Comparing Statistical and Analytical Routing Approaches for Delay-Tolerant Networks

Pedro R D'argenio, Juan A Fraire, Arnd Hartmanns, Fernando Raverta

Comparing Statistical and Analytical Routing Approaches for Delay-Tolerant Networks

Pedro R. D'Argenio[1,2], Juan A. Fraire[1,3], Arnd Hartmanns[4(✉)], and Fernando Raverta[1,2]

[1] CONICET, Córdoba, Argentina
[2] Universidad Nacional de Córdoba, Córdoba, Argentina
[3] Inria, Lyon, France
[4] University of Twente, Enschede, The Netherlands
a.hartmanns@utwente.nl

Abstract. In delay-tolerant networks (DTNs) with uncertain contact plans, the communication episodes and their reliabilities are known a priori. To maximize the end-to-end delivery probability, a bounded network-wide number of message copies are allowed. The resulting multi-copy routing optimization problem is naturally modelled as a Markov decision process with distributed information. The two state-of-the-art solution approaches are statistical model checking with scheduler sampling, and the analytical RUCoP algorithm based on probabilistic model checking. In this paper, we provide an in-depth comparison of the two approaches. We use an extensive benchmark set comprising random networks, scalable binomial topologies, and realistic ring-road low Earth orbit satellite networks. We evaluate the obtained message delivery probabilities as well as the computational effort. Our results show that both approaches are suitable tools for obtaining reliable routes in DTN, and expose a trade-off between scalability and solution quality.

1 Introduction

Delay-tolerant networks (DTNs) are time-evolving networks lacking continuous and instantaneous end-to-end connectivity [11,18]. The DTN domain comprises deep-space [9] and near-Earth communication [10], airborne networks [27], vehicular ad-hoc networks [5], mobile social networks [32], Internet of things scenarios [6], and underwater networks [40], among many others. A *bundle layer* overcomes the delay and disruption in DTNs by means of (i) a persistent storage on each DTN node and by (ii) assuming no immediate response from neighboring nodes [41]. As a result, *bundles* of data (a data unit in the Bundle Protocol [47])—and status information about the rest of the network—flow in a

This work was supported by Agencia I+D+i grants PICT-2017-1335 and PICT-2017-3894 (RAFTSys), DFG grant 389792660 as part of TRR 248, the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement 101008233 (MISSION), NWO VENI grant 639.021.754, and SeCyT-UNC grant 33620180100354CB (ARES).

E. Ábrahám and M. Paolieri (Eds.): QEST 2022, LNCS 13479, pp. 1–19, 2022.
https://doi.org/10.1007/978-3-031-16336-4_17

store-carry-and-forward fashion as transmission opportunities become available. Connectivity in DTNs is represented by means of *contacts*: an episode of time when a node is able to transfer data to another node.

Where contacts can be accurately predicted, the DTN is *scheduled* [22]; in *probabilistic* DTNs, the contact patterns can be dynamically inferred; no assumptions on future contacts can be made in *opportunistic* DTNs [11]. Recent work extended this classification to also consider *uncertain* DTNs, in which forthcoming connectivity can be described by probabilistic schedules available *a priori* [17,23,38,39,44,45]. Instead of a guaranteed contact plan, uncertain contact plans include information on the reliability (i.e. failure probability) of planned links. In other words, the materialization of contacts can differ from the original plan with a probability that can be computed/estimated in advance. Uncertain DTNs describe a plethora of practical scenarios: unreliable space networks [23], public vehicle networks with uncertain mobility patterns [35], interference-sensitive communication links in cognitive radio [46], or networks based on third-party carriers with limited but well-known availability [33].

This work summarises and compares existing routing solutions for uncertain DTNs. The state-of-the-art techniques are *lightweight scheduler sampling* (LSS) [17] and *routing under uncertain contact plans* (RUCoP) [44]. Both leverage Markov decision processes (MDPs), allow a bounded network-wide number of message copies to maximize the delivery probability, and properly assume that uncertain DTN nodes can only act on limited local knowledge. However, they are different in nature: LSS exploits simulation and statistical model checking techniques [1] whereas RUCoP is based on an analytical solution that exhaustively explores the MDP akin to probabilistic model checking [3,4]. Both are off-line approaches, as a central node is assumed to pre-compute the routing in advance and then distribute the required information to the DTN nodes.

We provide an extensive benchmarking framework to evaluate LSS and RUCoP comprising random networks (random contact assignment), binomial networks (multi-level tree contact topologies with controllable complexity), and realistic ring-road low-Earth orbit satellite networks. In these scenarios, we compare the resulting message delivery probability and computational effort in terms of time and memory consumption. Our results highlight the performance-cost trade-off between these two state-of-the-art routing techniques for uncertain DTN. We also report on our enhancements to encoding DTNs for use with LSS that significantly improve the cost/performance ratio of the approach.

Section 2 of this paper revises the background of DTNs, MDP, and modelling the routing problem. We dive into the details of the LSS and RUCoP techniques in Sect. 3, including a summary of our improvements to LSS for DTNs. We present, apply and analyze the benchmark framework in Sect. 4.

2 Background

This section describes the concept and context of DTNs and explains how to encode DTN routing with global and local information as MDP.

Scheduled vs. Uncertain DTNs. The term "DTN" was introduced in the context of interplanetary communication to designate time-evolving networks lacking continuous and instantaneous end-to-end connectivity [9]. The concepts and mechanisms devised to deal with the delays and disruptions of interplanetary communications can readily be applied to other domains characterized by long signal propagation time, frequent node occlusion, high node mobility, and/or reduced communication range and resources [24] such as airborne, vehicular, social, IoT, underwater and space networks [5,6,10,21,27,32,40]. DTN protocols like the Bundle Protocol [13,47] address the delays and disruptions by implementing the principles of *store-carry-and-forward* and *minimal end-to-end messaging exchange* for control or feedback [11]. The time-evolving and partitioned nature of DTNs favors representing connectivity by *contacts*: episodes of time where a node can transfer data to another node. Contacts can be classified [11] as *opportunistic* (no assumptions can be made on future contacts), *probabilistic* (contact patterns can be inferred from history, e.g. in social networks), and *scheduled* (contacts can be accurately predicted and documented in a *contact plan*).

A contact plan comprises the set of forthcoming contacts, and is a central element in scheduled DTN routing. The routing process is typically divided into *planning* (future episodes of communication are estimated to form the contact plan), *routing* (the plan is used to compute routes, either in a centralized (off-line) or decentralized (on-line) fashion [20]), and *forwarding* (effectively enqueuing the data for the correct next-hop node). Contact graph routing (CGR) [2] is the de-facto standard decentralized routing algorithm when a contact plan is available. It is the sole routing approach that has been flight-validated in deep-space [48] and near-Earth networked missions [34]. CGR optimizes delivery *time* by leveraging adaptations of Dijkstra's algorithm to the time dynamics of DTNs.

