipients (these must recover all datum units),
- and a sequence $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ of m contacts. During each contact $\sigma_c = (s_c, r_c) \in \mathcal{N}^2$, at most one datum unit can be transmitted from the sending node s_c to the receiving node r_c.

Objective

Finding a valid *transfer plan*, *i.e.* a function that indicates how to route the datum units from their sources to all recipient nodes.

Formal description

In this work, we consider only one datum (split into several identified datum units) to be delivered to all recipient nodes. A fixed amount of data can be transmitted during each contact. Formally, we consider:

- a set $\mathcal{N} = \{1, 2, \ldots, n\}$ of n nodes,
- a datum $\mathcal{D} = \{1, 2, \ldots, u\}$ of u datum units,
- each nodes $i \in \mathcal{N}$ stores a subset $\mathcal{O}_i \subseteq \mathcal{D}$ of datum units at the outset,
- a subset $\mathcal{R} \subseteq \mathcal{N}$ of recipients (these must recover all datum units),
- and a sequence $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ of m contacts. During each contact $\sigma_c = (s_c, r_c) \in \mathcal{N}^2$, at most one datum unit can be transmitted from the sending node s_c to the receiving node r_c.

Objective

Finding a valid *transfer plan*, *i.e.* a function that indicates how to route the datum units from their sources to all recipient nodes.

Definition

A *transfer plan* is a function:

$$\phi : \{1, 2, \ldots, m\} \mapsto \{\emptyset, \{1\}, \{2\}, \ldots, \{u\}\}$$

where $\phi(c)$ designates the datum unit received by r_c during contact σ_c.

Definition

Given a transfer plan ϕ, we associate with each node $i \in \mathcal{N}$, a set of *states* O_i^t defined by: $O_i^0 = \mathcal{O}_i$; $\forall c \in \{1, 2, \ldots, m\}$

$$O_{r_c}^c = O_{r_c}^{c-1} \cup \phi(c) \text{ and } \forall i \in \mathcal{N} \setminus \{r_c\}, O_i^c = O_i^{c-1}$$

Definition

A transfer plan ϕ is said to be *valid* if every node always transfer a datum unit it possesses, *i.e.* $\forall c \in \{1, 2, \ldots, m\}, \phi(c) \in \{\emptyset\} \cup \{\{k\} \mid k \in O_{s_c}^{c-1}\}$.

Definition

A *transfer plan* is a function:

$$\phi : \{1, 2, \dots, m\} \mapsto \{\emptyset, \{1\}, \{2\}, \dots, \{u\}\}$$

where $\phi(c)$ designates the datum unit received by r_c during contact σ_c.

Definition

Given a transfer plan ϕ, we associate with each node $i \in \mathcal{N}$, a set of *states* O_i^t defined by: $O_i^0 = \mathcal{O}_i$; $\forall c \in \{1, 2, \dots, m\}$

$$O_{r_c}^c = O_{r_c}^{c-1} \cup \phi(c) \text{ and } \forall i \in \mathcal{N} \backslash \{r_c\}, O_i^c = O_i^{c-1}$$

Definition

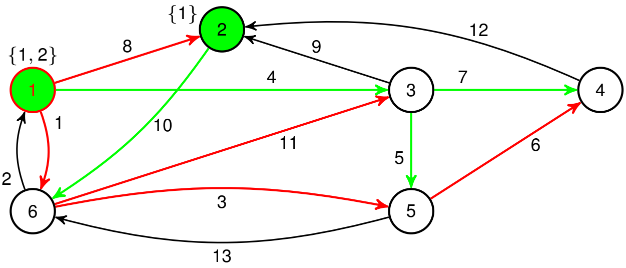
A transfer plan ϕ is said to be *valid* if every node always transfer a datum unit it possesses, *i.e.* $\forall c \in \{1, 2, \dots, m\}$, $\phi(c) \in \{\emptyset\} \cup \{\{k\} \mid k \in O_{s_c}^{c-1}\}$.

Objective function

Objective

Finding a transfer plan minimizing the dissemination length, *i.e.* the smallest index t at which every recipient is served. This problem is equivalent to find a set of arc-disjoint evolving branchings, whose roots are given by the source nodes, whose terminals are given by the recipient nodes, and such that the last transfer occurs at the earliest.

Evolving Graphs [Ferreira,2004]



$$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\} \; ; \mathcal{D} = \{1,2\} \; ; \mathcal{O}_1 = \{1, 2\} \; ; \mathcal{O}_2 = \{1\} \; ; \mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset \; ;$$
$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), (5, 4), \ldots, (5, 6)\}$$

Complexity

Theorem

The dissemination problem is strongly NP-Hard.

Theorem

The dissemination problem can be solved in polynomial time
if *$u = 1$ (only one datum unit) or* ***if*** *$|\mathcal{R}| = 1$ (only one recipient node).*

Complexity

Theorem

The dissemination problem is strongly NP-Hard.

Theorem

The dissemination problem can be solved in polynomial time ***if*** *$u = 1$ (only one datum unit) or* ***if*** *$|\mathcal{R}| = 1$ (only one recipient node).*

Guidelines

- Introduction
- Dominance rules
  - Minimal transfer plans
  - Strictly-active transfer plans
  - Example
- Constraint programming
- Computational results
- Conclusion

Minimal transfer plans

Definition

A transfer plan ϕ is *minimal* if every transfer $\phi(c)$, $c \in \{1, 2, \ldots, m\}$ is either *null* (*i.e.* $\phi(c) = \emptyset$) or *improving* (*i.e.* $O_{r_c}^{c-1} \subset O_{r_c}^{c}$).

