Modeling elements and solving techniques for the data dissemination problem

Ronan Bocquillon, Antoine Jouglet

Sorbonne Universités, Université de Technologie de Compiègne, CNRS, Heudiasyc UMR 7253, CS 60319, 60203 Compiègne cedex

{ *ronan.bocquillon ; antoine.jouglet* } *@hds.utc.fr*

Abstract

Systems of Systems (SoS) are collections of non-homogeneous, independent systems that interact to provide services. These systems can opportunistically share data during contacts that arise whenever two entities are close enough to each other. It is assumed in this paper that all contacts can be reliably predicted, *i.e.* the mobility of every system can be reliably estimated. A datum is split into several identified datum units to be delivered to a subset of recipient systems. During a contact, a given emitting system can transmit to a given receiving system one of the datum units that it possesses. The dissemination problem consists in finding a transfer plan which enables all the datum units to be transmitted from the sources (the systems that possess datum units from the beginning) to all the recipient systems. In this paper, we propose dominance-rule-based techniques for solving the data dissemination problem. In particular, we describe preprocessing procedures and some integer-linear-programming formulations to solve the problem.

Keywords: combinatorial optimization, dominance rules, preprocessing procedures, systems of systems, data transfer problem

1. Introduction

Systems of systems (or SoS) have been defined in many ways. One practical definition is that systems of systems are "*supersystems*" comprising other elements that are themselves complex, independent operational systems, all interacting to achieve a common goal [1]. If subsystems are not permanently connected, they must opportunistically make use of *contacts* that arise when entities are close enough to each other. These exchanges enable them to collaborate and route information from a subset of sources to a subset of recipient systems. This collaboration may be necessary, for instance, when contact durations are relatively short with respect to the volume of information to be disseminated. Here the information needs to be split up and possibly routed through non-recipient messenger systems whose role is to carry and forward data. Existing works have already looked at this kind of environment, in both opportunistic [2] and delay-tolerant networking [3]. Most of the time, no assumptions are made about the contacts that occur between the systems, although for many applications it is quite possible to make realistic predictions about node mobility and contacts. Such applications include satellite networks (where the trajectories of subsystems depend on straightforward physics), public transportation systems [4], and fleets of drones.

The present paper addresses this problem of making use of knowledge about possibilities of collaboration [5, 6] when information needs to be routed from sources to destinations within a given time horizon. The fundamental question is which elements of the information should be transferred from which system to which system when contacts occur.

This problem has exercised an increasing number of researchers over the last decade.

Alonso and Fall [7], for instance, proposed a linear formulation for computing a minimum delay transfer plan with respect to a set of nodes, a set of contacts and a set of messages. Available links need to be assigned to data such that every message can travel through the network from its sender to its receiver. The formulation incorporates constraints that are to be found in real applications, such as transmission delays, propagation delays and buffer capacities. As in most of the other works presented below, data transfers are modelled by unidentified numbers of bytes to be transmitted through a dynamic transportation graph. The problem can therefore be seen as a *dynamic* multi-commodity flow problem [8] in which messages are the commodities and edge capacities are time-varying. The main drawback here is that flow constraints implicitly forbid duplication of data, making such approaches unsuitable for multicast and multisource situations. In the present paper, we instead consider a set $\mathcal{N} = \{1, 2, \ldots, q\}$ of q interacting mobile nodes (systems) and a single set $\mathcal{D} = \{1, 2, \ldots, u\}$ of u identified "datum units", representing unitary, indivisible fragments of data, *e.g.* each datum unit might be a block of pixels corresponding to one high-resolution satellite picture. Initially, $\mathcal{D}$ (that we will sometimes refer to as "the datum") is distributed over different nodes $i \in \mathcal{N}$, the data sources, each of which holds a subset $O_i \subseteq \mathcal{D}$ of datum units. The datum must be transmitted within the allotted time to the subset $\mathcal{R} \subseteq \mathcal{N}$ of recipient nodes. The recipient nodes are required to obtain all the datum units. To our knowledge, no paper has so far addressed the multi-source case, despite its relevance if resulting algorithms are to be executed on-line, such as when routing tables need to be refreshed dynamically following new predictions on node mobility or connectivity.

Alonso's and Fall's works were subsequently extended by Jain *et al.* [9], who in particular proposed four oracles to compare the performances of routing procedures in terms of the amount of knowledge of network topology that they require. For example, the *contacts oracle* can answer any question regarding the contacts. Computational tests showed as expected that the greater the available knowledge, the better the performances. These oracles were extended by Zhao *et al.* [10] to take multicasting protocols into account (*e.g.* the *membership oracle* answers questions about group dynamics). In the present paper we consider that all the oracles are available.

In order to address higher-dimensional problems certain assumptions have been proposed. For example, Handorean *et al.* [11] defined *atomic contacts*, where contact durations (as opposed to inter-contact durations) are assumed to be instantaneous (both propagation and transmission delays are therefore disregarded). Hay and Giaccone [6] made the same assumption, and proposed a particularly interesting model that they called the *event-driven graph*. As the graph is *time-independent* and polynomial in size with respect to the number of contacts in the instance, very basic tools from graph theory can be used to solve numerous problems straightforwardly. So, for example, the authors solve shortest-path or max-flow subproblems to minimize the delay or to maximize the network throughput. In the present paper we also use the idea of atomic contacts. We define a sequence of contacts $\sigma = [\sigma_1, \sigma_2, \ldots, \sigma_m]$ of m ordered pairs $\sigma_c \in \mathcal{N}^2$ of nodes. During a contact $(s, r) \in \sigma$, the receiving node r can receive from the sending node s at most one datum unit $k \in \mathcal{D}$ that is held in s's buffer at that time (either from the outset or as a result of previous contacts). r is assumed to be in possession of k following the contact. In the following, s_c and r_c denote the sending and the receiving nodes at each contact $\sigma_c = (s_c, r_c) \in \sigma$. Buffers are assumed to be infinite and network failures are disregarded.

The *dissemination problem* is finding a transfer plan (a routing scheme) that minimizes the dissemination length, *i.e.* the number of contacts used to transmit the datum to all the recipient nodes. Instances of this problem are defined by one set of nodes $\mathcal{N} = \{1, 2, \ldots, q\}$, one datum $\mathcal{D} = \{1, 2, \ldots, u\}$, q sets $O_i \subseteq \mathcal{D}$ ($i \in \mathcal{N}$) corresponding to the datum units possessed by the nodes from the outset, one sequence σ of m contacts $(s_c, r_c) \in \mathcal{N}^2$, and one subset of recipients $\mathcal{R} \subseteq \mathcal{N}$. Every instance can be represented with an *evolving graph* [13], a time-dependent graph model proposed by Ferreira and described Section 2.1. The problem is strongly NP-hard [12].

The *delay-tolerant networking research group* deserves a mention for its extensive review of the literature. The large number of papers referenced on its website [`http://www.dtnrg.org/`] reflects a high level of interest in problems of routing in intermittently connected networks.

The remainder of the paper is organized as follows. In Section 2, we first formalize the notion of transfer plan and then introduce dominance rules for the dissemination problem. Sections 3 and 4 are devoted to algorithms that use dominance rules to detect irrelevant transfer plans. In Section 5 we propose a solving scheme that we then discuss and evaluate in Section 6.

2. Dominance Rules

The solving techniques discussed in this paper are based on a number of dominance rules (see [14] for a comprehensive paper on dominance rules) that dramatically improve the performance of enumeration algorithms. These dominance rules are defined and discussed in this section. The results form the basis for additional constraints and deduction algorithms to be presented in the following sections. However, we will first formalize the notion of a transfer plan, which provides a means of describing solutions to the problem. Let us recall that a transfer plan defines a routing scheme, by indicating which units have to be transmitted during the different contacts.

2.1. Transfer plans

A transfer plan is an application $\phi : \{1, 2, \ldots, m\} \mapsto \{\emptyset, \{1\}, \{2\}, \ldots, \{u\}\}$, where $\phi(c)$ indicates the datum unit received by r_c during contact $\sigma_c \in \sigma$. Where $\phi(c) = \emptyset$, nothing is transferred during contact σ_c. Hereinafter, T_ϕ denotes the target set $\{\emptyset, \{1\}, \{2\}, \ldots, \{u\}\}$ of ϕ. A transfer plan ϕ has a corresponding set of states $O_i^t \subseteq \mathcal{D}$, defined for each time index $t \in \{0, 1, \ldots, m\}$ and each node $i \in \mathcal{N}$, such that:

$$
\begin{array}{l}
(1)\ \forall i \in \mathcal{N},\ O_i^0 = O_i, \\
(2)\ \forall c \in \{1, 2, ..., m\},\ O_{r_c}^c = O_{r_c}^{c-1} \cup \phi(c), \\
(3)\ \forall c \in \{1, ..., m\},\ \forall i \in \mathcal{N} \backslash \{r_c\},\ O_i^c = O_i^{c-1}
\end{array}
$$

Thus, each state O_i^t contains the datum units obtained by node i during the first t contacts of sequence σ (in addition to the datum units held from the outset). The transfer plan is *valid* if nodes always send the units that they hold, *i.e.* $\forall \sigma_c \in \sigma$, we have $\phi(c) \in \{\emptyset\} \cup \{\{k\} \mid k \in O_{s_c}^{c-1}\}$.

A valid transfer plan ϕ has a *delivery length* $\lambda_i(\phi)$ for each node $i \in \mathcal{N}$, which corresponds to the smallest contact index t after which node i possesses every datum unit $k \in \mathcal{D}$, *i.e.* $\lambda_i(\phi) = \min \{t \in \{0, \ldots, m\} \mid O_i^t = \mathcal{D}\}$. If this index does not exist, then it is assumed below that $\lambda_i(\phi) = \infty$. The *dissemination length* $\lambda(\phi)$ of the transfer plan corresponds to the smallest index t at which all the recipient nodes are delivered, *i.e.* $\lambda(\phi) = \max_{i \in \mathcal{R}} \{\lambda_i(\phi)\}$. In this paper, we tackle the problem of finding a valid transfer plan minimizing $\lambda(\phi)$.

This problem is known to be NP-Hard in the strong sense, but it can be polynomially solved if $u = 1$ or $|\mathcal{R}| = 1$ (see [12] for a comprehensive study on the complexity of this problem).

(a) *instance definition*

$$\begin{cases} \mathcal{N} = \mathcal{R} = \{1, 2, \ldots, 6\}; \mathcal{D} = \{1, 2\} \\ O_1 = \{1, 2\}; O_2 = \{1\}; O_3 = \cdots = O_6 = \emptyset \\ \sigma = [(1, 6), (6, 1), (6, 5), \ldots, (5, 6)] \end{cases}$$

(b) *a solution of 3 arc-disjoint branchings*

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

(c) *evolving graph [13]*

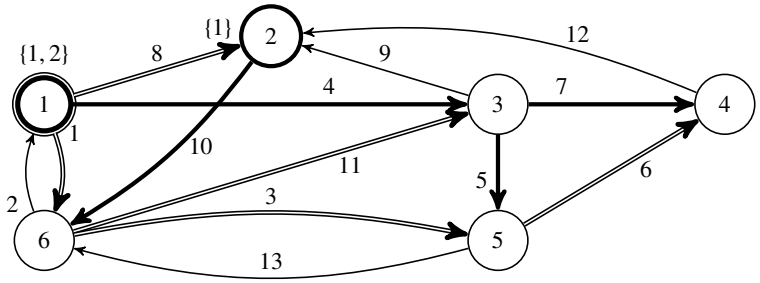


Figure 1: An instance of the problem, a solution and the associated evolving graph.

Instances of the problem can also be described by evolving graphs, *i.e.* multigraphs whose vertices represent nodes and whose arcs represent connections between these nodes. Each arc is labelled with time intervals which indicate the period in which the link is active. To address time constraints effectively, the notion of *path* is replaced by the notion of *journey*, which is an ordered set of arcs having *increasing* labels. For our needs, each contact is thus represented by an arc whose label is given by its position in the sequence. In Figure 1, $[\sigma_1 = (1, 6), \sigma_3 = (6, 5)]$ is a journey, since $3 \geq 1$. This represents the fact that node 6 is able to forward the datum unit it received from node 1 at time 1 to node 5 at time 3. However, $[\sigma_{13} = (5, 6), \sigma_1 = (1, 6)]$ is not a journey, since $1 < 13$. Given a datum unit $k \in \mathcal{D}$, a journey $[(i, u), \ldots, (v, j)]$ between a source node $i \in \mathcal{N} \,|\, k \in O_i$ and a recipient node $j \in \mathcal{R}$ therefore represents a store-carry-and-forward routing that enables unit k to be transmitted from i to j. It can even be shown that solving the problem is equivalent to finding a set of arc-disjoint branchings in this graph, where each datum unit is associated with a set of branchings rooted on corresponding source nodes and covering the recipients [12, 13], *cf.* Figure 1.

2.2. Dominant sets of solutions

We now propose and discuss the dominance rules.

First of all, it will be remarked that more than one transmission of the *same* unit to the *same* receiving node may occur within the *same* valid transfer plan. From an operational point of view, resources are wasted, and from a computational point of view, taking such solutions into account significantly enlarges the search space, which makes it desirable to disallow redundant transfers.

Hence the definitions below and the ensuing dominance rule.

Definition 2.1. *The transfer occurring at time c with respect to the transfer plan ϕ is said to be* null *if and only if no datum unit is transmitted throughout contact σ_c, i.e. $\phi(c) = \emptyset$.*

Definition 2.2. *The transfer occurring at time c with respect to transfer plan ϕ is said to be* improving *if and only if the receiver obtains a new datum unit during σ_c, i.e. $|O^{c-1}_{r_c}| < |O^c_{r_c}|$.*

Definition 2.3. *A transfer plan ϕ is said to be* minimal *if and only if all its transfers are either null or improving, i.e. no node receives the same datum unit more than once.*

Proposition 2.1. *The set of* minimal *transfer plans is dominant.*

Proof. Let ϕ be a non-minimal transfer plan. Consequently there exists at least one transfer which is neither null nor improving, *i.e.* $\exists c \in \{1, 2, \ldots, m\}$ with $\phi(c) \subseteq O_{r_c}^{c-1}$. The transfer plan ϕ' obtained by copying ϕ and by fixing $\phi'(c) = \emptyset$ has the same dissemination length, *i.e.* $\lambda(\phi') = \lambda(\phi)$. This process is to be repeated as long as the transfer plan is not minimal. □

The idea of a minimal transfer plan may be further reinforced by considering only minimal transfer plans in which *any* non-recipient node forwards *any* unit it receives at least once. From an operational point of view, this means avoiding transmitting data to any non-recipient node that is unable to contribute to data dissemination. Unfortunately, in practice, this kind of dominance rule is found to be less efficient than the minimality rule introduced above. Our numerical tests showed that it is beneficial to favour all improving transfers, rather than trying to reduce their number at a higher computational cost, even for non-recipient nodes.

In the light of these observations, we therefore propose the following dominance rule.

Definition 2.4. *A transfer plan ϕ is said to be* active *if there exists no contact $\sigma_c \in \sigma$ in which a datum unit $k \in \mathcal{D}$ is transmitted from $s_c \in \mathcal{N}$ to $r_c \in \mathcal{N}$, where the same transmission could have been done earlier using a non-improving transfer $\phi(c')$, i.e. $\forall c \in \{2, 3, \ldots, m\}$ such that $|O_{r_c}^{c-1}| < |O_{r_c}^{c}|$, $\nexists c' \in \{1, 2, \ldots, c-1\}$ with $r_{c'} = r_c$, $\phi(c) \subseteq O_{s_{c'}}^{c'-1}$, and $|O_{r_{c'}}^{c'-1}| = |O_{r_{c'}}^{c'}|$.*

Proposition 2.2. *The set of* active *transfer plans is dominant.*

Proof. Let ϕ be a non-active transfer plan, *i.e.* $\exists c, c' \in \{1, 2, \ldots, m\}$, and $\exists k \in \mathcal{D}$ such that $c' < c$, $r_c = r_{c'}$, $k \in O_{s_{c'}}^{c'-1}$, $k \notin O_{r_{c'}}^{c'-1}$, $|O_{r_{c'}}^{c'-1}| = |O_{r_{c'}}^{c'}|$, and $\phi(c) = \{k\}$. Let ϕ' refer to the transfer plan obtained by copying ϕ, and by setting $\phi'(c') = \{k\}$. The dissemination length of ϕ' is better than or equal to the dissemination length of ϕ. This process is repeated until ϕ' is active. $\phi'(c)$ can also be set to $\emptyset$ (at each iteration) if minimality properties have to be maintained. □

In an active transfer plan, an improving transfer may still be ignored in favour of another non-improving or null transfer. In particular, there may exist $c' \in \{1, 2, \ldots, m-1\}$ and $k \in \mathcal{D}$ such that $k \in O_{s_{c'}}^{c'-1}$, $k \notin O_{r_{c'}}^{c'}$ and $|O_{r_{c'}}^{c'-1}| = |O_{r_{c'}}^{c'}|$, if $\forall c \in \{c', \ldots, m\} \,|\, r_c = r_{c'}$, $\phi(c) \neq \{k\}$.

