Robust routing in deterministic delay-tolerant networks

Ronan Bocquillon, Antoine Jouglet

Robust routing in deterministic delay-tolerant networks

Ronan Bocquillon[1,2], Antoine Jouglet[1]

(1) Sorbonne Universités, Université de Technologie de Compiègne, UMR CNRS 7253 Heudiasyc, CS 60319, F-60203 Compiègne cedex (2) Université de Tours, LI (EA 6300), ROOT (ERL CNRS 6305), 64 av. Jean Portalis, F-37200 Tours

ronan.bocquillon@univ-tours.fr ; antoine.jouglet@hds.utc.fr

Abstract

A system of systems is a set of heterogeneous independent systems that share data in pursuit of a common goal. These systems form a delay-/disruption-tolerant network (DTN), where routing is based on the store-carry-and-forward paradigm. Systems can communicate whenever they are close enough to each other, in what are called contacts. We assume that the movements of these systems may be predicted in advance and we consider that a sequence of contacts is given at the outset. During a contact, a given emitting system can transfer to a given receiving system a fixed amount of data (termed datum unit) that it has in its possession. The dissemination problem is to find a transfer plan such that all the data can be transferred from a given subset of source systems to a given subset of recipient systems. In this paper we study the problem where communications may fail. We propose an algorithm for finding a robust transfer plan that minimizes the dissemination length.

Keywords: combinatorial optimization, robust optimization, constraint programming, delay- and disruption-tolerant networks, store-carry-and-forward routing, systems of systems

1. Introduction

Mo Jamshidi has defined *systems of systems* as large-scale integrated systems that are heterogeneous, independently operable on their own, and networked together for a common goal [17]. More precisely, in this paper, we consider a set of mobile systems which collaborate and share data to achieve a common objective. Due to their mobility, systems of systems may be sparse and lack continuous connectivity. They form a *delay-/disruption-tolerant network* (DTN) [10].

Systems can only communicate when they are close enough to each other. Such an opportunity is termed a *contact*. Messages are then transmitted from one system to another – one contact after another – and stored throughout the network in the hope that each message will reach its destinations. This is called *store-carry-and-forward* routing.

To minimize latency or to maximize the chances of a message being successfully transmitted to its destinations, one commonly used solution, termed *epidemic routing* [23], is to replicate messages and to spread many copies out over the network. Of course, this approach is not suitable where systems have limited memory capacity or where the bandwidth of links is low.

When systems produce contents that exceed the capacity of links, they must segment messages in order to transmit fragments (also termed *datum units*) separately. This raises the problem of which datum units to send during each contact (whenever systems meet). Belblidia *et al.* [3]

proposed, for example, a popularity-based decentralized heuristic which tends to homogenize the dissemination of each datum unit in the network.

This problem has exercised an increasing number of researchers over the last decade. Some [2, 9, 20] have proposed stochastic methods that estimate different probabilistic metrics to, *in fine*, decide which data are transmitted. Others [1, 13, 16, 21] have considered deterministic time-evolving networks and proposed algorithms that optimize a given criterion (such as delay or robustness). These approaches consider that the sequence of contacts is given at the outset, and therefore focus on applications where it is possible to make realistic predictions on systems' movements. Such applications include satellite networks (where the trajectory of all systems depends on straightforward physics) and public transportation systems. For example, Jain *et al.* [16] built an instance with 20 buses, equipped with wireless communication devices, making *scheduled* trips inside San Francisco.

Real-world applications include projects such as *Bluespots* [19] (where a small computer on a bus serves as a bluetooth content distribution station in a university public transit scenario), the *Disaster Monitoring Constellation* [24] (a multi-satellite Earth-imaging low-Earth-orbit sensor network where captured image swaths are stored onboard each satellite, and later downloaded from the satellite payloads to a ground station) and *DakNet*. The latter is a (very) low cost store-carry-and-forward wireless ad-hoc network for rural connectivity. It uses public buses (or other physical transports) to carry mobile access points between village kiosks and a hub with Internet access (a town). Data automatically uploads and downloads when the bus is in range of a kiosk or the hub. It has already been implemented in India and northern Cambodia [22]. Of course one can easily imagine that buses collaborate to spread the data more efficiently.

So-called *deterministic delay- and disruption-tolerant networks* are at the heart of the present paper. We examine the problem, commonly termed the *dissemination problem*, of making use of knowledge about systems' movements 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. A solution to this problem is termed a *transfer plan*.

We have previously addressed the case where transfers cannot fail [6, 7, 8]. We now propose to tackle a variant where *failures* are possible. We wish to find transfer plans guaranteeing that all datum units are correctly delivered to all recipient systems if no more than $\Gamma \in \mathbb{N}$ transmissions fail, regardless of those that actually fail (in accordance with the well-known approach proposed by Bertsimas and Sim [4]). By "failure" we of course mean anything that prevents a transfer terminating successfully (battery depletions, link problems, ...). In this way, we hope to cover a wider range of applications.

The rest of the paper is organized as follows. The *robust dissemination problem* is defined in Section 2. In Section 3 we describe an enumeration algorithm for finding all robust solutions. This algorithm is integrated into a constraint programming formulation in Section 4, and finally assessed in Section 5.

2. The robust dissemination problem

In this section the robust dissemination problem is formally defined. First we state the problem, then we discuss a well-known graph-based model to tackle it.

2.1. The robust dissemination problem

Let us now formally define the robust dissemination problem.

We first consider a set $\mathcal{N} = \{1, 2, \ldots, n\}$ of n interacting mobile systems, termed the *nodes*, and one *datum* $\mathcal{D} = \{1, 2, \ldots, u\}$ of u datum units. Each *datum unit* (also termed *unit*) is a unitary and indivisible fragment of data. At the outset, each node $i \in \mathcal{N}$ possesses a subset $O_i \subseteq \mathcal{D}$ of

units. For distinct nodes i, j the sets O_i, O_j do not have to be disjoint. Subset $\mathcal{R} \subseteq \mathcal{N}$ defines the nodes wishing to obtain the datum $\mathcal{D}$ (*i.e.* all the units) inside the given time horizon. For the sake of clarity, the term *source nodes* (or *sources*) will refer to the nodes $i \in \mathcal{N} \,|\, O_i \neq \emptyset$. The nodes in $\mathcal{R}$ will be called the *recipient nodes* (or *recipients*).

To ensure the dissemination of datum $\mathcal{D}$, nodes may exchange datum units whenever they are close enough to communicate. Such an opportunity is termed a *contact*. We assume that the trajectory of each node is deterministic and consequently that contacts are perfectly predictable. We actually consider that a *sequence of contacts* $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ of m ordered pairs of $\mathcal{N}^2$ is given. During contact $(s, r) \in \sigma$, the sending node s can send to the receiving node r at most one unit that it already possesses (either from the outset or as a result of previous contacts). After this contact, if the transmission has been successful, node r also possesses a copy of datum unit k. Here we assume that all buffers are infinite, so a node can potentially possesses a copy of all the datum units. Below, nodes s_c and r_c will denote the sender and the receiver in contact $\sigma_c \in \sigma$, *i.e.* $\sigma_c = (s_c, r_c)$.