Theorem

The set of *minimal* transfer plans is dominant.

Strictly-active transfer plans

Definition

A transfer plan ϕ is *strictly-active* if no transfer is null, while it could has been improving, *i.e.* $\forall c \in \{1, 2, \ldots, m\}$, if $\exists k \in \mathcal{D}$ such that $k \in O_{s_c}^{c-1}$ and $k \notin O_{r_c}^{c-1}$, then $\phi(c)$ is improving.

Theorem

The set of *strictly-active* transfer plans is dominant.

Theorem

The set of *minimal strictly-active* transfer plans is dominant.

Strictly-active transfer plans

Definition

A transfer plan ϕ is *strictly-active* if no transfer is null, while it could has been improving, *i.e.* $\forall c \in \{1, 2, \dots, m\}$, if $\exists k \in \mathcal{D}$ such that $k \in O_{s_c}^{c-1}$ and $k \notin O_{r_c}^{c-1}$, then $\phi(c)$ is improving.

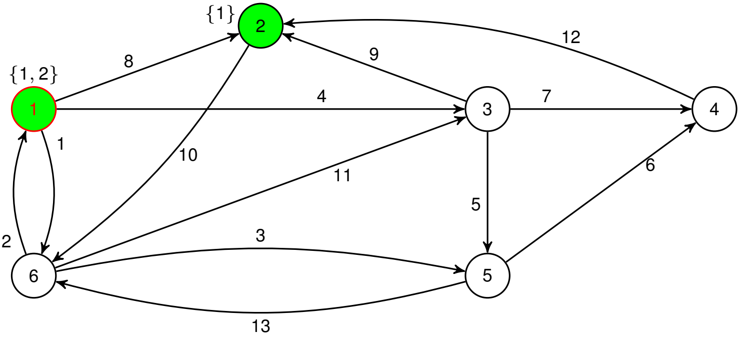
Theorem

The set of *strictly-active* transfer plans is dominant.

Theorem

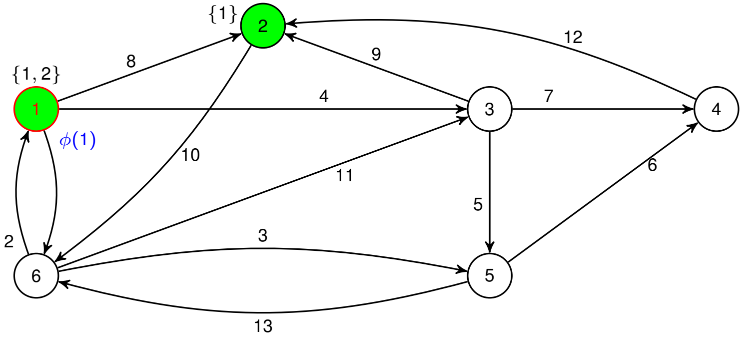
The set of *minimal strictly-active* transfer plans is dominant.

Example



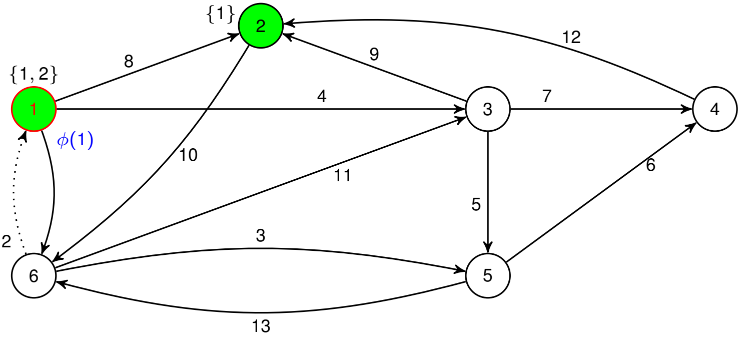
$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\}$; $\mathcal{D} = \{1{,}2\}$; $\mathcal{O}_1 = \{1, 2\}$; $\mathcal{O}_2 = \{1\}$; $\mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset$;
$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$

Example



$$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\} \,;\, \mathcal{D} = \{1,2\} \,;\, \mathcal{O}_1 = \{1, 2\} \,;\, \mathcal{O}_2 = \{1\} \,;\, \mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset \,;$$
$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$$
$$D(\phi(1)) = \{\{1\}, \{2\}\}$$

Example

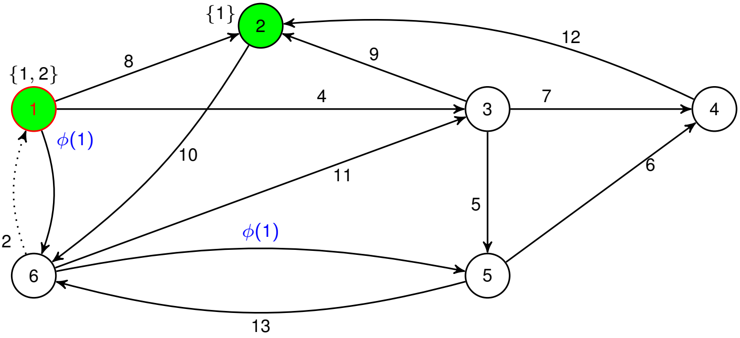


$$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\}\ ;\ \mathcal{D} = \{1, 2\}\ ;\ \mathcal{O}_1 = \{1, 2\}\ ;\ \mathcal{O}_2 = \{1\}\ ;\ \mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset\ ;$$