The dominance rule below strengthens the idea of an active transfer plan by ensuring that a non-profitable transfer can never be preferred to an improving transfer.

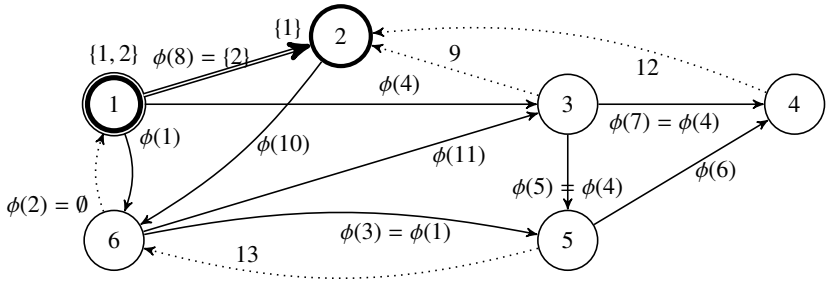
Definition 2.5. *A transfer plan ϕ is said to be* strictly-active *if no non-improving transfer could have been improving, i.e. $\forall c \in \{1, 2, \ldots, m\}$,* if $\exists k \in \mathcal{D}, k \in O_{s_c}^{c-1}, k \notin O_{r_c}^{c-1}$, then $|O_{r_c}^{c-1}| < |O_{r_c}^{c}|$.

Proposition 2.3. *The set of* strictly-active *transfer plans is dominant.*

Proof. Let ϕ be a non-strictly-active transfer plan, *i.e.* $\exists \sigma_c \in \sigma$, and $\exists k \in \mathcal{D}$ such that $k \in O_{s_c}^{c-1}$, $k \notin O_{r_c}^{c-1}$, and $|O_{r_c}^{c-1}| = |O_{r_c}^{c}|$. Let ϕ' denote the transfer plan obtained by copying ϕ and by fixing $\phi'(c) = \{k\}$. Transfer plan ϕ' has a dissemination length better than or equal to that of ϕ, that is to say $\lambda(\phi') \leq \lambda(\phi)$. The process is repeated until ϕ' is strictly-active. □

A strictly-active transfer plan is active (by definition). The sets of minimal-active and minimal-strictly-active transfer plans are dominant (by combining the above proofs).

The remainder of the paper will focus on algorithms designed to identify the non-minimal and the non-strictly-active transfer plans. We will now illustrate how these dominance rules can help us to solve the problem, by considering the instance depicted in Figure 1:



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

Figure 2: The knowledge collected using dominance-based deduction tools.

- **dominance-based deductions:** node 1 initially possesses all the datum units, whereas 6 starts with an empty buffer, *i.e.* $O_1 = \{1, 2\} \wedge O_6 = \emptyset$. Thus, $\phi(1) \neq \emptyset$ holds in every strictly-active transfer plan, *i.e.* $\phi(1) = \{1\} \vee \phi(1) = \{2\}$. It also means that node 6 possesses either datum unit 1 or datum unit 2 afterwards. Since node 1 initially possesses these two units, then $\phi(2)$ is null in all minimal strictly-active transfer plans. In practice, contact σ_2 can then be removed from the instance before any computation is done. With the same approach, it can be shown that $\phi(3) = \phi(1)$ and $\phi(8) = \{2\}$ hold in a minimal strictly-active transfer plan. Figure 2 illustrates the results that can be collected by applying these methods successively over each contact. Five contacts are seen to have been set.

- **delivery-requirement-based deductions:** This being said, a feasible transfer plan has still to be found. It needs to be decided how to continue the process, *e.g.* $\phi(1) = \{2\}$ or $\phi(1) = \{1\}$. Let us set $\phi(1) = \{2\}$. There exists only one journey for delivering datum unit 1 to nodes 6 and 5 with $\phi(4) = \phi(5) = \phi(7) = \phi(10) = \{1\}$. The only valid remaining solution is therefore with $\phi(6) = \{1\}$, *cf.* Figure 1. It should be remarked that a symmetric solution can be built with $\phi(1) = \{1\}$. These transfer plans are minimal, strictly-active and optimal.

Methods such as the one described above can be implemented within a branching algorithm, *e.g.* a branch-and-bound or a branch-and-cut algorithm. Decisions and backtracks constitute the branching stage (generation and selection of nodes), with local deductions performed at each node of the search tree to remove dominated solutions, *e.g.* through constraint propagation or cuts. The following two sections describe how these deduction techniques can be implemented, and then Section 5 focuses on a particular branching procedure that we have proposed.

3. Transfer Graph

In this section, we propose a graph model – the *transfer graph* – which enables a set of valid transfer plans to be represented, *i.e.* a subset of the search space associated with the dissemination problem. It is the data structure that forms the basis for some deduction algorithms that we will subsequently propose (Section 4). These procedures update the transfer graph so that the search space is dynamically reduced.

These techniques will be applied to pre-process the instances (to reduce their size), and within an integer-linear-programming framework, *i.e.* in a branch-and-cut procedure (*cf.* Section 5).

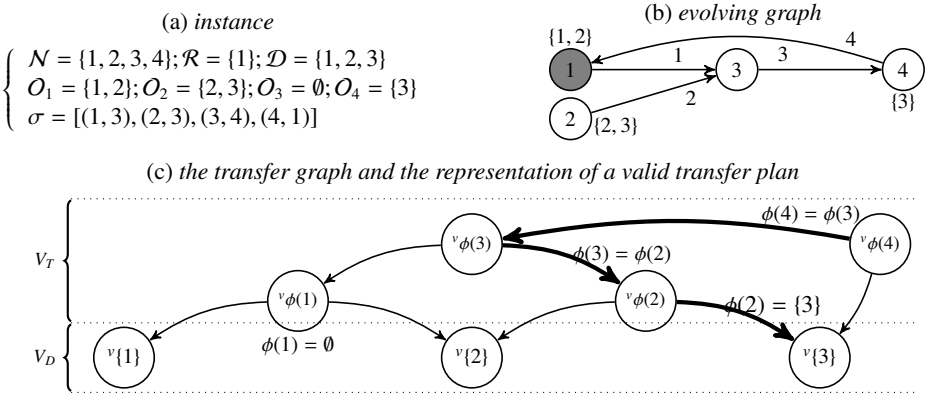


Figure 3: An instance, its evolving graph and the corresponding transfer graph.

3.1. About the transfer graph

First, let us recall that each state $O_i^t \subseteq \mathcal{D}$ with $i \in \mathcal{N}$ and $t \in \{0, 1, \ldots, m\}$ contains the subset of units obtained by node i after the first t contacts. Let us also recall that a transfer plan is valid if nodes transmit only datum units that they have possessed from the outset or that they have obtained as a result of previous contacts. Hence the following assertion:

$$\forall c \in \{1,2,\ldots,m\},\ O_{s_c}^{c-1} = O_{s_c} \cup \underbrace{\left(\bigcup_{\substack{t \in \{1,\ldots,c-1\} \\ r_t = s_c}} \phi(t) \right)}_{\substack{\text{the units obtained as a} \\ \text{result of former contacts}}}$$

From now on we will be looking at a graph – the so-called *transfer graph* – designed to take account of dependencies between transfers. It is defined as follows:

Definition 3.1. *Let us consider an instance of the dissemination problem. The associated transfer graph is a directed graph $G_\phi = (V, A)$ with $V = V_D \cup V_T$, $A = A_1 \cup \cdots \cup A_m \subseteq V_T \times V$, and where V and A are built as follows:*

1. $V_D = \{{}^v\{k\} \mid k \in \mathcal{D}\}$ *where vertex* ${}^v\{k\}$ *is associated with datum unit k,*
2. $V_T = \{{}^v\phi(c) \mid c \in \{1,\ldots,m\}\}$ *where vertex* ${}^v\phi(c)$ *is associated with transfer $\phi(c)$,*
3. $\forall c \in \{1,2,\ldots,m\}$,

$$A_c = \underbrace{\{({}^v\phi(c), {}^v\{k\}) \mid k \in O_{s_c}\}}_{\substack{\text{corresponds to the units} \\ \text{initially possessed by } s_c}} \cup \underbrace{\{({}^v\phi(c), {}^v\phi(t)) \mid t \in \{1,\ldots,c-1\} \text{ and } r_t = s_c\}}_{\substack{\text{corresponds to the units obtained by } s_c \\ \text{during the contacts which precede contact } \sigma_c}}$$

In such a graph, an arc between a vertex ${}^v\phi(c) \in V_T$ and a vertex ${}^v\{k\} \in V_D$ symbolises the fact that node s_c can transmit datum unit $k \in \mathcal{D}$ during contact σ_c because it has possessed that unit from the outset, *i.e.* $k \in O_{s_c}$. An arc between two vertices ${}^v\phi(c)$ and ${}^v\phi(t) \in V_T$ models the possibility that s_c will transmit during σ_c a unit that it has received during σ_t, as $t < c$, $r_t = s_c$.

Remark 3.1. *The transfer graph contains no circuits, and is polynomial in size of the instance with which it is associated. Indeed, there are exactly $u + m$ vertices and $O(mu + m^2)$ arcs.*

Remark 3.2. *The vertices in V_D are termed the* unit *vertices, and those in V_T are termed the* transfer *vertices. As we shall see below, the distinction between the unit vertices and the transfer vertices is rarely required. We frequently need to refer to a unit $\phi(c)$ that is transmitted during a contact σ_c without knowing precisely which unit $k \in \mathcal{D}$ it corresponds to, or even whether anything has really been transmitted. To clarify our notation, such an element is termed a* transfer value*. While a vertex ${}^v\{k\} \in V_D$ represents a unit and a vertex ${}^v\phi(c) \in V_T$ represents a transfer, we will use the notation ${}^vz \in V$ to refer to any kind of vertex.*

These definitions are illustrated in Figure 3. The set of *unit vertices* is $V_D = \{{}^v\{1\}, {}^v\{2\}, {}^v\{3\}\}$, while the set of *transfer vertices* is $V_T = \{{}^v\phi(1), {}^v\phi(2), {}^v\phi(3), {}^v\phi(4)\}$. The set of *transfer values* associated with vertices in $V = V_D \cup V_T$ is then $\{\{1\}, \{2\}, \{3\}, \phi(1), \phi(2), \phi(3), \phi(4)\}$. The arcs going out of vertex ${}^v\phi(4)$ are $({}^v\phi(4), {}^v\{3\})$ (because s_4 has possessed datum unit 3 from the outset) and $({}^v\phi(4), {}^v\phi(3))$ (contact σ_3 occurs before contact σ_4 and the receiver r_3 of that contact is s_4).

3.2. Transfer graph and subsets of transfer plans

The set A_c of arcs going out of each vertex ${}^v\phi(c) \in V_T$ defines – by construction – the set of all values that transfer $\phi(c)$ can take in a valid transfer plan. A path $[{}^v\phi(c_p), \ldots, {}^v\phi(c_1), {}^v\{k\}]$ of $p+1$ arcs from a transfer vertex ${}^v\phi(c_p) \in V_T$ to a unit vertex ${}^v\{k\} \in V_D$ defines a possible way to route datum unit k from the source node s_{c_1} to node r_{c_p}. A solution where $\phi(c_1) = \cdots = \phi(c_p) = \{k\}$ is valid – still by construction – and enables nodes $r_{c_1}, \ldots, r_{c_p}$ to receive unit k. More generally, an anti-branching rooted on a unit vertex ${}^v\{k\} \in V_D$ describes a routing to disseminate datum unit $k \in \mathcal{D}$. It corresponds to a set of branchings in the evolving graph.

As a result, a transfer plan can equally be defined as a subgraph of the transfer graph made up of u vertex-disjoint anti-branchings that are each rooted on a different unit vertex. It corresponds to a spanning subgraph of the transfer graph containing at most one arc going out of any vertex. This constraint actually originates from the fact that at most one datum unit can be transferred during a contact. Thus, selecting an arc $({}^v\phi(c), {}^v\{k\})$ is interpreted as $\phi(c) = \{k\}$, while selecting an arc $({}^v\phi(c_2), {}^v\phi(c_1))$ ensures that the unit transmitted during contact σ_{c_1} is transmitted during contact σ_{c_2}, that is to say $\phi(c_1) = \phi(c_2)$. In short, in this subgraph, transfer $\phi(c)$ is either considered as null if no arc out of vertex ${}^v\phi(c)$ has been selected, or equal to $\{k\}$ if vertices ${}^v\phi(c)$ and ${}^v\{k\}$ are linked by a path. If we look back at the example in Figure 3, the bolded arcs define a way to deliver datum unit 3 to recipient node 1. The anti-branchings associated with vertices ${}^v\{1\}$ and ${}^v\{2\}$ contain no arc because these datum units are never transmitted.

Remark 3.3. *Such a subgraph does* ***not*** *really need to be a union of anti-branchings. It prevents situations where a set includes an arc $({}^v\phi(c_2), {}^v\phi(c_1))$, but does not include an arc going out of vertex ${}^v\phi(c_1)$, i.e. the path starting from ${}^v\phi(c_2)$ does not end on a unit vertex. The meaning of this path – i.e. $\phi(c_2) = \phi(c_1) = \emptyset$ – can already be expressed by not selecting arc $({}^v\phi(c_2), {}^v\phi(c_1))$ at all. Taking such sets into account would therefore not enhance the expressiveness of the transfer graph, but would dramatically increase the number of subsets to be explored.*

Remark 3.4. *In Figure 3 it is worth nothing that the same transfer plan is obtained by selecting arc $({}^v\phi(4, {}^v\{3\})$ rather than arc $({}^v\phi(4), {}^v\phi(3)) \in A$. This is because the transfer plan represented by these two subgraphs is not minimal. Node 4 receives unit 3 during contact σ_3 even though it has possessed this datum unit from the outset. Thus, during contact σ_4, node 4 can transmit either the unit it has possessed from the outset, or the additional copy it has just received. Fortunately, a* minimal *transfer plan always has only one corresponding subgraph, and therefore we do not need to address this case.*

To represent particular subsets of transfer plans, we need to be able to arbitrarily disallow certain arcs or, alternatively, stipulate that certain arcs must be used. To this end, we introduce function $\chi_A : A \mapsto \mathcal{P}(\{\texttt{true}, \texttt{false}\})$, whose role is to indicate which arcs are allowed to be selected when a set of u anti-branchings (a transfer plan) is constructed. So, $\forall ({}^v\phi(c), {}^v z) \in A$,

- $\chi_A({}^v\phi(c), {}^v z) = \{\texttt{true}\}$ means that arc $({}^v\phi(c), {}^v z)$ must be selected and that node s_c must transmit transfer value z during contact σ_c,
- $\chi_A({}^v\phi(c), {}^v z) = \{\texttt{false}\}$ means that arc $({}^v\phi(c), {}^v z)$ cannot be selected and that node s_c must not transmit transfer value z during contact σ_c,
- $\chi_A({}^v\phi(c), {}^v z) = \{\texttt{true}, \texttt{false}\}$ means that no decision has been taken yet, that is to say there is no additional constraint upon this transfer.