A *transfer plan* is a function $\phi : \{1, 2, \ldots, m\} \mapsto \{\emptyset, \{1\}, \{2\}, \ldots, \{u\}\}$ where $\phi(c)$ designates the datum unit sent by node s_c during contact σ_c. Note that $\phi(c) = \emptyset$ if nothing is transferred during contact σ_c. Such a transfer is said to be *null*. Thus, for a given transfer plan ϕ and a given set of failures, we can compute the subset of datum units possessed by any node at any time. Formally, given an indicator function $\mathtt{S} : \{1, 2, \ldots, m\} \mapsto \{0, 1\}$ such that $\mathtt{S}(c) = 0$ stands for a success and $\mathtt{S}(c) = 1$ for a failure of transmission $\phi(c)$, we define the subset $O_i^t(\mathtt{S}, \phi) \subseteq \mathcal{D}$ of units possessed by node $i \in \mathcal{N}$ at time $t \in \{0, 1, \ldots, m\}$ as follows:

$$
\begin{aligned}
&(1)\ \forall i \in \mathcal{N},\ O_i^0 = O_i,\\
&(2)\ \forall c \in \{1, 2, ..., m\} \,|\, \mathtt{S}(c) = 1,\ \forall i \in \mathcal{N},\ O_i^c = O_i^{c-1},\\
&(3)\ \forall c \in \{1, 2, ..., m\} \,|\, \mathtt{S}(c) = 0,\ O_{r_c}^c = O_{r_c}^{c-1} \cup [\, \phi(c) \cap O_{s_c}^{c-1} \,],\\
&(4)\ \forall c \in \{1, 2, ..., m\} \,|\, \mathtt{S}(c) = 0,\ \forall i \in \mathcal{N} \backslash \{r_c\},\ O_i^c = O_i^{c-1}.
\end{aligned}
$$

The transfer plan ϕ is *valid* if all intended transmissions are possible in the most favorable case where no failure occurs. If we denote by $\mathtt{S}^*$ the constant function $\mathtt{S}^*(c) = 0$, then ϕ is valid if and only if $\forall \sigma_c \in \sigma$, $\phi(c) \in \{\emptyset\} \cup \{\{k\} \,|\, k \in O_{s_c}^{c-1}(\mathtt{S}^*, \phi)\}$.

A valid transfer plan ϕ has a *delivery length* $\lambda_i^\Gamma(\phi)$, for each node $i \in \mathcal{N}$, which corresponds to the smallest time index t at which it can be guaranteed that i possesses all the datum units if at most $\Gamma \in \mathbb{N}$ transmissions fail (regardless of which Γ transmissions fail), *i.e.*

$$
\begin{aligned}
\lambda_i^\Gamma(\phi) = \min \{\, t \in \{0, 1, \ldots, m\} \text{ such that for any indicator function} \\
\mathtt{S} \in \{1, 2, \ldots, m\} \mapsto \{0, 1\} \text{ with } \textstyle\sum_{c=1}^{m} \mathtt{S}(c) \leq \Gamma, O_i^t(\mathtt{S}, \phi) = \mathcal{D} \,\}
\end{aligned}
$$

If this index does not exist, then it is assumed that $\lambda_i^\Gamma(\phi) = \infty$. The dissemination length $\lambda^\Gamma(\phi) = \max_{i \in \mathcal{R}} \{\lambda_i^\Gamma(\phi)\}$ corresponds to the time index at which we can guarantee that all the recipient nodes possess all the datum units if at most Γ transmissions fail.

Given a set of nodes $\mathcal{N} = \{1, 2, \ldots, n\}$, a set of datum units $\mathcal{D} = \{1, 2, \ldots, u\}$, the subset $O_i \subseteq \mathcal{D}$ of datum units possessed by each node $i \in \mathcal{N}$ at the outset, a subset $\mathcal{R} \subseteq \mathcal{N}$ of recipient nodes, a sequence of contacts $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_m)$ such that $\forall c \in \{1, 2, \ldots, m\}$, $\sigma_c \in \mathcal{N}^2$ and a limit $\Gamma \in \mathbb{N}$ on the number of failures, the *robust dissemination problem* is to find a valid transfer plan ϕ minimizing the dissemination length $\lambda^\Gamma(\phi)$.

Remark 2.1. *Note here that many other objectives could have been studied, e.g. we could have minimized the number of contacts that are really tapped to optimize the energy consumption of the network. We leave this for future works.*

(a) instance definition

$$\begin{cases} \mathcal{N} = \mathcal{R} = \{1,2,3,4\};\ \mathcal{D} = \{1,2\} \\ O_1 = \{1,2\};\ O_2 = \{1\};\ O_3 = O_4 = \emptyset \\ \sigma = ((1,3),(3,4),(1,3),\ldots,(2,4)) \end{cases}$$

(b) a solution of 3 arc-disjoint branchings

$$\begin{cases} \phi(3) = \phi(7) = \{1\} \\ \phi(1) = \phi(2) = \phi(5) = \{2\} \\ \phi(4) = \phi(6) = \emptyset \end{cases}$$

(c) evolving graph [12]

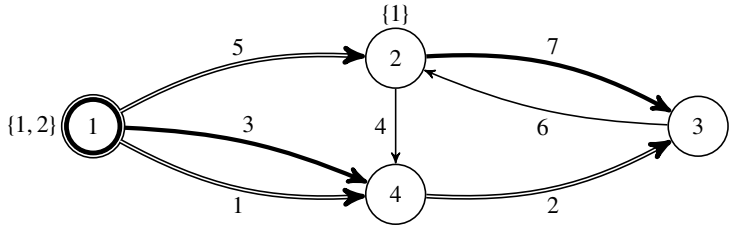


Figure 1: An instance of the non-robust dissemination problem ($\Gamma = 0$), a solution and the associated evolving graph.

Γ is a key parameter that characterizes the minimum level of robustness that any solution must guarantee. A transfer plan ϕ is Γ-robust if and only if $\lambda^{\Gamma}(\phi) \neq \infty$. Γ-robust solutions are the only admissible solutions for the robust dissemination problem, which means that an instance may be *feasible* with a given value of Γ, but *infeasible* with a greater value of that parameter. However, Γ-robust transfer plans are always p-robust for all $0 \leq p \leq \Gamma$.

The robust dissemination problem is NP-Hard in the strong sense, since the dissemination problem (a strongly NP-Hard problem [8]) is a special case of the robust dissemination problem where $\Gamma = 0$.

2.2. Evolving graphs

An instance of the robust dissemination problem is described by a multigraph whose vertices represent the set of nodes $\mathcal{N}$, and whose arcs represent the sequence of contacts σ. Each arc is labeled with the position in σ of the contact to which it corresponds. Such a graph is commonly called an evolving graph [11]. To appropriately take account of time constraints, the concept of *path* is replaced by the concept of *journey*, which is a sequence of arcs having *increasing* labels (a sequence of arcs that respect the order imposed by σ).

In Figure 1, $[\sigma_1 = (1,4), \sigma_2 = (4,3)]$ is a journey (since $1 \leq 2$). This represents the fact that node 4 can forward the unit it receives from node 1 at time 1 to node 3 at time 2. Nevertheless, $[\sigma_6 = (3,2), \sigma_4 = (2,4)]$ is not a journey (because $6 > 4$). More generally, 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 $j \in \mathcal{R}$ represents a store-carry-and-forward routing to transfer unit k from i to j.

Thus, a store-carry-and-forward routing to transfer a given unit $k \in \mathcal{D}$ to all the recipient nodes $\mathcal{R}$ is defined in the evolving graph by a set of arc-disjoint branchings that are rooted on the source nodes of unit k, and such that all the recipient nodes are covered. It is recalled that a *branching* is an evolving graph in which, for a vertex u called the *root* and any other vertex v, there is exactly one journey from u to v.

For example, in Figure 1, the double arcs define a branching to disseminate datum unit 2 from the unique source node 1 to all the other nodes. Similarly, the bold arcs define a set of branchings to disseminate datum unit 1 from the two source nodes 1 and 2 to all the other nodes. Note that, in the following, such a set of branchings is simply termed a *covering branching*, since it can

be seen as a single branching if we imagine a virtual root node that transfers datum unit k to the source nodes (1 and 2) at time 0.

Solving the non-robust dissemination problem ($\Gamma = 0$) is therefore equivalent to finding u arc-disjoint covering branchings of this type (*one* per datum unit) [8].

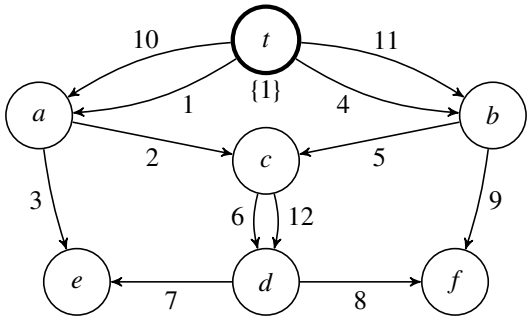
An intuitive extension to the robust case consists in searching for exactly $\Gamma + 1$ arc-disjoint covering branchings *per datum unit*. The resulting transfer plan (built from the $(\Gamma + 1) \times u$ covering branchings) is Γ-robust. It defines how $\Gamma+1$ copies of each datum unit can be transferred (along independent and contact-disjoint journeys) to the recipient nodes. Γ failures will never be enough to prevent any given node receiving a given datum unit. However, it is worth noting that this approach is over-protective. The existence of $(\Gamma + 1) \times u$ covering branchings is a sufficient but not a necessary condition for a Γ-robust solution to exist.