$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$$

$$D(\phi(1)) = \{\{1\}, \{2\}\}, \phi(2) = \emptyset$$

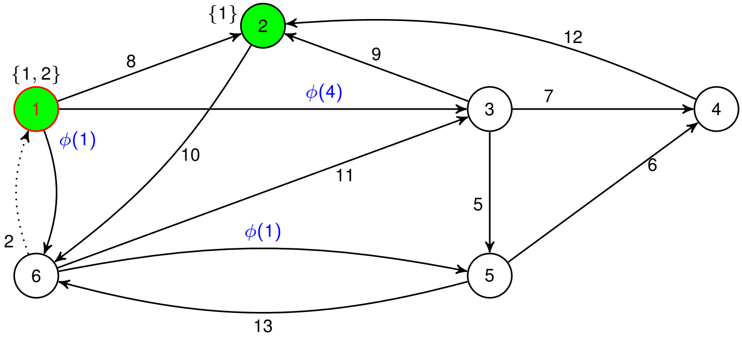
Example



$$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\} \; ; \; \mathcal{D} = \{1{,}2\} \; ; \; \mathcal{O}_1 = \{1, 2\} \; ; \; \mathcal{O}_2 = \{1\} \; ; \; \mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset \; ;$$
$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$$

$$D(\phi(1)) = \{\{1\}, \{2\}\},$$
$$\phi(2) = \emptyset, \; \phi(1) = \phi(3)$$

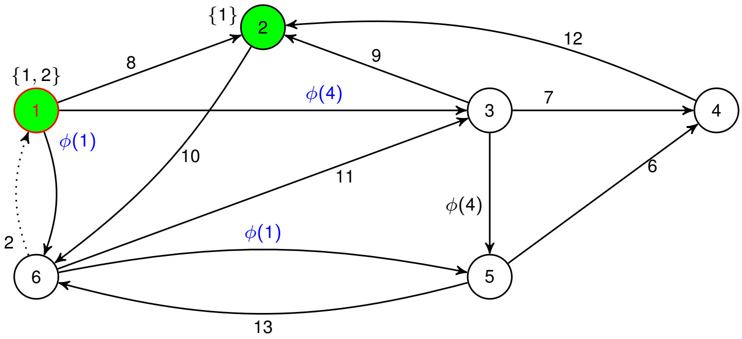
Example



$$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\}\ ; \mathcal{D} = \{1,2\}\ ; \mathcal{O}_1 = \{1, 2\}\ ; \mathcal{O}_2 = \{1\}\ ; \mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset\ ;$$
$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$$

$$D(\phi(1)) = D(\phi(4)) = \{\{1\}, \{2\}\},$$
$$\phi(2) = \emptyset, \phi(1) = \phi(3)$$

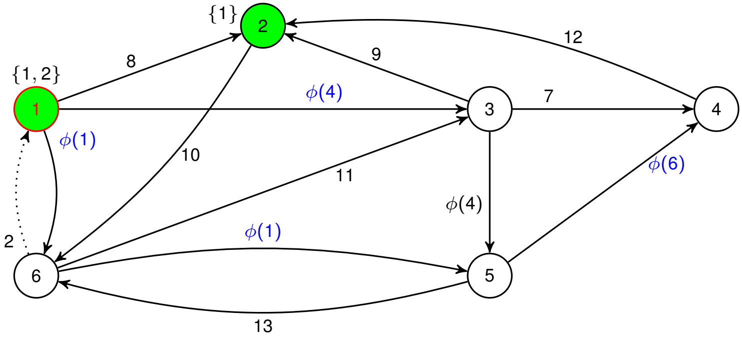
Example



$$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\}\ ;\ \mathcal{D} = \{1{,}2\}\ ;\ \mathcal{O}_1 = \{1, 2\}\ ;\ \mathcal{O}_2 = \{1\}\ ;\ \mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset\ ;$$
$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$$

$$D(\phi(1)) = D(\phi(4)) = \{\{1\}, \{2\}\}, \phi(2) = \emptyset, \phi(1) = \phi(3),$$
$$D(\phi(5)) = \{\phi(4), \emptyset\}$$

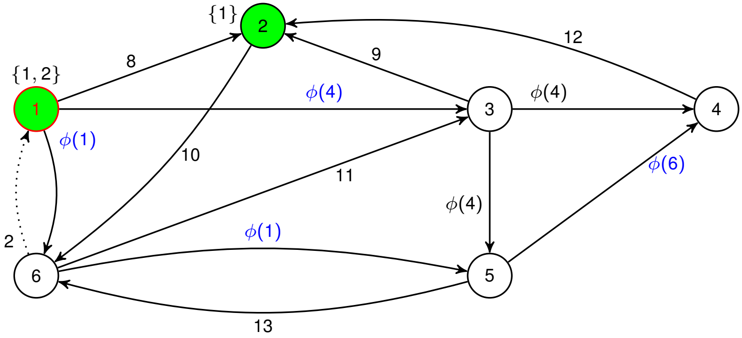
Example



$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\}$; $\mathcal{D} = \{1, 2\}$; $\mathcal{O}_1 = \{1, 2\}$; $\mathcal{O}_2 = \{1\}$; $\mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset$;
$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$