As shown in Figure 4, *cf.* the instance in Figure 3, this property (function) is able to represent various constraints – *e.g.* $\chi_A({}^v\phi(3), {}^v\phi(2)) = \{\texttt{true}\}$ expresses constraint $\phi(3) = \phi(2)$, whereas the combination of $\chi_A({}^v\phi(1), {}^v\{1\}) = \{\texttt{false}\}$ and $\chi_A({}^v\phi(1), {}^v\{2\}) = \{\texttt{false}\}$ forces transfer $\phi(1)$ to be null, *i.e.* $\phi(1) = \emptyset$. Below, in the light of Remark 3.3, a transfer $\phi(c)$ will be considered null if and only if all arcs $a \in A$ leaving or entering ${}^v\phi(c) \in V_T$ are such that $\chi_A(a) = \{\texttt{false}\}$.

3.3. Additional graph properties and complex subsets of transfer plans

The transfer graph, or specifically function χ_A, represents *a set of valid transfer plans*. If all arcs $a \in A$ are such that $\chi_A(a) = \{\texttt{true}, \texttt{false}\}$, *i.e.* the whole transfer graph is retained, then all valid solutions are represented. On the other hand, if χ_A includes constraints, *e.g.* a constraint that some arcs cannot be selected, then some transfer plans can no longer be built, and therefore in this case only a subset of transfer plans is represented.

Nevertheless, it remains impossible to represent more complex subsets of transfer plans, *e.g.* the transfer plans such that $\phi(3) = \{2\}$ (since there is no arc between vertices ${}^v\phi(3)$ and ${}^v\{2\}$), or the set of transfer plans such that a given node receives a given datum unit between two given dates (since time windows are not represented in the graph).

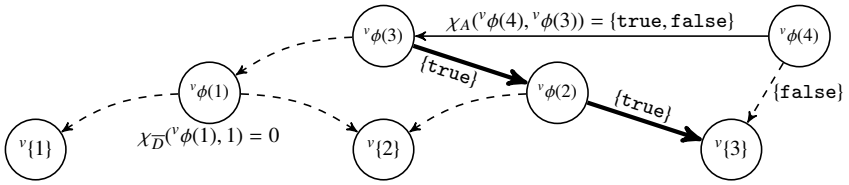
However, these sets of transfer plans are all defined by sets of constraints which can easily be represented with the additional vertex properties we introduce below:

- $\chi_\phi : V \mapsto \mathcal{P}(T_\phi)$ (recalling that $T_\phi = \{\emptyset, \{1\}, \{2\}, \ldots, \{u\}\}$) : this property specifies the domain of every transfer value – that is the set of values that transfer values are allowed to have. For example, $\chi_\phi({}^v\phi(c)) = \{\emptyset, \{1\}, \{2\}\}$ indicates that transfer $\phi(c)$ can either be null, equal to $\{1\}$, or equal to $\{2\}$. In short,
  - $\forall {}^v\phi(c) \in V_T, \chi_\phi({}^v\phi(c))$ is the current domain of $\phi(c)$ and $\phi(c) \in \chi_\phi({}^v\phi(c))$ holds. Thus, $\chi_\phi({}^v\phi(c)) = \{\emptyset\}$ means that transfer $\phi(c)$ has to be null, while $\emptyset \notin \chi_\phi({}^v\phi(c))$ means that transfer $\phi(c)$ cannot be null.
  - $\forall {}^v\{k\} \in V_D, \chi_\phi({}^v\{k\}) = \{\{k\}\}$ by convention.

  This property is used to represent various kinds of constraints. Looking back at the example in Figure 4, $\chi_\phi({}^v\phi(1)) = \{\emptyset\}$ ensures that transfer $\phi(1)$ is null, and $\chi_\phi({}^v\phi(4)) = \{\emptyset, \{3\}\}$ implies that transfer $\phi(4)$ is either null, or equal to $\{3\}$. Transfer $\phi(2)$ has been set to $\{2\}$. Note that this same property is used when a transfer is shown to be improving in all strictly-active minimal solutions represented by the transfer graph.

- $\chi_{\underline{D}}$ and $\chi_{\overline{D}} : V \times \mathcal{N} \mapsto \{0, \ldots, m, \infty\}$: these properties specify an earliest and a latest date at which each node is allowed to receive each transfer value ($\infty = m + 1$ in practice):

(a) *the transfer graph and some of its properties*



(b) *additional vertex properties*

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

(c) *the two transfer plans represented by this transfer graph*

$$\phi_1 : \begin{cases} \phi_1(1) = \emptyset \\ \phi_1(3) = \phi_1(2) = \{3\} \\ \phi_1(4) = \emptyset \end{cases} \quad \phi_2 : \begin{cases} \phi_2(1) = \emptyset \\ \phi_2(3) = \phi_2(2) = \{3\} \\ \phi_2(4) = \phi_2(3) = \{3\} \end{cases}$$

Figure 4: The transfer graph and its properties.

- $\forall {}^v z \in V, \forall i \in \mathcal{N}, \chi_{\underline{D}}({}^v z, i)$ forbids node i to receive transfer value z too soon. Formally, $\forall t \in \{0, 1, \ldots, m\}$, if $t < \chi_{\underline{D}}({}^v z, i)$, then $z \not\subseteq O_i^t$ holds,
- $\forall {}^v z \in V, \forall i \in \mathcal{N}, \chi_{\overline{D}}({}^v z, i)$ imposes a deadline on node i to receive transfer value z, *i.e.* $\forall t \in \{0, 1, \ldots, m\}$, if $t \geq \chi_{\overline{D}}({}^v z, i)$, then $z \subseteq O_i^t$ holds.

Node i is allowed to receive transfer value z during contacts occurring between dates $\chi_{\underline{D}}({}^v z, i)$ and $\chi_{\overline{D}}({}^v z, i)$. If we look back at the example in Figure 4, node 1 is constrained to possess $\phi(1)$ from the outset as $\chi_{\overline{D}}({}^v\phi(1), 1) = 0$, *i.e.* $\phi(1) \subseteq O_1$ must hold, regardless of the value of $\phi(1)$ (the fact that $\phi(1)$ is null as a result of χ_ϕ does not matter here). Moreover, node 3 is not allowed to receive unit 3 before contact σ_2, since $\chi_{\underline{D}}({}^v\{3\}, 3) = 2$, and is even not required to possess this unit at the end of a transfer plan as $\chi_{\overline{D}}({}^v\{3\}, 3) = \infty$.

Remark 3.5. *The properties focus on different aspects of the problem. Although they are linked by constraints, the different properties have their particular features and are not interchangeable. Redundancy is not a problem in practice, but deduction algorithms must be designed to take account of it, for example in ensuring that $\forall c \in \{1, 2, \ldots, m\}$, all arcs $a \in A$ leaving or entering vertex ${}^v\phi(c) \in V_T$ are such that $\chi_A(a) = \{\texttt{false}\}$ whenever $\chi_\phi({}^v\phi(c)) = \{\emptyset\}$, and vice versa...*

Altogether, the transfer graph shown in Figure 4, *cf.* the instance in Figure 3, represents the set $\{\phi_1, \phi_2\}$ of valid transfer plans where the transfer arising during contact σ_1 is null, and where datum unit 3 is transmitted from node 2 to node 4 during contacts σ_2 and σ_3. The transfer that occurs at time 4 is either null, *i.e.* $\phi_1(4) = \emptyset$, or improving, *i.e.* $\phi_2(4) = \phi_2(3) = \phi_2(2) = \{3\}$. Transfer plan ϕ_2 is seen not to be a solution, since it does not fulfil delivery requirements, *i.e.* datum unit 3 has not been obtained by recipient node 1 when the last contact ends.

What represents what – a short summary of Sections 3.1 to 3.3:.

1. the transfer graph *represents* the whole set of valid transfer plans,
2. a spanning subgraph of the transfer graph (described by function χ_A) thus *represents* a subset of valid transfer plans,
3. a spanning subgraph of the transfer graph which includes at most one arc going out of each vertex *represents* one valid transfer plan, *cf.* Section 3.2,

4. the search is limited to the sets of anti-branchings (rooted on unit vertices) in order to avoid equivalent solutions, *cf.* Remarks 3.3 and 3.4,
5. vertex properties $\chi_\phi + \chi_{\underline{D}} + \chi_{\overline{D}}$ enable us to refine the subsets of transfer plans represented by the transfer graph and χ_A, *cf.* Section 3.3.

3.4. The transfer graph within a branching procedure

The transfer graph and its properties can be used to represent a state in a branching algorithm which searches for a valid transfer plan. The properties can be used to separate a set of transfer plans during the branching stage – *e.g.* given a unit $k \in \mathcal{D}$ and a contact $\sigma_c \in \sigma$, the branches $\chi_\phi({}^v\phi(c)) = \{\{k\}\}$ and $\{k\} \notin \chi_\phi({}^v\phi(c))$ separate the transfer plans according to whether $\phi(c) = \{k\}$ or $\phi(c) \neq \{k\}$. The properties are always initialized with the constraints of the problem only, so that the whole set of valid transfer plans, *i.e.* the whole search space, is represented when the solving procedure starts. For example, for any unit $k \in \mathcal{D}$ and any node $i \in \mathcal{N}$, assuming that α refers to the first contact where $i = r_\alpha$, $\chi_{\underline{D}}({}^v\{k\}, i) = \chi_{\overline{D}}({}^v\{k\}, i) = 0$ **if** $k \in O_i$; $\chi_{\underline{D}}({}^v\{k\}, i) = \alpha$ and $\chi_{\overline{D}}({}^v\{k\}, i) = m$ **if** $k \notin O_i$ and $i \in \mathcal{R}$; $\chi_{\underline{D}}({}^v\{k\}, i) = \alpha$ and $\chi_{\overline{D}}({}^v\{k\}, i) = \infty$ otherwise.

Besides, the transfer graph can also be used to apply deductive elements locally. The properties can be updated to express new knowledge, *e.g.* value $\emptyset$ can be removed from set $\chi_\phi({}^v\phi(c))$ if it is shown that transfer $\phi(c)$ cannot be null in any dominant solution remaining at the current node. This enables dominated transfer plans to be removed where possible.

4. Deductive elements

In this section, we propose several elements of deduction, based either on the dominance rules presented in Section 2 or on the problem itself. Given a transfer graph, we would like to update its properties (χ_A, χ_ϕ, $\chi_{\underline{D}}$ and $\chi_{\overline{D}}$) to reduce the size of the search space (so that infeasible and dominated solutions can be ignored during the search for an optimal valid transfer plan).

4.1. Finding non-valid and non-minimal transfer plans

Let us first consider the following proposition.

Proposition 4.1. *Let $k \in \mathcal{D}$ be a datum unit and $\sigma_c \in \sigma$ be a contact,*

1. [minimality] *Node r_c cannot receive multiple copies of unit k in a minimal solution, which means that it is not necessary to consider the transmission of unit k to node r_c during σ_c if it is known that r_c possesses k before σ_c, i.e. $c > \chi_{\overline{D}}({}^v\{k\}, r_c)$ implies $\{k\} \notin \chi_\phi({}^v\phi(c))$.*
2. [validity] *The validity constraint ensures that the sending node s_c always possesses the datum unit it sends, i.e. $c \leq \chi_{\underline{D}}({}^v\{k\}, s_c)$ implies $\{k\} \notin \chi_\phi({}^v\phi(c))$.*
3. [properties] *Node r_c can receive unit k only if it is allowed to possess this unit, which leads to the following property-consistency rule: $c < \chi_{\underline{D}}({}^v\{k\}, r_c)$ implies $\{k\} \notin \chi_\phi({}^v\phi(c))$.*

These statements can be generalized to any transfer value. For example, in a minimal transfer plan, transfer value $\phi(c)$, *whatever its value*, cannot be forwarded to a node that has already received this element. Hence the corollary.

Corollary 4.1. *Let $\sigma_c \in \sigma$ be a contact and $({}^v\phi(c), {}^vz) \in A$ be an arc out of vertex ${}^v\phi(c)$,*

1. [minimality] $c > \chi_{\overline{D}}({}^vz, r_c) \Longrightarrow \chi_A({}^v\phi(c), {}^vz) = \{\texttt{false}\}$
2. [validity] $c \leq \chi_{\underline{D}}({}^vz, s_c) \Longrightarrow \chi_A({}^v\phi(c), {}^vz) = \{\texttt{false}\}$
3. [properties] $c < \chi_{\underline{D}}({}^vz, r_c) \Longrightarrow \chi_A({}^v\phi(c), {}^vz) = \{\texttt{false}\}$

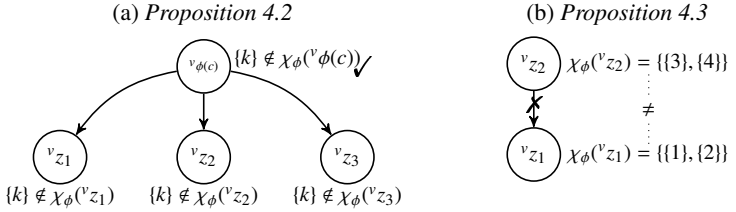


Figure 5: Deduction tools for strengthening the domain χ_ϕ of each transfer.

The following deduction rules are designed to refine the domain χ_ϕ of each transfer, using the information we have about other transfers. Figure 5 illustrates the propositions below. Definition 4.1 is only introduced for the sake of clarity.

Definition 4.1. $\Gamma : V \mapsto V$ *indicates the remaining successors of each vertex, which corresponds to the set of values that transfers can still have in accordance with* χ_A*:*

- $\forall^v\phi(c) \in V_T$, $\Gamma(^v\phi(c)) = \{\, ^vz \in V$ *such that arc* $(^v\phi(c), ^vz) \in A$ *and* $\texttt{true} \in \chi_A(^v\phi(c), ^vz)\,\}$,
- *and* $\forall^v\{k\} \in V_D$*, we set* $\Gamma(^v\{k\}) = \emptyset$ *by convention,*

Proposition 4.2. *For any datum unit* $k \in \mathcal{D}$ *and contact* $\sigma_c \in \sigma$*, transfer* $\phi(c)$ *cannot be equal to value* $\{k\}$ *if none of the transfer values that may be transmitted during contact* σ_c *can have that value, i.e.* $[\forall^vz \in \Gamma(^v\phi(c)), \{k\} \notin \chi_\phi(^vz)]$ *implies* $\{k\} \notin \chi_\phi(^v\phi(c))$*.*

Proposition 4.3. *For any contact* $\sigma_c \in \sigma$ *and any transfer value* z *that may be transmitted during that contact, i.e. such that* $^vz \in \Gamma(^v\phi(c))$*, transfer* $\phi(c)$ *cannot be equal to* z *if the domains of these transfer values are conflicting, i.e.* $\chi_\phi(^vz) \cap \chi_\phi(^v\phi(c)) \subseteq \{\emptyset\}$ *implies* $\chi_A(^v\phi(c), ^vz) = \{\texttt{false}\}$*.*

These propositions and corollaries are the basis for all the deduction algorithms discussed hereafter in this paper. The aim will usually be to make use of bounds $\chi_{\underline{D}}$, $\chi_{\overline{D}}$ and Corollary 4.1 to remove arcs (to set χ_A-properties to $\{\texttt{false}\}$) in the transfer graph, so as to reduce the domain of transfers (the χ_ϕ-properties) using Proposition 4.2. Non-minimal transfers can be removed if it can be proved that the recipient nodes possess all the transfer values the sending nodes are able to send. This way, it can be proved that some transfers are always null in a dominant solution.

The procedure for applying these deductive elements is described in Algorithm 1, whose effectiveness will depend on the quality of bounds $\chi_{\underline{D}}$ and $\chi_{\overline{D}}$. We shall therefore devote the following subsection to consistency procedures designed to strengthen these bounds.

4.2. Elementary reasonings

In this subsection we propose algorithms designed to strengthen bounds $\chi_{\overline{D}}$ and $\chi_{\underline{D}}$. This will enable us to prove that some transfers are null in any dominant solution (*cf.* Subsestion 4.1).

Bottom-up deductive reasoning. In this paragraph, our aim is making use use of knowledge related to earlier contacts (corresponding to the vertices at the bottom of the transfer graph) in order to deduce information about later contacts (located at the top of the transfer graph).

