Comparing Statistical, Analytical, and Learning-Based Routing Approaches for Delay-Tolerant Networks

PEDRO R. D'ARGENIO, Universidad Nacional de Córdoba, Argentina and CONICET, Argentina
JUAN A. FRAIRE, Inria, INSA Lyon, Université de Lyon, France, CONICET, Argentina, and Universidad Nacional de Córdoba, Argentina
ARND HARTMANNS, University of Twente, The Netherlands
FERNANDO RAVERTA, Universidad Nacional de Córdoba, Argentina and CONICET, Argentina

In delay-tolerant networks (DTNs) with uncertain contact plans, the communication episodes and their reliabilities are known a priori. To maximise 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. In this paper, we provide an in-depth comparison of three solution approaches: statistical model checking with scheduler sampling, the analytical RUCoP algorithm based on probabilistic model checking, and an implementation of concurrent Q-learning. 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 all three approaches are suitable tools for obtaining reliable routes in DTN, and expose a trade-off between scalability and solution quality.

ACM Reference Format:
Pedro R. D'Argenio, Juan A. Fraire, Arnd Hartmanns, and Fernando Raverta. 2024. Comparing Statistical, Analytical, and Learning-Based Routing Approaches for Delay-Tolerant Networks . *ACM Trans. Model. Comput. Simul.* 1, 1, Article 1 (January 2024), 26 pages. https://doi.org/10.1145/3665927

1 INTRODUCTION

Delay-tolerant networks (DTNs) are time-evolving networks lacking continuous and instantaneous end-to-end connectivity [18, 29]. The DTN domain has attracted the community's interest since its introduction in 2003 [14]. Most existing deep space initiatives like NASA's Lunanet, ESA's Moonlight, and IOAG's Future Mars Communications Architecture make DTN protocols a core requirement for their networking architecture [38, 39, 55]. Beyond the interest in networking at deep-space distances [1, 28], the DTN domain currently motivates research in satellite-aerial 6G integrated networks [73, 74], vehicular networks comprising unmanned aerial vehicles (UAV) [27], maritime [86], and terrestrial vehicles (a.k.a., VDTN) [52, 63, 78, 80], social networks [82, 84], and disaster recovery deployments [87]. While some of these contributions involve security [63, 79] and energy optimization [86], the core research challenge rests on time-evolving multi-hop routing [26, 27, 32, 59, 80, 84, 87]. DTN is notable for its *overlay layer* approach, adeptly handling data with

Authors are ordered alphabetically.
Authors' addresses: Pedro R. D'Argenio, Universidad Nacional de Córdoba, Córdoba, Argentina and CONICET, Córdoba, Argentina, pedro.dargenio@unc.edu.ar; Juan A. Fraire, Inria, INSA Lyon, Université de Lyon, Lyon, France and CONICET, Córdoba, Argentina and Universidad Nacional de Córdoba, Córdoba, Argentina, juanfraire@unc.edu.ar; Arnd Hartmanns, University of Twente, Enschede, The Netherlands, a.hartmanns@utwente.nl; Fernando Raverta, Universidad Nacional de Córdoba, Córdoba, Argentina and CONICET, Córdoba, Argentina, fernando.raverta@unc.edu.ar.

1049-3301/2024/1-ART1
https://doi.org/10.1145/3665927

temporal storage, as outlined in RFC 4838 [18] and its latest protocol update in 2022 [77]. The overlay 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 neighbouring nodes [68]. As a result, *bundles* of data (a data unit in the Bundle Protocol [18, 76])—and status information about the rest of the network—flow in a 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* [34]; in *probabilistic* DTNs, the contact patterns can be dynamically inferred; no assumptions on future contacts can be made in *opportunistic* DTNs [18]. Recent work extended this classification to also consider *uncertain* DTNs, in which forthcoming connectivity can be described by probabilistic schedules available *a priori* [22, 36, 61, 62, 71, 72]. 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 [36], public vehicle networks with uncertain mobility patterns [57], interference-sensitive communication links in cognitive radio [75], or networks based on third-party carriers with limited but well-known availability [53].

This work summarises and compares routing solutions for uncertain DTNs. The state-of-the-art techniques are *lightweight scheduler sampling* (LSS) [22] and *routing under uncertain contact plans* (RUCoP) [71]. Both leverage Markov decision processes (MDPs), allow a bounded network-wide number of message copies to maximise 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 [2] whereas RUCoP is based on an analytical solution that exhaustively explores the MDP akin to probabilistic model checking [4, 5]. Given the success of artificial intelligence/machine learning methods, in particular of *reinforcement learning* [81], in complex decision-making problems, we additionally compare LSS and RUCoP with a new adaptation of Q-learning [83] to the problem of finding routing strategies in uncertain DTNs. Like LSS, Q-learning (QL) gathers data about the MDP via simulation, but like RUCoP it stores information about the values of the available choices in each explored state. Q-learning however may not need to explore all states, thus possibly residing between the two other methods in terms of memory usage. All three methods 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, RUCoP, and QL 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 three 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, and our adaptation of QL to the problem of DTN nodes having only local knowledge (following the *concurrent Q-learning* approach).

Structure. This paper is organised as follows: Sect. 2 of this paper revises the background of DTNs, MDP, and modelling the routing problem. We dive into the details of the LSS, RUCoP, and QL techniques in Sect. 3, including a summary of our improvements to LSS for DTNs and an exposition

of how we implement concurrent Q-learning to tackle the local-knowledge setting. We present, apply and analyse the benchmark framework in Sect. 4.

Prior versions. This paper is an extended version of our conference paper presented at QEST 2022 [24]. The conference version compared LSS and RUCoP; we newly implemented Q-learning as a third method and expanded our experiments to compare all three methods in a consistent way. We have also expanded Sect. 2, particularly some explanations on DTN, and the description of the algorithms in Sect. 3 (with the description of Q-learning newly added).

2 BACKGROUND

In this section we describe the concept and context of DTNs, introduce the modelling formalism of Markov decision processes (MDP), and explain how to encode DTN routing with global and local information as MDP.

2.1 Delay-Tolerant Networks

The term "DTN" was introduced in the context of interplanetary communication to designate time-evolving networks lacking continuous and instantaneous end-to-end connectivity [14]. The concepts and mechanisms devised to deal with the delays and disruptions of interplanetary communications can readily be applied to other domains characterised by long signal propagation time, frequent node occlusion, high node mobility, and/or reduced communication range and resources [32] such as airborne, vehicular, social, IoT, underwater and space networks [6, 7, 16, 33, 43, 51, 67]. DTN protocols like the Bundle Protocol [21, 76] 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 [18]. The Bundle protocol is currently standardized by the Internet Engineering Task Force (IETF) and the Consultative Committee for Space Data Systems (CCSDS), bridging the gap between space and terrestrial networks. Also, several Bundle Protocol software stacks have been developed [68]. Interplanetary Overlay Network (ION) [12] and Micro-Planetary Communication Network (μPCN) [30] are two implementations that are specifically targeted for the space environment.

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. During a contact, data is transmitted by DTN node A (the contact's sending node) at data rate R such that data will be received by node B (the contact's receiving node). The time values can be expressed either in absolute units (e.g., Gregorian Coordinated Universal Time, UTCG) or in relative time with respect to a reference epoch. Contacts can be classified [18] 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*). Contacts are, by definition, unidirectional (unidirectional transmission is not uncommon in space communications). Bidirectional communication is represented in a contact plan by a pair of unidirectional contacts.

A contact plan comprises the set of forthcoming contacts, and is a central element in scheduled DTN routing. As shown in Fig. 1, 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 centralised (off-line) or decentralised (on-line) fashion [35]), and *forwarding* (effectively enqueuing the data for the correct next-hop node). Contact graph routing (CGR) [3, 32] is the de-facto standard decentralised routing algorithm when a contact plan is available. It is the sole routing approach that has been flight-validated in deep-space [85] and near-Earth networked missions [56]. CGR optimises delivery *time* by leveraging adaptations of Dijkstra's algorithm to the time dynamics of DTNs. CGR has received increasing attention in recent years [32],

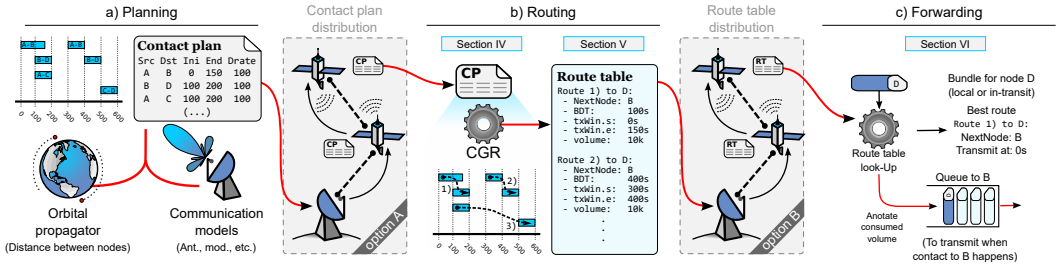


Fig. 1. CGR routing framework in scheduled space DTNs. a) Orbital propagators and communication models allow the computation of a contact plan, which may be kept at a central control center or distributed to DTN nodes (option A). b) Routing uses CGR to compute a route table from the contact plan, which may be distributed to DTN nodes if computed in a central control center (option B). c) DTN nodes consult the route table to decide upon the proper outbound queue for forwarding outbound data. [32]

including source routing extensions [54], routing loop prevention [10], adaptations for Low-Earth Orbit (LEO) satellites [17], overbooking management [9], opportunistic enhancements [13], and route table management strategies [37].