The limitation of the contact plan structure and associated routing algorithms like CGR is that they assume that connectivity episodes are guaranteed. Instead, an *uncertain contact plan* comprises contacts whose materialization can differ from the original plan with a given probability available a priori [45]. Reasons include well-known failure modes of the DTN nodes, or an incomplete/inaccurate knowledge of the system status by the time the schedule was computed. Uncertain contact plans gave raise to a new type of DTNs coined *uncertain DTNs* [17,38,39,44,45] that exploit time-dependent probabilistic information of the forthcoming communication opportunities. Instead of a single copy sent via the fastest path like CGR, uncertain DTNs can use the uncertainty information in the contact plan to optimally route *multiple copies* of the data to increase its successful delivery probability (SDP).

Markov decision processes (MDPs) provide a mathematical framework capturing the interaction between non-deterministic and probabilistic choices [19,42], making them appropriate for modelling decision making under probabilistically quantified uncertainty. In its simplest form, an MDP $\mathcal{M}$ is a tuple $(S, Act, \mathbf{P}, s_0)$ where S is a finite set of states with initial state $s_0 \in S$, Act is a finite set of actions, and $\mathbf{P} : S \times Act \times S \to [0,1]$ is a transition probability function such that $\sum_{s' \in S} \mathbf{P}(s, \alpha, s') \in \{0, 1\}$ for all $s \in S$ and $\alpha \in Act$. If $\sum_{s' \in S} \mathbf{P}(s, \alpha, s') = 1$,

α is *enabled* in s, and $\mathbf{P}(s, \alpha, s')$ gives the probability that the next state is s' conditioned on the system being in state s and action α being chosen.

A *reachability problem* is characterized as follows: given a set of *goal states* $B \subseteq S$, maximize the probability that a state in B is reached from the initial state s_0. That is, we want to calculate $Pr^{\max}_{s_0}(reach(B))$. In our application, B is the set of states in which a bundle has been successfully delivered. Moreover, we are also interested in determining the decisions—namely, the *policy* or *scheduler*—that lead to such a maximizing value. A *scheduler* is a function $\pi : S \to Act$ that defines the decision that resolves a possible non-determinism. This problem can be solved e.g. by using value iteration on the Bellman equations [4].

Encoding Uncertain Contact Plans. Consider the example contact plan with nodes A, B, C, and D in Fig. 1. It spans a window of five time slots, t_0 to t_4. We also assume an ending time t_5. The possible contacts in each slot are depicted by an arrow labelled with the contact failure probability. In time slot t_1, for instance, node C is in reach of node B with transmission failure probability of 0.1 (and success probability of 0.9).

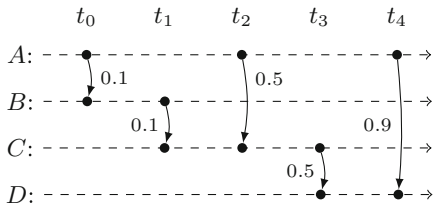


Fig. 1. Uncertain contact plan.

Suppose we want to transmit a bundle from A to D. To increase the probability of success, we allow two copies throughout the network. A state of the MDP consists of the number of copies that each node holds at a given time slot. Initially, at the beginning of t_0, node A has the two copies while the others have none, represented by state $[A^2\, B^0\, C^0\, D^0 \,|\, t_0]$ in Fig. 2. At this point, node A has three options: (i) sending only one copy to node B, represented by action "$A \xrightarrow{1} B$" leaving from state $[A^2\, B^0\, C^0\, D^0 \,|\, t_0]$, (ii) sending two copies to B (action "$A \xrightarrow{2} B$"), or (iii) keeping the two copies (action "A stores"). In the first case, the successful transmission leads to state $[A^1\, B^1\, C^0\, D^0 \,|\, t_1]$ where A has kept one copy and the other has reached B. Since success probability is 0.9, we have

$$\mathbf{P}(\,[A^2\, B^0\, C^0\, D^0 \,|\, t_0],\ A \xrightarrow{1} B\,,[A^1\, B^1\, C^0\, D^0 \,|\, t_1]\,) = 0.9.$$

Failing to transmit moves us to the next time slot without altering the number of copies in each node. Therefore

$$\mathbf{P}(\,[A^2\, B^0\, C^0\, D^0 \,|\, t_0],\ A \xrightarrow{1} B\,,\ [A^2\, B^0\, C^0\, D^0 \,|\, t_1]\,) = 0.1.$$

Action $A \xrightarrow{1} B$ is the black transition out of $[A^2\, B^0\, C^0\, D^0 \,|\, t_0]$ in Fig. 2 where the solid line represents the successful transmission while the dotted arrow represents the failing event. The situation is

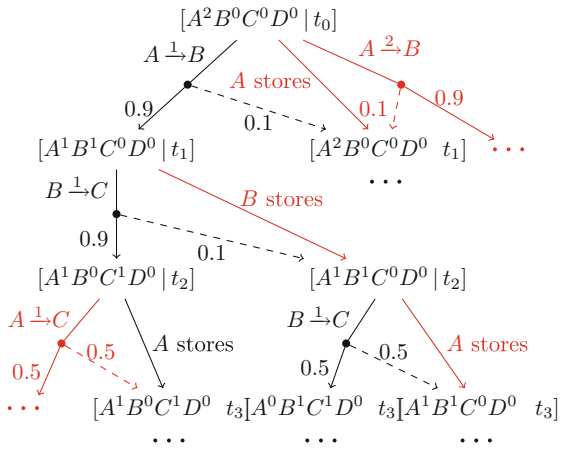


Fig. 2. MDP modelling the plan of Fig. 1. (Color figure online)

analogous for action $A \xrightarrow{2} B$ (red transition on the right), while for storing the two bundles there is no possibility of failure, so we have

$$\mathbf{P}(\,[A^2\,B^0\,C^0\,D^0\,|\,t_0],\ A \text{ stores},\,[A^2\,B^0\,C^0\,D^0\,|\,t_1]\,) = 1.$$

The construction is similar for the rest of the MDP. Figure 2 depicts it partially; we indicate with "..." where the MDP needs to continue.

We assume that the sending node can determine whether a transmission was successful or not; in case of success, it deletes the transmitted number of copies, while in case of failure, it keeps them. This ensures that the entire network contains the intended number of copies at any time, and is possible and typical in LEO constellations. We refer to this assumption as *acknowledged communication* (a.k.a. custody transfer in the Bundle Protocol [24]). The alternative is *fully unreliable* communication where transmitted copies are lost upon failure, which is natural in deep-space communication.