$D(\phi(1)) = D(\phi(4)) = \{\{1\}, \{2\}\}$, $\phi(2) = \emptyset$, $\phi(1) = \phi(3)$,
$D(\phi(5)) = \{\phi(4), \emptyset\}$, $D(\phi(6)) = \{\phi(1), \phi(4)\}$

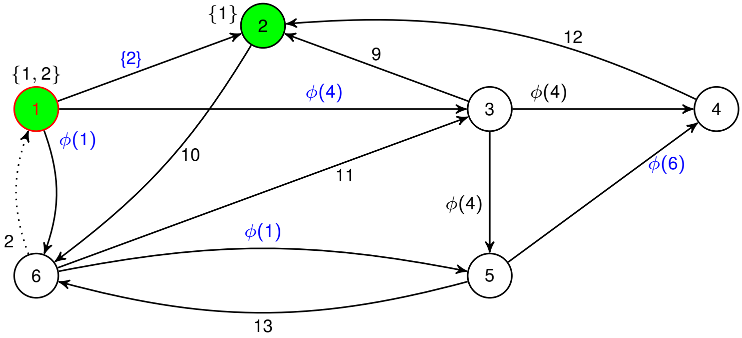
Example



$$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\} \,;\, \mathcal{D} = \{1,2\} \,;\, \mathcal{O}_1 = \{1, 2\} \,;\, \mathcal{O}_2 = \{1\} \,;\, \mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset \,;$$
$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$$

$$D(\phi(1)) = D(\phi(4)) = \{\{1\}, \{2\}\}, \phi(2) = \emptyset, \phi(1) = \phi(3),$$
$$D(\phi(5)) = \{\phi(4), \emptyset\}, D(\phi(6)) = \{\phi(1), \phi(4)\}, D(\phi(7)) = \{\phi(4), \emptyset\}$$

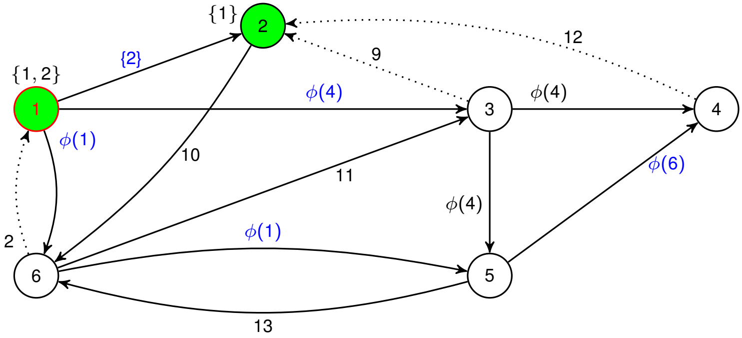
Example



$$\mathcal{N} = \mathcal{R} = \{1, 2, \dots, 6\} \,;\, \mathcal{D} = \{1,2\} \,;\, \mathcal{O}_1 = \{1, 2\} \,;\, \mathcal{O}_2 = \{1\} \,;\, \mathcal{O}_3 = \dots = \mathcal{O}_6 = \emptyset \,;$$
$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \dots, (5, 6)\}$$

$$D(\phi(1)) = D(\phi(4)) = \{\{1\}, \{2\}\}, \phi(2) = \emptyset, \phi(1) = \phi(3), D(\phi(5)) = \{\phi(4), \emptyset\},$$
$$D(\phi(6)) = \{\phi(1), \phi(4)\}, D(\phi(7)) = \{\phi(4), \emptyset\}, \phi(8) = \{2\}$$

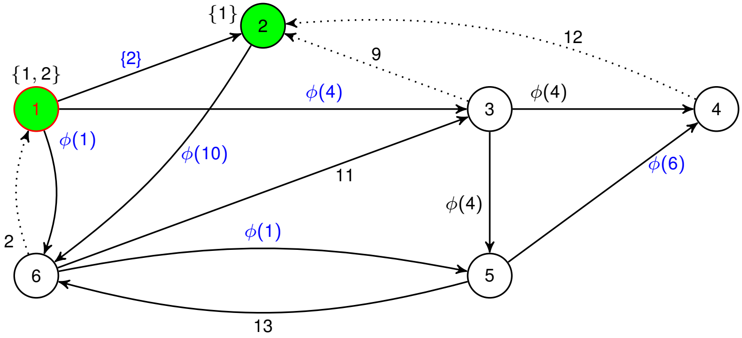
Example



$$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\}\ ;\ \mathcal{D} = \{1,2\}\ ;\ \mathcal{O}_1 = \{1, 2\}\ ;\ \mathcal{O}_2 = \{1\}\ ;\ \mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset\ ;$$
$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$$

$$D(\phi(1)) = D(\phi(4)) = \{\{1\}, \{2\}\},\ \phi(2) = \emptyset,\ \phi(1) = \phi(3),$$
$$D(\phi(5)) = \{\phi(4), \emptyset\},\ D(\phi(6)) = \{\phi(1), \phi(4)\},\ D(\phi(7)) = \{\phi(4), \emptyset\},$$
$$\phi(8) = \{2\},\ \phi(9) = \phi(12) = \emptyset$$