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 [72]. Reasons include well-known failure modes of the DTN nodes, source of interference that could not be foreseen in advance (especially when operating over unlicensed frequency bands), or incomplete/inaccurate knowledge of the system status by the time the schedule was computed. Uncertain contact plans gave rise to a new type of DTNs coined *uncertain DTNs* [22, 61, 62, 71, 72] that exploit time-dependent probabilistic information of the forthcoming communication opportunities. Note that previous works, such as opportunistic contact plan extensions sought in [13], tackled the problem of reacting to a new, unexpected contact, which was not included in the original contact plan. Instead, an uncertain contact plan is a data structure that comprises the complete contact forecast but labels each contact with a probability of occurring or failing.

Forwarding in an uncertain DTN is also different than scheduled DTNs. 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). Multi-copy forwarding is one of the core strategies of probabilistic DTNs. Indeed, uncertain DTNs are in-between scheduled DTNs (there is a contact plan with topology forecast available *a priori*) and probabilistic DTNs (contacts in the contact plan have a probability of occurring, also known *a priori*).

2.2 Markov Decision Processes

Markov decision processes (MDPs) provide a mathematical framework capturing the interaction between non-deterministic and probabilistic choices [31, 69], making them appropriate for modelling decision making under probabilistically quantified uncertainty.

Definition 2.1. 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 $\alpha \in Act$.

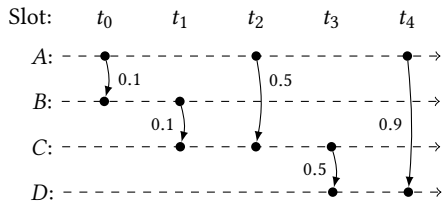


Fig. 2. Uncertain contact plan.

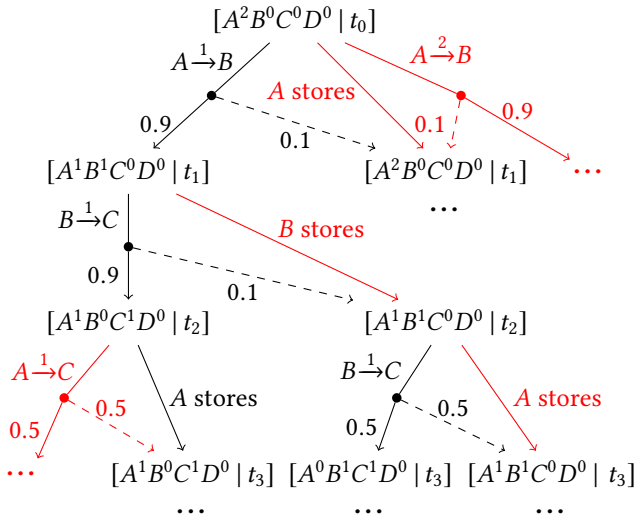


Fig. 3. MDP modelling the plan of Fig. 2 (excerpt).

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. We also write $Act(s)$ for the set of actions enabled in s.

A *reachability problem* is characterised as follows: given a set of *goal states* $B \subseteq S$, maximise 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 [5].

2.3 Encoding Contact Plans

Consider the example uncertain contact plan with nodes A, B, C, and D in Fig. 2. 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).

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 $[A^2\,B^0\,C^0\,D^0 \mid t_0]$ in Fig. 3. 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 $[A^2\,B^0\,C^0\,D^0 \mid t_0]$, (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 $[A^1\,B^1\,C^0\,D^0 \mid t_1]$ where A has kept one copy and the other has reached B. Since success probability is 0.9, we have

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

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

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

Action $A \xrightarrow{1} B$ is the black transition out of $[A^2\,B^0\,C^0\,D^0 \mid t_0]$ in Fig. 3 where the solid line represents the successful transmission while the dotted arrow represents the failing event. The situation is 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

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

The construction is similar for the rest of the MDP. Fig. 3 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, which is possible and typical in LEO constellations. We refer to this assumption as *acknowledged communication* (a.k.a. custody transfer in the Bundle Protocol [32]). The alternative is *fully unreliable* communication, where transmitted copies are lost upon failure, which is natural in deep-space communication.

The assumption of 'acknowledged communication' plays a pivotal role in maintaining the integrity of the network by ensuring the consistency of the number of message copies. It is particularly critical for RUCoP's operational model, as it relies on a fixed number of copies for accurate state representation within its MDP framework. While this principle is equally pertinent for the LSS and QL methods, which also operate on the same MDP structure, the sampling-based nature of these approaches may obscure its impact in practical scenarios. We maintain this hypothesis throughout our evaluation to ensure model consistency and clear comparative analysis. However, we acknowledge this assumption's potential limitations and propose exploring its relaxation and the subsequent practical implications as an avenue for future research.

2.4 Global and Local Information

For the MDP described above, the maximizing scheduler for goal set $\mathcal{G} = \{\ [A^a\,B^b\,C^c\,D^d \mid t_5] \mid d \geq 1 \wedge (a+b+c+d) \leq 2\ \}$ describes the optimal routing decisions and is represented by the black choices in Fig. 3. 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. 2. 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. 3: 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 $[A^1\, B^1\, C^0\, D^0 \mid t_2]$ or $[A^1\, B^0\, C^1\, D^0 \mid t_2]$. 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* [19, 40, 41].

3 ROUTING IN UNCERTAIN DTNS

The optimal *global* scheduler can be computed using any probabilistic model checker such as Prism [58], Storm [50], or mcsta of the Modest Toolset [46]: 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 [47] or by using an iterative algorithm such as a sound variant of value iteration [44, 48, 70], 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 [71]). Furthermore, none of them provides a solution for the local-information problem. We now summarise the two pre-existing MDP-based approaches for optimal DTN routing under uncertain contact plans, RUCoP and LSS, and our adaptation of Q-learning.

All three 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 centralised node. While we have a model of the entire MDP at hand, LSS and Q-learning are in fact *model-free* algorithms: They only need to be able to observe states, obtain the set of actions enabled in a state, and *sample* a successor state for the selected action. RUCoP, in contrast, requires the full information about the actions' probability distributions. LSS and Q-learning thus do not make use of all the information that is available in our model-based approach.

3.1 RUCoP

RUCoP [72] (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 (that is, the goal $\mathcal{G}$ defined above). It then walks backwards in the contact plan, selecting only the maximizing transitions according to the Bellman equations.

More precisely, RUCoP first determines the goal states in the last time slot and then iterates through the time slots starting from the last but one. In turns, each iteration walks through every state s explored in the previous time slot (say t_{i+1}) and proceeds in two phases. The first phase builds all possible transitions that reach s without failing in any possible contact (some storing could also take place). The second phase iterates through all these transitions determining the predecessor state (which is then included in the set of explored states at time slot t_i), finding the best decision explored so far at this state (i.e. the candidate optimizing transition) and calculating the probability at the state to eventually deliver the message successfully using that decision. Thus, when it terminates, RUCoP delivers the set of all states that reach a goal state (i.e., a message arrives to target) with positive probability, the probability of doing so from each state, and the best decision in each state. The latter one can be turned into the routing tables. For a comprehensive and detailed exploration of the RUCoP algorithm, readers are encouraged to consult our previous work [72].

This reference offers an in-depth explanation and full listing of the algorithm's steps, providing valuable insights for those seeking a deeper understanding of its mechanisms and applications.

In its general form, RUCoP 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. 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 in general. However, it is based on a global view of the system which, as mentioned in Sect. 2.4, let nodes make routing decisions based on information not available locally. 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 $[A^1\, B^1\, C^0\, D^0 \mid t_2]$ and $[A^1\, B^0\, C^1\, D^0 \mid t_2]$ will be both taken from $T(A, 1, t_2)$ which in turn is obtained from the decision in state $[A^1\, B^0\, C^0\, D^0 \mid t_2]$ 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 [72].

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 [2]) 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 parallelise and distribute on multi-core systems and compute clusters.

Sampling schedulers. Lightweight scheduler sampling [60] (LSS) extends SMC to MDP, keeping its constant memory usage: Given an MDP M, it

(1) randomly selects a set Σ of m schedulers, each identified by a fixed-size integer (e.g. of 32 bits as in our implementation),
(2) 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

(3) performs a standard SMC analysis on $M|_{\sigma_{max}}$ to provide an estimate $\hat{p}_{\sigma_{max}}$ for $Pr^{\max}_{s_0}(reach(B))$.

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 $Pr^{\max}_{s_0}(reach(B))$ 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 a simplified variant of the *smart sampling* [25] approach to select σ_{max} in step 2: 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.

Constant-memory schedulers. 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 (see Sect. 3.3).

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.

3.2.1 Local information. As described above, LSS produces global-information schedulers. However, it can be adapted to sample from local-information schedulers only [22]: 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 information locally available to node N: the number of locally-stored copies and the current time slot. We refer to LSS with local-information schedulers as L-LSS, whereas we denote by LSS the original global-information technique.