Global and Local Information. For the MDP described above, the maximizing scheduler for goal set $B = \{\,[A^a\,B^b\,C^c\,D^d\,|\,t_5] \mid d \geq 1\,\}$ describes the optimal routing decisions. This scheduler, however, is based on a *global* view of the system: decisions are taken based on the current state of the whole network. This implies that distributed nodes need to know where all copies are in the network at any moment, including remote and potentially disconnected nodes. This is impossible to achieve in practice in highly partitioned DTNs. Nodes must therefore decide based on partial local knowledge. To illustrate, consider time slot t_2 in the example of Fig. 1. Here, node A has two possible decisions: storing or forwarding to C. Consider precisely the situation in which A has one copy and the second copy is already on its way. A's optimal decision depends on whether the other copy is on B or C at time t_2, reflecting the optimal decisions on Fig. 2: A stores if C already has the other copy and A forwards to C if B has the copy. However, it is most likely that A is not able to know whether the second copy is in B or C, in which case A's decision should be the same regardless if it is in state $[A^1\,B^1\,C^0\,D^0\,|\,t_2]$ or $[A^1\,B^0\,C^1\,D^0\,|\,t_2]$. This type of problem, in which decisions in an MDP associated to a distributed system may only be based on *local* knowledge, is known as *distributed scheduling* [12,25,26].

3 Routing in Uncertain DTNs

The optimal *global* scheduler can be computed using any probabilistic model checker such as Prism [36], Storm [16], or mcsta of the Modest Toolset [30]: we compactly describe the MDP and the goal set in the tool's higher-level input language; then the tool generates and stores in memory the MDP's entire state space, solves the reachability problem by solving the linear program induced by the Bellman equations or by using an iterative algorithm such as a sound variant of value iteration [28,31,43], and writes the induced scheduler to file. Probabilistic model checkers, however, are generic tools that solve arbitrarily structured MDP without optimizations for the DTN routing application. For complex networks, they will quickly encounter the state space explosion problem and run

out of memory (see [44]). Furthermore, none of them provides a solution for the local-information problem. We now summarize the two existing MDP-based approaches for optimal DTN routing under uncertain contact plans, RUCoP and LSS. Both can also produce schedulers based on *local* information only, and approach the routing process in an off-line fashion: the routing decisions are pre-computed in a centralized node.

3.1 RUCoP

RUCoP [45] (routing under uncertain contact plans) provides an analytical solution to find the routing decisions optimising SDP for an uncertain contact plan.

The first observation exploited by RUCoP is that, due to the inclusion of the current time slot value in the states, the MDP for an uncertain contact plan is acyclic. RUCoP thus only constructs the "optimal" part of the MDP by following the Bellman equations backwards. In our example from the previous section, it starts at any state in t_5 in which D contains at least one copy. It then walks backwards in the contact plan, selecting only the maximizing transitions according to the Bellman equations. In its general form, RUCoP (i) considers the possibility that multiple nodes can transmit to each other in one time slot, which may produce a cycle in the MDP. However, since cyclic transmission would only lower the SDP, RUCoP can break all such cycles and keep the MDP acyclic. It also in general (ii) takes a target node and builds the optimal part of the MDP for any possible transmitting source rather than restricting to a single source node as in our example. The full RUCoP algorithm is in 2-EXPTIME: its runtime is exponential in the number of nodes and doubly exponential in the number of copies. This makes RUCoP highly expensive in time and memory. However, for memory optimization, RUCoP not only constructs the optimal part of the MDP backwards in an on-the-fly manner, but also writes all information that is not going to be necessary for further calculations to disk. In particular, only the states at the current time slot are necessary for calculating the states at the preceding time slot and the respective connecting optimal transitions.

RUCoP delivers optimal routing decisions for acknowledged communication in general. However, it is based on a global view of the system. To find local-information schedulers, we need to use its L-RUCoP (local RUCoP) variant. It works as follows: Suppose that, to increase reliability, n copies of the bundle are used. L-RUCoP builds a table $T(N, c, t_i)$ that assigns to each node N holding c copies ($1 \leq c \leq n$) at time t_i the best decision based on local knowledge. This decision is taken from running RUCoP on c copies (instead of n), which basically amounts to supposing that N holds c copies and no copy is on the other nodes. Thus, for our example, the decision for states $[A^1\, B^1\, C^0\, D^0 \,|\, t_2]$ and $[A^1\, B^0\, C^1\, D^0 \,|\, t_2]$ will be both taken from $T(A, 1, t_2)$ which in turn is obtained from the decision in state $[A^1\, B^0\, C^0\, D^0 \,|\, t_2]$ derived from running RUCoP with one single copy. On top of this basic idea, L-RUCoP also exploits extra knowledge that may be available in certain occasions. For instance, at time t_1 in our example, A knows if B holds a copy depending on whether the transmission at time t_0 was successful or not. In this case, L-RUCoP looks ahead using the

appropriate RUCoP instance on the state with the *available* knowledge where, just like before, all information about the *other* (unknown) copies is assumed to be 0. In the example, at time t_1, the entry $T(A, 1, t_1)$ will be filled with the information retrieved from RUCoP for two copies on state $[A^1\, B^1\, C^0\, D^0 \mid t_1]$ since A knows B has received the copy. The interested reader may find the details of L-RUCoP as well as the full specification of RUCoP in [45].

3.2 LSS

Given a discrete-time Markov chain (DTMC), i.e. an MDP where every state has at most one enabled action, Monte Carlo simulation or *statistical model checking* (SMC [1]) can be used to estimate the probabilities for reachability problems: We (pseudo-)randomly sample n paths—*simulation runs*—through the DTMC, identify each success (that reaches a goal state) with 1 and every failure with 0, and return the average as an estimate of the reachability probability. The result is correct up to a statistical error and confidence depending on n. Compared to probabilistic model checking, SMC needs only constant memory, assuming that we can effectively simulate the MDP from a high-level description so that we do not need to store its entire state space. As a simulation-based approach, SMC is easy to parallelize and distribute on multi-core systems and compute clusters.

Lightweight scheduler sampling [37] (LSS) extends SMC to MDP: Given an MDP M, it (i) randomly selects a set Σ of m schedulers, each identified by a fixed-size integer (e.g. of 32 bits), (ii) employs some heuristic (that involves simulating the DTMCs $M|_\sigma$ resulting from combining M with a scheduler $\sigma \in \Sigma$) to select the $\sigma_{max} \in \Sigma$ that appears to induce the highest probability, and finally (iii) performs a standard SMC analysis on $M|_{\sigma_{max}}$ to provide an estimate $\hat{p}_{\sigma_{max}}$ for $Pr_{s_0}^{\max}(reach(B))$. However, note that—unless we are lucky and Σ happens to include an optimal scheduler and the heuristic identifies it as such—$\hat{p}_{\sigma_{max}}$ is an underapproximation of $Pr_{s_0}^{\max}(reach(B))$ only, and subject to the statistical error of the SMC analysis. The effectiveness of LSS depends on the probability mass of the set of near-optimal schedulers among the set of all schedulers that we sample Σ from: It works well if a randomly selected scheduler is somewhat likely to be near-optimal, but usually fails in cases where many decisions need to be made in exactly one right way in order to get a successful path at all. We use the *smart sampling* [15] approach to select σ_{max} in step (ii): We start by performing 1 simulation run for each of the m schedulers, then discard the $\lceil \frac{m}{2} \rceil$ worst of them; in the next round, we perform 2 runs for each of the approx. $\frac{m}{2}$ remaining schedulers, and again discard the worst half. We continue until only one scheduler remains, which is σ_{max}. In this way, the number of simulation runs, and thus the runtime, needed for LSS grows only logarithmically in m.