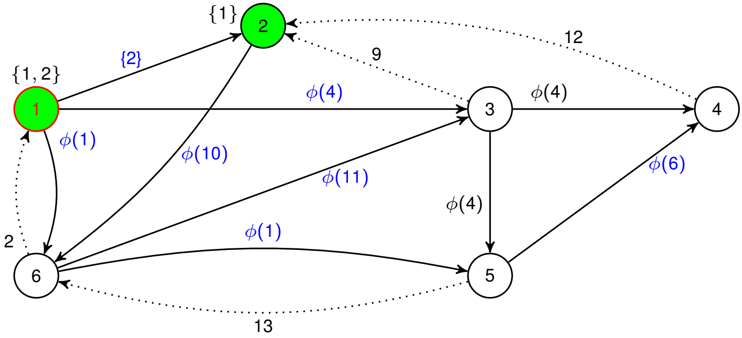
Example



$$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\}\ ; \mathcal{D} = \{1, 2\}\ ; \mathcal{O}_1 = \{1, 2\}\ ; \mathcal{O}_2 = \{1\}\ ; \mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset\ ;$$
$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$$

$$D(\phi(1)) = D(\phi(4)) = \{\{1\}, \{2\}\},\ \phi(2) = \emptyset,\ \phi(1) = \phi(3),$$
$$D(\phi(5)) = \{\phi(4), \emptyset\},\ D(\phi(6)) = \{\phi(1), \phi(4)\},\ D(\phi(7)) = \{\phi(4), \emptyset\},$$
$$\phi(8) = \{2\},\ \phi(9) = \phi(12) = \emptyset,\ D(\phi(10)) = \{\{1\}, \{2\}\}$$

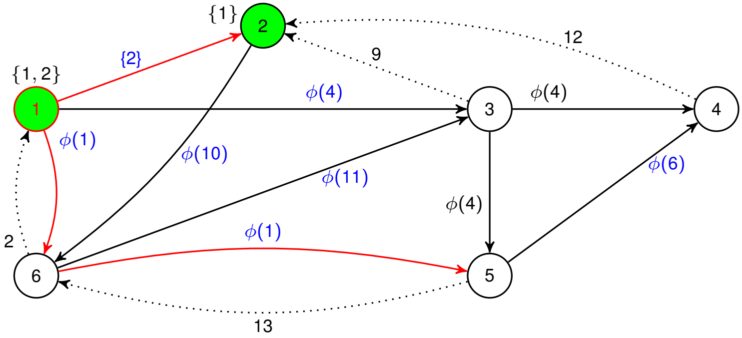
Example



$$\mathcal{N} = \mathcal{R} = \{1, 2, \dots, 6\}\ ;\ \mathcal{D} = \{1,2\}\ ;\ \mathcal{O}_1 = \{1, 2\}\ ;\ \mathcal{O}_2 = \{1\}\ ;\ \mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset\ ;$$
$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \dots, (5, 6)\}$$

$$D(\phi(1)) = D(\phi(4)) = \{\{1\}, \{2\}\},\ \phi(2) = \emptyset,\ \phi(1) = \phi(3),$$
$$D(\phi(5)) = \{\phi(4), \emptyset\},\ D(\phi(6)) = \{\phi(1), \phi(4)\},\ D(\phi(7)) = \{\phi(4), \emptyset\},\ \phi(8) = \{2\},$$
$$\phi(9) = \phi(12) = \emptyset,\ D(\phi(10)) = \{\{1\}, \{2\}\},\ \phi(13) = \emptyset,\ D(\phi(11)) = \{\phi(1), \phi(10)\}$$

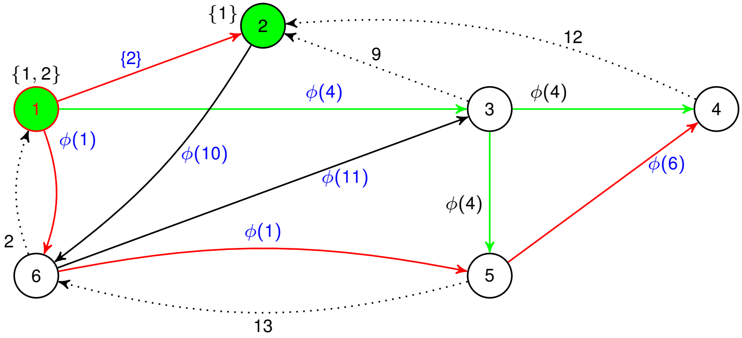
Example



$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\}$; $\mathcal{D} = \{1, 2\}$; $\mathcal{O}_1 = \{1, 2\}$; $\mathcal{O}_2 = \{1\}$; $\mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset$;
$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$

$D(\phi(1)) = D(\phi(4)) = \{\{1\}, \{2\}\}$, $\phi(2) = \emptyset$, $\phi(1) = \phi(3)$,
$D(\phi(5)) = \{\phi(4), \emptyset\}$, $D(\phi(6)) = \{\phi(1), \phi(4)\}$, $D(\phi(7)) = \{\phi(4), \emptyset\}$, $\phi(8) = \{2\}$,
$\phi(9) = \phi(12) = \emptyset$, $D(\phi(10)) = \{\{1\}, \{2\}\}$, $\phi(13) = \emptyset$, $D(\phi(11)) = \{\phi(1), \phi(10)\}$