To avoid conflicts where two nodes need to make a decision at the same time, assume that the high-level modelling of the MDP as a system of multiple independently executing nodes is *good for distributed scheduling* [22]. In words, an MDP is good if, in all states where there is a nontrivial choice between actions from multiple nodes,

(1) this choice cannot influence whether we directly move to a goal state,
(2) no node has a local choice involving at least one transition in which another node is involved, and
(3) no transition can change variables that are visible to a node not involved in the transition.

The MDPs for DTN routing that we create from contact plans are good for distributed scheduling by construction.

Implementing L-LSS schedulers. 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.

3.2.2 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 [22]. 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. 2.3 in the MODEST modelling language [45], and an implementation of LSS and L-LSS in the MODES simulator/statistical model checker [11] 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. By analysing the graph of contacts over time, we precompute the predicates rc_{src} and rc_{dest} that, given a node X and a time slot, return *true* iff an earlier sequence of contacts exists from the source to X or a later one from X to the target, respectively. We then include only those contacts for X at time slot t in the model for which $rc_{src}(X,t) \wedge rc_{dest}(X,t)$. 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 [22, 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.

3.3 Q-Learning

Reinforcement learning [81] is a machine learning approach to train agents to take actions maximising a reward in uncertain environments. Mathematically, the agent in its environment can be described as an MDP: the agent chooses actions; the environment determines the states and is responsible for the probabilistic outcomes of the actions. One specific and popular reinforcement learning method is *Q-learning* [83]. It maintains a Q-function

$$Q\colon S \times Act \to [0,1]$$

stored in explicit form (as a so-called *Q-table*) initialised to 0 everywhere. Using a learning rate parameter α, a discounting factor $\gamma \in (0,1]$, and a probability ϵ, k learning *episodes* are performed as follows, starting from s being the MDP's initial state s_0:

(1) Perform option (a) with probability ϵ and (b) with probability $1 - \epsilon$:
  (a) Select a from $Act(s)$ uniformly at random (*exploration*).
  (b) Select $a := \arg\max_{a' \in Act(s)} Q(s, a')$ (*exploitation*).
(2) Select $s' \in \{ s'' \in S \mid \mathbf{P}(s, a, s'') > 0 \}$ at random according to the probabilities given by $\mathbf{P}$.
(3) Set $r := 1$ if $s' \in B$ and 0 otherwise.
(4) Update $Q(s, a) := Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a' \in Act(s')} Q(s', a') - Q(s, a))$.
(5) If $s' \in B$ or s' has probability 0 of reaching a goal state, end the episode; else set $s := s'$ and go to step 1.

Typically, α and ϵ decrease over time from the first to the last episode. A higher ϵ allows the algorithm to explore various actions, avoiding premature convergence to sub-optimal solutions. As the algorithm learns more about the environment, it becomes beneficial to gradually reduce ϵ, leading to a focus on exploiting the best-known strategies. Similarly, α determines the impact of new information on the existing knowledge. A high initial α allows for rapid learning but can destabilize due to noisy data or outliers. Reducing α over time helps stabilize the learning process as the algorithm converges, integrating new information more conservatively [83]. As an optimisation, we skip the Q-table updates for states with only one enabled action.

An episode is very similar to a simulation run, and we again assume that we can effectively perform the above operations using a high-level description of the MDP so that we do not need to store its entire state space. The main differences to simulation as used for SMC and LSS are that we update the Q-function to estimate the "quality" of taking action a from state s as $Q(s, a)$ and follow an "ϵ-greedy" strategy: initially, when ϵ is high, we mostly explore randomly; over time, we make it more and more likely to follow what looks like the best action to improve our estimate of its quality. RL traditionally optimises for expected discounted rewards, thus the discounting factor γ; for reachability problems as we consider them in this paper, we set γ to 1 and only obtain a reward upon reaching a goal state (as above). As long as (1) we are guaranteed to visit every state infinitely often (i.e. if ϵ never becomes 0), (2) parameters α and ϵ decrease over time from the first to the last episode, and (3) these series of parameters fulfill some variant of the stochastic approximation conditions, $\max_{a \in Act(s_0)} Q(s_0, a)$ converges towards $Pr^{\max}_{s_0}(reach(B))$ [81, 83].

Memory and runtime. Q-learning is similar to LSS in that it uses simulation runs; consequently, both are so-called model-free techniques. Its memory usage, however, is in $O(|S| \cdot |Act|)$ (for the Q-table) and thus more similar to probabilistic model checking and RUCoP. For many models, however, Q-learning only explores—and thus stores a Q-value for—a subset of S. This happens some parts of the state space have a very low probability of being reached from s_0 within the specified number of episodes. Additionally, no Q-values need to be stored for states where $|Act(s)| = 1$.

In terms of runtime, RUCoP is, on an abstract level, incomparable to LSS and Q-learning: RUCoP's runtime depends on the number of relevant states of the MDP and the number of message copies, whereas the time spent in LSS and Q-learning depends on the number of simulation runs performed. For LSS using our smart sampling approach, we need $O(m \cdot \log m)$ runs (where m is the number of schedulers to sample), while Q-learning needs $O(k)$ runs (where k is the number of episodes to learn from). Each run (episode) in Q-learning is however slightly more computationally expensive than in LSS due to the computations involving the Q-table needed for ϵ-greedy scheduling and the learning in step 4.

3.3.1 Local information. Q-learning as described above ranges over global-information schedulers, training a single global agent to make the choices. In the local-information setting of DTN routing, we instead have a multi-agent reinforcement learning (MARL) [15] problem. In particular, our

agents—the nodes of the DTN—act cooperatively, attempting to achieve the common goal of end-to-end message delivery, with partial information. Where we use MDP with distributed scheduling, the conceptually corresponding model typically used to capture the same scenario in machine learning is that of decentralised partially observable MDP (Dec-POMDP) [65]. Like distributed scheduling, finding optimal schedulers in Dec-POMDP is an intractable problem, already being NEXP-complete in the finite-horizon case [8].

One straightforward adaptation of Q-learning to the multi-agent local-information setting is the *concurrent learning* approach [66]. Here, each agent (in our case: DTN node) learns on its own, keeping and updating its own Q-function and only observing that part of the current state that contains the agent's local information. As a result, however, each learners' environment now includes the other learners, making the environment non-stationary—which violates a main assumption of the Q-learning algorithm: As one agent learns, it modifies its choices, which can ruin the quality of other agents' learned choices (and thus invalidate the information stored in their Q-tables) by invalidating the observations on which they are based. Consequently, convergence and optimality are no longer guaranteed [20]. We nevertheless follow the concurrent learning approach for our experiments with DTNs in this paper, due to its simplicity.

An MDP resulting from the parallel composition of multiple components/agents/DTN nodes gives rise to a product state space as shown in Fig. 3, where every state is a combination of the local states of all nodes. This results in an explosion of the state space as the number of nodes grows. With concurrent Q-learning, however, each node's learner only sees the local component state space. Thus, instead of being exponential in the number of nodes as in global Q-learning, the number of Q-values stored in concurrent Q-learning grows only linearly with the number of components. This makes concurrent Q-learning particularly interesting to compare with L-RUCoP, which explores the entire (backwards reachable) product state space of exponential size, and LSS, which has constant memory usage.

The way we implement concurrent Q-learning is by running one copy of the Q-learning algorithm described at the beginning of Sect. 3.3 per node N with the following modifications:

(1) In all computations involving the Q-table, instead of working with global states s, s', etc., we work with their projections $s|_N$, $s'|_N$, etc.
(2) For step 1, we first need to choose at a global level one of the nodes to make the scheduling choice. Due to the MDP being good for distributed scheduling, at most one node will have a nontrivial choice in each state; that is the one node we let perform the Q-learning steps for this state.
(3) When we reach a goal state, we let all nodes perform the Q-learning steps for this last step with reward 1 (i.e. we do not make any attempt at a distributed reward assignment).

In the remainder of this paper, we write QL for Q-learning with global information, and L-QL for our variant of concurrent Q-learning.

3.3.2 Our implementation. We have newly implemented Q-learning for MDP with global and local information as part of the MODEST tool for this paper, significantly extending an earlier prototype implementation for stochastic hybrid systems [64]. In contrast to a pure Q-learning implementation to approximate $Pr^{\max}_{s_0}(reach(B))$ that would return $\max_{a \in Acts_0} Q(s_0, a)$ after k learning episodes have been performed, MODEST instead follows up on the learning with a standard SMC analysis on $M|_{\sigma_Q}$ to provide an estimate $\hat{p}_{\sigma_Q}$ for $Pr^{\max}_{s_0}(reach(B))$ (like in step 3 of the LSS approach). Here, σ_Q is the scheduler that maps each state s to action choice $a = \arg\max_{a' \in Act(s)} Q(s, a')$. We found that the Q-values tend to identify a good scheduler long before they converge to a reasonable approximation of $Pr^{\max}_{s_0}(reach(B))$; by separating the scheduler-finding and scheduler-evaluation tasks in this way, we tend to get better final results with fewer episodes. This is in fact similar to the *deep statistical*

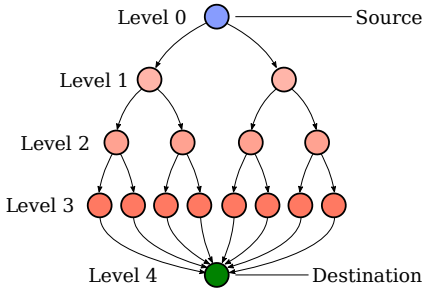


Fig. 4. Binomial tree.

model checking technique [42], which however uses deep neural networks to approximate the Q-function in place of our explicit table-based representation, thereby gaining scalability but losing monotonicity and the convergence guarantee that we have in the global-information case and not necessarily performing better than LSS [49].