This last point is very important and actually justifies this study. Since solving the robust case is *not* equivalent to solving the initial problem simply by considering $\Gamma + 1$ times more datum units, we cannot directly use the algorithms that we proposed for the non-robust case.

Proposition 2.1. *The existence of at least* $(\Gamma+1)\times u$ *mutually arc-disjoint covering branchings in the evolving graph (*$\Gamma + 1$ *covering branchings per datum unit) is a sufficient, but not necessary, condition for a* Γ*-robust transfer plan to exist.*

Proof. The condition is sufficient, because Γ failures are not enough to invalidate $\Gamma + 1$ mutually arc-disjoint covering branchings.

To prove this is not necessary, we consider the following instance:



We assume that $\mathcal{R} = \mathcal{N} = \{t, a, \ldots, f\}$ (all nodes are recipients). The valid transfer plan ϕ such that $\forall c \in \{1, 2, \ldots, 12\}$, $\phi(c) = \{1\}$ is 1-robust since no single failure can prevent the correct delivery of unit 1. Yet, we can show that there do not exist two arc-disjoint covering branchings rooted at node t (the source) in this evolving graph. To demonstrate this, let us first suppose that there are two arc-disjoint covering branchings B_1 and B_2 rooted at t. Each covering branching has exactly 6 arcs. Thus, $\{B_1, B_2\}$ is necessarily a partition of the set $A = \{a_1, a_2, \ldots, a_{12}\}$ of arcs. a_{10} and a_{11} cannot be in the same covering branching, otherwise vertices c, d, e and f could not be reached. Let us suppose that $a_{10} \in B_1$ and $a_{11} \in B_2$. This means that arc $a_4 \in B_1$ and $a_1 \in B_2$. To reach node c, since a_2 cannot follow a_{10} and a_5 cannot follow a_{11} in a valid journey, we have to put a_2 in the covering branching containing a_1, so $a_2 \in B_2$, and a_5 in the covering branching containing a_4, so $a_5 \in B_1$. To reach node d, we have to put a_6 in one covering branching and a_{12} in the other one. If $a_6 \in B_1$ and $a_{12} \in B_2$, then a_7 have to be in B_1 (since it cannot follow a_{12} in a valid journey) and a_3 have to be in B_2 (it can follow a_1). Thus, none of a_8, a_9 can be added to B_2, because a_9 cannot follow a_{11} and a_8 cannot follow a_{12} in a valid journey. The same contradiction arises with $a_{10} \in B_2$, $a_{11} \in B_1$ or with $a_6 \in B_2$, $a_{12} \in B_1$. □

3. Robust optimization

In this section we propose a branching algorithm for enumerating all valid Γ-robust transfer plans. The procedure is based on a necessary and sufficient condition of robustness that we will proceed to discuss. We also propose a polynomial-time algorithm to check that a given solution is Γ-robust. This branching algorithm will be incorporated into the branch-and-bound procedure described in Section 4.

3.1. Necessary and sufficient condition for a transfer plan to be Γ*-robust*

In the worst case, all the failures prevent the same node $r \in \mathcal{R}$ receiving the same unit $k \in \mathcal{D}$ (the failures invalidate all the journeys going from a source of k to node r in the evolving graph). Thus, if we wish to prove that a given valid transfer plan ϕ is Γ-robust, it will be enough to show that for all datum units $k \in \mathcal{D}$ and all recipient nodes $r \in \mathcal{R}$, the number of contacts that have to fail to prevent r from receiving k is strictly greater than Γ. With this aim in view, for each unit $k \in \mathcal{D}$ and each node $r \in \mathcal{R}$, we consider the flow network $G(\phi, k, r) = (X, U)$ built as follows.

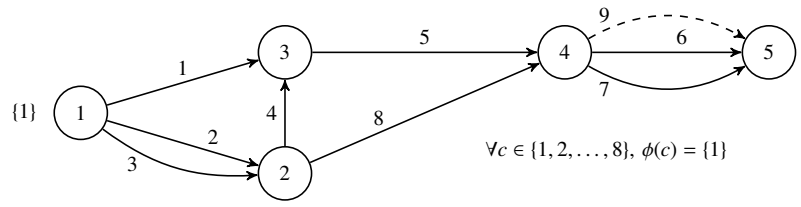
1. First, we add a source vertex $src \in X$ and a sink node $snk \in X$.
2. For each node $i \in \mathcal{N}$, we add $\mu_i + 1$ nodes $\{i_0, i_1, \ldots, i_{\mu_i}\} \subseteq X$, where $\mu_i = |\{\, \sigma_c \in \sigma \,|\, r_c = i \,\}|$ is the number of contacts whose receiver is i.
3. We add one arc $(src, i_0) \in U$ for each source node $i \in \mathcal{N} \,|\, k \in \mathcal{O}_i$, and one arc $(r_{\mu_r}, snk) \in U$. These arcs all have an infinite capacity.
4. For each node $i \in \mathcal{N}$, we add arcs $\{(i_0, i_1), (i_1, i_2), \ldots, (i_{\mu_i - 1}, i_{\mu_i})\}$ with infinite capacities.
5. For each contact $\sigma_c = (i, j) \in \sigma$ such that $\phi(c) = \{k\}$, we add an arc $(i_x, j_y) \in U$ of capacity 1. x denotes the number of contacts occurring before σ_c and whose receiver is i. y is set so that σ_c is the y^{th} contact whose receiver is j.

The units of flow traversing a vertex $i_x \in X$ (*cf.* item 2) correspond to the copies of datum unit k that are transferred to node $i \in \mathcal{N}$ during the first x contacts in σ of which it is the receiver. These units of flow come from vertex $i_{z-1} \in X$ through an arc of infinite capacity (this represents the fact that nodes can stores datum units over time, *cf.* item 4) and/or from another vertex $j_y \in X$ through an arc of capacity 1 (this corresponds to a transfer during a contact, *cf.* item 5). The arcs discussed in item 3 just ensure that the sources of datum unit k can store as many copies of k as needed. Note finally that vertex $snk \in X$ collects the units of flow corresponding to what node r receives throughout the transfer plan.

A similar graph was proposed in [8] to prove that the non-robust dissemination problem *with one recipient* is polynomial and is equivalent to a max-flow problem (the graph has been slightly simplified because we are considering only one unit at a time). We can prove, in the same way, that a *src*-*snk* flow in $G(\phi, k, r)$ defines a set of arc-disjoint journeys for transmitting unit k, from one or several sources, to node r, in the evolving graph. These journeys are obtained from the saturated arcs representing a contact in $G(\phi, k, r)$, *i.e.* the arcs having a flow and a capacity of 1. Each unit of flow entering snk in $G(\phi, k, r)$ corresponds to a journey from a source of k to node r in the evolving graph. We therefore know that Γ failures cannot be enough to prevent node r from receiving unit k during transfer plan ϕ if the maximum amount of flow traversing $G(\phi, k, r)$ is greater than or equal to $\Gamma + 1$ (*i.e.* if there are at least $\Gamma + 1$ arc-disjoint journeys from a source of k to node r in the evolving graph).

In Figure 2b, we report the flow network $G(\phi, k = 1, r = 5)$ corresponding to the transfer plan ϕ described in Figure 2a. Only one unit of flow can traverse the network (using, for example, the bold arcs). This means that the transfer plan is not robust to failures (*i.e.* ϕ is not even 0-robust).