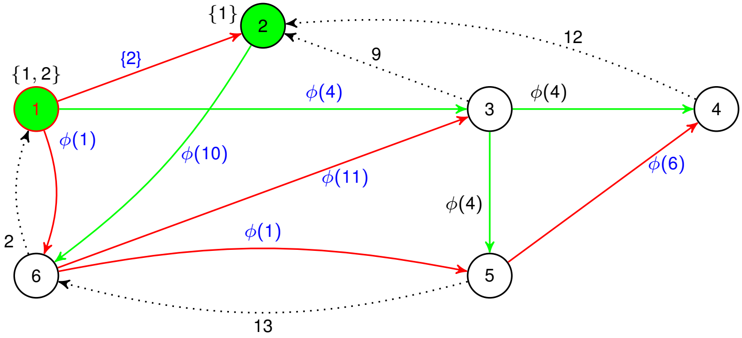
Example



$$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\} \; ; \mathcal{D} = \{1,2\} \; ; \mathcal{O}_1 = \{1, 2\} \; ; \mathcal{O}_2 = \{1\} \; ; \mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset \; ;$$
$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$$

$$D(\phi(1)) = D(\phi(4)) = \{\{1\}, \{2\}\}, \phi(2) = \emptyset, \phi(1) = \phi(3),$$
$$D(\phi(5)) = \{\phi(4), \emptyset\}, D(\phi(6)) = \{\phi(1), \phi(4)\}, D(\phi(7)) = \{\phi(4), \emptyset\}, \phi(8) = \{2\},$$
$$\phi(9) = \phi(12) = \emptyset, D(\phi(10)) = \{\{1\}, \{2\}\}, \phi(13) = \emptyset, D(\phi(11)) = \{\phi(1), \phi(10)\}$$

Example



$$\mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\} \,;\, \mathcal{D} = \{1,2\} \,;\, \mathcal{O}_1 = \{1, 2\} \,;\, \mathcal{O}_2 = \{1\} \,;\, \mathcal{O}_3 = \cdots = \mathcal{O}_6 = \emptyset \,;$$
$$\sigma = \{(1, 6), (6, 1), (6, 5), (1, 3), (3, 5), \ldots, (5, 6)\}$$

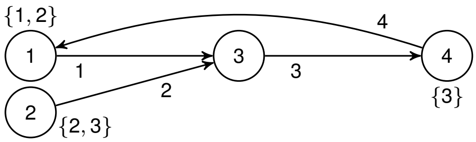
$$D(\phi(1)) = D(\phi(4)) = \{\{1\}, \{2\}\}, \phi(2) = \emptyset, \phi(1) = \phi(3),$$
$$D(\phi(5)) = \{\phi(4), \emptyset\}, D(\phi(6)) = \{\phi(1), \phi(4)\}, D(\phi(7)) = \{\phi(4), \emptyset\}, \phi(8) = \{2\},$$
$$\phi(9) = \phi(12) = \emptyset, D(\phi(10)) = \{\{1\}, \{2\}\}, \phi(13) = \emptyset, D(\phi(11)) = \{\phi(1), \phi(10)\}$$

Guidelines

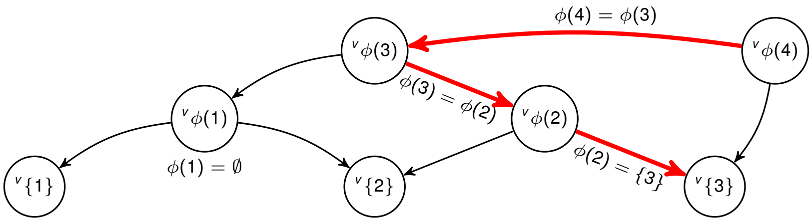
- Introduction
- Dominance rules
- Constraint programming
  - Preprocessings
  - Branching algorithm
  - Additional features
- Computational results
- Conclusion

The transfer graph

An instance of the dissemination problem

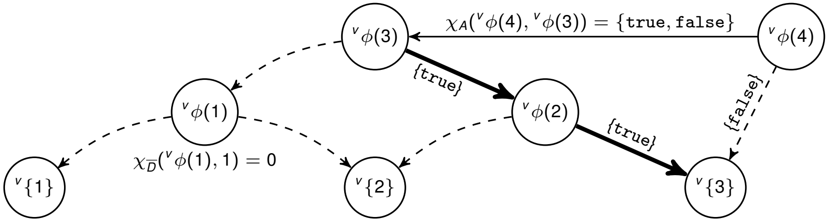


The associated transfer graph



The transfer graph

Transfer graph and subsets of transfer plans



Additional vertex/arc properties

- $\chi_\phi(^v\{3\}) = \{\{3\}\}$
- $\chi_\phi(^v\phi(1)) = \{\emptyset\}$
- $\chi_\phi(^v\phi(2)) = \{\{2\}\}$
- $\chi_\phi(^v\phi(4)) = \{\emptyset, \{3\}\}$
- $\chi_{\underline{D}}(^v\{1\}, 3) = \infty$
- $\chi_{\overline{D}}(^v\{2\}, 1) = 0$
- $\chi_{\underline{D}}(^v\{3\}, 3) = 2$
- $\chi_{\overline{D}}(^v\{3\}, 3) = \infty$

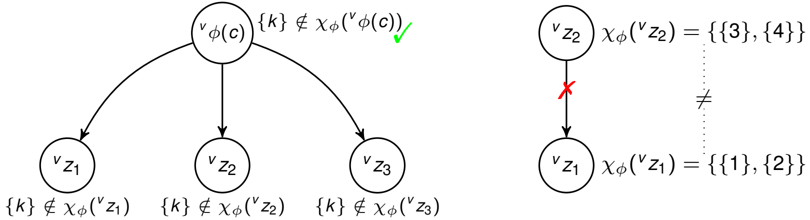
Elementary reasonings

Preprocessing procedures

For each contact $\sigma_c \in \sigma$, we try to show that all the transfer values possessed by s_c are also possessed by node r_c when contact σ_c occurs. If so, the contact is removed in accordance with the minimality rule.