4 EVALUATION

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

4.1 Benchmark Set

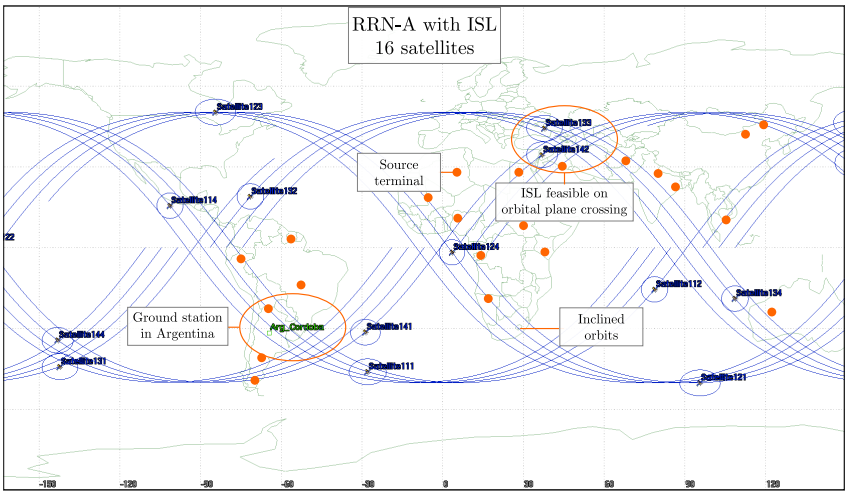
The proposed benchmark comprises random, binomial, and ring-road network configurations, which we describe below. While the random and binomial networks are artificial examples to study the behaviour of the three methods, the ring-road network is a realistic case study.

4.1.1 Random networks. This is the first and simplest model of the benchmark. It uses a uniform distribution of contacts among a configurable number of network nodes and contact plan duration. The primary objective of this benchmark suite is to facilitate an evaluation that is broad-based and academically oriented, focusing on general principles rather than being tethered to specific network deployment scenarios. We use 10 random topologies with 8 nodes, each covering a duration of 100 seconds. Time is discretised into episodes of 10 seconds. 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 [62]. 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.

4.1.2 Binomial networks. We have devised a family of contact plans with a binomial topology to gain insights into how increasing the topological complexity affects the different routing algorithms. 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:

(1) $1 + 2^{L-2}$ nodes have contacts with two neighbors;
(2) $\sum_{i=1}^{i<L-2} 2^i$ nodes have contacts with three neighbors; and
(3) 1 final destination node has 2^{L-2} contacts.

The resulting tree is illustrated in Fig. 4. Contacts between consecutive levels are also consecutive in the time dimension, that is, the order of the contacts corresponds to enumerating the arrows in Fig. 4 left-to-right, top-to-bottom. A node on the i-th level will have a total of 2^{L-2-i} paths to



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. 5. RRN satellite constellation topology, parameters and orbital tracks [36].

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.

4.1.3 Ring road networks. Finally, for our evaluation targeting an industry-focused use case, we employed a realistic satellite topology derived from high-precision orbital propagators. This topology provides a practical and authentic context, enabling us to assess the applicability and effectiveness of our solutions in real-world industrial settings. Specifically, we consider a low-Earth orbit Walker constellation of 16 satellites as proposed and described in [36]. 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. 5. Routes can involve multiple hops between satellites and ground terminals. The scenario spans 24 hours 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 versions (L-LSS and L-RUCoP). We evaluate them in terms of the SDP of the computed scheduler, and the computational resources

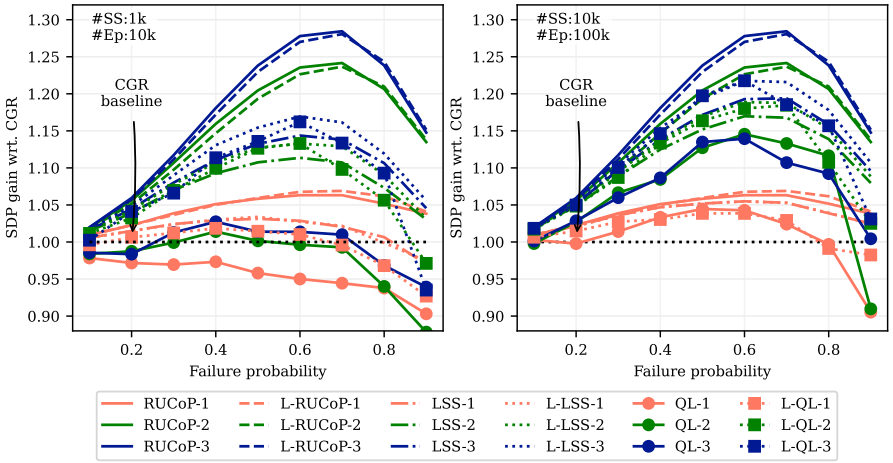


Fig. 6. SDP gain over CGR in random networks.

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 have 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.

4.2.1 Random networks. The SDPs we obtained for random networks are illustrated in Fig. 6. 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. QL-based algorithms show a more varied performance which depend on the number of learning episodes and if the algorithm is based on global or local information.

We ran QL and L-QL in two configurations, one training using $k = 10000$ learning episodes and the other using $k = 100000$ episodes. This is indicated with "`#Ep`" in our figures. We can see in Fig. 6 that, for this case study, the increase on learning episodes yields an improvement on the SDP which could be up to 14% in the global case, though generally milder for the L-QL variants. Also, notice that the L-QL techniques perform relatively well in general, while the global QL cases mostly underperformed CGR when the number of episodes was low. Both phenomena are explained by the combination of the size of the Q-table and the exploration budget. The Q-table for QL grows exponentially with the number of nodes in the network, while it grows linearly for L-QL. Therefore, a very large number of episodes would be required to explore each position of the Q-table sufficiently many times in QL. Instead, being the Q-table exponentially smaller, the likelihood of visiting each of its positions in L-QL with the same budget is much higher, hence, the learning speed.

We also 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. 6, 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 $\approx 1.8\%$, with $\approx 5.8\%$ 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.

To make a fair comparison we grouped the LSS techniques with 1000 schedulers sampling together with the QL techniques with 10000 learning episodes as they require comparable computational effort. Similarly, we grouped the LSS with 10000 schedulers sampling together with the QL under 100000 learning episodes.

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%. L-QL shows slightly lower results particularly for higher failure probabilities ($p_f \geq 0.7$). We observe that the lower the number of copies, the smaller the difference between L-RUCoP, L-LSS and L-QL. In particular the single-copy case is almost identical in SDP for L-RUCoP and L-LSS. Interestingly, the single-copy case provides limited or no gain with respect to the CGR baseline in these simple topologies. A similar effect was reported for Opportunistic CGR in [13].

Regarding the processing and memory footprint for random networks, all the techniques we study always complete in less than 20 seconds, using less than 20 MB of memory. Also, we observe 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.

4.2.2 Binomial networks. The results obtained for binomial networks are plotted in Fig. 7. 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. 7 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. 7). 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.

With respect to QL, in general, the resultig SDP values are comparable to LSS on levels 4 and 5 and, in general, the local variants with multiple copies (namely, L-QL-2 and L-QL-3) have responded with results closer to L-LSS with the same number of copies (some times slightly better, some

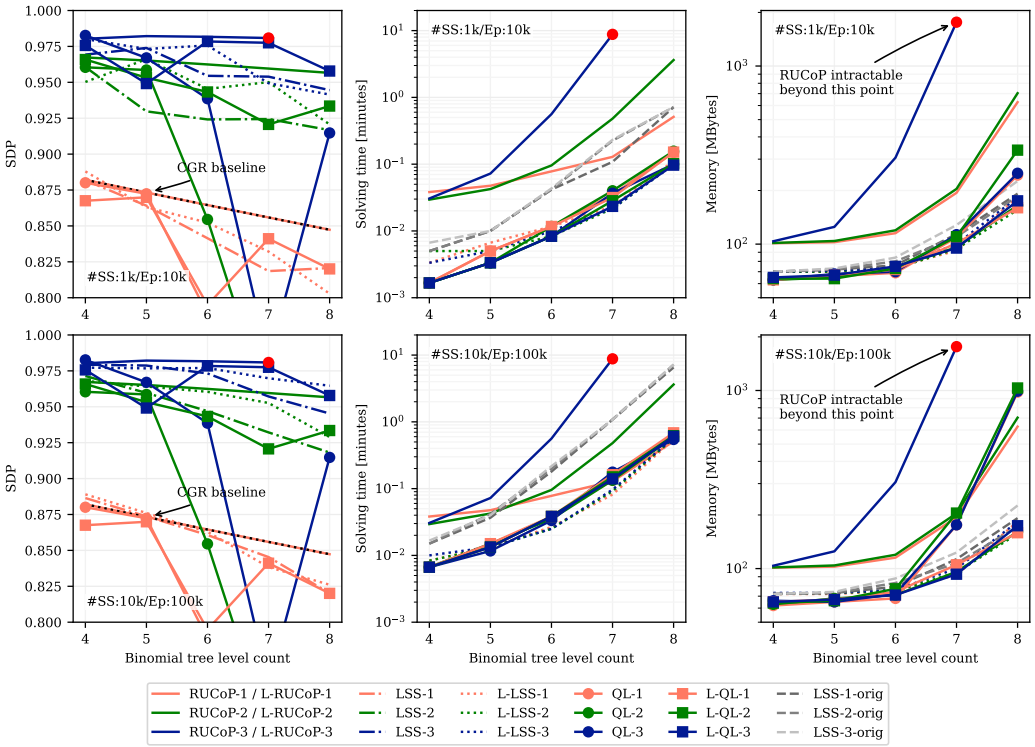


Fig. 7. SDP, solving time, and memory for binomial networks with varying complexity.