Proposition 4.4. *Let* $i \in \mathcal{N}$ *be a node and* $t \in \{0, 1, \ldots, m\}$ *a time index. Let* $^v\phi(c) \in V_T$ *be a transfer vertex and* $^vz_1, ^vz_2, \ldots, ^vz_\alpha \in \Gamma(^v\phi(c))$ *its successors (transfer values* $z_1, \ldots, z_\alpha \in T_\phi$ *thus correspond to what node* s_c *may transmit to node* r_c *during contact* σ_c*). If* i *possesses* $z_1, z_2, \ldots,$ *and* z_α *at time* t*, then it necessarily also possesses* $\phi(c)$ *at time* t*. Whatever the value chosen for transfer* $\phi(c)$ *from among* $z_1, z_2, \ldots, z_\alpha$*, or* $\emptyset$*, node* i *possesses that element at time* t*. As a result,* $[\forall^vz \in \Gamma(^v\phi(c)), t \geq \chi_{\overline{D}}(^vz, i)]$ *implies that* $\chi_{\overline{D}}(^v\phi(c), i) \leq t$ *holds.*

Algorithm 1 MINIMALITY+VALIDITY CONSISTENCY

Require: transfer graph $G_\phi = (V, A, \chi_A, \chi_\phi, \chi_{\underline{D}}, \chi_{\overline{D}})$;
Require: vertex ${}^v\phi(c)$ to which these deduction rules will be applied ;
1: # *consistency rules in accordance with Proposition 4.1.*
2: **for all** $k \in \mathcal{D}$ **do**
3: **if** $[c > \chi_{\overline{D}}({}^v\{k\}, r_c)] \vee [c < \chi_{\underline{D}}({}^v\{k\}, r_c)] \vee [c \leq \chi_{\underline{D}}({}^v\{k\}, s_c)]$ **then**
4: **remove** $\{k\}$ **from** $\chi_\phi({}^v\phi(c))$;
5: # *R is the union over* ${}^v z \in \Gamma({}^v\phi(c))$ *of sets* $\chi_\phi({}^v z)$ *– computed line 16 and used line 19.*
6: $R \leftarrow \emptyset$;
7: # *each of the successors of vertex* ${}^v\phi(c)$ *is analyzed.*
8: **for all** ${}^v z \in \Gamma({}^v\phi(c))$ **do**
9: # *consistency rules in accordance with Corollary 4.1.*
10: **if** $[c > \chi_{\overline{D}}({}^v\phi(c), r_c)] \vee [c > \chi_{\overline{D}}({}^v z, r_c)] \vee [c < \chi_{\underline{D}}({}^v z, r_c)] \vee [c \leq \chi_{\underline{D}}({}^v z, s_c)]$ **then**
11: **set** $\chi_A({}^v\phi(c), {}^v z) = \{\texttt{false}\}$;
12: # *consistency rules in accordance with Proposition 4.3.*
13: **else if** $\forall \{k\} \in \chi_\phi({}^v\phi(c)), \{k\} \notin \chi_\phi({}^v z)$ **then**
14: **set** $\chi_A({}^v\phi(c), {}^v z) = \{\texttt{false}\}$;
15: # *R is computed traversing* $\Gamma({}^v\phi(c))$.
16: **else add** $\chi_\phi({}^v z)$ **to** R ;
17: **end for**
18: # *consistency rules in accordance with Proposition 4.2.*
19: **for all** $\{k\} \in \chi_\phi({}^v\phi(c)) \backslash R$ **do remove** $\{k\}$ **from** $\chi_\phi({}^v\phi(c))$;
20: # *this contact is "removed" from the model if possible.*
21: **if** $[\Gamma({}^v\phi(c)) = \emptyset] \vee [\chi_\phi({}^v\phi(c)) = \emptyset]$ **then**
22: **set** $\phi(c) = \emptyset$; and **update** the transfer graph:
23: • **set** $\chi_\phi({}^v\phi(c)) = \{\emptyset\}$; • **for all** $i \in \mathcal{N}$ **do set** $\chi_{\underline{D}}({}^v\phi(c), i) = \chi_{\overline{D}}({}^v\phi(c), i) = 0$;
24: • **for all** $a \in A$ leaving or entering ${}^v\phi(c)$ **do set** $\chi_A(a) = \{\texttt{false}\}$;

From a practical point of view, this result leads to Algorithm 2. This algorithm visits a transfer vertex ${}^v\phi(c) \in V_T$ and tries to strengthen the bounds $\chi_{\overline{D}}({}^v z, r_c)$, ${}^v z \in \Gamma({}^v\phi(c))$ associated with the recipient node r_c of transfer $\phi(c)$ and each transfer value $z \in T_\phi$ that could be transmitted during that transfer. In particular, if it is shown that r_c necessarily possesses a transfer value z at time $c-1$ (*i.e.* if $c - 1 \geq \chi_{\overline{D}}({}^v z, r_c)$), then Corollary 4.1 enables us to remove arc $({}^v\phi(c), {}^v z)$ by setting $\chi_A({}^v\phi(c), {}^v z) = \{\texttt{false}\}$. If all arcs going out of ${}^v\phi(c)$ can be removed in this way, then it proves that transfer $\phi(c)$ is null in any dominant solution.

The bounds are refined in accordance with Proposition 4.4 in function **bottom-up-subroutine** with $i = r_c$ and $t = c - 1$. ${}^v z$ still refers to the vertex whose bounds must be refined. This function returns `true` if node $i = r_c$ possesses ${}^v z$ at time $t = c - 1$ and puts a mark on all visited nodes to avoid redundant calculations. If the best known bound $\chi_{\overline{D}}({}^v z', r_c)$ of a vertex $z' \in \Gamma(z)$ is greater than $c - 1$, *i.e.* a condition to deduce that $c - 1 \geq \chi_{\overline{D}}({}^v z', r_c)$ is not fulfilled, we attempt to refine it recursively, *cf.* line 14. This algorithm is depicted in Figure 6.

Remark 4.1. *Although Algorithm 2 traverses vertices in depth, information is propagated from lower to upper vertices. The same outcome might also be achieved by sequentially calling function* ***bottom-up-subroutine*** *with transfer vertices* ${}^v\phi(1), {}^v\phi(2), \ldots,$ *to* ${}^v\phi(c)$*, but this would compel us to visit all the vertices "below" vertex* ${}^v\phi(c)$*, whatever the situation in hand.*

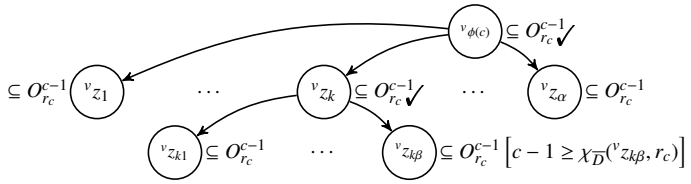


Figure 6: The bottom-up procedure, *cf.* Proposition 4.4.

Algorithm 2 BOTTOM-UP CONSISTENCY

Require: transfer graph $G_\phi = (V, A, \chi_A, \chi_\phi, \chi_{\underline{D}}, \chi_{\overline{D}})$;
Require: vertex ${}^{v}\phi(c)$ to which this deduction rule will be applied ;
1: *# the recursive procedure is initialized.*
2: **unmark** all vertices ; **boolean** $b \leftarrow$ `true` ;
3: *# the recursive procedure is performed over each successor.*
4: **for all** ${}^{v}z \in \Gamma({}^{v}\phi(c))$ **do**
5: $b \leftarrow b$ **and bottom-up-subroutine** $({}^{v}z, r_c, c-1)$;
6: **if** $b =$ `true` **then**
7: *# it has been proved that $\phi(c)$, whatever its value, is possessed by r_c at time $c-1$.*
8: **apply** $\chi_{\overline{D}}({}^{v}\phi(c), r_c) \leq c-1$;

1: **function bottom-up-subroutine** $({}^{v}z \in V, i \in \mathcal{N}, t \in \{0, 1, \ldots, m\})$: **boolean**
2: **if** $t \geq \chi_{\overline{D}}({}^{v}z, i)$ **then**
3: *# transfer value z is possessed by i at time t.*
4: **return** `true` ;
5: **if** ${}^{v}z \in V_D$ **then**
6: *# vertex ${}^{v}z$ represents a datum unit and Proposition 4.4 cannot be applied.*
7: **put** a mark on ${}^{v}z$; and **return** `false` ;
8: *# the successors of ${}^{v}z$ must be analyzed to check whether z is possessed by i at time t.*
9: **for all** ${}^{v}z' \in \Gamma({}^{v}z)$ **such that** $t < \chi_{\overline{D}}({}^{v}z', i)$ **do**
10: **if** ${}^{v}z'$ is marked **then**
11: *# the procedure has already failed to prove that z' is possessed by i at time t.*
12: **put** a mark on ${}^{v}z$; and **return** `false` ;
13: *# the procedure now attempts to prove that z' is possessed by i at time t.*
14: **else if** ${}^{v}z'$ is *not* marked **and bottom-up** $({}^{v}z', i, t) =$ `false` **then**
15: **put** a mark on ${}^{v}z$; and **return** `false` ;
16: **end for**
17: *# it has been proved that z is possessed by i at time t using Proposition 4.4.*
18: **apply** $\chi_{\overline{D}}({}^{v}z, i) \leq t$; and **return** `true` ;
19: **end function**

Top-down deductive reasoning. In this paragraph, our aim is making use of knowledge related to later contacts (and therefore shown at the top of the transfer graph) to deduce information about earlier contacts (located at the bottom of the transfer graph).

For example, let us consider the case where a node $i \in \mathcal{N}$ is the receiver in *only* two contacts $\sigma_{c1} = (s1, i)$ and $\sigma_{c2} = (s2, i)$ (*cf.* Figure 7). Let us assume that sending nodes $s1$ and $s2$ possess two units 1 and $2 \in \mathcal{D}$, *i.e.* $O_{s1}^{c1-1} = O_{s2}^{c2-1} = \{1, 2\}$, and that node i did not possess any units at the

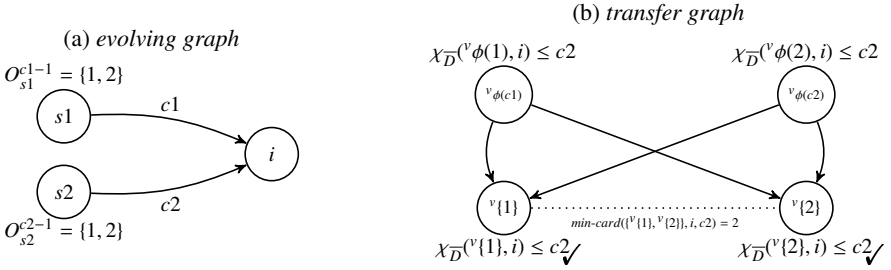


Figure 7: The top-down procedure, *cf.* Proposition 4.5.

outset, *i.e.* $O_i = \emptyset$. Note that a unit is always transmitted during σ_{c1} in a strictly-active transfer plan, and that $\chi_\phi({}^v\phi(c1)) = \{\{1\}, \{2\}\}$ in the transfer graph. This remark holds for contact σ_{c2} too. As the units transmitted to node i must necessarily be different in a minimal transfer plan, we are able to deduce that node i possesses datum units 1 and 2 after these two contacts in all strictly-active minimal transfer plans, *i.e.* $\chi_{\overline{D}}({}^v\{1\}, i) \leq c2$ and $\chi_{\overline{D}}({}^v\{2\}, i) \leq c2$. We have actually shown that node i possesses at least 2 datum units from among subset $Z = \{1, 2\}$ after contact σ_{c2} in any dominant solution. Thus, $\phi(c1) \cup \phi(c2) = \{1, 2\}$ and $Z \subseteq O_i^{c_2}$ ($Z = O_i^{c_2}$ in this case).

The approach can be generalized to any set of transfer values.

Definition 4.2. *Given a set ${}^vZ \subseteq V$ of vertices in the transfer graph, a node $i \in \mathcal{N}$, and a time index $t \in \{0, \ldots, m\}$, min-card$({}^vZ, i, t)$ correspond to a* lower bound *on the smallest number of transfer values $z \in Z$ that node i possesses after the t first contacts in a minimal strictly-active transfer plan (the smallest number of transfer values $z \in Z$ such that $z \subseteq O_i^t$ in such a solution).*

In Section 4.3 we will discuss how *min-card* may be evaluated.

Proposition 4.5. *Let ${}^vZ \subseteq V$ be a set of vertices, $i \in \mathcal{N}$ a node, and let $t \in \{0, 1, \ldots, m\}$ denote a time index. If min-card$({}^vZ, i, t) = |{}^vZ|$ (if node i possesses at least $|{}^vZ|$ transfer values from among a set of $|{}^vZ|$ transfer values at time t), then node i necessarily possesses all the transfer values in Z at time t – that is $[\forall {}^vz \in {}^vZ, \chi_{\overline{D}}({}^vz, i) \leq t]$ holds.*

From a practical point of view, this result leads to Algorithm 3. This procedure visits a transfer vertex ${}^v\phi(c) \in V_T$ and tries to prove that transfer $\phi(c)$ is null in any dominant solution. With this aim in view, it attempts once again to show that any transfer value which could be transferred during contact σ_c is already possessed by node r_c at time $c - 1$ in a dominant transfer plan, that is $\forall {}^vz \in \Gamma({}^v\phi(c)), c - 1 \geq \chi_{\overline{D}}({}^vz, r_c)$. This can sometimes be achieved using Proposition 4.5 with ${}^vZ = \Gamma({}^v\phi(c))$, $i = r_c$ and $t = c - 1$. It involves evaluating *min-card*$(\Gamma({}^v\phi(c)), r_c, c - 1)$.

The algorithm is quite straightforward. It should be noted, however, that strengthening the bound $\chi_{\overline{D}}({}^vz, r_c)$ of a vertex ${}^vz \in V$ can enable a better lower bound *min-card*$(\Gamma({}^vz), r_c, c - 1)$ to be computed, and thus enable Proposition 4.5 to be applied with ${}^vZ = \Gamma({}^v\phi(c))$, $i = r_c$ and $t = c - 1$. The same deductive step is repeated from the upper to the lower vertices, the aim being to find matches between subsets of transfer values which are possessed by node r_c at time $c - 1$ and subsets of transfer values which correspond to former contacts, or ideally to subsets of units.

Strict-activity-based deductive reasoning. Let us now turn to the strict-activity-based deduction rule and first recall that a transfer plan is strictly-active if and only if no transfer that might have been improving is not improving (Definition 2.5). This means that a transfer is always improving

Algorithm 3 Top-Down Consistency

Require: transfer graph $G_\phi = (V, A, \chi_A, \chi_\phi, \chi_{\underline{D}}, \chi_{\overline{D}})$;
Require: vertex ${}^v\phi(c)$ to which this deduction rule will be applied ;
1: # *the recursive procedure is launched at vertex* ${}^v\phi(c)$ *with* $i = r_c$ *and* $t = c - 1$.
2: **top-down** $({}^v\phi(c), r_c, c-1)$;

1: **function top-down** $({}^v\phi(c) \in V_T, i \in \mathcal{N}, t \in \{0, 1, \ldots, m\})$
2: # *computing the smallest number of transfer values* ${}^vz \in {}^vZ$ *possessed by node* i *at time* t.
3: $bound \leftarrow$ *min-card*$({}^vZ, i, t)$;
4: **if** $bound = |{}^vZ|$ **then**
5: # *it is shown that node* i *possesses all transfer values* ${}^vz \in {}^vZ$ *at time* t.
6: $\forall {}^vz \in {}^vZ$, **apply** $\chi_{\overline{D}}({}^vz, i) \leq t$; and $\forall {}^vz \in {}^vZ \cap V_T$, **top-down** $({}^vz, i, t)$;
7: **else if** $bound = |{}^vZ| - 1$ **and** $r_c = i$ **and** $t = c - 1$ **then**
8: # *it is shown that node* r_c *will possess all transfer values* ${}^vz \in {}^vZ$ *after contact* σ_c.
9: $\forall {}^vz \in {}^vZ$, **apply** $\chi_{\overline{D}}({}^vz, i) \leq c$; and $\forall {}^vz \in {}^vZ \cap V_T$, **top-down** $({}^vz, i, c)$;
10: **end function**

if the sending node possesses a unit that the receiving node does not possess. If we look back at the example in Figure 3b [see page 7] for instance, transfer $\phi(3)$ is improving in a strictly-active minimal solution, since node 3 possesses at least 2 units from among $\{1, 2, 3\}$ in such a solution, whereas node 4 only possesses unit 3 when contact σ_3 occurs.