(a) a transfer plan ϕ to deliver datum unit 1 from node 1 to node 5



(b) the corresponding flow network $G(\phi, k = 1, r = 5)$

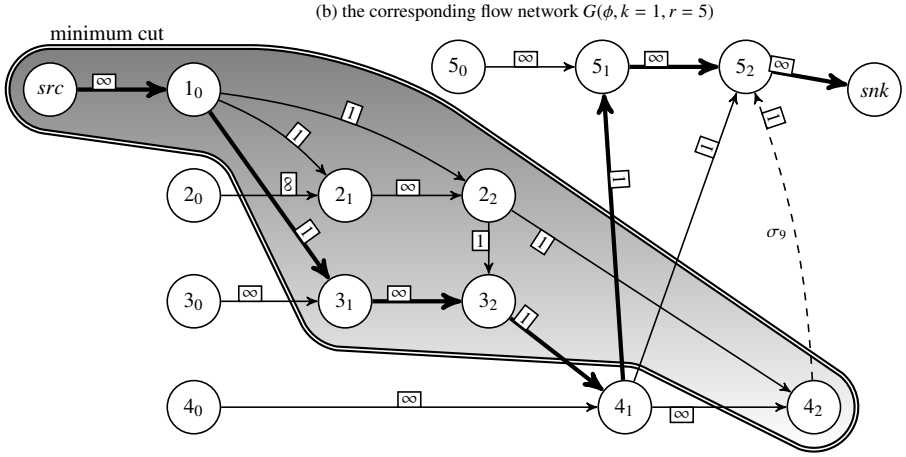


Figure 2: The flow network used to show the robustness of a transfer plan.

The minimum cut (depicted in gray) obtained with the Ford-Fulkerson [12] algorithm informs us that node 5 cannot receive unit 1 if contact $\sigma_5 = (3, 4)$ fails. This contact is represented by the only arc, namely $(3_2, 4_1)$, which is in the cut-set of that minimum cut. It means that all the journeys from node 1 to node 5 in the evolving graph use contact σ_5. On the other hand, adding one contact $(4, 5)$ at time 9, such that $\phi(9) = \{1\}$, would improve the robustness of the solution, *cf.* the augmenting path $(src, 1_0, 2_1, 2_2, 4_2, 5_2, snk)$. The two arc-disjoint journeys in the evolving graph would be $(\sigma_1, \sigma_5, \sigma_6)$ and $(\sigma_2, \sigma_8, \sigma_9)$.

Formally, this results in the following proposition.

Proposition 3.1. *Let ϕ be a valid transfer plan. ϕ is Γ-robust if and only if, for all datum units $k \in \mathcal{D}$ and all recipient nodes $r \in \mathcal{R}$, there exists a src-snk flow of value $\Gamma + 1$ in transportation network $G(\phi, k, r)$.*

Proof. We want to show that, given a datum unit $k \in \mathcal{D}$ and a recipient node $r \in \mathcal{R}$, Γ failures are not enough to prevent r from receiving k if and only if there exists a *src-snk* flow of value $\Gamma + 1$ in $G(\phi, k, r)$. We will assume below that the maximum amount of flow traversing the network is finite. Note though that it will be infinite (and therefore greater than $\Gamma + 1$) if and only if r is a source of k, *i.e.* if and only if even an infinite number of failures would not be enough to prevent r possessing k at the end of the sequence of contacts σ.

The condition is obviously sufficient, since Γ failures cannot be enough to invalidate the $\Gamma + 1$ journeys, in the evolving graph, which correspond to a flow of $\Gamma+1$ in the flow network $G(\phi, k, r)$, and which enable node r to receive unit k.

To show the condition is necessary, we use the well-known max-flow/min-cut theorem, which states that, in network $G(\phi, k, r)$, the maximum amount of flow passing from *src* to *snk* is equal to the minimum capacity of a *src*-*snk* cut, *i.e.* to the minimum total capacity of the arcs which if removed would disconnect *src* from *snk*. Only arcs representing a contact, with a *finite* capacity of 1, can belong to the cut-set of such a cut. It follows that a minimum *src*-*snk* cut in $G(\phi, k, r)$ corresponds to a minimum set of contacts which if removed would disconnect the sources of unit k from node r in the evolving graph. If ϕ is Γ-robust, the cardinal of this set is always greater than or equal to $\Gamma + 1$. □

On a practical level, Proposition 3.1 gives a polynomial time algorithm for checking whether a transfer plan is Γ-robust. For each datum unit and each recipient, we solve a max-flow problem to verify that there exist $\Gamma + 1$ mutually arc-disjoint journeys linking a source of this datum unit to this recipient in the evolving graph. This test runs in $O(u|\mathcal{R}|(\Gamma + 1)(n + m))$ time since, for each pair $(k, r) \in \mathcal{D} \times \mathcal{R}$, we need to find $\Gamma + 1$ augmenting paths in a network that contains at most $1 + n + 2m$ arcs.

This proposition is the basis of the enumeration procedure that we propose in the following section. It enumerates the solutions such that $\Gamma + 1$ journeys connect the sources of each datum unit to each recipient node in the evolving graph.

3.2. Enumeration procedure

The most naive method for enumerating the set of all Γ-robust transfer plans is to enumerate the set of all valid transfer plans and to retain only Γ-robust solutions. To this end, we propose to set the transfers in the order of the sequence σ, from $\phi(1)$ to $\phi(m)$. At each node of the search tree, the first transfer $\phi(c)$ that is not yet fixed is selected, and one branch is created for each possible value, that is one branch for each datum unit $k \in O_{s_c}^{c-1}(\mathbb{S}^*, \phi)$ (failures are ignored at this point), and one additional branch for the null transfer. The algorithm proposed in Section 3.1 is used to filter non-robust solutions.

Unfortunately, such an approach means exploring the entire search space (exponential in size). To avoid this, we propose generalizing the dominance rules that we proposed in [7]. It is recalled that *dominance rules* yield conditions on which a subset of the search space considered to solve the problem can be ignored. A *dominant* set of solutions is a subset of the search space that – if the search space contains at least one *feasible* solution – contains at least one *optimal* solution. Thus, enumerating any dominant set of solutions is sufficient to solve the problem. Of course, the smaller this set, the better. We refer to [18] for a comprehensive study on this concept.

Robust-minimal transfer plans

First, it will be remarked that some transmissions may be redundant. A *same* node can receive more than $\Gamma + 1$ times the *same* unit. 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. It is obvious that such transfers are not desirable.

Hence the definitions below and the ensuing dominance rule.

Definition 3.1. *Transfer $\phi(c) = \{k\}$, $c \in \{1, 2, \ldots, m\}$, $k \in \mathcal{D}$, is* robust-improving *if and only if it improves the level of robustness associated with unit k and node r_c (i.e. the minimum number of failures required to prevent r_c from receiving datum unit k) without, however, exceeding* Γ.

In practice, a transfer $\phi(c) = \{k\}$ is then robust-improving if and only if the maximum amount of flow from *src* to *snk* in transportation network $G(\phi, k, r_c)$ does not exceed $\Gamma + 1$ and is larger when we consider transfer $\phi(c)$ than when we only consider the first $c - 1$ contacts (than when the capacities of the arcs corresponding to a contact σ_t with $t \geq c$ are set to 0).

Definition 3.2. *A transfer plan ϕ is* robust-minimal *if and only if all its transfers are null or robust-improving.*

In Figure 2, transfer $\phi(6)$ is robust-improving, because it "opens" the first journey to node 5 in the evolving graph. Transfer $\phi(7)$ is not robust-improving, since adding or removing arc $(4_1, 5_2)$ does not increase the amount of flow that can traverse $G(\phi, 1, 5)$. If we set $\phi(7) = \emptyset$, ϕ becomes robust-minimal. It remains robust-minimal if we consider $\phi(9) = \{1\}$ (since this transfer is robust-improving).

Proposition 3.2. *The set of* robust-minimal *transfer plans is dominant.*

Proof. Let ϕ be a valid non-robust-minimal transfer plan. So there exists at least one transfer $\phi(c)$ which is neither null nor robust-improving. Thus, the transfer plan ϕ' – obtained by copying ϕ and by setting $\phi'(c) = \emptyset$ – has exactly the same dissemination length as ϕ, *i.e.* $\lambda^\Gamma(\phi') = \lambda^\Gamma(\phi)$. This process is to be repeated as long as the new transfer plan is not robust-minimal. □

Robust-strictly-active transfer plans

In many cases, some transmissions can be postponed without affecting the optimilaity of the solution. This flexibility significantly enlarges the search space and is not desirable.