times slightly worst) However, for higher levels, in particular for the global case, QL significantly underperformed the other methods in most of the cases, most likely due to the fact that this technique may get trapped on local maxima. Another thing to observe here is that all QL variants reported the same results regardless whether the number of learning episodes is increased from 10000 to 1000000, implying that, in this case, a relatively low number of episodes is sufficient to saturate the learning curve.

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 minutes of computation time, compared to less than 10 seconds for LSS with 1000 schedulers, or 1 minute 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 [22], i.e. without the improvements described in Sect. 3.2.2, are also plotted in Fig. 7, in gray dashed lines. These improvements reduce LSS runtime by up to $\approx 600\%$ (from 117 down to 17 seconds). A reduction of $\approx 6\%$ 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. Also, notice that the different QL methods show similar time performance as the LSS variants. With respect to memory usage, it is mostly also comparable for low number of episodes. However, increasing to 100000, the memory usage significantly increase in some cases for the higher level instances of the binomial network. In any case, we already know there is no gain to increase the numbers of episodes (and hence the memory usage) since the results for this family of models do not change.

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. Instead, QL has more difficulties to deliver consistently near optimal values and good results are more sparse. 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.

4.2.3 Ring road networks. We have evaluated all downlink source-destination pairs in the realistic RRN network. Fig. 8 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 hours (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) and similarly, the #Ep parameter is set to 10000 and 10000 episodes.

The SDP plots in Fig. 8 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. 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 $\approx 60\%$, for failure probabilities of ≈ 0.6, and contact plans of 3 hours. Interestingly, the gap is reduced to $\approx 30\%$ 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 binomial and random topologies, and may need to be adapted to a variant yet more specifically tailored to the DTN routing application. On the contrary QL seems to perform better in this case study than in the previous one. Though still appreciated the effects of local maxima, in which curves for QL do not show smooth curves, in the majority of the cases it reports significantly better SDP that the respective LSS cases, in particular for the local cases. Notably, in most of the cases, L-QL-1 and L-QL-2 report SDP values closer to L-RUCoP-1 and L-RUCoP-2 than to L-LSS-1 and L-LSS-1, respectively.

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. 6, 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.

Fig. 9 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. 8). 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 minutes (for the 3-hour contact plan, with 3 copies), while LSS and QL typically delivers a result in less than 1 minute. 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

[1]Nodes 1 and 7 correspond to nodes 8 and 15 in the contact plan used in [36].

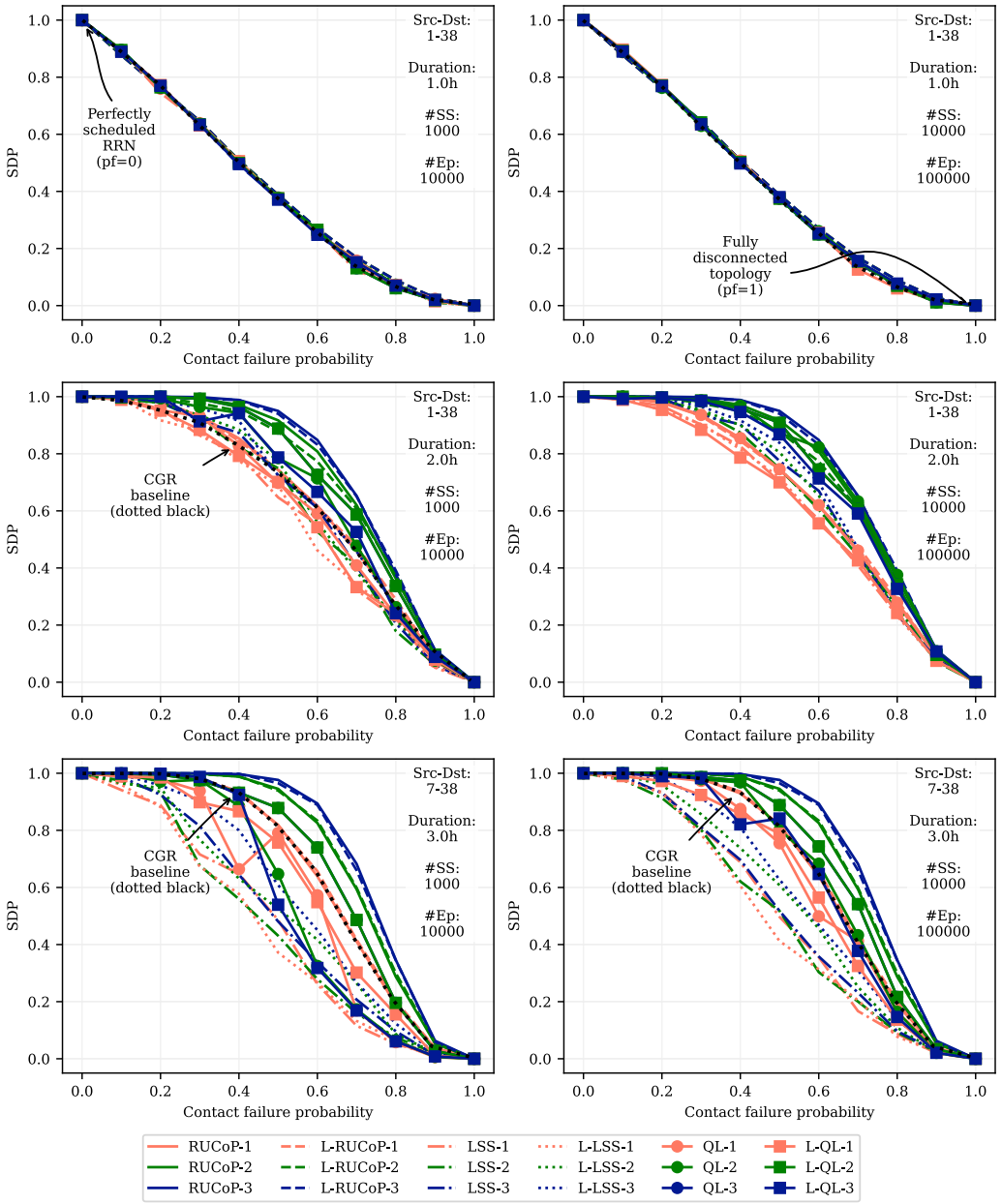


Fig. 8. SDP for RRN for different source-target nodes and plan durations.

nature of LSS, where no decision trees need to be stored as in RUCoP. The memory usage of QL is comparative to LSS in the majority of the cases, using slightly more memory for the more demanding cases. This is expected as it needs to store the Q-table. 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.