The key to LSS is the constant-memory representation of schedulers as (32-bit) integers. It enables LSS' constant memory usage in the size of the MDP, which sets it apart from simulation-based machine learning techniques such as reinforcement learning, which need to store learned information (e.g. Q-tables) for each visited state. Let $i \in \mathbb{Z}_{32}$ identify scheduler σ_i. Then, upon encountering a state s with $k > 1$ enabled actions while simulating $M|_{\sigma_i}$, LSS selects the $(\mathcal{H}(i.s) \bmod k)$-th

action, where $i.s$ is the concatenation of the binary representations of s and i, and $\mathcal{H}$ is a (usually simple non-cryptographic) hash function that maps its inputs to a fixed-size integer so that, ideally, the resulting values are uniformly distributed over the output space. This selection procedure is deterministic, so we can reproduce the decision for state s at any time knowing i. For nontrivial $\mathcal{H}$, it is also highly unpredictable: changing i, e.g. by modifying a certain bit, may result in a different decision for many states.

Local Information. As described above, LSS produces global-information schedulers. However, it can be adapted to sample from local-information schedulers only [17]: When having to make a decision on node N, instead of feeding $i.s$ into $\mathcal{H}$, we use $N.i.s|_N$ instead, where $s|_N$ contains only the locally available information: the number of locally-stored copies and the current time slot. To avoid conflicts where two nodes need to make a decision at the same time, certain restrictions apply to the high-level modelling of the MDP as a system of multiple independently executing nodes; we refer the interested reader to [17] for details. We refer to LSS with local-information schedulers as L-LSS.

A scheduler found to be good via L-LSS can in principle be implemented, e.g. on the satellites themselves, by simply replicating the L-LSS decision procedure: each node knows its identifier N, the number of copies it stores, and can translate the current time into a time slot in the contact plan. The only data that needs to be transmitted to the node is the integer identifying the scheduler.

Our Improvements to LSS for DTN. For our comparison in Sect. 4, we use the implementation of DTN routing with LSS and L-LSS of [17]. It consists of two parts: a CP2MODEST Python script that converts a contact plan into a high-level description of the MDP as described in Sect. 2 in the MODEST modelling language [29], and an implementation of LSS and L-LSS in the MODES simulator/statistical model checker [7] of the MODEST TOOLSET. We use the latter as-is, but have added preprocessing based on decisions already implemented in RUCoP to the former in order to produce more succinct MDP models as follows:

1. *Useful contacts only.* A contact may be useless for transmitting a message from the source to the target node because it leads to a dead-end, i.e. a situation where a message copy is transmitted to node X in time slot t but there is no sequence of contacts reaching the target from X after t. Similarly, there may not be any sequence of contacts from the source to X before t: then X is guaranteed not to have any copies in t. We analyse the contact graph for such situations and drop all useless contacts. This reduces the amount of decisions in the MDP, and thus the number of schedulers to sample from, without excluding any scheduler with positive message delivery probability. Consequently, (near-)optimal schedulers are more likely to be sampled.
2. *Forcing to send.* With the same motivation, when we are in node X's last useful contact, it would be useless to keep any copies. Thus, for such contacts, the only option that we generate now is to send all available copies.
3. *Forcing to receive.* Like a node deciding to store all copies at a contact, i.e. choosing not to send, the previous translation allowed the receiving node to

ignore the incoming transmission (which would consequently look like a failure to the sender). While this allowed some interesting collaborations between nodes to share non-local information [17, Sect. 5.3], we are not interested in such special behaviours, and consequently omit the option to ignore an incoming message. This again reduces the scheduler sampling space.

4. *Skipping empty slots.* The previous translation generated a "clock tick" action to advance time from t to $t+1$ in all nodes for every time slot, even if that slot had no contacts. To improve simulation runtime, we now omit these actions for empty slots and directly skip ahead to the next slot with a contact.

All combined, these improvements eliminate many useless schedulers from the sample space, making (L-)LSS noticeably more likely to find good ones; they also simplify the model, improving the runtime and memory consumption of MODES. We will showcase the difference on one of our benchmarks in Sect. 4.

4 Evaluation

In order to evaluate the performance-cost trade-off of LSS and RUCoP in uncertain DTNs, we created a benchmark set consisting of three use cases to compute SDP metrics and the associated computational cost.

4.1 Benchmark Set

Random networks use a uniform distribution of contacts among a configurable number of network nodes and contact plan duration. We use 10 random topologies with 8 nodes, each covering a duration of 100 s. Time is discretized into episodes of 10 s. In each episode, the connectivity between nodes (i.e. the presence of contacts) is decided based on a contact density parameter of 0.2, similar to [39]. We assume an all-to-all traffic pattern, run each of the routing algorithms 100 times on each of the 10 networks, and report the averages.

Binomial Networks. To gain insights into how increasing the topological complexity affects the routing algorithms, we devised a family of contact plans with a binomial topology. They are easy to scale up in a controlled manner that preserves the characteristics of the topology. The topology is a binomial tree. The higher the number of levels in the tree, the more complex the routing problem is to solve. Specifically, a binomial topology with L levels implies: (i) $1+2^{L-2}$ nodes have contacts with two neighbors; (ii) $\sum_{i=1}^{i<L-2} 2^i$ nodes have contacts with three neighbors; and (iii) 1 final destination node has 2^{L-2} contacts. The resulting tree is illustrated in Fig. 3. Contacts between consecutive levels

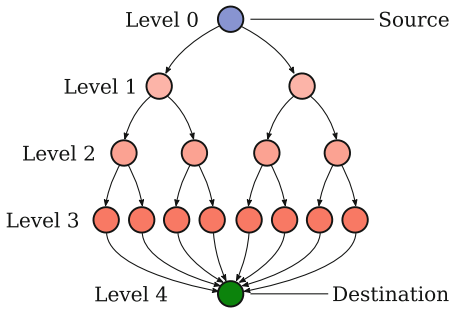
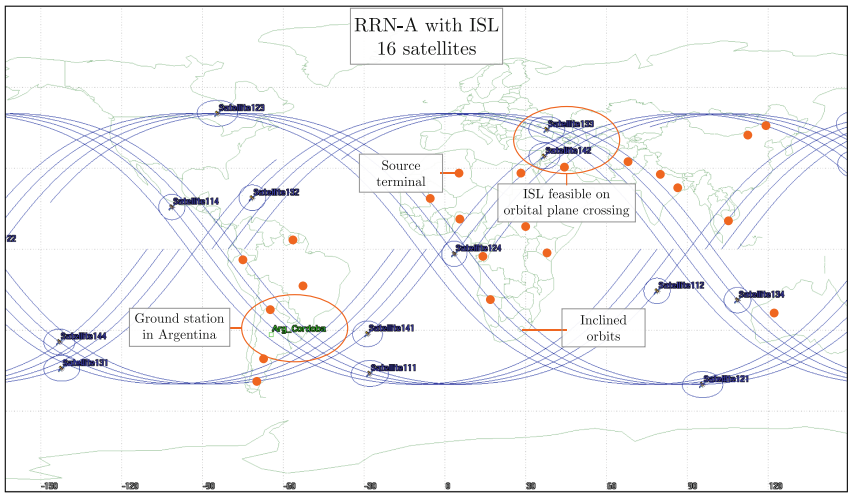


Fig. 3. Binomial tree.