Hence the following dominance rule.

Definition 3.3. *A transfer plan ϕ is* robust-strictly-active *if and only if all transfers are robust-improving when possible, i.e. $\forall c \in \{1, 2, \ldots, m\}$, if $\exists k \in O_{s_c}^{c-1}$ such that $\phi(c) = \{k\}$ is robust-improving, then $\phi(c)$ is robust-improving.*

Proposition 3.3. *The set of* strictly-active *transfer plans is dominant.*

Proof. Let ϕ be a non-robust-strictly-active transfer plan, *i.e.* there exists a non-robust-improving transfer $\phi(c)$, $c \in \{1, 2, \ldots, m\}$, and a datum unit $k \in \mathcal{D}$ such that $\phi(c) = \{k\}$ would be robust-improving. Let then ϕ' be 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, *i.e.* $\lambda^\Gamma(\phi') \leq \lambda^\Gamma(\phi)$. This process is to be repeated as long as the transfer plan is not robust-strictly-active. □

Proposition 3.4. *The set of* robust-minimal *and* robust-strictly-active *solutions is also dominant (by combining the above proofs).*

Enumerating robust transfer plans

Proposition 3.4 states that enumerating robust-minimal and robust-strictly-active transfer plans is sufficient to solve the robust dissemination problem (other transfer plans can be ignored). Therefore – if we use a sequential branching algorithm – we can skip all the branches that do not correspond either to a null transfer or to a robust-improving transfer.

At each node of the search tree, the earliest transfer $\phi(c)$ that is not yet fixed is selected, and one branch is created for each possible transfer $\phi(c) = \{k\}$ ($k \in O_{s_c}^{c-1}$) that "opens" a new journey from a source in k to node r_c in the evolving graph. If no such transfer exists, then $\phi(c)$ is set to $\emptyset$. A transfer plan is Γ-robust when every recipient has received $\Gamma + 1$ copies of each unit.

If the flows are stored from one node of the search tree to another, then testing whether a given transfer is robust-improving takes $O(n + m)$ time. Only one iteration of the Ford-Fulkerson [12] algorithm is required, because a single new arc in $G(\phi, k, r_c)$ cannot enable two or more new augmenting paths to be found at the same time (*i.e.* a single new transfer in the evolving graph cannot "opens" two or more new journeys from a source of k to r_c at the same time).

Remark 3.1. *In practice, only one flow network can be stored in memory. The network defined in Section 3.1 has a structure which depends on the instance that we are examining. In particular, it is only the capacities that differ between two graphs $G(\phi_1, k_1, r_1)$ and $G(\phi_2, k_2, r_2)$ (i.e. the capacities depend only on the partial transfer plan, the datum unit and the recipient node under consideration). Therefore only the capacities – associated with each unit $k \in \mathcal{D}$ and each node $i \in \mathcal{N}$ – and the resulting flows need to be stored in a reversible (backtrack-able) data structure. The structure of the graph itself can be static.*

4. A constraint programming approach

In this section we implement the approach proposed in Section 3 for solving the robust dissemination problem. We use constraint programming to be able to utilize pre-implemented tools (filtering algorithms) that make it easy to develop an efficient branch-and-bound procedure. We also propose *ad hoc* methods to speed up the solving of the problem. All of this will be assessed in Section 5.

4.1. Model

In this model, all transfers are required to be robust-improving. Therefore, every recipient node $r \in \mathcal{R}$ must receive exactly $\Gamma + 1$ copies of each unit $k \in \mathcal{D}$. At the outset, we assume that r possesses $\Gamma + 1$ (resp. 0) copies of k if $k \in O_r$ (resp. $k \notin O_r$).

For each node $i \in \mathcal{N}$, $\mathcal{T}_i$ is defined as the set of time indexes at which the state of i can change, *i.e.* $\mathcal{T}_i = \{0\} \cup \{c \in \{1, 2, \ldots, m\} \,|\, r_c = i\}$. In addition, $\forall t \in \{0, \ldots, m\}$, $\mathcal{T}_i(t)$ refers to 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 of our model are defined as follows:

- $\forall k \in \mathcal{D}$, $\forall c \in \{1, \ldots, m\}$, $x_{k,c} = 1$ if datum unit k is transmitted from node s_c to node r_c during contact σ_c, and $x_{k,c} = 0$ otherwise.
- $\forall i \in \mathcal{N}$, $\forall k \in \mathcal{D}$, $\forall t \in \mathcal{T}_i$, $y_{i,k,t}$ indicates the number of copies of datum unit k possessed by node i after contact σ_t. Thus, the domain of these variables is $\{0, 1, \ldots, \Gamma + 1\}$.
- $\forall c \in \{1, 2, \ldots, m\}$, $a_c = 1$ if transfer $\phi(c)$ is robust-improving, and $a_c = 0$ otherwise.
- $\forall i \in \mathcal{N}$, $\forall k \in \mathcal{D}$, variable $\lambda_{i,k}$ represents the delivery length associated with datum unit k and node i, *i.e.* the date from which node i possesses $\Gamma+1$ copies of datum unit k. Its domain is $\mathcal{T}_i \cup \{\infty\}$. $\lambda_{i,k} = \infty$ means that i does not recover k during the transfer plan. In practice, we consider $\infty = m + 1$.
- $\forall i \in \mathcal{R}$, variable $\lambda_i = \max_{k \in \mathcal{D}} \{\lambda_{i,k}\}$ represents the delivery length of recipient node i, *i.e.* the date from which we are sure that node i possesses all datum units if at most Γ failures occur. Its domain is $\mathcal{T}_i$ (i has to be served).
- Finally, variable $\lambda = \max_{i \in \mathcal{R}} \{\lambda_i\}$ represents the dissemination length of the transfer plan. Its domain is $\bigcup_{i \in \mathcal{R}} \mathcal{T}_i \subseteq \{0, 1, \ldots, m\}$.

Remark 4.1. *y-variables are indexed with sets $\mathcal{T}_i$ rather than set $\{0, \ldots, m\}$. However, we may easily know whether a node $i \in \mathcal{N}$ possesses a datum unit $k \in \mathcal{D}$ at any index $t \in \{0, 1, \ldots, m\}$, since $y_{i,k,\mathcal{T}_i(t)} = 1$ if and only if node i possesses datum unit k after the first t contacts.*

Minimizing the dissemination length leads to the following objective.

$$\lambda^* = \min \lambda \tag{1}$$

The constraints are written as follows (these constraints alone are **not** sufficient to ensure that the transfer plan is Γ-robust):

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

$$\forall i \in \mathcal{N},\ \forall k \in \mathcal{D} \,|\, k \in O_i,\ y_{i,k,0} = \Gamma + 1 \tag{2}$$

$$\forall i \in \mathcal{N},\ \forall k \in \mathcal{D} \,|\, k \notin O_i,\ y_{i,k,0} = 0 \tag{3}$$

- The transfer plan must be valid (sending nodes must possess the units that they transmit):

$$\forall k \in \mathcal{D},\ \forall c \in \{1, 2, \ldots, m\},\ x_{k,c} \leq y_{s_c,k,\mathcal{T}_i(c-1)} \tag{4}$$

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

$$\forall k \in \mathcal{D},\ \forall c \in \{1, 2, \ldots, m\},\ y_{r_c,k,\mathcal{T}_i(c-1)} + x_{k,c} = y_{r_c,k,c} \tag{5}$$

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

$$\forall c \in \{1, 2, \ldots, m\},\ \sum_{k \in \mathcal{D}} x_{k,c} = a_c \tag{6}$$

- $\lambda_{i,k}$ is the delivery length associated with datum unit k and node i:

$$\forall i \in \mathcal{N},\ \forall k \in \mathcal{D},\ \forall t \in \mathcal{T}_i,\ [\, y_{i,k,t} < \Gamma + 1 \,] \iff [\, \lambda_{i,k} > t \,] \tag{7}$$

Implicit constraints.