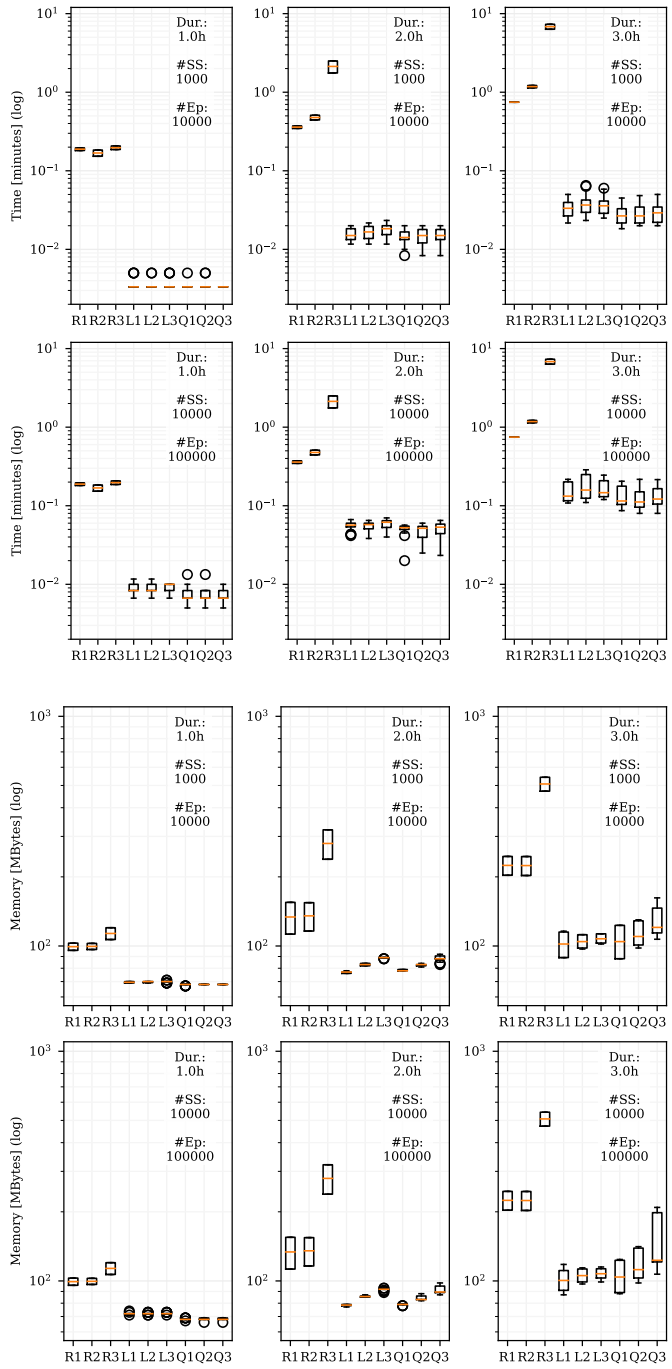


Fig. 9. Solving time (top) and memory (bottom) for RRN for different source-target nodes, contact plan durations, and numbers of schedulers sampled (R: RUCoP, L: LSS). Q: QL

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. However, contrarily to the artificially build binomial networks, QL reported much better SDP values in this real case study, usually outperforming LSS and quite often getting results closer to those of the analytical approach. Nevertheless, the QL approach still remains inconsistent at the time of delivering good results. 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 minutes [34], solving times of 20 minutes, 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, statistical, and learning-based routing approaches for uncertain DTNs. All three methods presented here leverage MDP models. While RUCoP performs an exhaustive and optimal exploration of the solution space, LSS exploits SMC with sampling for optimization and QL presents an alternative to LSS where the search for optimizing schedulers is done through learning rather than sampling. We improved the DTN models for LSS for efficiency and we also introduced here two variants of the Q-learning method which have been newly implemented in the Modest Toolset. We thoroughly compared the three 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, LSS, and QL. On the one hand, all schemes provide routes that deliver up to 1.8 times the data volume achievable by the baseline CGR approach. In general, both variants of RUCoP and LSS and the local variant of QL have consistently delivered good results, normally outperforming CGR. However, we touched the tractability limits of RUCoP in binomial networks of 8 levels. While RUCoP failed to deliver, LSS and QL were 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-hour contact plans. Notably QL, and particularly the local variant, has performed better than LSS for the realistic case study, providing thus a possible alternative to RUCoP. In any case, further studies are needed in this respect.

Even though LSS, RUCoP, and QL 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, all approaches assume non-congested links: routing in uncertain *and* congested DTNs is an open research topic. Also the integration of uncertain and Opportunistic CGR [13] 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 10.5281/zenodo.11214677 [23].

ACKNOWLEDGMENTS

This work was supported by Agencia I+D+i grants PICT-2017-1335, PICT-2017-3894 (RAFTSys), and PICT 2022-09-00580 (CoSMoSS), 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), the Interreg North Sea project STORM_SAFE, NWO VENI grant 639.021.754, and SeCyT-UNC grants 33620180100354CB (ARES) and 33620230100384CB (MECANO).

REFERENCES

[1] Mohammed Y Abdelsadek, Aizaz U Chaudhry, Tasneem Darwish, Eylem Erdogan, Gunes Karabulut-Kurt, Pablo G Madoery, Olfa Ben Yahia, and Halim Yanikomeroglu. 2022. Future space networks: Toward the next giant leap for humankind. *IEEE Transactions on Communications* 71, 2 (2022), 949–1007.

[2] Gul Agha and Karl Palmskog. 2018. A Survey of Statistical Model Checking. *ACM Trans. Model. Comput. Simul.* 28, 1 (2018), 6:1–6:39. https://doi.org/10.1145/3158668

[3] G. Araniti, N. Bezirgiannidis, E. Birrane, I. Bisio, S. Burleigh, C. Caini, M. Feldmann, M. Marchese, J. Segui, and K. Suzuki. 2015. Contact graph routing in DTN space networks: overview, enhancements and performance. *IEEE Comms. Magazine* 53, 3 (March 2015), 38–46. https://doi.org/10.1109/MCOM.2015.7060480

[4] Christel Baier, Luca de Alfaro, Vojtech Forejt, and Marta Kwiatkowska. 2018. Model Checking Probabilistic Systems. In *Handbook of Model Checking*, Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem (Eds.). Springer, 963–999. https://doi.org/10.1007/978-3-319-10575-8_28

[5] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. MIT Press.

[6] Nabil Benamar, Kamal D. Singh, Maria Benamar, Driss El Ouadghiri, and Jean-Marie Bonnin. 2014. Routing protocols in Vehicular Delay Tolerant Networks: A comprehensive survey. *Computer Communications* 48 (2014), 141 – 158. https://doi.org/10.1016/j.comcom.2014.03.024

[7] F. Z. Benhamida, A. Bouabdellah, and Y. Challal. 2017. Using delay tolerant network for the Internet of Things: Opportunities and challenges. In *2017 8th International Conference on Information and Communication Systems (ICICS)*. IEEE, 252–257. https://doi.org/10.1109/IACS.2017.7921980

[8] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The Complexity of Decentralized Control of Markov Decision Processes. *Math. Oper. Res.* 27, 4 (2002), 819–840. https://doi.org/10.1287/moor.27.4.819.297

[9] N. Bezirgiannidis, C. Caini, and V. Tsaoussidis. 2016. Analysis of contact graph routing enhancements for DTN space communications. *Int. Journal of Satellite Coms. and Networking* 34, 5 (2016), 695–709.

[10] Edward Birrane, Scott Burleigh, and Niels Kasch. 2012. Analysis of the contact graph routing algorithm: Bounding interplanetary paths. *Acta Astronautica* 75 (2012), 108 – 119. https://doi.org/10.1016/j.actaastro.2012.02.004

[11] Carlos E. Budde, Pedro R. D'Argenio, Arnd Hartmanns, and Sean Sedwards. 2020. An efficient statistical model checker for nondeterminism and rare events. *Int. J. Softw. Tools Technol. Transf.* 22, 6 (2020), 759–780. https://doi.org/10.1007/s10009-020-00563-2

[12] Scott Burleigh. 2007. Interplanetary Overlay Network: An Implementation of the DTN Bundle Protocol. In *4th IEEE Consumer Communications and Networking Conference, CCNC 2007, Las Vegas, NV, USA, January 11-13, 2007*. IEEE, 222–226. https://doi.org/10.1109/CCNC.2007.51

[13] S. Burleigh, C. Caini, J. Messina, and M. Rodolfi. 2016. Toward a unified routing framework for delay-tolerant networking. In *2016 IEEE Int. Conf. on Wireless for Space and Extreme Environments (WiSEE)*. IEEE, 82–86.

[14] S. Burleigh, A. Hooke, L. Torgerson, K. Fall, V. Cerf, B. Durst, K. Scott, and H. Weiss. 2003. Delay-tolerant Networking: An Approach to Interplanetary Internet. *Comm. Mag.* 41, 6 (June 2003), 128–136. https://doi.org/10.1109/MCOM.2003.1204759

[15] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Trans. Syst. Man Cybern. Part C* 38, 2 (2008), 156–172. https://doi.org/10.1109/TSMCC.2007.913919

[16] Carlo Caini, H. Cruickshank, S. Farrell, and M. Marchese. 2011. Delay- and Disruption-Tolerant Networking (DTN): An Alternative Solution for Future Satellite Networking Applications. *Proc. IEEE* 99, 11 (Nov 2011), 1980–1997. https://doi.org/10.1109/JPROC.2011.2158378

[17] C. Caini and R. Firrincieli. 2012. Application of Contact Graph Routing to LEO satellite DTN communications. In *2012 IEEE International Conference on Communications (ICC)*. 3301–3305. https://doi.org/10.1109/ICC.2012.6363686

[18] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. 2007. *Delay-Tolerant Networking Architecture*. RFC 4838. RFC Editor. http://www.rfc-editor.org/rfc/rfc4838.txt http://www.rfc-editor.org/rfc/rfc4838.txt.

[19] Ling Cheung, Nancy A. Lynch, Roberto Segala, and Frits W. Vaandrager. 2006. Switched PIOA: Parallel composition via distributed scheduling. *Theor. Comput. Sci.* 365, 1-2 (2006), 83–108. https://doi.org/10.1016/j.tcs.2006.07.033

[20] Caroline Claus and Craig Boutilier. 1998. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In *15th National Conference on Artificial Intelligence and 10th Innovative Applications of Artificial Intelligence Conference (AAAI, IAAI)*, Jack Mostow and Chuck Rich (Eds.). AAAI Press / The MIT Press, 746–752.

[21] Consultative Committee for Space Data Systems (CCSDS). 2015. CCSDS Bundle Protocol Specification (Blue Book, Recommended Standard CCSDS 734.2-B-1). https://public.ccsds.org/Pubs/734x2b1.pdf.