RRN with ISL - 16 LEO satellites - 1 ground station (dst) - 22 ground terminals (src)					
Name	**T. Anomaly [deg]**	**Altitude [km]**	**Arg. Perigee [deg]**	**Inclination [deg]**	**RAAN [deg]**
Satellite111,12,13,14	0, 90, 180, 270	500	0	50	0
Satellite121,22,23,24	23, 113, 203, 293	500	0	50	90
Satellite131,32,33,34	45, 135, 225, 315	500	0	50	180
Satellite141,42,43,44	68, 158, 248, 338	500	0	50	270
Name	**Latitude [deg]**	**Longitude [deg]**	**Altitude [km]**		
Arg_Cordoba	-31.5242	-64.4636	0.724		

Fig. 4. RRN satellite constellation topology, parameters and orbital tracks [23].

are also consecutive in the time dimension, that is, the order of the contacts corresponds to enumerating the arrows in Fig. 3 left-to-right, top-to-bottom. A node on the i-th level will have a total of 2^{L-2-i} paths to the destination. Therefore, the larger the level count, the more nodes are in the network and the more paths per node have to be evaluated. For example, a binomial topology of 6 levels results in 32 nodes with up to 32 simple paths. When considering the forwarding of 3 copies, a total of $91,000$ possible actions need to be considered.

Ring Road Networks. Finally, we use a realistic satellite topology exported from high-precision orbital propagators. Specifically, we consider a low-Earth orbit Walker constellation of 16 satellites as proposed and described in [23]. Satellites act as data mules by receiving data from 22 isolated ground terminals, storing the data, and delivering it to a ground station placed in Argentina. We use an all-to-one traffic pattern. The satellites are equipped with inter-satellite links (ISLs), so contacts are possible in orbit. The dynamics of the topology and the specific orbital and ground parameters are depicted in Fig. 4. Routes can involve multiple hops between satellites and ground terminals. The scenario spans 24 h and is sliced into 1440 time slots, each of 60 s. Within a time slot, we consider a contact feasible if communication is possible for more than 30 s.

4.2 Analysis

Our evaluation results present compelling evidence of the trade-off between the LSS and RUCoP approaches, both in their global (LSS and RUCoP) and local

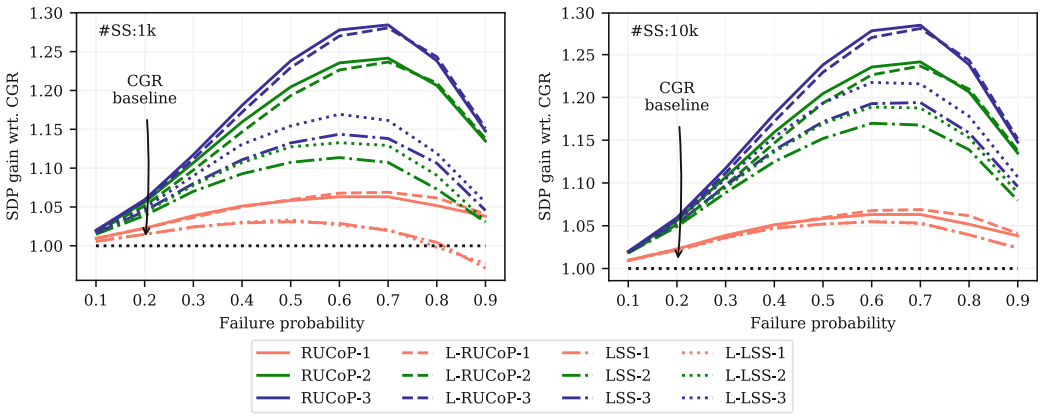


Fig. 5. SDP gain over CGR in random networks.

versions (L-LSS and L-RUCoP). We evaluate them in terms of the SDP of the computed scheduler, and the computational resources used: processing time and memory consumption. Plain single-copy CGR is used as a baseline. We write "(L-)RUCoP-c" and "(L-)LSS-c" for the respective method when allowing c copies. We used an Intel Core i5-5300U (2 cores, 4 threads, 2.3–2.9 GHz) system with 12 GB of memory running 64-bit Ubuntu 18.04.5 for all experiments.

Random Networks. The SDPs we obtained for random networks are illustrated in Fig. 5. To facilitate the interpretation of the outcomes, we plot the curves with respect to the SDP delivered by CGR. Indeed, CGR is the baseline of comparison as it assumes a perfect contact plan that does not drift from reality. As the contact plan becomes more uncertain, the RUCoP- and LSS-based schemes provide increasingly better SDPs. This holds up to the point where the failure probability is such that the partitioning of the topology dominates (i.e. $p_f \approx 0.8$), a situation in which delivery of data becomes much more difficult. Still, in these cases, RUCoP and LSS perform noticeably better than CGR.

We ran LSS and L-LSS in two configurations, one sampling $m = 1000$ and one sampling $m = 10000$ schedulers. We indicate m as "`#SS`", the number of sampled schedulers, in our figures. From Fig. 5, we observe that increasing m from 1000 to 10000 does not improve the SDP drastically in these random networks. In particular, averaged along all failure probabilities, sampling $m = 10000$ schedulers improves SDP by ≈1.8%, with ≈5.8% being the maximum gain registered at $p_f = 0.7$. We explain this limited improvement with the simplicity of the random topologies, which are easily explored with few schedulers.

When compared to L-RUCoP, L-LSS is, on average, 3% and 1% worse in terms of SDP, for 1000 and 10000 schedulers, respectively. The larger difference is observed at $p_f \approx 0.7\%$ and 3 copies, where L-RUCoP outperforms L-LSS by 10%. We observe that the lower the number of copies, the smaller the difference between L-RUCoP and L-LSS, with the single-copy case almost identical in SDP. Interestingly, the single-copy case provides limited or no gain with respect