This leads to the following definition and the ensuing proposition.

Definition 4.3. *Given a set* ${}^vZ \subseteq V$ *of vertices in the transfer graph, a node* $i \in \mathcal{N}$*, and a time index* $t \in \{0, \ldots, m\}$*,* $max\text{-}card({}^vZ, i, t)$ *is an* upper bound *on the number of transfer values* $z \in Z$ *that node* i *possesses after the first* t *contacts in a strictly-active minimal transfer plan, that is the greatest number of transfer values* $z \in Z$ *such that* $z \subseteq O_i^t$ *in such a solution.*

In Section 4.3 we will discuss how *max-card* may be evaluated.

Proposition 4.6. *Let* $\sigma_c \in \sigma$ *be a contact, and* ${}^vZ \subseteq V$ *a set of vertices. If node* s_c *necessarily possesses more units than node* r_c *when contact* σ_c *occurs, then transfer* $\phi(c)$ *is always improving in a strictly-active minimal transfer plan. Thus, if* $min\text{-}card({}^vZ, s_c, c-1) > max\text{-}card({}^vZ, r_c, c-1)$*, then* $\emptyset \notin \chi_\phi({}^v\phi(c))$ *in a strictly-active minimal solution.*

This condition is tested with ${}^vZ = V_D$ in practice, *cf.* Algorithm 4.

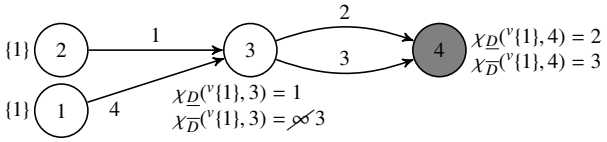
Delivery requirements. This paragraph looks at enforcing the delivery constraint, which states that every node $j \in \mathcal{N}$ has to obtain every unit $k \in \mathcal{D}$ before time $t = \chi_{\overline{D}}({}^v\{k\}, j)\}$ (except where $t = \infty$). To this end, we can utilize contacts $\sigma_c = (i, j)$ which occur before time t between a node $i \in \mathcal{N}$ and j. For $\phi(c) = \{k\}$ to be valid, other conditions have also to be fulfilled: $c > \chi_{\underline{D}}({}^v\{k\}, i)$ (node i has to possess unit k when contact σ_c occurs), $\chi_{\underline{D}}({}^v\{k\}, j) \leq c \leq \chi_{\overline{D}}({}^v\{k\}, j)$ (node j has to be allowed to obtain k at time c), and $\{k\} \in \chi_\phi({}^v\phi(c))$ (transfer $\phi(c) = \{k\}$ has to be allowed).

In consequence, if node j does not possess unit k from the start and if only one node $i \in \mathcal{N}$ is able to transfer k to node j, an implicit constraint forces node i itself to obtain that unit early enough to be transmitted to j on time. For example, in Figure 8, node 3 is the only node that can transmit unit 1 to recipient node 4 within time interval $\{\chi_{\underline{D}}({}^v\{1\}, 4) = 2, \ldots, \chi_{\overline{D}}({}^v\{1\}, 4) = 3\}$. So, node 3 has to obtain unit 1 before contact σ_3 at the latest. This can be formulated as follows.

Algorithm 4 STRICT-ACTIVITY CONSISTENCY

Require: transfer graph $G_\phi = (V, A, \chi_A, \chi_\phi, \chi_{\underline{D}}, \chi_{\overline{D}})$;
Require: vertex ${}^v\phi(c)$ to which this deduction rule will be applied ;
1: *# the two bounds are evaluated – cf. Subsection 4.3 for details.*
2: *min-s* $\leftarrow$ *min-card*$(V_D, s_c, c-1)$; **and** *max-r* $\leftarrow$ *max-card*$(V_D, r_c, c-1)$;
3: *# Proposition 4.5 still holds.*
4: **if** *min-s* $= |V_D|$ **then**
5: **for all** ${}^v\{k\} \in V_D$ **apply** $\chi_{\overline{D}}({}^v\{k\}, s_c) \leq c-1$;
6: *# Proposition 4.6 is then applied.*
7: **if** *min-s* $>$ *max-r* **then remove** $\emptyset$ **from** $\chi_\phi({}^v\phi(c))$;

(a) Node 3 has to obtain unit 1 before time 3 to forward it to node 4.



(b) In consequence, contact 4 occurs too late and contact 1 becomes necessary.

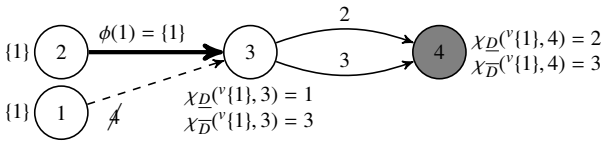


Figure 8: The delivery-requirements procedure – *cf.* Proposition 4.7.

Proposition 4.7. *Let $k \in \mathcal{D}$ be a datum unit, and $j \in \mathcal{N}$ a node that is required to obtain k during the transfer plan, i.e. $k \notin O_j$, $\chi_{\overline{D}}({}^v\{k\}, j) \leq m$. Let $\mathcal{N}^{j,k} = \{\, i \in \mathcal{N}$ such that $\exists \sigma_c = (i, j) \in \sigma$ with $c > \chi_{\underline{D}}({}^v\{k\}, i)$, $c \geq \chi_{\underline{D}}({}^v\{k\}, j)$, $c \leq \chi_{\overline{D}}({}^v\{k\}, j)$ and $\{k\} \in \chi_\phi({}^v\phi(c))\,\}$ denotes the set of nodes that can transmit unit k to node j in a valid transfer plan. If $\mathcal{N}^{j,k} = \{i\}$ is a singleton, that is if only one node i can transmit unit k to node j, then $\chi_{\overline{D}}({}^v\{k\}, i) < \chi_{\overline{D}}({}^v\{k\}, j)$ and $\chi_{\underline{D}}({}^v\{k\}, i) < \chi_{\underline{D}}({}^v\{k\}, j)$ hold. If $\mathcal{N}^{j,k} = \emptyset$, i.e. if there are no nodes able to transmit the unit to node j, then there is no solution fulfilling the set of constraints represented by the transfer graph.*

On Figure 8a, Proposition 4.7 is applied with $k = 1$, $j = 4$ and $i = 3$. The sequence of contacts is $\sigma = [(2, 3), (1, 3), (3, 4), (3, 4)]$. The only predecessor of node 4 is $\mathcal{N}^{4,1} = \{3\}$. Thus, $\chi_{\overline{D}}({}^v\{1\}, 3)$ can be adjusted to $\chi_{\overline{D}}({}^v\{1\}, 4)$. Thereafter, as shown in Figure 8b, the proposition is applied with $j = 4$ and $i = 1$. As contact σ_4 is no longer consistent with $\chi_{\overline{D}}({}^v\{1\}, 3)$, node 2 becomes the only node able to transmit datum unit 1 to node 3. Transfer $\phi(1)$ can even be set to $\{1\}$, because only one contact $\sigma_1 = (1, 3) \in \sigma$ exists.

From a practical point of view, Proposition 4.7 leads to Algorithm 5. This procedure ensures that bounds $\chi_{\underline{D}}$ and $\chi_{\overline{D}}$ are consistent (in the sense of Propositon 4.7) for a given unit $k \in \mathcal{D}$ and all the nodes in $\mathcal{N}$. Note that S is a stack containing the pair of nodes $(i, j) \in \mathcal{N}^2 \,|\, \mathcal{N}^{j,k} = \{i\}$, *i.e.* the nodes $\{j \in \mathcal{N} \,|\, k \notin O_j \wedge \chi_{\overline{D}}({}^v\{k\}, j) \leq m\}$ whose bounds $\chi_{\underline{D}}({}^v\{k\}, j)$ and $\chi_{\overline{D}}({}^v\{k\}, j)$ may need to be strengthened. Instruction **fail** notifies the calling function that there is no solution fulfilling the set of constraints represented by the transfer graph. Thus, if DELIVERY-REQUIREMENTS is used within a branching algorithm, it will prune the current branch and trigger a backtrack.

Algorithm 5 DELIVERY-REQUIREMENTS CONSISTENCY

Require: transfer graph $G_\phi = (V, A, \chi_A, \chi_\phi, \chi_{\underline{D}}, \chi_{\overline{D}})$;
Require: datum unit $k \in \mathcal{D}$ to which this deduction rule will be applied ;
1: # *the set of nodes whose bounds have to be checked is possessed in stack* S.
2: $S \leftarrow \{(i, j) \,|\, \mathcal{N}^{j,k} = \{i\}\}$;
3: # *the bounds are updated in accordance with Proposition 4.7.*
4: **while** $S \neq \emptyset$ **do**
5: $(i, j) \leftarrow S.back()$; $S.pop()$;
6: **apply** $\chi_{\underline{D}}({}^v\{k\}, i) < \chi_{\underline{D}}({}^v\{k\}, j)$ **and** $\chi_{\overline{D}}({}^v\{k\}, i) < \chi_{\overline{D}}({}^v\{k\}, j)$;
7: # *unique transfers are even forced.*
8: **if** $\exists! \sigma_c = (i, j) \in \sigma \,|\, c > \chi_{\underline{D}}({}^v\{k\}, i),\ c \geq \chi_{\underline{D}}({}^v\{k\}, j),\ c \leq \chi_{\overline{D}}({}^v\{k\}, j)$,
9: and $\{k\} \in \chi_\phi({}^v\phi(c))$ **then apply** $\phi(c) = \{k\}$;
10: # *stack* S *is then updated if required.*
11: **update** $\mathcal{N}^{i,k}$; and **push** (z, i) **if** $\mathcal{N}^{i,k} = \{z\}$; or **fail if** $\mathcal{N}^{i,k} = \emptyset$;
12: **end while**

Global deductive elements. The deduction procedures described above are heuristically orchestrated in Algorithm 6. The vertices of the transfer graph are sequentially processed to reduce the domain of possibilities associated with each transfer (*cf.* function χ_ϕ). If the domain of a transfer becomes a singleton, this means that the transfer has been decided. If it becomes empty, or if the other properties are inconsistent, *e.g.* if $\exists {}^v z \in V,\ \exists i \in \mathcal{N} \,|\, \chi_{\underline{D}}({}^v z, i) > \chi_{\overline{D}}({}^v z, i)$, it means that no transfer plan fulfils the constraints represented by the transfer graph.

During each iteration $c \in \{1, 2, \ldots, m\}$, there is first an attempt to refine the bound $\chi_{\overline{D}}({}^v z, r_c)$ of each successor ${}^v z \in \Gamma({}^v\phi(c))$ of vertex ${}^v\phi(c) \in V_T$ in the transfer graph (using both bottom-up and top-down deductions). The aim is to prove that some transfer values which can be transferred by node s_c during contact σ_c are possessed by node r_c when the contact occurs (in any minimal and strictly-active solution). Subsequently, the minimality consistency algorithm is charged with updating χ_A and χ_ϕ. The other consistency algorithms are run independently.

In this way, transfers can often be shown to be null in any dominant solution. The procedure can be repeated as long as changes are occurring in DELIVERY-REQUIREMENTS. Nevertheless, this is seen to be inefficient in practice (most of the transfers being fixed from the first call). We have tested Algorithm 6 as a preprocessing procedure designed to remove fruitless contacts, so that the size m of sequence σ can be reduced before the problem is solved. Besides, we have tested it as a propagation procedure, whose goal is to set more variables during each branching stage of a branch-and-cut algorithm (during the solving of an integer-linear-programming model described in Section 5). However, before going any further, we must discuss how *min-card* and *max-card* bounds can be computed in practice.

4.3. Evaluating maximum and minimum cardinal bounds

In this section, we will look at how to compute an upper and a lower bound on the number of transfer values possessed by a given node at a given time (*cf.* Definitions 4.3 and 4.2). The input data will be one instance of the dissemination problem, one transfer graph (one subset of transfer plans), one subset ${}^v Z \subseteq V$ of its vertices, one node $i \in \mathcal{N}$ and one time index $t \in \{0, \ldots, m\}$.

With regard to *max-card*, note that only the case ${}^v Z = V_D$ will be considered, *i.e.* we will have to find an upper bound of the number of *units* possessed by node i at time t (*cf.* Algorithm 4).

Algorithm 6 GLOBAL CONSISTENCY – HEURISTIC

Require: transfer graph $G_\phi = (V, A, \chi_A, \chi_\phi, \chi_{\underline{D}}, \chi_{\overline{D}})$;

 # the main loop aims to find which transfers are necessarily null.
for $c : 1 \to m$ **do**
  # we try to show transfer values possessed by s_c are possessed by r_c when σ_c occurs.
  BOTTOM-UP $(G_\phi, {}^v\phi(c))$; TOP-DOWN $(G_\phi, {}^v\phi(c))$;
  # the strict-activity consistency procedure is run quite independently.
  STRICT-ACTIVE $(G_\phi, {}^v\phi(c))$;
  # the redundant transfers are finally removed in accordance with the minimality rule.
  MINIMALITY+VALIDITY $(G_\phi, {}^v\phi(c))$;
end for
 # the delivery constraints are then propagated as necessary.
while at least one change occurs **do**
  for all $k \in \mathcal{D}$ **do** DELIVERY-REQUIREMENTS (G_ϕ, k) ;

Let us consider the problem of evaluating *max-card*(V_D, i, t). We recall that the *distinct* datum units (described by V_D) that a node $i \in \mathcal{N}$ can receive before time $t \in \{0, 1, \ldots, m\}$ are associated with as many vertex-disjoint paths in the transfer graph. Each path has to start at a transfer vertex ${}^v\phi(c) \in V_T$ $(c \leq t)$ where $r_c = i$, and end at a unit vertex ${}^v\{k\} \in V_D$. These paths must also fulfil the constraints resulting from functions $\chi_A, \chi_\phi, \chi_{\underline{D}}$ and $\chi_{\overline{D}}$ (*cf.* summary, page 10).

Therefore, by relaxing the constraints expressed with functions $\chi_\phi, \chi_{\underline{D}}$ and $\chi_{\overline{D}}$, the problem in hand can be formulated as the problem of finding the greatest number of arc-and-vertex-disjoint paths from a transfer vertex ${}^v\phi(c) \in V_T \,|\, c \leq t$ and $r_c = i$ to a unit vertex ${}^v\{k\} \in V_D$. This search is limited to the subgraph defined by χ_A (the arcs $a \in A \,|\, \texttt{true} \in \chi_A(a)$). To solve the problem, we compute a maximum flow in the following graph $G = (X, U)$ (*cf.* Figure 9, instance Figure 3):

1. we consider a source vertex *source* $\in X$ and a sink vertex *sink* $\in X$,
2. with each vertex ${}^vz \in V$ in the transfer graph, we associate two vertices z_s and $z_e \in X$, plus one arc $(z_s, z_e) \in U$ between these two vertices, in the transportation network,
3. with each arc $a = ({}^vx, {}^vy) \in A$ where $\texttt{true} \in \chi_A(a)$ in the transfer graph, we associate an arc $(x_e, y_s) \in U$ in the transportation network,
4. we add an arc from the source node *source* to each vertex $z_s \in X$ which is associated with a transfer value possessed by node r_c at time t (where $\chi_{\overline{D}}({}^vz, i, t) \leq t$ in the transfer graph),
5. we add an arc from each unit vertex $\{k\}_e \in X$ to the sink node *sink*,
6. a node $z \in X$ that is not descendant of *source*, or not ascendant of *sink*, is removed,
7. the capacity of each arc $a \in U$ is finally set to 1.

Flows represent arc-disjoint paths (as all capacities are set to 1) and define a valid assignment of the transfer values possessed by node i at time t. The value of each transfer $\phi(c)$, $c \in \{1, \ldots, m\}$ is unambiguous (the paths are vertex disjoint) as at most one unit can flow out of vertex $\phi(c)_e \in X$. The constraints expressed by χ_A are fulfilled by construction. Hence the proposition below.