[22] Pedro R. D'Argenio, Juan A. Fraire, and Arnd Hartmanns. 2020. Sampling Distributed Schedulers for Resilient Space Communication. In *NASA Formal Methods - 12th International Symposium, NFM 2020, Proceedings (LNCS, Vol. 12229)*, Ritchie Lee, Susmit Jha, and Anastasia Mavridou (Eds.). Springer, 291–310. https://doi.org/10.1007/978-3-030-55754-6_17

[23] Pedro R. D'Argenio, Juan A. Fraire, Arnd Hartmanns, and Fernando Raverta. 2024. Comparing Statistical, Analytical, and Learning-Based Routing Approaches for Delay-Tolerant Networks (Artifact). Zenodo. https://doi.org/10.5281/zenodo.11214677

[24] Pedro R. D'Argenio, Juan A. Fraire, Arnd Hartmanns, and Fernando D. Raverta. 2022. Comparing Statistical and Analytical Routing Approaches for Delay-Tolerant Networks. In *19th International Conference on the Quantitative Evaluation of Systems (QEST) (Lecture Notes in Computer Science, Vol. 13479)*, Erika Ábrahám and Marco Paolieri (Eds.). Springer, 337–355. https://doi.org/10.1007/978-3-031-16336-4_17

[25] Pedro R. D'Argenio, Axel Legay, Sean Sedwards, and Louis-Marie Traonouez. 2015. Smart sampling for lightweight verification of Markov decision processes. *Int. J. Softw. Tools Technol. Transf.* 17, 4 (2015), 469–484. https://doi.org/10.1007/s10009-015-0383-0

[26] Sanjay K Dhurandher, Jagdeep Singh, Petros Nicopolitidis, Raghav Kumar, and Geetanshu Gupta. 2022. A blockchain-based secure routing protocol for opportunistic networks. *Journal of Ambient Intelligence and Humanized Computing* (2022), 1–13.

[27] Zhaoyang Du, Celimuge Wu, Tsutomu Yoshinaga, Xianfu Chen, Xiaoyan Wang, Kok-Lim Alvin Yau, and Yusheng Ji. 2021. A routing protocol for UAV-assisted vehicular delay tolerant networks. *IEEE Open Journal of the Computer Society* 2 (2021), 85–98.

[28] Dalia I Elewaily, Hesham A Ali, Ahmed I Saleh, and Mohamed M Abdelsalam. 2024. Delay/Disruption-Tolerant Networking-based the Integrated Deep-Space Relay Network: State-of-the-Art. *Ad Hoc Networks* 152 (2024), 103307.

[29] Kevin Fall. 2003. A Delay-tolerant Network Architecture for Challenged Internets. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (Karlsruhe, Germany) *(SIGCOMM '03)*. ACM, New York, NY, USA, 27–34. https://doi.org/10.1145/863955.863960

[30] Marius Feldmann and Felix Walter. 2015. μPCN—A bundle protocol implementation for microcontrollers. In *2015 International Conference on Wireless Communications & Signal Processing (WCSP)*. IEEE, 1–5.

[31] Jerzy Filar and Koos Vrieze. 1996. *Competitive Markov Decision Processes*. Springer-Verlag, Berlin, Heidelberg.

[32] Juan A. Fraire, Olivier De Jonckère, and Scott C. Burleigh. 2021. Routing in the Space Internet: A contact graph routing tutorial. *Journal of Network and Computer Applications* 174 (2021), 102884. https://doi.org/10.1016/j.jnca.2020.102884

[33] Juan A. Fraire, Marius Feldmann, and Scott C. Burleigh. 2017. Benefits and challenges of cross-linked ring road satellite networks: A case study. In *IEEE International Conference on Communications, ICC 2017*. IEEE, 1–7. https://doi.org/10.1109/ICC.2017.7996778

[34] J. A. Fraire and J. M. Finochietto. 2015. Design challenges in contact plans for disruption-tolerant satellite networks. *Communications Magazine, IEEE* 53, 5 (May 2015), 163–169. https://doi.org/10.1109/MCOM.2015.7105656

[35] Juan A. Fraire and Elías L. Gasparini. 2021. Centralized and Decentralized Routing Solutions for Present and Future Space Information Networks. *IEEE Netw.* 35, 4 (2021), 110–117. https://doi.org/10.1109/MNET.011.2100102

[36] Juan A. Fraire, Pablo G. Madoery, Scott C. Burleigh, Marius Feldmann, Jorge M. Finochietto, Amir Charif, Nacer-Eddine Zergainoh, and Raoul Velazco. 2017. Assessing Contact Graph Routing Performance and Reliability in Distributed Satellite Constellations. *J. Comput. Networks Commun.* 2017 (2017), 2830542:1–2830542:18. https://doi.org/10.1155/2017/2830542

[37] Juan A Fraire, Pablo G Madoery, Amir Charif, and Jorge M Finochietto. 2018. On route table computation strategies in Delay-Tolerant Satellite Networks. *Ad Hoc Networks* 80 (2018), 31–40.

[38] Pietro Giordano, Antoine Grenier, Paolo Zoccarato, Lorenzo Bucci, Alexander Cropp, Richard Swinden, D Gomez Otero, Wael El-Dali, William Carey, Ludovic Duvet, et al. 2021. Moonlight navigation service-how to land on peaks of eternal light. In *Proceedings of the 72nd International Astronautical Congress (IAC), Dubai, United Arab Emirates*. 25–29.

[39] Pietro Giordano, Richard Swinden, Cheryl Gramling, Juan Crenshaw, and Javier Ventura-Traveset. 2023. LunaNet Position, Navigation, and Timing Services and Signals, Enabling the Future of Lunar Exploration. In *Proceedings of the 36th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2023)*. 3577–3588.

[40] Sergio Giro. 2010. *On the Automatic Verification of Distributed Probabilistic Automata with Partial Information*. Ph. D. Dissertation. Universidad Nacional de Córdoba, Argentina.

[41] Sergio Giro, Pedro R. D'Argenio, and Luis María Ferrer Fioriti. 2014. Distributed probabilistic input/output automata: Expressiveness, (un)decidability and algorithms. *Theor. Comput. Sci.* 538 (2014), 84–102. https://doi.org/10.1016/j.tcs.2013.07.017

[42] Timo P. Gros, Holger Hermanns, Jörg Hoffmann, Michaela Klauck, and Marcel Steinmetz. 2020. Deep Statistical Model Checking. In *40th IFIP WG 6.1 International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE) (Lecture Notes in Computer Science, Vol. 12136)*, Alexey Gotsman and Ana Sokolova (Eds.). Springer, 96–114. https://doi.org/10.1007/978-3-030-50086-3_6

[43] Lav Gupta, Raj Jain, and Gabor Vaszkun. 2015. Survey of important issues in UAV communication networks. *IEEE Communications Surveys & Tutorials* 18, 2 (2015), 1123–1152.

[44] Serge Haddad and Benjamin Monmege. 2018. Interval iteration algorithm for MDPs and IMDPs. *Theor. Comput. Sci.* 735 (2018), 111–131. https://doi.org/10.1016/j.tcs.2016.12.003

[45] Ernst Moritz Hahn, Arnd Hartmanns, Holger Hermanns, and Joost-Pieter Katoen. 2013. A compositional modelling and analysis framework for stochastic hybrid systems. *Formal Methods Syst. Des.* 43, 2 (2013), 191–232. https://doi.org/10.1007/s10703-012-0167-z

[46] Arnd Hartmanns and Holger Hermanns. 2014. The Modest Toolset: An Integrated Environment for Quantitative Modelling and Verification. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8413)*. Springer, 593–598. https://doi.org/10.1007/978-3-642-54862-8_51

[47] Arnd Hartmanns, Sebastian Junges, Tim Quatmann, and Maximilian Weininger. 2023. A Practitioner's Guide to MDP Model Checking Algorithms. *CoRR* abs/2301.10197 (2023). https://doi.org/10.48550/arXiv.2301.10197

[48] Arnd Hartmanns and Benjamin Lucien Kaminski. 2020. Optimistic Value Iteration. In *Computer Aided Verification - 32nd International Conference, CAV 2020, Los Angeles, CA, USA, July 21-24, 2020, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 12225)*. Springer, 488–511. https://doi.org/10.1007/978-3-030-53291-8_26

[49] Arnd Hartmanns and Michaela Klauck. 2022. The Modest State of Learning, Sampling, and Verifying Strategies. In *11th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation. Adaptation and Learning(ISoLA) (Lecture Notes in Computer Science, Vol. 13703)*, Tiziana Margaria and Bernhard Steffen (Eds.). Springer, 406–432. https://doi.org/10.1007/978-3-031-19759-8_25

[50] Christian Hensel, Sebastian Junges, Joost-Pieter Katoen, Tim Quatmann, and Matthias Volk. 2022. The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transf.* 24, 4 (2022), 589–610. https://doi.org/10.1007/s10009-021-00633-z

[51] J. Hom, L. Good, and Shuhui Yang. 2017. A survey of social-based routing protocols in Delay Tolerant Networks. In *2017 International Conference on Computing, Networking and Communications (ICNC)*. 788–792. https://doi.org/10.1109/ICCNC.2017.7876231

[52] Shaobo Huang, Jinsong Gui, Tian Wang, and Xiong Li. 2021. Joint mobile vehicle–UAV scheme for secure data collection in a smart city. *Annals of Telecommunications* 76 (2021), 559–580.