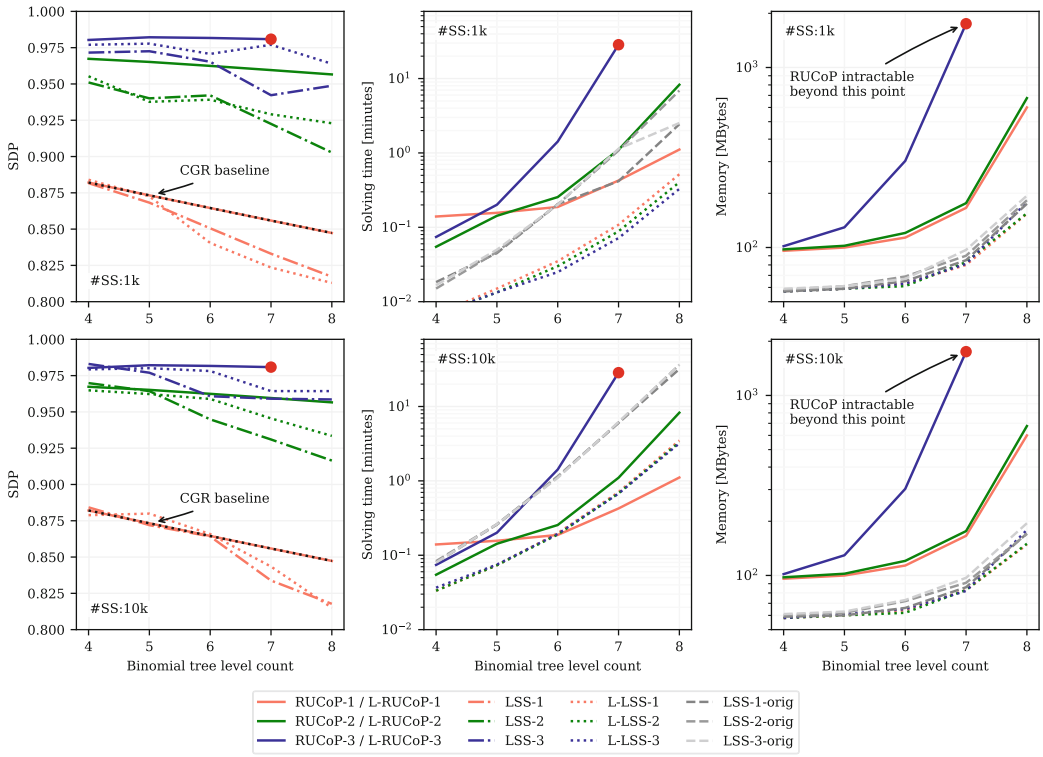


Fig. 6. SDP, solving time, and memory for binomial networks with varying complexity.

to the CGR baseline in these simple topologies. A similar effect was reported for Opportunistic CGR in [8].

Regarding the processing and memory footprint for random networks, all the techniques we study always complete in less than 20 s, using less than 20 MB of memory. Also, we observed that the runtime and memory values were rather stable and independent of the failure probability. In the following, we thus leverage the more complex binomial and ring-road topologies for a more detailed time and memory consumption assessment.

Binomial Networks Analysis. The results obtained for binomial networks are plotted in Fig. 6. All links in the topology were set to a failure probability of 0.1 in this case. Instead, we vary the tree level count from 4 to 8 (i.e. 8 to 128 nodes, and 13 to 449 paths), to evaluate the performance of RUCoP and LSS with increasing topological complexity, and thus, increasing routing decision making difficulty. Results are expressed, from left to right in the figure, in terms of SDP, solving time, and required memory.

In the binomial topologies, the CGR baseline is always equal to RUCoP with one copy (RUCoP-1) since the path with the earliest delivery time is also the one with highest SDP. On the other hand, the global view of RUCoP can be directly implemented with a limited local view. This is because each node can only reach two exclusive neighbors, which means that the local information is already enough to take a globally-optimal decision (i.e. the amount of copies to

send to one of the two next hop nodes). As a result, L-RUCoP and RUCoP plots in Fig. 6 are presented in a single curve (solid line).

On the one hand, the SDP plots show that LSS is rather close to RUCoP when leveraging 10000 schedulers, especially for low level counts (with less than 0.01% difference). In the worst-case scenario with 8 levels, L-LSS is only 3% below L-RUCoP for the single and dual copy scenarios. However, due to memory exhaustion, RUCoP (and thus L-RUCoP) fails to deliver a valid routing schedule for 8 levels and 3 copies (its limit highlighted by the red circle in Fig. 6). We verify that for this case, more than 15 million actions need to be considered in the MDP. Another observation from these plots is that the delivery probability when using dual copies increases from ≈0.88 to ≈0.97 (i.e. by 10%) for 4 levels and from ≈0.85 to ≈0.96 (i.e. by 13%) for 8 levels. However, due to the binomial nature of the topology, having a third copy provides limited or no advantage.

Regarding the time and memory requirements in the binomial topologies, RUCoP proves to be by far the most demanding approach. In the worst case solved for 3 copies (7 levels), RUCoP needs 28 min of computation time, compared to less than 10 s for LSS with 1000 schedulers, or 1 min with 10000 schedulers. This is a notable difference considering the similar performance in terms of SDP. Solving time and memory plots of the original LSS as in [17], i.e. without the improvements described in Sect. 3.2, are also plotted in Fig. 6, in gray dashed lines. These improvements reduce LSS runtime by up to ≈600% (from 117 down to 17 s). A reduction of ≈6% in memory is also achieved. Indeed, in memory utilization, RUCoP quickly escalates up to more than 1 GB to keep track of the MDP decision tree, while lightweight schedulers never require more than 100 MB, even for the most complex binomial topologies.

In summary, for binomial topologies, LSS and L-LSS with 10k schedulers closely follow RUCoP and L-RUCoP in delivery probability and solving effort for simple trees. As the topology's complexity rises (notably for more than 7 levels), RUCoP exhausts the available memory. Even in these challenging cases, LSS is able to deliver a valid solution with minimal runtime and memory footprint.

Ring Road Networks Analysis. We have evaluated all downlink source-destination pairs in the realistic RRN network. Figure 7 present some representative cases for the different behaviors we observed. In this figure, node 38 as the destination stands for the mission control center on ground, while node 1 and 7 are remote nodes sending data via the ring-road satellites[1]. For these nodes, we present the computation of the routing schedule for varying contact plan sizes, spanning durations from 1 to 3 h (plots from top to bottom). The `#SS` parameter is again varied to 1000 and 10000 schedulers, to gain sensitivity on the improvement of the sampling technique (plots from left to right).

The SDP plots in Fig. 7 show that the longer the contact plan, the more noticeable the difference between the analytic and statistical approaches (i.e. curves separate progressively). In particular, there is barely any difference for any failure probabilities for the shorter contact plan with 1 h of scheduling horizon.

[1] Nodes 1 and 7 correspond to nodes 8 and 15 in the contact plan used in [23].