No simple constraints exist that would explicitly ensure that the transfer plan is Γ-robust. We ensure this through the branching stage. As mentioned in Section 3.2, to solve the problem, we sequentially fix the transfers. This corresponds to setting x-variables from $x_{k,1}$ ($\forall k \in \mathcal{D}$) to $x_{k,m}$.

1. At each node of the search tree, the earliest transfer $\phi(c)$ which is not yet fixed (the smallest index $c \in \{1, 2, \ldots, m\}$ such that $\exists k \in \mathcal{D}$ where $x_{k,c}$ is not yet fixed) is selected.
2. Next, one branch is created for each robust-improving transfer. For each datum unit $k \in O_{s_c}^{c-1}$, if transfer $\phi(c) = \{k\}$ is robust-improving, then a branch is added where $x_{k,c} = 1$ and $\forall p \in \mathcal{D} \backslash \{k\}$, $x_{p,c} = 0$.
3. Finally, if no robust-improving transfers have been found, then $\phi(c)$ is set to $\emptyset$, *i.e.* $\forall k \in \mathcal{D}$, we set $x_{k,c} = 0$.

To check whether a given transfer is robust-improving, we use the test defined in Section 3.2. Note that the capacities of the transportation network are given by the x-variables for transfers occurring before $\phi(c)$, and must be set to 0 for other transfers. In this way, we ensure that the transfer plan is Γ−*robust* (because other constraints ensure that each recipient receives $\Gamma + 1$ copies of each unit).

This approach ensures that the transfer plan is both *robust-minimal* (every transfer is either null or robust-improving) and *robust-strictly-active* (since robust-improving transfers cannot be postponed).

Remark 4.2. *A transfer $\phi(c) = \{k\}$ is necessarily robust-improving if there exist more arc-disjoint journeys from a source of datum unit k to node s_c than from a source of k to node r_c. In such a case no flows need to be computed and a_c can be set to* 1*. In practice, we can use the following constraint:* $\forall k \in \mathcal{D}$, $\forall c \in \{1, 2, \ldots, m\}$, ***if*** $[\, y_{s_c,k,\mathcal{T}_{s_c}(c-1)} > y_{r_c,k,\mathcal{T}_{r_c}(c-1)} \,]$ ***then*** $[\, a_c = 1 \,]$.

Ad hoc procedures.

As regards the robust dissemination problem, we have discussed the key concepts and models. The solving process may be speeded up through the use of *ad hoc* procedures (typically through the use of specific *filtering algorithms*).

To start with, note that almost all approaches that were proposed for the non-robust case [6] can be generalized straightforwardly to the robust case. These remain applicable provided that datum units *and* their copies are taken into consideration, *i.e.* provided that we correctly take into account the requirement that each recipient receives $\Gamma + 1$ copies of each datum unit, and not just one. In Sections 4.2 through 4.4 we summarize the key ideas behind all these approaches, but we refer the reader to [6] for further details (*cf.* Propositions 5.1, 5.2, 5.3 and 5.5). In Section 4.5 we propose and discuss a new *ad hoc* procedure.

4.2. Lower bounds

First, the so-called *weak lower bound* becomes:

Proposition 4.1. *Let $i \in \mathcal{R}$ be a recipient node, and $\alpha = (\Gamma + 1) \times (u - |O_i|)$ be the number of datum units (including copies) that i has to receive during the transfer plan. Let $\sigma_x \in \sigma$ denote the α^{th} contact $\sigma_c = (s_c, i)$ during which a unit $k \in \mathcal{D} \backslash O_i$ can be transferred to node i (i.e. such that variable $x_{k,c}$ is not set to 0). When this index does not exist (when it remains less than α contacts fulfilling the condition), we consider that $x = \infty$. x is a lower bound for $\lambda_i^{\Gamma}(\phi)$ and $\lambda^{\Gamma}(\phi)$ (i.e. $\lambda \geq \lambda_i \geq x$ in the model).*

This lower bound is trivial, but it can be improved if one can prove that there exists no valid transfer plan allowing node i to receive the α datum units it needs with only the first α available contacts. We can actually add some contacts while the condition is not respected. Each test can run in polynomial time by solving an assignment problem. This corresponds the so-called *strong lower bound*, which is tighter, but which is also more complicated to compute.

4.3. Symmetry-breaking constraints

Let us consider the example on the left of Figure 3. At time $t = 3$, datum units 1 and 2 share the same sources (nodes 1, 2 and 3), *i.e.* $\forall i \in \mathcal{N}$, $1 \in O_i^3$ if and only if $2 \in O_i^3$. Thus, the role of these datum units can be swapped in the rest of the sequence (see the right half of the figure). That is to say, the sub-branchings (in the evolving graph) corresponding to datum units 1 and 2, and with contacts occurring after contact σ_3, can be swapped. The dissemination length is not affected by this operation. As a result, it is sufficient to consider one option (either $\phi(4) = \{1\}$ or $\phi(4) = \{2\}$) out of the two to solve the robust dissemination problem, *e.g.* we can arbitrarily decide to transmit the datum unit with the lowest index first.

In practice, at each node of the search tree, whenever we are setting a transfer $\phi(c)$, we can check whether two units k_1 and $k_2 \in \mathcal{D}$ (with $k_1 < k_2$) have the same sources. If so, $x_{k_2,c}$ can be set to 0, *i.e.* the branch such that $\phi(c) = \{k_2\}$ can be ignored, because the branch where $\phi(c) = \{k_1\}$ will be explored.

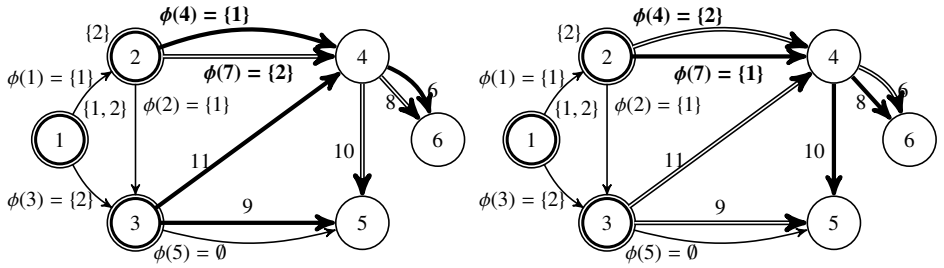


Figure 3: A symmetry appears between units 1 and 2 when we set $\phi(4)$.

4.4. Memorization

Let us now move to another symmetry-breaking technique. The latter involves registering the state of the nodes visited in the search tree. This enables branches to be pruned by detecting dominance relationships among the transfer plans under construction.

At each node of the search tree, whenever we are setting a transfer $\phi(c)$, we can check whether another transfer plan ϕ' such that $\forall i \in \mathcal{N}$, $O_i^{c-1}(\mathbb{S}^*, \phi) \subseteq O_i^{c-1}(\mathbb{S}^*, \phi')$ has already been visited. If so, the current branch can be pruned. As a matter of fact, from any solution ϕ that is reachable from the current node, we can build a better or equivalent solution ϕ'' by choosing the first $c-1$ transfers of ϕ' and the last $m-c+1$ transfers of ϕ. Transfer plan ϕ can be ignored because ϕ'' has already been visited.

Testing whether $\forall i \in \mathcal{N}$, $O_i^{c-1}(\mathbb{S}^*, \phi) \subseteq O_i^{c-1}(\mathbb{S}^*, \phi')$ is trivial if these sets are represented by bit vectors (only a few boolean operations are sufficient to check it). This computational efficiency makes it possible to store thousands of states and thereby, to improve the performance of the solver significantly.

4.5. The look-ahead procedure

In some circumstances, the solver manages to prove that some transfers are necessary even before the sequential branching algorithm needs to choose a value for these transfers (by combining some of the constraints in the model). The corresponding x-variables are then set to 1. In such a case, the first $t \in \{0, \dots, m-1\}$ transfers are fixed and $\phi(t+1)$ is the first unfixed transfer, but there exists a transfer $\phi(c) = \{k\}$, $c \in \{t+2, t+3, \dots, m\}$, $k \in \mathcal{D}$, that has been set by another filtering procedure. Like any other transfer, $\phi(c)$ must be robust-improving. This constrains some transfers $\phi(x)$, $x \in \{t+1, \dots, c-1\}$ that have not yet been fixed.