[53] CL Hwang, Frank A Tillman, and MH Lee. 1981. System-Reliability Evaluation Techniques for Complex/Large Systems: A Review. *IEEE Transactions on Reliability* 30, 5 (1981), 416–423.

[54] Edward J Birrane III. 2012. Building routing overlays in disrupted networks: inferring contacts in challenged sensor internetworks. *International Journal of Ad Hoc and Ubiquitous Computing* 11, 2-3 (2012), 139–156.

[55] David J Israel, Kendall D Mauldin, Christopher J Roberts, Jason W Mitchell, Antti A Pulkkinen, D Cooper La Vida, Michael A Johnson, Steven D Christe, and Cheryl J Gramling. 2020. Lunanet: a flexible and extensible lunar exploration communications and navigation infrastructure. In *2020 IEEE Aerospace Conference*. IEEE, 1–14.

[56] Andrew Jenkins, Sebastian Kuzminsky, Kevin K. Gifford, Robert L. Pitts, and Kelvin Nichols. 2010. Delay/Disruption-Tolerant Networking: Flight test results from the international space station. In *2010 IEEE Aerospace Conference*. IEEE, 1–8. https://doi.org/10.1109/AERO.2010.5446948

[57] Ravi Kalaputapu and Michael J. Demetsky. 1995. Modeling schedule deviations of buses using automatic vehicle-location data and artificial neural networks. *Transportation Research Record* 1497 (1995), 44 – 52.

[58] M. Kwiatkowska, G. Norman, and D. Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11) (Lecture Notes in Computer Science, Vol. 6806)*, G. Gopalakrishnan and S. Qadeer (Eds.). Springer, 585–591.

[59] Tuan Le. 2021. Multi-hop routing under short contact in delay tolerant networks. *Computer Communications* 165 (2021), 1–8.

[60] Axel Legay, Sean Sedwards, and Louis-Marie Traonouez. 2014. Scalable Verification of Markov Decision Processes. In *Software Engineering and Formal Methods - SEFM 2014 Collocated Workshops: HOFM, SAFOME, OpenCert, MoKMaSD, WS-FMDS, Grenoble, France, September 1-2, 2014, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 8938)*. Springer, 350–362. https://doi.org/10.1007/978-3-319-15201-1_23

[61] P. Madoery, F. Raverta, J. Fraire, and J. Finochietto. 2017. On the Performance Analysis of Disruption Tolerant Satellite Networks Under Uncertainties. In *Proceedings of the 2017 XVII RPIC Workshop*.

[62] P. G. Madoery, F. D. Raverta, J. A. Fraire, and J. M. Finochietto. 2018. Routing in Space Delay Tolerant Networks under Uncertain Contact Plans. In *2018 IEEE International Conference on Communications (ICC)*. 1–6. https://doi.org/10.1109/ICC.2018.8422917

[63] Jabar Mahmood, Zongtao Duan, Yun Yang, Qinglong Wang, Jamel Nebhen, and Muhammad Nasir Mumtaz Bhutta. 2021. Security in vehicular ad hoc networks: challenges and countermeasures. *Security and Communication Networks* 2021 (2021), 1–20.

[64] Mathis Niehage, Arnd Hartmanns, and Anne Remke. 2021. Learning optimal decisions for stochastic hybrid systems. In *19th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, S. Arun-Kumar, Dominique Méry, Indranil Saha, and Lijun Zhang (Eds.). ACM, 44–55. https://doi.org/10.1145/3487212.3487339

[65] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs*. Springer. https://doi.org/10.1007/978-3-319-28929-8

[66] Liviu Panait and Sean Luke. 2005. Cooperative Multi-Agent Learning: The State of the Art. *Auton. Agents Multi Agent Syst.* 11, 3 (2005), 387–434. https://doi.org/10.1007/s10458-005-2631-2

[67] Jim Partan, Jim Kurose, and Brian Neil Levine. 2007. A Survey of Practical Issues in Underwater Networks. *SIGMOBILE Mob. Comput. Commun. Rev.* 11, 4 (Oct. 2007), 23–33. https://doi.org/10.1145/1347364.1347372

[68] Wolf-Bastian Pöttner, Johannes Morgenroth, Sebastian Schildt, and Lars Wolf. 2011. Performance comparison of DTN bundle protocol implementations. In *Proceedings of the 6th ACM workshop on Challenged networks*. ACM, 61–64.

[69] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st ed.). John Wiley & Sons, Inc., New York, NY, USA.

[70] Tim Quatmann and Joost-Pieter Katoen. 2018. Sound Value Iteration. In *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 10981)*. Springer, 643–661. https://doi.org/10.1007/978-3-319-96145-3_37

[71] F. D. Raverta, R. Demasi, P. G. Madoery, J. A. Fraire, J. M. Finochietto, and P. R. D'Argenio. 2018. A Markov Decision Process for Routing in Space DTNs with Uncertain Contact Plans. In *2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*. IEEE, 189–194. https://doi.org/10.1109/WiSEE.2018.8637330

[72] Fernando D. Raverta, Juan A. Fraire, Pablo G. Madoery, Ramiro A. Demasi, Jorge M. Finochietto, and Pedro R. D'Argenio. 2021. Routing in Delay-Tolerant Networks under uncertain contact plans. *Ad Hoc Networks* 123 (2021), 102663. https://doi.org/10.1016/j.adhoc.2021.102663

[73] Partha Pratim Ray. 2022. A review on 6G for space-air-ground integrated network: Key enablers, open challenges, and future direction. *Journal of King Saud University-Computer and Information Sciences* 34, 9 (2022), 6949–6976.

[74] Nasir Saeed, Heba Almorad, Hayssam Dahrouj, Tareq Y Al-Naffouri, Jeff S Shamma, and Mohamed-Slim Alouini. 2021. Point-to-point communication in integrated satellite-aerial 6G networks: State-of-the-art and future challenges. *IEEE Open Journal of the Communications Society* 2 (2021), 1505–1525.

[75] Anant Sahai, Rahul Tandra, Shridhar Mubaraq Mishra, and Niels Hoven. 2006. Fundamental design tradeoffs in cognitive radio systems. In *Proceedings of the First International Workshop on Technology and Policy for Accessing Spectrum*. ACM, 2.

[76] K. Scott and S. Burleigh. 2007. *Bundle Protocol Specification*. RFC 5050. RFC Editor. http://www.rfc-editor.org/rfc/rfc5050.txt

[77] K. Scott, S. Burleigh, and E. Birrane. 2022. *Bundle Protocol Version 7*. RFC 9171. RFC Editor. http://www.rfc-editor.org/rfc/rfc9171.txt http://www.rfc-editor.org/rfc/rfc9171.txt.

[78] Hamayoun Shahwani, Syed Attique Shah, Muhammad Ashraf, Muhammad Akram, Jaehoon Paul Jeong, and Jitae Shin. 2022. A comprehensive survey on data dissemination in Vehicular Ad Hoc Networks. *Vehicular Communications* 34 (2022), 100420.

[79] Atul Sharma, Nitin Goyal, and Kalpna Guleria. 2021. Performance optimization in delay tolerant networks using backtracking algorithm for fully credits distribution to contrast selfish nodes. *The Journal of Supercomputing* 77 (2021), 6036–6055.

[80] Amit Kumar Singh and Rajendra Pamula. 2021. An efficient and intelligent routing strategy for vehicular delay tolerant networks: An efficient routing strategy for VDTN. *Wireless Networks* 27 (2021), 383–400.

[81] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction* (second ed.). The MIT Press.

[82] Saif Ullah and Amir Qayyum. 2022. Socially-aware adaptive delay tolerant Network (DTN) routing protocol. *PloS one* 17, 1 (2022), e0262565.

[83] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-Learning. *Mach. Learn.* 8 (1992), 279–292. https://doi.org/10.1007/BF00992698

[84] Libing Wu, Shuqin Cao, Yanjiao Chen, Jianqun Cui, and Yanan Chang. 2021. An adaptive multiple spray-and-wait routing algorithm based on social circles in delay tolerant networks. *Computer Networks* 189 (2021), 107901.

[85] J. Wyatt, S. Burleigh, R. Jones, L. Torgerson, and S. Wissler. 2009. Disruption Tolerant Networking Flight Validation Experiment on NASA's EPOXI Mission. In *Advances in Satellite and Space Communications, 2009. SPACOMM 2009. First International Conf. on.* 187–196. https://doi.org/10.1109/SPACOMM.2009.39

[86] Tingting Yang, Lingzheng Kong, Nan Zhao, and Ruijin Sun. 2021. Efficient energy and delay tradeoff for vessel communications in SDN based maritime wireless networks. *IEEE Transactions on Intelligent Transportation Systems* 22, 6 (2021), 3800–3812.

[87] Zhiyan A Younis, Adnan Mohsin Abdulazeez, Subhi RM Zeebaree, Rizgar R Zebari, and Diyar Qader Zeebaree. 2021. Mobile Ad Hoc Network in Disaster Area Network Scenario: A Review on Routing Protocols. *International Journal of Online & Biomedical Engineering* 17, 3 (2021).