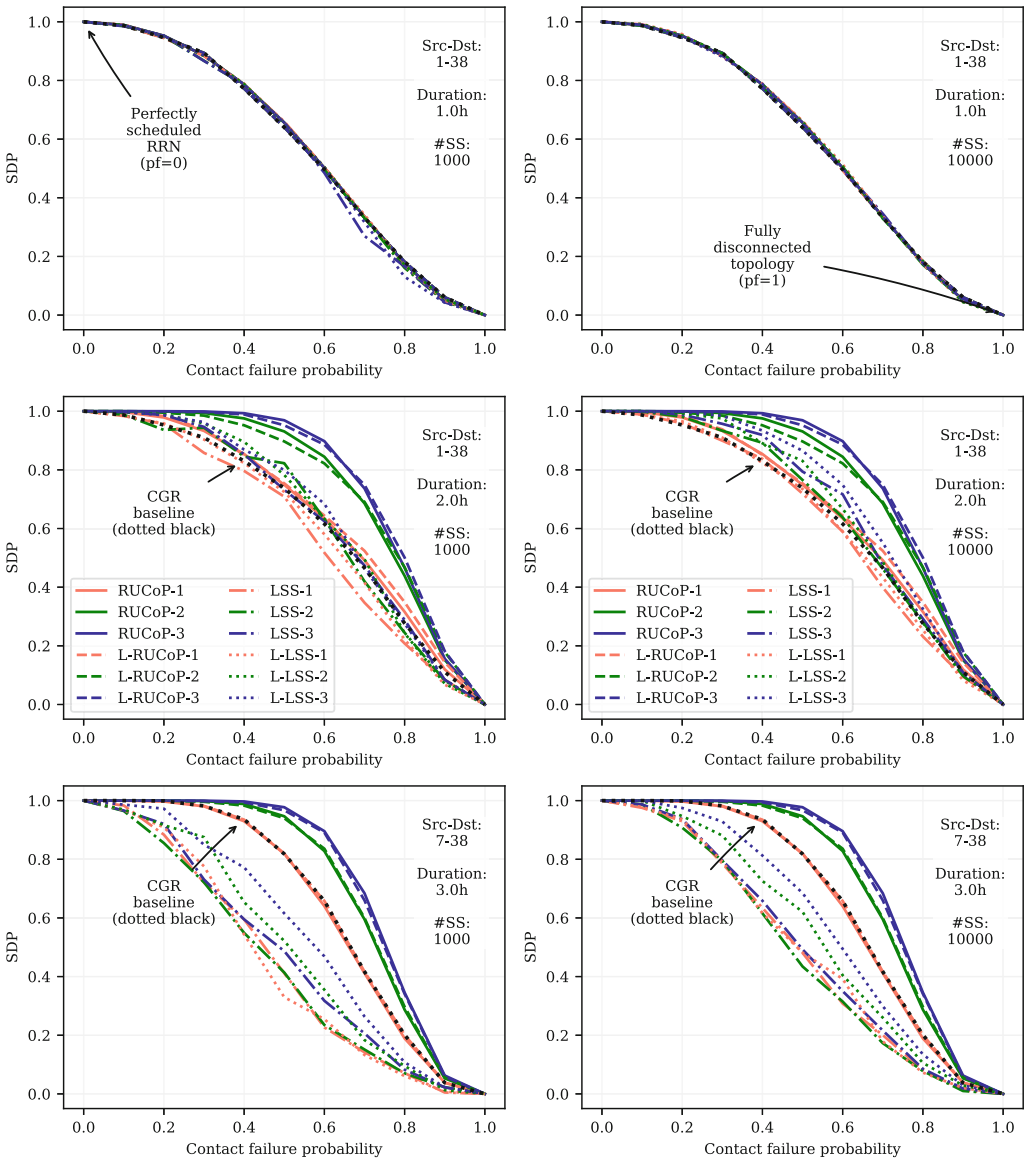


Fig. 7. SDP for RRN for different source-target nodes and plan durations.

However, we observe that L-RUCoP is notably superior to L-LSS for the 2 h and 3 h plans, especially for failure probabilities between 0.4 and 0.8. Specifically, we observe that the gap between RUCoP and LSS can be as large as ≈60%, for failure probabilities of ≈0.6, and contact plans of 3 h. Interestingly, the gap is reduced to ≈30% if we raise the number of schedulers to 10000 in LSS, indicating that this case is right on the boundary of what can effectively be solved via LSS. Nevertheless, both LSS and L-LSS perform worse than the CGR baseline even when leveraging multiple copies in schedules larger than 2 h. This is compelling evidence that the uninformed sampling strategy of LSS may not be fully adequate for realistic RRN topologies, even though it performed pretty well in generic

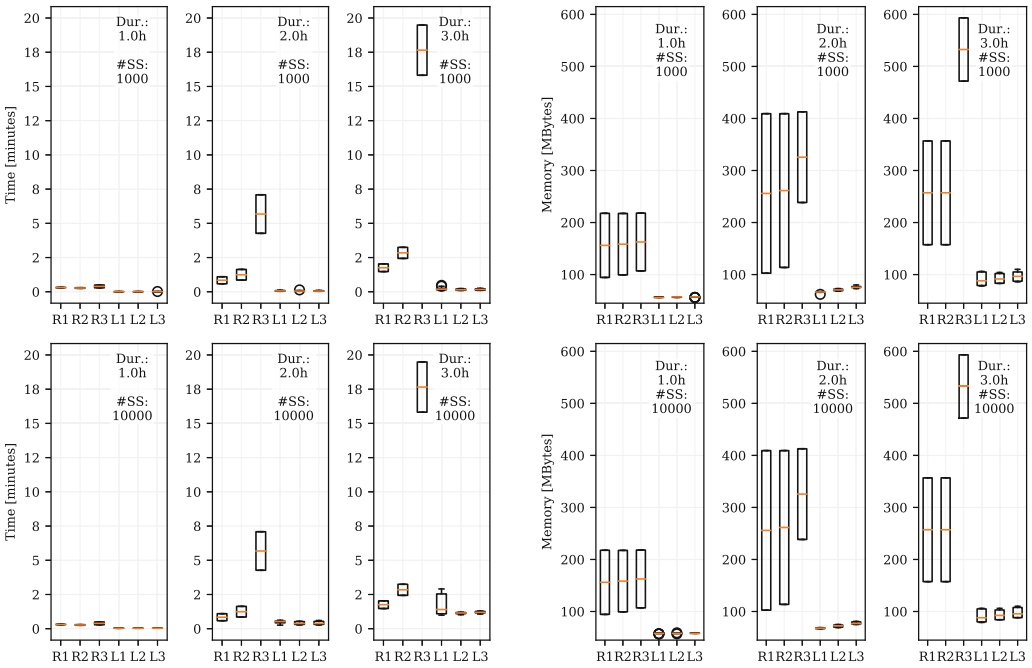


Fig. 8. Solving time (left) and memory (right) for RRN for different source-target nodes, contact plan durations, and numbers of schedulers sampled (R: RUCoP, L: LSS).

binomial and random topologies, and may need to be adapted to a variant yet more specifically tailored to the DTN routing application.

Also, we observe that LSS and L-LSS are typically close, but L-LSS frequently presents better SDP than the global LSS. This was also observed in Fig. 5, but in a much more subtle manner. We explain this phenomenon with the fact that L-LSS has a reduced space of schedulers to be sampled from, which increases the chances of finding a better routing policy.