For example, in Figure 4, the solver may deduce that $\phi(6) = \{1\}$ and $\phi(9) = \{1\}$ at the start (since there are only two contacts left for sending two copies of unit 1 to node 6). In Figure 4a, the branch $\phi(1) = \{2\}$ is explored. We assume that the solver has deduced, for any reason, that $\phi(3)$ is necessarily equal to $\{2\}$. We can easily show that this branch leads to no dominant transfer plan, so it can be pruned immediately. In Figure 4b, the branch $\phi(1) = \{1\}$ is explored. We can show that unit 1 must be transferred to nodes 4 and 5 during contacts σ_5 and σ_7, so variables $x_{1,5}$ and $x_{1,7}$ can both be set to 1.

Unfortunately, these constraints are not explicit in the model and specific algorithms must be developed to make such deductions. In this context, we have proposed the so-called *look-ahead* procedure, which aims to:

1. trigger a backtrack when one can prove that there exists no dominant solution in which transfer $\phi(c)$ is robust-improving,

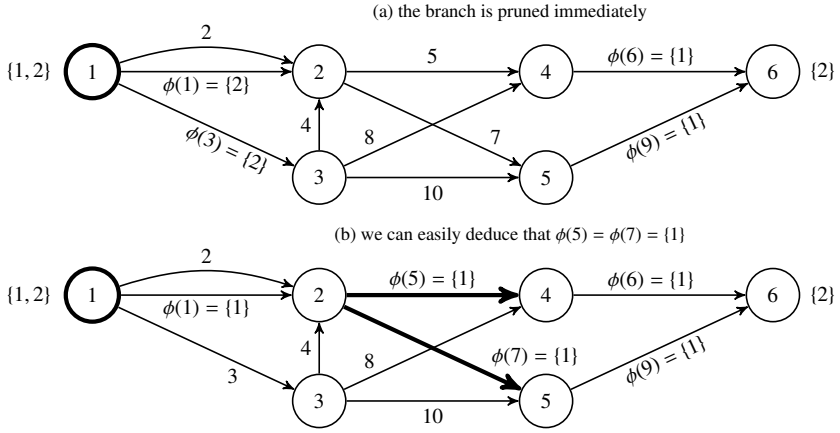


Figure 4: The look-ahead procedure ($\Gamma = 1$, $6 \in \mathcal{R}$ and $t = 1$).

2. prove that some transfers $\phi(x)$, $x \in \{t + 1, \ldots, c - 1\}$ are necessary, and therefore to set the corresponding x-variables to 1.

Space constraints prevent us from describing the look-ahead procedure thoroughly, so readers are referred to [5] for further details. Note here that the procedure relies on a careful study of the minimum cuts of the transportation networks $G(\phi, k, r)$, $k \in \mathcal{D}$, $r \in \mathcal{N}$, that are described in Section 3.1. The procedure is integrated into the branching algorithm described in Section 3.2.

5. Computational results

In this section, computational results are reported and discussed. We begin by describing our benchmark, built out of the instances generated for the non-robust dissemination problem, then we look at the results obtained using the different algorithms described in Section 4.

All the instances and the `C++` source code discussed below are available online [14, 15].

5.1. About the benchmark

To build a "hard" benchmark we propose using the instances that we previously generated for the non-robust dissemination problem.

It is recalled here that instances of the non-robust dissemination problem were generated by a destruction-construction heuristic designed to increase the hardness of an instance. This procedure is initialized with an instance chosen at random, then seeks to generate harder and harder instances by removing the least relevant contacts (*i.e.* the contacts removed by a trivial preprocessing procedure), so that new ones can be added elsewhere without increasing the size of the instance. A few contacts are randomly renewed in order to introduce diversity. At each iteration, the instance is solved with CPLEX. After an arbitrary number of iterations, the instance that required the most CPU time is returned. This was shown to produce some challenging instances of which we would like to keep the structure. For further details we refer to [7].

Given Propositions 2.1 and 3.1, we expect an instance of the robust dissemination problem to be as difficult as an instance of the dissemination problem having roughly Γ times as many units

name	*from*	*nbinst*	Γ	u	n	$\overline{rec}$	$\overline{src}$	$\overline{m}$
1r2u20n	4u20n	41	1	2	20	18	1	366
1r2u50n	4u50n+5u50n	49	1	2	50	44	1	717
1r2u100n	4u100n	20	1	2	100	87	2	1720
1r5u10n	10u10n	16	1	5	10	6	2	197
1r25u10n	50u10n	16	1	25	10	6	2	750
1r50u10n	100u10n	6	1	50	10	7	4	2000
2r2u50n	2r5u50s	23	2	2	50	50	1	726
2r3u10n	10u10n	16	2	3	10	6	2	197
2r17u10n	50u10n	16	2	17	10	6	2	750
2r34u10n	100u10n	6	2	34	10	7	4	2000
3r2u10n	10u10n	16	3	2	10	6	2	197
3r13u10n	50u10n	16	3	13	10	6	2	750
3r25u10n	100u10n	6	3	25	10	7	4	2000

Table 1: The benchmark generated for the robust dissemination problem.

(keeping constant the number of nodes and the number of contacts). Consequently, we propose reusing the instances generated for the dissemination problem, reducing the number of datum units so that the previous number of units is roughly equal to $(\Gamma + 1) \times u$ in the new instance.

We are faced with a familiar trade-off between a larger value of $(\Gamma + 1) \times u$, where feasible transfer plans are harder to find, but proofs of infeasibility are usually easier, and a smaller value of $(\Gamma + 1) \times u$, where feasible transfer plans are easier to find, but proofs of infeasibility may be harder. It will consequently be difficult to prove that any solution is optimal.

The benchmark is described in Table 1. Each class is characterized by the number *nbinst* of instances it contains, the required level of robustness Γ, the number of units u, and the number of nodes n in these instances. The average number $\overline{rec}$ of recipients $i \in \mathcal{R}$, the average number $\overline{src}$ of source nodes $i \in \mathcal{N}$ such that $O_i \neq \emptyset$, and the average number $\overline{m}$ of contacts in every class are also reported. In column *from*, we indicate the class(es) (originally used for the dissemination problem) from which the new instances are built.

5.2. Numerical results

Like in [6, 7], the computations reported in this paper were all performed on a server equipped with 16×6 cores (running at 2.67Ghz) and 1TB RAM. Algorithms were all implemented in `C++`. Constraint-programming models were solved using *CP-Optimizer*, a commercial solver developed by IBM-Ilog. The multithreading features proposed by the library were systematically deactivated, because the use of concurrent optimizers led to unstable results (different executions of the same computation often led to different results in terms of CPU time), which prevented reliable comparisons between our methods. Instances were therefore solved using a *deterministic* sequential algorithm with a one-hour time limit.

All our results tables (see below) feature the following metrics:

1. *solved* (-%) indicates the percentage of instances solved to optimality (a feasible solution was proved to be optimal or the instance was proved to be infeasible).
2. *feas* (-%) indicates the percentage of instances where optimality was not proved, but for which the solver found at least one feasible solution within the one-hour time limit.
   It follows that $100\% - solved - feas$ indicates the ratio (-%) of instances for which the solver neither found a feasible solution nor showed the instance to be infeasible.