Proposition 4.8. *Let f_M be the maximum flow through G. f_M = max-card (V_D, i, t) is an upper bound on the greatest number of datum units that node $i \in \mathcal{N}$ possesses at time t.*

Strict-activity and minimality constraints are not relevant when computing *max-card*$({}^vZ, i, t)$, since redundant and postponed transfers can only lead to a reduction in the number of transfer values received by node i.

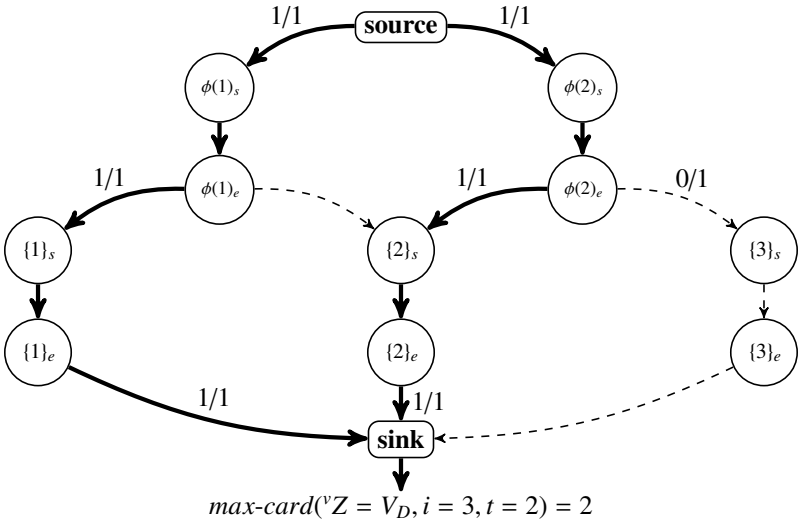


Figure 9: Example of a transportation network used to evaluate *max-card*.

Let us now consider the problem of evaluating *min-card*$({}^vZ, i, t)$. It might be tempting to use the same approach and to try to transform the smallest cardinal bound problem into a well-known flow-based problem. Unfortunately, this does not work. In such a situation, the best solution will always be the null transfer plan in which no units are transferred at all, and consequently the lower bound will always be null. To get better bounds, solutions must be required to be minimal and strictly-active (which prevents unjustified null transfers occurring). Unfortunately, this kind of constraint cannot be introduced into a flow problem. It can even be proved that computing an *exact* value *min-card*$({}^vZ, i, t)$, *i.e.* the *precise* number of transfer values in ${}^vZ$ that node i possesses at time t in the worst case, is a strongly NP-hard problem if solutions have to be strictly-active.

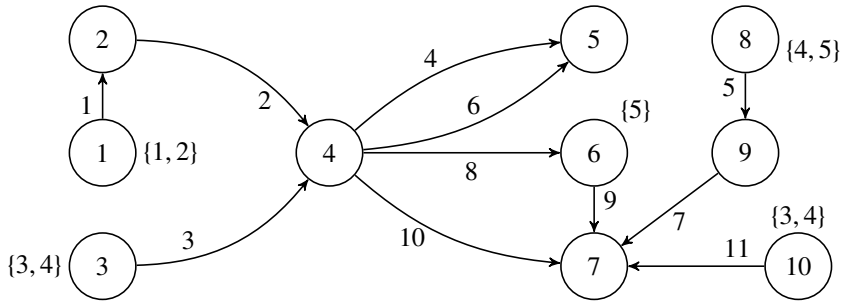
Since we were unable to find a satisfactory heuristic for the problem, we propose a procedure for polynomially transforming an instance of the smallest cardinal bound problem, where ${}^vZ \subseteq V_D$ does not necessarily hold, into another instance where ${}^vZ \subseteq V_D$ holds, *i.e.* where *datum units only* need to be considered. In this case, the integer-linear-programming model defined in Section 5 can be adapted to our needs (with few changes). The evaluation of *min-card* in the transformed instance also gives a lower bound of *min-card*$({}^vZ, i, t)$ in the original instance.

Unfortunately, the transformation is not always relevant and the method is then limited to some cases that we describe below.

Let us consider the instance depicted in Figure 10. We wish to compute the smallest number of transfer values from among set ${}^vZ = \{\phi(1), \phi(3)\}$ possessed by node $i = 5$ after the first $t = 6$ contacts in any dominant solution, *i.e. min-card*$(\{{}^v\phi(1), {}^v\phi(3)\}, 5, 6)$.

As we stated above, in order to evaluate *min-card* (using integer linear programming) we wish to reduce the original problem to an easier problem where ${}^vZ \subseteq V_D$. With this goal in mind, we propose adding a *virtual* datum unit $\{A\}$ to represent transfer value $\phi(1)$. $\{A\}$ is then transmitted to node 1 at time 1 (like $\phi(1)$) via a fictitious source node s_1, while contact σ_1 is removed from the instance. Transfer $\phi(3)$ is replaced by virtual datum unit $\{B\}$ in the same way. Thereafter, any contact (resp. node) whose representative arc (resp. vertex) does not belong to a journey leading to node 5, or which occurs after contact σ_6, is removed from the instance, since it cannot help node 5 to obtain new datum units on time. The new instance is depicted in Figure 11a.

(a) *the evolving graph*



(b) *the associated transfer graph*

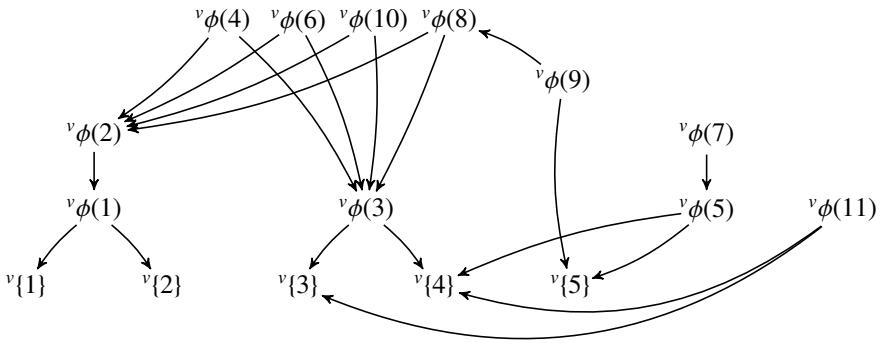


Figure 10: An instance for which we want to extract knowledge about cardinals.

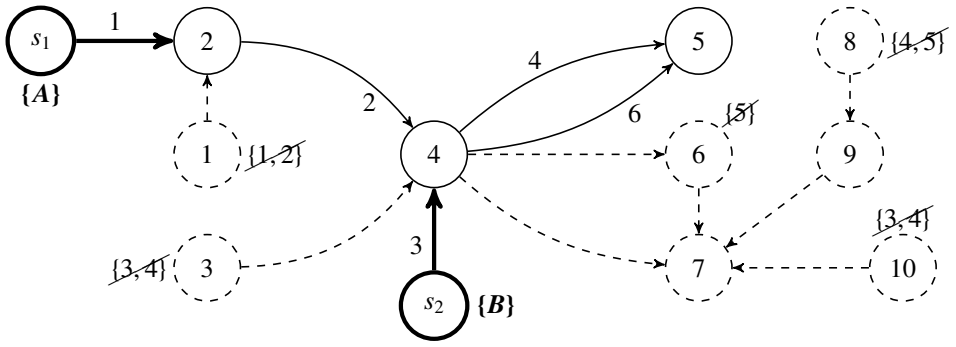
min-card($\{^v\phi(1), ^v\phi(3)\}, 5, 6$) in the original instance is equal to *min-card*($\{^v\{A\}, ^v\{B\}\}, 5, 6$) = 2 in the new instance (it is possible to compute this value because we are only considering datum units). Node 2 receives unit $\{A\}$ instead of transfer value $\phi(1)$ (during σ_1), but both represent one unit in the set $\{\{1\}, \{2\}\}$. Thereafter, node 4 receives $\{B\}$ instead of transfer value $\phi(3)$ (during σ_3), but both represent one unit in the set $\{\{3\}, \{4\}\}$. Since the rest of the instance is not changed, units $\{A\}$ and $\{B\}$ play the same role as transfer values $\phi(1)$ and $\phi(3)$ in the original instance. Therefore, $\phi(1) \subseteq O_5^6$ and $\phi(2) \subseteq O_5^6$ hold in all dominant solutions.

In the general case, because of the transfer graph, the transformations can be automated using the following procedure :

1. To "replace" a transfer $\phi(c)$ by a *virtual* unit, it is sufficient to remove all the descendants of vertex $^v\phi(c)$ in the original transfer graph, so that it becomes a leaf in the new transfer graph (a vertex representing a datum unit). This virtual datum unit thus aggregates all the choices that need to be made regarding the removed vertices. This operation is repeated for all the vertices $^vz \in {}^vZ \cap V_T$. This way, all the transfer values in ^vZ are units in the new instance.
2. Thereafter, the deletion of the irrelevant entities consists in removing the contacts occurring too late ($\{^v\phi(c) \in V_T \mid t < c\}$), together with the transfer values whose representative vertex is not a descendant of a vertex associated with node i in the transfer graph (*i.e.* $\{\, ^vz \in V$ such that $\nexists ^v\phi(c') \in V_T,\ c' \leq t,\ r_{c'} = i$ and ^vz is a descendant of $^v\phi(c')\,\}$).

This transformation is shown in Figure 11b.

(a) *the evolving graph*



(b) *the associated transfer graph*

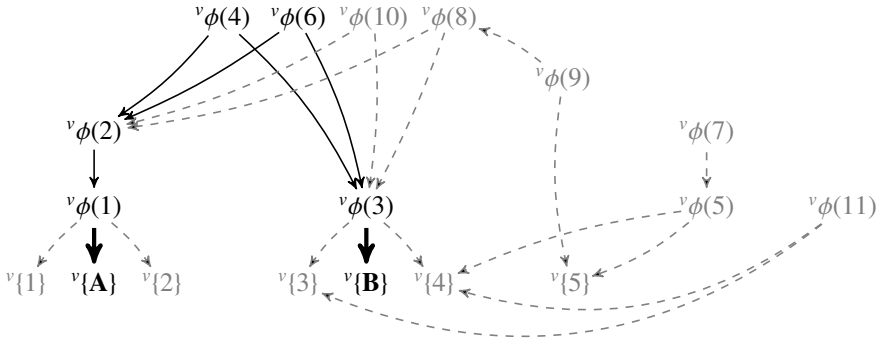


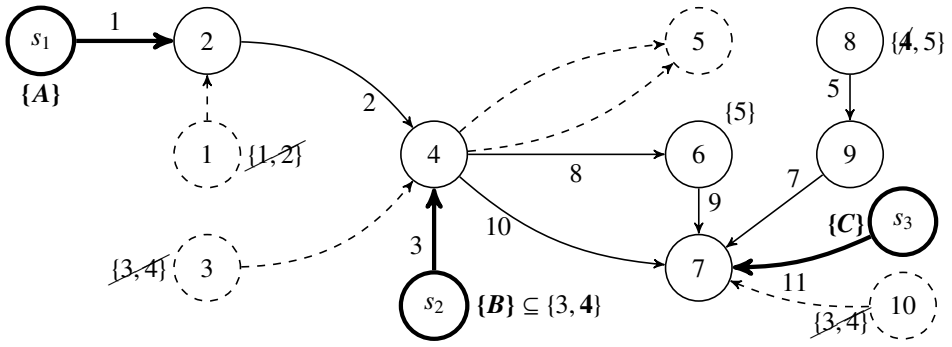
Figure 11: A valid transformation for computing *min-card*($\{^v\phi(1), ^v\phi(3)\}, 5, 6$).

These operations can transform other transfer vertices into leaves. For example, in Figure 12, the transformation of transfer $\phi(3)$ into unit $\{B\}$ also transforms transfer $\phi(11)$ into unit $\{C\}$.

In some cases, not all of the successors of a vertex are removed – see vertices $^v\phi(5)$ and $^v\{4\}$ for example. In such situations, the evaluation of *min-card* in the transformed instance does not lead to a lower bound of *min-card* in the original instance: extra transformations are required to solve the problem. The set of valid choices concerning these transfers has been implicitly reduced and some solutions are thus ignored. In this example, it would imply $\phi(5) = \phi(7) = \{5\}$, and then $\phi(8) = \phi(9)$. We would therefore find $\phi(9) = \{A\}$, $\phi(10) = \{B\}$, or vice versa, and finally conclude that bound *min-card*($\{^v\phi(1), ^v\phi(3)\}, 7, 11) = 2$. Nevertheless, the minimal strictly-active solution $\phi(1) = \phi(2) = \phi(8) = \{1\}$, $\phi(3) = \phi(10) = \{3\}$, $\phi(5) = \phi(7) = \{4\}$, $\phi(9) = \{5\}$, $\phi(11) = \emptyset$ proves that node 7 can possess fewer than 2 units at time 11 in a dominant transfer plan. The best bound is therefore *min-card*($\{^v\phi(1), ^v\phi(3)\}, 7, 11) = 1$, *e.g.* with $\phi(3) \subseteq O_5^{11}$ and $\phi(1) \not\subseteq O_5^{11}$.

For the bound to computed consistently, transfer $\phi(5)$ needs to be transformed into a fourth virtual unit $\{D\}$ by removing the remaining descendant $^v\{5\}$ of vertex $^v\phi(5)$. In practice, such cases are never addressed, since they generally lead to *complex transformations* (triggered in chain) and *poor bounds*. Every transformation is in fact a relaxation of some minimality constraints, in the sense that *dependent* sets of choices are being assimilated to *independent* units.

(a) *the evolving graph*



(b) *the associated transfer graph*

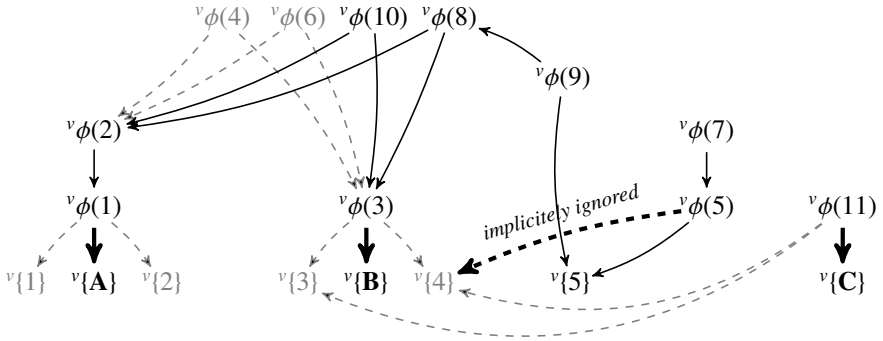


Figure 12: An inconsistent transformation for computing *min-card*($\{^v\phi(1), ^v\phi(3)\}, 7, 11$).

For example, the transformation of transfers $\phi(3)$ and $\phi(11) \subseteq \{3, 4\}$ into virtual units $\{B\}$ and $\{C\}$ implicitly allows that $\phi(3) = \phi(11)$, although $r_3 = r_{11} = 10$. Note that the cases in which two transfer values in ^vZ have hierarchical relationships are not addressed either. Transforming ancestors implies deleting others, so that the target value *min-card*($^vZ, i, t$) = $|^vZ|$ becomes unattainable from the start.

Proposition 4.9. *Let ^vZ_T denote the set of virtual datum units in the transformed instance which corresponds to the set of transfer values ^vZ in the original instance. Let min-card$_T$($^vZ_T, i, t$) refer to the smallest number of transfer values in Z_T that node i possesses at time t in a dominant transfer plan (computed in the transformed instance). min-card$_T$($^vZ_T, i, t$) = min-card($^vZ, i, t$) is a lower bound of the smallest number of transfer values possessed by node i after the first t contacts in the original instance.*

In the following section we propose an integer-linear-programming model designed to solve the dissemination problem. It is similar to the model we use to compute *min-card$_T$*($^vZ_T, i, t$).

5. Solving the Dissemination Problem

In Section 2 we proposed several dominance rules which enable the search space to be reduced. Deduction procedures based on these results were discussed in Sections 3 and 4. In this section we propose an integer-linear-programming model to solve the problem.