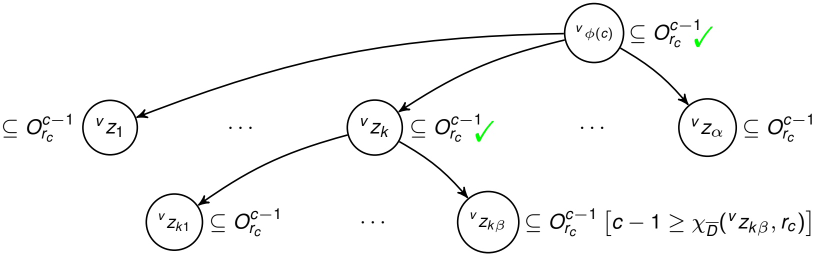
Elementary reasonings

Domain consistency



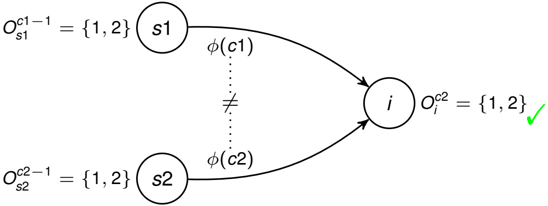
Elementary reasonings

Bottom-up procedure



Elementary reasonings

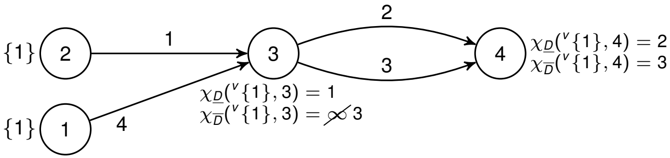
Top-down procedure



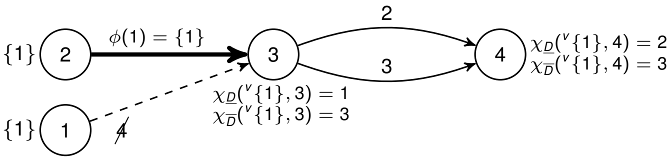
Elementary reasonings

Delivery consistency

Node 3 must receive datum unit 1 before time 3 to forward it to node 4.



Thus σ_4 occurs too late and transfer $\phi(1) = \{1\}$ is necessary.



Branching algorithm

The model contains:

- some variables to represent the transfer plan;
- some variables to represent what nodes possess;
- some variables to compute the dissemination length;
- some constraints to bind the variables, and to express the problem;
- some constraints to express the dominance rules.

The transfers are set *sequentially*. At each node of the search tree, the solver selects the smallest index $c \in \{1, 2, \ldots, m\}$ for which the value of transfer $\phi(c)$ has not yet been decided, then creates one branch per possible value. The order in which these branches are visited is heuristic. We first seek to identify the most "critical" transfers in terms of *feasibility* or, in case of a tie, we seek to balance the dissemination of the datum units.

Lower bounds

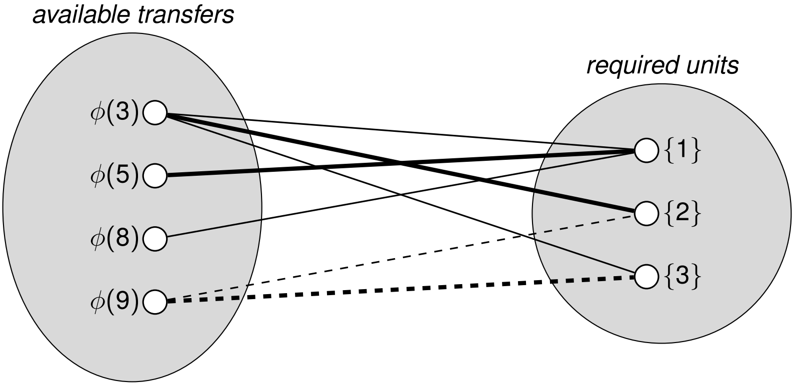
The first lower bound is based on the fact that each recipient node $i \in \mathcal{R}$ needs to receive $\alpha = u - |\mathcal{O}_i|$ datum units.

Proposition

Let $\sigma_x \in \sigma$ be the α^{th} contact $\sigma_c = (s_c, i) \in \sigma$ during which a datum unit $k \in \mathcal{D} \backslash \mathcal{O}_i$ can be transferred to node i. x is a valid lower bound.