Figure 8 presents the computational resources required to obtain the discussed SDP results for ring-road networks. This figure is computed based on the computational effort of solving several downlinking node pairs (instead of the two example pairs discussed in Fig. 7). The results confirms once again that RUCoP is able to deliver network performance at the expense of significantly higher memory and runtime. In particular, the runtimes for the analytical approach can reach up to ≈20 min (for the 3-h contact plan, with 3 copies), while LSS typically delivers a result in less than 1 min. We thus postulate that the 3 h contact plan is as challenging for RUCoP as the 7-level binomial topology, i.e. that larger contact plans are likely intractable for RUCoP. Memory-wise, we observe similar ratios. While RUCoP needs as much as 600 MB of memory for the worst-case scenario, LSS consistently uses about 100 MB. Again, this is due to the simulation nature of LSS, where no decision trees need to be stored as in RUCoP. Interestingly, LSS also showed a limited computational cost sensitivity to increasing L-LSS from 1000 to 10000. This is likely due to the possibility of using multiple CPU threads concurrently to perform the exploration in

LSS. Indeed, LSS can exploit parallelization intensively: each scheduler can be evaluated independently in separate threads. However, in RUCoP, the calculations for each time slot strongly depend on the successor time slot, which limits parallelization.

In summary, the evaluation over realistic ring-road networks showed that there is still room for improvement on scheduler sampling techniques to cope with more heterogeneous or application-specific topologies. In our particular satellite constellation, L-RUCoP provided delivery probabilities up to 60% higher than LSS, at higher computational costs. The reported runtimes and memory usages anyway appear reasonable for this kind of satellite application. In particular, since satellites revisit ground stations at most every ≈90 min [22], solving times of 20 min, as measured for RUCoP, are by all means acceptable.

5 Conclusions

This paper provides the first extensive comparison of the state-of-the-art analytical and statistical routing approaches for uncertain DTNs. While both RUCoP and LSS leverage MDP models, the former performs an exhaustive and optimal exploration of the solution space whereas the latter exploits SMC with sampling for optimization. We improved the DTN models for LSS for efficiency. We thoroughly compared the two approaches in a new benchmarking framework comprising random, binomial, and realistic satellite network topologies.

The outcomes provided quantitative evidence of the performance of the global- and local-information flavors of RUCoP and LSS. On the one hand, both schemes provide routes that deliver up to 1.8 times the data volume achievable by the baseline CGR approach. However, we touched the tractability limits of RUCoP in binomial networks of 8 levels. While RUCoP failed to deliver, LSS was able to solve the problem with just 5% of the memory footprint. We attribute part of this success to the improvements made to LSS for DTNs in this paper. Last but not least, the analysis on realistic satellite networks showed that despite the good performance of LSS, its applicability to case-specific topologies could enjoy further refinement. Such work is indeed needed seeing that RUCoP already stressed the computational resources for 3-h contact plans.

Even though LSS and RUCoP stand on the frontier of the state-of-the-art of routing in uncertain DTNs, a few challenges remain to be tackled. On the one hand, both approaches assume non-congested links: routing in uncertain *and* congested DTNs is an open research topic. Also the integration of uncertain and Opportunistic CGR [8] is appealing future work. Finally, the evaluation of the routing schedules obtained from the presented use cases in realistic DTN protocol simulations is currently being investigated by the authors.

Data Availability. A dataset with the models and tools needed to replicate our experimental evaluation is archived and available at *DOI* 10.4121/20334687 [14].

References

1. Agha, G., Palmskog, K.: A survey of statistical model checking. ACM Trans. Model. Comput. Simul. **28**(1), 6:1-6:39 (2018). https://doi.org/10.1145/3158668
2. Araniti, G., et al.: Contact graph routing in DTN space networks: overview, enhancements and performance. IEEE Comms. Mag. **53**(3), 38–46 (2015). https://doi.org/10.1109/MCOM.2015.7060480
3. Baier, C., de Alfaro, L., Forejt, V., Kwiatkowska, M.: Model checking probabilistic systems. In: Handbook of Model Checking, pp. 963–999. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-10575-8_28
4. Baier, C., Katoen, J.: Principles of Model Checking. MIT Press, Cambridge (2008)
5. Benamar, N., Singh, K.D., Benamar, M., Ouadghiri, D.E., Bonnin, J.M.: Routing protocols in vehicular delay tolerant networks: a comprehensive survey. Comput. Commun. **48**, 141–158 (2014). https://doi.org/10.1016/j.comcom.2014.03.024
6. Benhamida, F.Z., Bouabdellah, A., Challal, Y.: Using delay tolerant network for the Internet of Things: Opportunities and challenges. In: 2017 8th International Conference on Information and Communication Systems (ICICS), pp. 252–257, April 2017. https://doi.org/10.1109/IACS.2017.7921980
7. Budde, C.E., D'Argenio, P.R., Hartmanns, A., Sedwards, S.: An efficient statistical model checker for nondeterminism and rare events. Int. J. Softw. Tools Technol. Transf. **22**(6), 759–780 (2020). https://doi.org/10.1007/s10009-020-00563-2
8. Burleigh, S., Caini, C., Messina, J., Rodolfi, M.: Toward a unified routing framework for DTN. In: 2016 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), pp. 82–86, Sept 2016
9. Burleigh, S., et al.: Delay-tolerant networking: an approach to interplanetary internet. Comm. Mag. **41**(6), 128–136 (2003). https://doi.org/10.1109/MCOM.2003.1204759
10. Caini, C., Cruickshank, H., Farrell, S., Marchese, M.: Delay- and disruption-tolerant networking (DTN): an alternative solution for future satellite networking applications. Proc. IEEE **99**(11), 1980–1997 (2011). https://doi.org/10.1109/JPROC.2011.2158378
11. Cerf, V., et al.: Delay-tolerant networking architecture. RFC 4838, RFC Editor, April 2007. http://www.rfc-editor.org/rfc/rfc4838.txt
12. Cheung, L., Lynch, N.A., Segala, R., Vaandrager, F.W.: Switched PIOA: parallel composition via distributed scheduling. Theor. Comput. Sci. **365**(1–2), 83–108 (2006). https://doi.org/10.1016/j.tcs.2006.07.033
13. Consultative Committee for Space Data Systems (CCSDS): CCSDS bundle protocol specification (blue book, recommended standard CCSDS 734.2-B-1), September 2015. https://public.ccsds.org/Pubs/734x2b1.pdf
14. D'Argenio, P.R., Fraire, J.A., Hartmanns, A., Raverta, F.: Comparing statistical and analytical routing approaches for delay-tolerant networks (artifact). 4TU.ResearchData (2022). https://doi.org/10.4121/20334687
15. D'Argenio, P., Legay, A., Sedwards, S., Traonouez, L.-M.: Smart sampling for lightweight verification of Markov decision processes. Int. J. Softw. Tools Technol. Transf. **17**(4), 469–484 (2015). https://doi.org/10.1007/s10009-015-0383-0
16. Dehnert, C., Junges, S., Katoen, JP., Volk, M.: A STORM is coming: a modern probabilistic model checker. In: Majumdar, R., Kunčak, V. (eds) CAV 2017. LNCS, vol. 10427, pp. 592–600. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63390-9_31