5.1. Integer linear programming

The integer-linear-programming model (ILP) we propose is quite straightforward. The model is based on time-indexed boolean variables that describe a transfer plan. For each node $i \in \mathcal{N}$, we define $\mathcal{T}_i = \{0\} \cup \{c \in \{1, \ldots, m\} \,|\, r_c = i\}$, the set of time indexes at which the state of node i can change, *i.e.* at which node i can receive a datum unit. $\mathcal{T}_i(t)$, $t \in \{0, 1, \ldots, m\}$, refers to the last contact occurring before time t where node i is the receiver, *i.e.* $\mathcal{T}_i(t) = \max\{t' \in \mathcal{T}_i \,|\, t' \leq t\}$.

The variables are defined as follows:

- $\forall k \in \mathcal{D}$, $\forall c \in \{1, 2, \ldots, m\}$, $x_{k,c} = 1$ if datum unit k is transmitted from node s_c to node r_c during contact σ_c, and $x_{k,c} = 0$ otherwise.
- $\forall i \in \mathcal{N}$, $\forall k \in \mathcal{D}$, $\forall t \in \mathcal{T}_i$, $y_{i,k,t} = 1$ if node i possesses datum unit k after contact σ_t, and $y_{i,k,t} = 0$ otherwise.
- $\forall t \in \{0, 1, \ldots, m\}$, $z_t = 1$ if a node $i \in \mathcal{R}$ is not completely served after the first t contacts, and $z_t = 0$ otherwise.

Remark 5.1. *y-variables are indexed with sets $\mathcal{T}_i$ instead of set $\{0, 1, \ldots, m\}$. However, we may easily know whether a node $i \in \mathcal{N}$ possesses a datum unit $k \in \mathcal{D}$ at any index $t \in \{0, 1, \ldots, m\}$. Indeed, $y_{i,k,\mathcal{T}_i(t)} = 1$ if and only if node i possesses datum unit k after the first t contacts.*

The model thus contains um x-variables, $u.(2m + q)$ y-variables and $m + 1$ z-variables.

Minimizing the dissemination length leads to the following objective function.

$$\lambda^* = \min \sum_{t=0}^{m} z_t \tag{1}$$

Given a time index $t \in \{0, \ldots, m\}$, variable z_t is null if and only if all the recipient nodes have been served at time t, *i.e.* $\forall i \in \mathcal{R}$, $\forall k \in \mathcal{D}$, $y_{i,k,\mathcal{T}_i(t)} = 1$. Hence the following constraints.

$$\forall i \in \mathcal{R},\ \forall t \in \mathcal{T}_i,\ \forall k \in \mathcal{D},\ z_t \geq 1 - y_{i,k,t} \tag{2}$$

Equations (2) can be aggregated in different ways:

$$\forall t \in \{0, 1, \ldots, m\},\ \forall i \in \mathcal{R},\ z_t \geq 1 - \frac{1}{u} \sum_{k \in \mathcal{D}} y_{i,k,\mathcal{T}_i(t)} \tag{2-a}$$

$$\text{or / and } \forall t \in \{0, 1, \ldots, m\},\ \forall k \in \mathcal{D},\ z_t \geq 1 - \frac{1}{|\mathcal{R}|} \sum_{i \in \mathcal{R}} y_{i,k,\mathcal{T}_i(t)} \tag{2-b}$$

$$\text{or / and } \forall t \in \{0, 1, \ldots, m\},\ z_t \geq 1 - \frac{1}{u.|\mathcal{R}|} \sum_{i \in \mathcal{R}} \sum_{k \in \mathcal{D}} y_{i,k,\mathcal{T}_i(t)} \tag{2-c}$$

The efficiency of the variants will be discussed in Section 6.

Other constraints bind the x-variables (the decision variables) to y and z-variables (the auxiliary variables) and also ensure that each constraint is respected (*e.g.* the transfer plan must be valid and the recipient nodes must be served):

- All recipient nodes have to be served before the end of the time horizon:

$$z_m = 0 \tag{3}$$

- Each node $i \in \mathcal{N}$ initially possesses a subset O_i of datum units:

$$\forall i \in \mathcal{N},\ \forall k \in \mathcal{D} \,|\, k \in O_i,\ y_{i,k,0} = 1 \tag{4}$$
$$\forall i \in \mathcal{N},\ \forall k \in \mathcal{D} \,|\, k \notin O_i,\ y_{i,k,0} = 0 \tag{5}$$

- The transfer plan must be valid, *i.e.* sending nodes possess the datum unit they transfer:

$$\forall k \in \mathcal{D},\ \forall c \in \{1, 2, \ldots, m\},\ x_{k,c} \leq y_{s_c,k,\mathcal{T}_i(c-1)} \tag{6}$$

- Nodes possess a datum unit from the time they receive it:

$$\forall k \in \mathcal{D},\ \forall c \in \{1, \ldots, m\},\ y_{r_c,k,c} \leq y_{r_c,k,\mathcal{T}_i(c-1)} + x_{k,c} \tag{7}$$
$$\forall k \in \mathcal{D},\ \forall c \in \{1, \ldots, m\},\ y_{r_c,k,c} \geq 1/2 \left[y_{r_c,k,\mathcal{T}_i(c-1)} + x_{k,c} \right] \tag{8}$$

- At most one datum unit can be transferred during each contact:

$$\forall c \in \{1, \ldots, m\},\ \sum_{k \in \mathcal{D}} x_{k,c} \leq 1 \tag{9}$$

5.2. Additional constraints

Together, constraints (7) and (8) ensure that variable $y_{r_c,k,c}$ is equal to 1 if $y_{r_c,k,\mathcal{T}_i(c-1)} = 1$ or $x_{k,c} = 1$, and to 0 otherwise. Node r_c possesses unit k after contact σ_c if it already possessed unit k after contact σ_{c-1} or if it received the unit during contact σ_c. Yet, if we seek a minimal transfer plan, since both conditions cannot be true at the same time, constraints (7) and (8) can be replaced by the following constraint:

$$\forall k \in \mathcal{D},\ \forall c \in \{1, 2, \ldots, m\},\ y_{r_c,k,\mathcal{T}_i(c-1)} + x_{k,c} = y_{r_c,k,c} \tag{10}$$

It is also possible to strengthen the model such that transfer plans are active, but unfortunately this leads to complex equations and poor numerical results in practice. However, it is quite easy to guarantee that solutions are strictly-active. This can be achieved by ensuring that transfer plans are minimal and that any transfer $\phi(c)$ cannot be null if the sending node s_c possesses at least one datum unit that the receiving node r_c does not possess (since $\phi(c) \neq \emptyset$ and $O^c_{r_c} > O^{c-1}_{r_c}$ are equivalent in a minimal transfer plan), *i.e.*

$$\forall k \in \mathcal{D},\ \forall c \in \{1, \ldots, m\},\ y_{s_c,k,\mathcal{T}_{s_c}(c-1)} - y_{r_c,k,\mathcal{T}_{r_c}(c-1)} \leq \sum_{k' \in \mathcal{D}} x_{k',c} \tag{11}$$

Altogether, the model contains $1 + uq + m + 4um$ constraints – including constraints (2) through (11) (except constraints (2-a) through(2-c) and constraints (7)+(8)).

Below we discuss the effectiveness of preprocessing procedures, *cf.* Section 4, and we compare the different models proposed in Section 5.

6. Computational Results

In this section, computational results are reported and discussed. In Section 6.1, the algorithms used to generate benchmarks are described. Instance classes are built with respect to the number of nodes q and the number of units u (the number of contacts m is chosen such that most of the instances are feasible). The models described in Section 5 are assessed in Section 6.2, while the preprocessing procedures proposed in Sections 3 and 4 are discussed in Section 6.3.

6.1. Generation of benchmarks

In this section, we describe a destruction / construction algorithm to generate difficult instances for the problem. Please not that these instances are available on our website [15].

Random instances are generally easy to solve. That is why we developed an iterative procedure which attempts to increase the "hardness" of an instance by renewing the least relevant contacts, *i.e.* the contacts removed by a trivial preprocessing procedure. Initialized with a random instance, the process is intended to generate harder and harder instances by removing irrelevant contacts, so that new ones can be added elsewhere without increasing the size of the problem (specifically the number of contacts). Note that a few random contacts are also renewed to introduce diversity in the instances visited during the process.

At each iteration, the instance is solved by a given solver (in practice we use the branch-and-cut algorithm described in Subsection 5.1). After a given number of iterations, the instance that required the most CPU time is returned (this is assumed to be the most "difficult" instance).

The generated instances were classified by difficulty with respect to the number of nodes q and units u. The number of contacts m was chosen accordingly. Thereafter, the instances were solved using a variety of solvers, *e.g.* a MIP-based solver with and without a preprocessing procedure, and the union of the twenty hardest instances obtained for each solver was retained (the instances which could not be solved by at least one of these solvers were ignored).

184 *instances* were selected in this way, resulting in eight *classes*. These classes are described by the number *nbinst* of instances they contain, the number u of units and the number q of nodes characterizing these instances. The average number of receivers $\overline{rec} = |\mathcal{R}|$, the average number of sources $\overline{src} = |\{i \in \mathcal{N} \,|\, O_i \neq \emptyset\}|$ and the average number of contacts $\overline{m}$ of these instances are also reported. The eight classes were *grouped* by difficulty. The first group contains the easiest instances. The second and third group contain the instances having respectively many nodes but few datum units, and few nodes but many datum units.

	name	*nbinst*	u	q	$\overline{rec}$	$\overline{src}$	$\overline{m}$
1	**3u10n**	36	3	10	10	1	135
	4u20n	41	4	20	18	1	366
2	**4u50n**	26	4	50	39	1	710
	4u100n	20	4	100	87	2	1720
	5u50n	23	5	50	50	1	726
3	**10u10n**	16	10	10	6	2	197
	50u10n	16	50	10	6	2	750
	100u10n	6	100	10	7	4	2000

In the following subsections, we provide and discuss some computational results that show the efficiency of our solving algorithms. These computations were performed on a server equipped

with 16×6 cores (each running at 2.67Ghz) and 1To RAM. All the algorithms were implemented via the `C++` programming language. MILP are solved by CPLEX, *i.e.* a commercial solver developed by IBM-ILOG. The multithreading features proposed by the library were been deactivated, since the use of concurrent optimizers led to unstable results. For example, different executions of the same computations gave very different results in terms of CPU time, which prevents us making reliable comparisons between our methods. Every instance was therefore solved using a pure-sequential and *deterministic* algorithm with a one-hour time-limit.

The same information is to be found in the different tables of results, namely:

1. *solved* (-%) indicates the ratio of instances solved by the given solver.
2. $feas$ (-%) indicates the ratio of instances that remained unsolved but for which the solver found at least one feasible solution within the one-hour time-limit.
   Note that $100\% - solved - feas$ (-%) therefore indicates the ratio of instances for which the solver neither found a feasible solution nor showed the instance to be infeasible.
3. gap (-%) indicates the average relative gap between the best known lower bound and the best feasible solution. This metric concerns only the $feas\%$ of the instances which remained unsolved but for which the solver found at least one feasible solution.
4. cpu (-s) indicates the average solution time for the given solver, including all instances and thus all time-limits. cpu therefore tends to 3600s if *solved* tends to 0.0%.

Remark 6.1. *In each section, different sets of parameters are compared and the best strategies selected. In our opinion, selections should be made with respect to the three groups rather than the eight classes. In this way, the benchmark cannot be learned by heart.*

6.2. About the integer-linear-programming model

The most trivial model is given in Section 5.1. This MILP-based model is defined by equations (1) through (9), and is termed **std** in the following. To decide which constraint from among (2), (2-a), (2-b) and (2-c) is most suitable, we compare these four cases, *cf.* Table 1.

There is no clear dominance between the four models. Indeed, in every group, the best results per class are always obtained with *different* models. The influence of the parameter appears to be quite random at a first glance. Nevertheless, we decided to use constraint (2-a) in the following, because it leads to the best results on average, *i.e.* across all classes.

Model **std** can be refined. If transfer plans are required to be minimal, constraints (7) and (8) can be replaced by constraint (10), giving rise to a new model denoted as **min**. If transfer plans are required to be minimal and strictly-active, constraint (11) can be added to model **min**, giving rise to a model **min+st/act**. The results obtained with these models are reported in Table 1.

Model **min+st/act** is seen to outperform both **std** and **min** in the first group, while model **min** dominates the two others on bigger instances. None of them, however, gives convincing results, and limits appear as the number of nodes and, more importantly, the number of units increases.

As we will confirm below, the difficulty of instances is better explained by the number of units than by the number of nodes. With a solving rate greater than 90% for the three solvers, the instances in group 2 (characterized by many systems but few units) seem manageable while, instances of the third group (few systems but many datum units) seem out of reach for the moment. This being said, we could state that **min** is the best model out of the three, because it is the only one that successfully solves some of the most difficult instances. However, this model's disappointing results when dealing with easier classes, *e.g.* **3u10n** and **4u20n**, make us reluctant to make such a categorical statement.

		std // constraint (2)				**std // constraint** (2-a)			
	name	*solved*	*feas*	*cpu*	*gap*	*solved*	*feas*	*cpu*	*gap*
1	**3u10n**	100	-	0.80	-	100	-	0.64	-
	4u20n	100	-	9.9	-	100	-	12.3	-
2	**4u50n**	100	-	39.7	-	100	-	49.6	-
	4u100n	100	-	148	-	100	-	224	-
	5u50n	91.3	8.7	520	9.2	95.7	4.3	332	12.6
3	**10u10n**	50.0	25.0	2362	21.8	68.8	12.5	1614	22.2
	50u10n	0.00	12.5	3587	2.7	0.00	6.3	3585	1.1
	100u10n	0.00	0.00	3593	-	0.00	0.00	3592	-
	avg	82.6	4.3	724	13.8	84.8	2.2	645	14.5
		std // constraint (2-b)				**std // constraint** (2-c)			
	name	*solved*	*feas*	*cpu*	*gap*	*solved*	*feas*	*cpu*	*gap*
1	**3u10n**	100	-	1.1	-	100	-	0.96	-
	4u20n	100	-	17.3	-	100	-	19.6	-
2	**4u50n**	100	-	84.1	-	100	-	50.8	-
	4u100n	100	-	110	-	100	-	165	-
	5u50n	95.7	4.3	398	14.5	95.7	4.3	392	12.5
3	**10u10n**	50.0	25.0	2087	16.5	56.3	18.8	1969	17.2
	50u10n	0.00	18.8	3586	1.8	0.00	18.8	3586	8.2
	100u10n	0.00	0.00	3589	-	0.00	0.00	3589	-
	avg	83.2	4.3	688	10.7	83.7	3.8	679	12.7
		min // constraint (2-a)				**min+st/act // constraint** (2-a)			
	name	*solved*	*feas*	*cpu*	*gap*	*solved*	*feas*	*cpu*	*gap*
1	**3u10n**	100	-	0.91	-	100	-	0.30	-
	4u20n	100	-	14.1	-	100	-	4.2	-
2	**4u50n**	100	-	30.2	-	100	-	50.7	-
	4u100n	95.0	5.0	240	4.6	90.0	10.0	919	10.6
	5u50n	95.7	4.3	266	14.0	95.7	4.3	696	3.3
3	**10u10n**	81.3	12.5	1317	20.1	93.8	6.3	706	10.2
	50u10n	56.3	18.8	2563	2.6	0.00	6.3	3591	16.5
	100u10n	33.3	16.7	3116	0.28	0.00	0.00	3597	-

Table 1: Computational results achieved with CPLEX and different models.

These three options (**std**, **min** and **min+st/act** with constraint (2-a)) will thus continue to be considered in the next section. The superiority of model **min** will then be clearly established.

6.3. About the preprocessing procedures

The main conclusion of the previous section is probably that standard solvers show their limits quite soon for this kind of problem. The deduction algorithms proposed in Section 4 significantly

improve the solving performances when applied within a preprocessing procedure whose role is to remove useless contacts or, conversely, to detect unavoidable transfers. To show this, we will now investigate different procedures that are based on Algorithm 6, *cf.* Section 4.2.