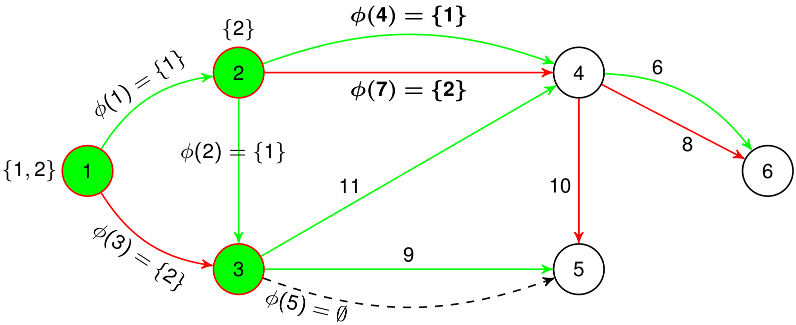
Lower bounds

Assignment problem



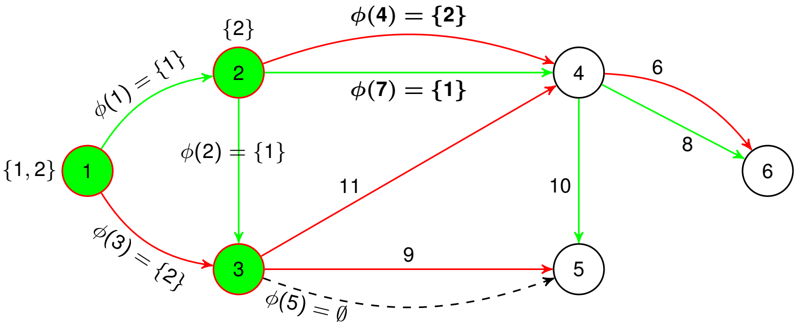
Symmetry-breaking techniques

Symmetric sub-branchings



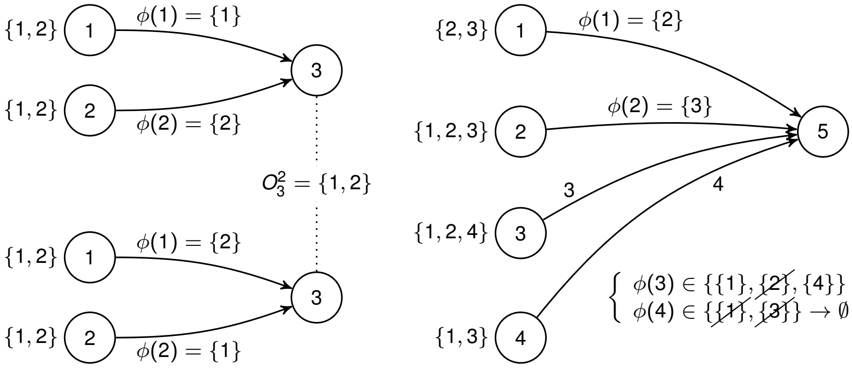
Symmetry-breaking techniques

Symmetric sub-branchings



Symmetry-breaking techniques

Consecutive contacts



Nogood recording

When we set transfer $\phi(t)$, we compute the subset of datum units possessed by each node. If we have already built a transfer plan such that all nodes possessed exactly the same datum units (or even more) at time t, then we can prune the current node.

Guidelines

- Introduction
- Dominance rules
- Constraint programming
- Computational results
  - CPLEX
  - Preprocessings
  - CP-Optimizer
- Conclusion

CPLEX

	name	*solved* (%)	*feas* (%)	*cpu* (s)	*gap* (%)
1	**3u10n**	100	-	0.91	-
	4u20n	100	-	14.1	-
2	**4u50n**	100	-	30.2	-
	4u100n	95.0	5.0	240	4.6
	5u50n	95.7	4.3	266	14.0
3	**10u10n**	81.3	12.5	1317	20.1
	50u10n	56.3	18.8	2563	2.6
	100u10n	33.3	16.7	3116	0.28

CPLEX+Preprocessings

	name	*solved*	*feas*	*cpu*	*gap*	*prep*	*rem*	*fcd*
1	**3u10n**	100	-	0.47	-	0.42	49.0	5.3
	4u20n	100	-	2.0	-	1.4	26.9	6.7
2	**4u50n**	100	-	4.1	-	2.5	21.0	6.7
	4u100n	100	-	20.1	-	5.7	20.3	5.6
	5u50n	100	-	19.4	-	2.7	13.1	7.4
3	**10u10n**	93.8	6.3	464	10.17	12.9	14.6	15.5
	50u10n	68.8	6.3	1691	1.06	156	8.8	8.9
	100u10n	66.7	0.00	3305	-	601	1.6	1.8

CP-Optimizer+Preprocessings

	name	*algorithm*	*solved*	*feas*	*cpu*
1	**3u10n**	sym+ngr+wlb	100	-	0.39
	4u20n	sym+ngr+wlb	100	-	1.3
2	**4u50n**	sym+ngr+wlb	100	-	2.7
	4u100n	sym+ngr+wlb	100	-	88.5
	5u50n	sym+ngr+wlb	100	-	20.7
3	**10u10n**	sym+ngr+slb	100	-	36.1
	50u10n	sym+ngr+slb	87.5	12.0	640
	100u10n	sym+ngr+slb	100	-	1220

Guidelines

- Introduction
- Dominance rules
- Constraint programming
- Computational results
- **Conclusion**

Conclusion

- An *extensible*, *intuitive* and *generic* framework.
- Efficient and "user-friendly" solvers.
- Ongoing research – the robust dissemination problem !