	none			wlb			slb		
	solved	*feas*	*cpu*	*solved*	*feas*	*cpu*	*solved*	*feas*	*cpu*
1r2u20n	73.2	26.8	280	78.0	22.0	117	75.6	24.4	503
1r2u50n	12.2	83.7	77.9	14.3	81.6	68.8	14.3	81.6	327
1r2u100n	20.0	65.0	1.6	25.0	60.0	1.2	25.0	60.0	2.7
1r5u10n	6.3	62.5	15.9	31.3	37.5	6.2	31.3	37.5	26.2
1r25u10n	0.00	93.8	-	43.8	50.0	4.1	43.8	50.0	47.8
1r50u10n	0.00	100	-	50.0	50.0	19.1	50.0	50.0	97.8
2r2u50n	0.00	60.9	-	21.7	39.1	531	34.8	43.5	1.3
2r3u10n	12.5	75.0	1222	50.0	37.5	235	50.0	37.5	357
2r17u10n	0.00	93.8	-	50.0	43.8	35.5	50.0	43.8	326
2r34u10n	0.00	100	-	50.0	50.0	12.4	50.0	50.0	53
3r2u10n	43.8	56.3	195	68.8	31.3	165	56.3	37.5	168
3r13u10n	0.00	87.5	-	56.3	31.3	114	50.0	37.5	48
3r25u10n	0.00	100	-	50.0	50.0	10.4	50.0	50.0	39
average	20.2	69.6	254	42.9	47.0	114	42.5	48.6	251
	wlb+mem			**wlb+mem+sym**			**wlb+mem+sym+la**		
	solved	*feas*	*cpu*	*solved*	*feas*	*cpu*	*solved*	*feas*	*cpu*
1r2u20n	100	-	0.41	100	-	0.31	100	-	0.23
1r2u50n	67.3	30.6	200	73.5	24.5	257	81.6	16.3	240
1r2u100n	55.0	45.0	3.7	60.0	40.0	110	65.0	35.0	162
1r5u10n	43.8	37.5	149	56.3	25.0	18.4	56.3	25.0	11.1
1r25u10n	43.8	50.0	4.1	50.0	37.5	26.6	50.0	37.5	27.4
1r50u10n	50.0	50.0	18.8	83.3	16.7	23.2	83.3	16.7	23.6
2r2u50n	34.8	34.8	150	34.8	34.8	58.4	43.5	30.4	691
2r3u10n	87.5	12.5	398	93.8	6.3	91.8	93.8	6.3	51.4
2r17u10n	50.0	43.8	30.6	68.8	25.0	79.0	68.8	25.0	87.0
2r34u10n	50.0	50.0	12.1	66.7	33.3	23.7	66.7	33.3	14.5
3r2u10n	100	-	0.28	100	-	0.20	100	-	0.20
3r13u10n	56.3	31.3	49.7	62.5	25.0	1.6	62.5	25.0	1.6
3r25u10n	50.0	50.0	12.0	66.7	33.3	10.0	66.7	33.3	10.2
average	66.0	27.9	94.0	72.5	21.1	77.9	75.3	18.6	112

Table 2: Computational results obtained with CP-Optimizer.

3. *cpu* (-s) is the average solution time. This metric is calculated only on the *solved*% of the instances that were solved to optimality within the one-hour time-limit.

All comparisons between algorithms are therefore made with respect to metrics that have a clear hierarchy of relevance, where *solved* is the most important information. If *solved* is the same, the next most important information to take into account is *feas* and finally *cpu*. Note that value "-" means that the corresponding metric is meaningless (*e.g. feas* is always equal to 0% if *solved* = 100%) or cannot be calculated (*e.g. cpu* cannot be calculated if *solved* = 0%).

Column **none** in Table 2 contains the computational results obtained using the model described in Section 4.1 and the branching algorithm described in Section 3.2 (with no *ad hoc* procedures).

We observe that only 20% of the benchmark was solved to optimality, but that feasible robust transfer plans were found for approximately 70% of the remaining instances. This shows that the benchmark is sufficiently challenging, and that the solver (CP-Optimizer) can run into difficulties, even for small instances.

To achieve better results, we need to consider the *ad hoc* procedures proposed in Sections 4.2 through 4.4, namely the weak (**wlb**) or the strong (**slb**) lower bounds, the symmetry-breaking constraints (**sym**), memorization (**mem**) and/or the look-ahead procedure (**la**).

When these procedures are considered one by one, we remark that the weak lower bound is the element that has the greatest impact on the performance of the solver (see column **wlb** in Table 2). However, we might wonder whether the strong lower bound really is relevant for this problem, given that the percentage of instances solved to optimality tends to be worse in column **slb** than in column **wlb**. The strong lower bound is tighter, but also has a higher time complexity than the weak lower bound. As is often the case, there is a balance to be struck between heavy computations and tight bounds *versus* light computations and weak bounds. Experimental results show that the weak lower bound is more effective, in practice, than the strong lower bound.

Memorization and symmetry-breaking constraints are also seen to be very effective. Using these features (together with the weak lower bound) enables more than 70% of the benchmark to be solved (see columns **wlb+mem** and **wlb+mem+sym** in Table 2). The impact of the look-ahead procedure is certainly more limited, but is worth mentioning (*cf.* **wlb+mem+sym+la**).

6. Conclusion

This paper investigates the problem of transferring data through a deterministic delay- and disruption-tolerant network. We address the problem of finding a Γ-robust transfer plan (routing scheme), *i.e.* a valid transfer plan that guarantees that the recipient nodes always receive all the datum units, even if some transfers – at most Γ – fail. We believe that this approach covers a wide range of constraints that are to be found in real applications.

We propose a constraint-programming-based algorithm This algorithm rests on a necessary and sufficient condition for a transfer plan to be robust. Promising numerical results are reported. We show that specific *ad hoc* filtering algorithms, *e.g.* some lower bounds, are *required* for the model to be solved efficiently.

Acknowledgement

These works are financed by the *Conseil Régional de Picardie*, and carried out in the framework of *Labex MS2T* (Maîtrise des systèmes-de-systèmes technologiques), 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] J. Alonso and K. Fall. A Linear Programming Formulation of Flows over Time with Piecewise Constant Capacity and Transit Times. Technical report, Intel Research, Berkeley, 2003.

[2] E. Altman, G. Neglia, F. De Pellegrini, and D. Miorandi. Decentralized Stochastic Control of Delay Tolerant Networks. In *The 28th Conference on Computer Communications, IEEE INFOCOM 2009*, pages 1134–1142. IEEE, 2009.

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

[4] D. Bertsimas and M. Sim. The Price of Robustness. *Operations Research*, 52(1):35–53, 2004.

[5] R. Bocquillon. *Data distribution optimization in a system of collaborative systems*. PhD thesis, Université de Technologie de Compiègne, 2015.

[6] R. Bocquillon and A. Jouglet. A constraint-programming-based approach to solve the data dissemination problem. *Computers and Operations Research*, 78:278–289, 2017.

[7] R. Bocquillon and A. Jouglet. Modeling elements and solving techniques for the data dissemination problem. *European Journal of Operational Research*, 256(3):713–728, 2017.

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

[9] B. Burns, O. Brock, and B. N. Levine. MV routing and capacity building in disruption tolerant networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 398–408. IEEE, 2003.

[10] 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 2003*, page 27, New York, NY, USA, 2003. ACM Press.

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

[12] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

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

[14] `C++` source code. `https://www.hds.utc.fr/~rbocquil/dokuwiki/_media/dp_code.zip`.

[15] Instances for both the robust and the original dissemination problem. `https://www.hds.utc.fr/~rbocquil/dokuwiki/_media/dp_instances.zip`.

[16] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. *ACM SIGCOMM Computer Communication Review*, 34(4):145, 2004.

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

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

[19] J. LeBrun and C.-N. Chuah. Bluetooth content distribution stations on public transit. In *Proceedings of the 1st international workshop on Decentralized resource sharing in mobile computing and networking, MobiShare 2006*, page 63, New York, NY, USA, 2006. ACM Press.

[20] A. Lindgren, A. Doria, and O. Schelén. Probabilistic Routing in Intermittently Connected Networks. *SIGMOBILE Mobile Computing and Communications Review*, 7(3):19–20, 2003.

[21] S. Merugu, M. Ammar, and E. Zegura. Routing in Space and Time in Networks with Predictable Mobility. Technical report, Georgia Institute of Technology, 2004.

[22] A. Pentland, R. Fletcher, and A. Hasson. DakNet: rethinking connectivity in developing nations. *Computer*, 37(1):78–83, 2004.

[23] A. Vahdat and D. Becker. Epidemic Routing for Partially Connected Ad Hoc Networks. Technical report, Duke University, 2000.

[24] W. D. Wood, Lloyd Ivancic, W. M. Eddy, D. Stewart, J. Northam, C. Jackson, and A. da Silva Curiel. Use of the Delay-Tolerant Networking Bundle Protocol from Space. In *59th International Astronautical Congress and Exhibition*, Glasgow, Scotland, United Kingdom, 2008.