Let us recall that Algorithm 6 is based on a model known as the *transfer graph* that encapsulates knowledge about each transfer $\phi(c)$, $c \in \{1, \ldots, m\}$, and each state O_i^t, $i \in \mathcal{N}$, $t \in \{0, \ldots, m\}$. We are then able to apply different dominance-rule-based propagation algorithms to deduce new knowledge and, in particular, to reveal the transfers that are always null or improving in a strictly-active minimal transfer plan. The main routine processes each transfer $\phi(c)$, $c \in \{1, \ldots, m\}$. First, it calls Bottom-Up and Top-Down to determine which datum units possessed by node s_c have also been received by node r_c. Then, Minimality+Validity is charged with using the eponymous dominance rule in order to avoid redundant transfers. Strict-Active and Delivery-Requirements also detect transfers that are necessary or impossible. The knowledge, thus enriched, can finally be passed on to the mathematical model using polynomial algorithms, *i.e.* some variables can be fixed at the root node, *e.g.* $x_{k,c} = 0$, $\forall k \in \mathcal{D}$, if $\phi(c)$ is null in all dominant solutions.

As usual, a tradeoff has to be found between the time spent running the preprocessing procedure and the time saved during the branch-and-cut procedure as a result of the preprocessing. This is especially true for routines Top-Down and Strict-Active, since both generally consume a significant amount of time solving the cardinal-related subproblems discussed in Section 4.3. To empirically evaluate the compromise, we propose five more or less aggressive strategies, that we have labelled **minimal**, **light**, **normal**, **aggressive** and **maximal**. These strategies are built with the intuitive idea that consistency algorithms are more efficient on earlier contacts than on later ones, because cardinal subproblems involving few transfers are much smaller. We therefore vary the number of contacts after which the most computationally costly routines are skipped.

Each strategy will therefore be characterized by the maximum amount *pre.tl* of time that can be spent in the preprocessing procedure, the ranges *td.range* and *sa.range* of contacts to which the Top-Down and Strict-Active routines will respectively be applied, and the maximum number *dr.lim* of times the while-loop associated with Delivery-Requirements can be run. The functions evaluating *min-card* and *max-card*, *cf.* Section 4.3, are limited in time by parameters *minc.tl* and *maxc.tl* respectively. Bottom-Up and Minimality+Validity are called without any restrictions.

The strategies are described below. Note that *td.range* = 0.25 means that top-down deductive reasoning is applied to the first quarter of transfers (from $c = 0$ to $c = \lfloor 0.25 \times m \rfloor$). The numerical results obtained using the different strategies are reported in Tables 2 and 3. The columns denoted by *rem* and fcd respectively indicate the average ratio of contacts which have been *removed*, *i.e.* necessarily null, or alternatively *forced*, *i.e.* compelled to be improving or even fixed. Thus, *rem* and fcd measure the efficiency of the different procedures. They should be considered together with the average amount *prep* of time required to execute the procedures.

	minimal	**light**	**normal**	**aggr.**	**maximal**
pre.tl (-s)	25	25	25	600	2400
do.bottom.up	yes	yes	yes	yes	yes
do.minimality	yes	yes	yes	yes	yes
td.range	-	0.50	1.00	1.00	1.00
sa.range	-	0.15	0.50	1.00	1.00
dr.lim	2	2	2	4	4
minc.tl	-	0.06s	0.06s	0.10s	0.10s
maxc.tl	-	0.20s	0.20s	0.30s	0.30s

		standard		min		min+st/act		efficiency		
	name	*solved*	*cpu*	*solved*	*cpu*	*solved*	*cpu*	*prep*	*rem*	*fcd*
1	***3u10s***	100	0.70	100	0.78	100	0.26	0.00	15.2	-
	4u20s	100	8.6	100	11.3	100	3.3	0.01	4.9	-
2	***4u50s***	100	31.3	100	33.5	100	46.5	0.02	5.6	-
	4u100s	95.0	237	95.0	230	95.0	600	0.08	5.1	-
	5u50s	100	270	95.7	270	95.7	433	0.02	2.3	-
3	***10u10s***	62.5	1819	87.5	997	81.3	956	0.00	9.8	-
	50u10s	0.00	3589	56.3	2343	0.00	3593	0.05	5.5	-
	100u10s	0.00	3590	33.3	3208	0.00	3591	0.29	1.9	-

strategy **minimal**

		standard		min		min+st/act		efficiency		
	name	*solved*	*cpu*	*solved*	*cpu*	*solved*	*cpu*	*prep*	*rem*	*fcd*
1	***3u10s***	100	0.48	100	0.47	100	0.52	0.42	49.0	5.3
	4u20s	100	2.2	100	2.0	100	2.6	1.4	26.9	6.7
2	***4u50s***	100	9.0	100	4.1	100	17.3	2.5	21.0	6.7
	4u100s	100	32.0	100	20.1	100	275	5.7	20.3	5.6
	5u50s	100	19.6	100	19.4	100	107	2.7	13.1	7.4
3	***10u10s***	75.0	1184	93.8	482	93.8	546	0.62	19.0	6.4
	50u10s	6.3	3523	68.8	1647	0.00	3583	5.0	8.8	3.8
	100u10s	0.00	3594	33.3	3112	0.00	3591	25.3	0.76	1.3

strategy **light**

		standard		min		min+st/act		efficiency		
	name	*solved*	*cpu*	*solved*	*cpu*	*solved*	*cpu*	*prep*	*rem*	*fcd*
1	***3u10s***	100	1.8	100	1.9	100	2.0	1.8	62.8	8.9
	4u20s	100	6.5	100	6.1	100	6.8	5.8	34.3	10.6
2	***4u50s***	100	16.7	100	10.8	100	24.9	9.8	23.6	11.2
	4u100s	100	60.0	100	35.3	100	271	21.7	20.3	9.4
	5u50s	100	39.0	100	15.5	100	171	11.5	13.3	12.0
3	***10u10s***	81.3	1017	100	107	100	313	3.5	14.4	12.3
	50u10s	0.00	3590	62.5	1721	0.00	3581	25.1	4.9	5.9
	100u10s	0.00	3590	16.7	3267	0.00	3587	25.3	0.76	1.3

strategy **normal**

Table 2: Computational results achieved with preprocessing procedures and CPLEX – *part* 1

Remark 6.2. *The most aggressive strategy,* ***maximal****, is characterized by a specific propagation algorithm which is executed before* Bottom-Up *for all contact indexes, that is at the beginning of Algorithm 6's main loop. It attempts to prove that recipient node r_c has necessarily obtained all the units in accordance with Proposition 4.5, by testing if min-card$(V_D, r_c, c-1) = u$. Combined with* Bottom-Up *and* Minimality+Validity*, this leads to the deletion of contact σ_c. Although this approach is inefficient in practice, we retained it as part of* ***maximal****, because it is only designed to upper-bound the number of deductions that can be done with our approach.*

		standard		min		min+st/act		efficiency		
	name	*solved*	*cpu*	*solved*	*cpu*	*solved*	*cpu*	*prep*	*rem*	*fcd*
1	***3u10s***	100	3.0	100	3.1	100	3.2	3.0	64.9	12.1
	4u20s	100	17.0	100	17.1	100	17.7	16.4	37.9	11.4
2	***4u50s***	100	42.2	100	36.2	100	49.0	35.0	24.9	12.2
	4u100s	100	170	100	137	100	322	128	21.6	9.8
	5u50s	100	74.5	100	54.2	100	173	48.8	13.4	12.5
3	***10u10s***	75.0	1020	93.8	464	100	277	12.9	14.6	15.5
	50u10s	0.00	3589	68.8	1691	0.00	3584	156	8.8	8.9
	100u10s	0.00	3592	66.7	3305	0.00	3591	601	1.6	1.8

strategy **aggressive**

		standard		min		min+st/act		efficiency		
	name	*solved*	*cpu*	*solved*	*cpu*	*solved*	*cpu*	*prep*	*rem*	*fcd*
1	***3u10s***	100	3.5	100	3.5	100	3.4	3.5	65.1	12.1
	4u20s	100	25.6	100	25.0	100	25.4	25.0	40.2	11.4
2	***4u50s***	100	64.1	100	58.3	100	66.9	57.1	25.9	12.2
	4u100s	100	233	100	215	100	319	206	21.9	9.8
	5u50s	100	109	100	87.1	100	196	84.0	13.4	12.5
3	***10u10s***	75.0	1029	93.8	533	100	337	19.7	14.7	15.5
	50u10s	0.00	3593	68.8	1724	0.00	3590	195	8.8	8.9
	100u10s	0.00	3600	0.00	3597	0.00	3600	2402	2.0	1.8

strategy **maximal**

Table 3: Computational results achieved with preprocessing procedures and CPLEX – *part* 2

Minimal. In this strategy, only the most basic procedures are run, *e.g.* no transfer can be forced since routine STRICT-ACTIVE is deactivated. The number of null transfers detected is weak too. Although the amount *prep* of time required by these procedures is insignificant and almost linear with respect to the number of contacts, this strategy does not improve the behaviour of the solver (the results reported in Tables 1 and 2 are similar). In fact, we may suppose that these deductions are automatically deduced from the model by CPLEX's preprocessing engine.

Light / Normal / Aggressive. The effects of the preprocessing procedures become significant when TOP-DOWN and STRICT-ACTIVE are acting, *e.g.* if we look at the first class, about 50% of the contacts have been removed and about 5% of the transfers have been forced in only 0.4 seconds. The procedure seems to break the combinatorial bound on the number of contacts or the number of nodes. It unfortunately *collapses* when the number of datum units is large (*rem* and fcd drop when u is greater than 4 or 5).

This behaviour might be explained by the fact that evaluating *min-card*/*max-card* can actually be seen as solving the subproblems of a decomposition of our problem. These subproblems are characterized by fewer contacts, the transfer values, and only one recipient at a time. We thus try to solve smaller problems in order to deduce information regarding the original master problem. These problems are effectively smaller in size, but the number of transferable units increases very fast when the number of units of the original problem becomes large. The proposed method for evaluating *min-card* depends on solving an NP-Hard problem, which is far from straightforward

		selected		computational results				comparison // maximal		
	name	*model*	*strat.*	*solved*	$feas$	*cpu*	*gap*	Δ_{rem}	Δ_{fcd}	Δ_{prep}
1	**3u10s**	min	light	100	-	0.47	-	75.3	44.0	11.9
	4u20s	min	light	100	-	2.0	-	66.9	58.1	5.6
2	**4u50s**	min	light	100	-	4.1	-	81.2	54.6	4.4
	4u100s	min	light	100	-	20.1	-	92.7	57.1	2.8
	5u50s	min	light	100	-	19.4	-	97.8	59.0	3.2
3	**10u10s**	min	agg.	93.8	6.3	464	10.17	99.6	100	65.6
	50u10s	min	agg.	68.8	6.3	1691	1.06	100	99.9	79.7
	100u10s	min	agg.	66.7	0.00	3305	-	81.1	97.7	25.0

$\Delta_{rem} = {}^{rem(strat.)}/_{rem(maximal)}\ (\times 100) \cdot \Delta_{fcd} = {}^{fcd(strat.)}/_{fcd(maximal)}\ (\times 100) \cdot \Delta_{prep} = {}^{prep(strat.)}/_{prep(maximal)}\ (\times 100)$

Table 4: The preprocessing strategy selected in every group.

in such situations. This explains why the CPU time required to run preprocessing procedures, *cf.* column $prep$, exponentially increases with the number of units.

Maximal. The final strategy, **minimal**, is much too aggressive. The computational overhead of the preprocessing procedures is unjustified. This observation is highlighted in Table 4, where the selected strategies are reported. The last columns indicate ratios between the other strategies and **maximal**. If we consider class **5u50n**, almost all contacts removed (resp. around 60% of contacts forced) by the most aggressive strategy are also removed (resp. forced) by strategy **light**, with a computational cost that is around 30 times less. So, strategy **maximal** should not be used.

Note in addition that model **min** is clearly dominant when preprocessing procedures are used, except for class **3u10n** which is better solved with model **min+st/act**.

It will be remarked that computational performances can be significantly improved using the deduction rules proposed in Section 4. We have shown how these can be used within a preprocessing procedure designed to simplify the instances. Given the positive results achieved, we attempted to apply these deduction rules dynamically too, by propagating the constraints during the branching stage, as a constraints-programming engine usually does. However, this gave very poor results that are not worth reporting here.

This approach actually falls within the the framework of constraint propagation, a powerful *constraint-programming* tool that is unsuitable for linear programming. Implementing Algorithm 6 using this kind of approach seemed unnatural and excessively complex. Matching the variables in our model and the functions in the transfer graph ($\chi_A, \chi_\phi, \chi_{\underline{D}}, \chi_{\overline{D}}$) is not at all straightforward. This prevents us from making full use of the information obtained using the deduction procedures in the model, *e.g.* what kind of relationship is it that binds the arcs of the transfer graph – and more specifically function χ_A – and the x-variables ?

In fact, the notion of *minimal transfer plan* is seen to be very well integrated into the model (through constraint (10)), while the concept of *strictly-active transfer plan* continues to yield disappointing results. We plan to study constraint programming to tackle these problems.

7. Conclusions and Perspectives

In this paper, the problem of transferring data within a deterministic delay-tolerant network or a system of systems is investigated. An integer-linear-programming formulation is proposed for solving the so-called *dissemination problem*. We also propose some dominance-rule-based preprocessing procedures, enabling promising computational results to be achieved. These preprocessing procedures are designed to detect useless contacts and required transfers, so that instances can be simplified (and the model strengthened).

In the future, we plan to investigate constraint programming so that these dominance rules can be more deeply integrated within the procedure, *e.g.* during the branch-and-bound procedure.

Acknowledgement

These works are financed by the *Conseil Régional de Picardie* and carried out in the framework of *Labex MS2T*, funded by the French government through the *Investments for the Future* program, and managed by the *National Agency for Research* (ANR-11-IDEX-0004-02).

References

[1] M. Jamshidi, System of Systems Engineering: Principles and Applications, Boca Raton, Taylor & Francis, 2008.

[2] N. Belblidia, M. Dias De Amorim, L. H. M. K. Costa, J. Leguay, V. Conan, PACS: Chopping and shuffling large contents for faster opportunistic dissemination, in: 2011 8th International Conference on Wireless On-Demand Network Systems and Services, WONS 2011, IEEE, 2011, pp. 9–16.

[3] K. Fall, A delay-tolerant network architecture for challenged internets, in: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '03, ACM Press, New York, USA, 2003, p. 27.

[4] A. Pentland, R. Fletcher, A. Hasson, DakNet: Rethinking Connectivity in Developing Nations, Computer 37 (1) (2004) 78–83.

[5] S. Merugu, M. Ammar, E. Zegura, Routing in Space and Time in Networks with Predictable Mobility, Tech. rep., Georgia Institute of Technology (2004).

[6] D. Hay, P. Giaccone, Optimal routing and scheduling for deterministic delay tolerant networks, 2009 Sixth International Conference on Wireless On-Demand Network Systems and Services (2009) 27–34.

[7] J. Alonso, K. Fall, A Linear Programming Formulation of Flows over Time with Piecewise Constant Capacity and Transit Times, Tech. rep., Intel Research, Berkeley (2003).

[8] S. Even, A. Itai, A. Shamir, On the Complexity of Timetable and Multicommodity Flow Problems, SIAM Journal on Computing 5 (4) (1976) 691–703.

[9] S. Jain, K. Fall, R. Patra, Routing in a delay tolerant network, in: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '04, ACM Press, New York, USA, 2004, p. 145.

[10] W. Zhao, M. Ammar, E. Zegura, Multicasting in delay tolerant networks, in: Proceeding of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking - WDTN '05, ACM Press, New York, USA, 2005, pp. 268–275.

[11] R. Handorean, C. Gill, G.-c. Roman, Accommodating Transient Connectivity in Ad Hoc and Mobile Settings, Lecture Notes in Computer Science 3001 (2004) 18–23.

[12] R. Bocquillon, A. Jouglet, J. Carlier, The data transfer optimization problem in a system of systems, European Journal of Operational Research 244 (2) (2015) 392–403.

[13] A. Ferreira, Building a reference combinatorial model for MANETs, IEEE Network 18 (5) (2004) 24–29.

[14] A. Jouglet, J. Carlier, Dominance rules in combinatorial optimization problems, European Journal of Operational Research 212 (3) (2011) 433–444.

[15] Instances – `https://www.hds.utc.fr/~rbocquil/dokuwiki/_media/dp_instances.zip`