On the Tractability of Yen's Algorithm and Contact Graph Modeling in Contact Graph Routing

Olivier De Jonckère*, Juan A. Fraire†‡, Scott Burleigh§
*Technische Universität Dresden, Dresden, Germany
†Univ Lyon, Inria, INSA Lyon, CITI, F-69621 Villeurbanne, France
‡CONICET - Universidad Nacional de Córdoba, Córdoba, Argentina
§D3TN U.S. Corp., Florida, 444 Brickell Avenue, Miami, FL 33131, USA

***Abstract*—Contact Graph Routing (CGR), later standardized as Schedule-Aware Bundle Routing (SABR) by the Consultative Committee for Space Data Systems (CCSDS), is commonly implemented through modifications of Dijkstra and Yen's algorithms. The contribution of this paper lies in the detailed analysis and measurement of the effects caused by Yen's algorithm on CGR. It is observed that Yen's algorithm significantly reduces the scalability of CGR, rendering it unsuitable for scenarios involving numerous nodes numbering in the thousands or even hundreds. Furthermore, our analysis sheds light on how Yen's algorithm introduces unpredictable performance in CGR, resulting in substantial variations in memory usage and processing demands during the scheduling of individual bundles. This critical examination of the algorithm's behavior highlights operational risks, as potential adversaries could exploit this vulnerability by strategically forging and transmitting only a few bundles, effectively paralyzing the network.**

***Index Terms*—Contact Graph Routing, Bundle Protocol, Schedule-Aware Bundle Routing**

I. Introduction

The prediction of space network size growth, as discussed in the "The future Mars Communications Architecture" report by the Interagency Operations Advisory Group (IOAG) [1], necessitates the scalability of the underlying routing mechanisms.

As indicated by the IOAG, the Delay-Tolerant Network (DTN) architecture [2] comprises relay orbiters, user vehicles (both in orbit and on the surface), relevant Earth stations, and Mission Operations Centers (MOCs) operating as DTN nodes. Implementing the bundle protocol [3] within this architecture enables the retention of bundles, which serve as the protocol transfer unit, until subsequent intervals of connectivity.

Schedule-Aware Bundle Routing (SABR) [4] standardized by the Consultative Committee for Space Data Systems (CCSDS) is also identified by the IOAG as essential for the service management function. SABR is derived from the Contact Graph Routing (CGR) implementation [5]. CGR is currently part of the Interplanetary Overlay Network (ION) [6], NASA's reference implementation of the DTN stack.

CGR is a deterministic routing algorithm that utilizes delay-tolerant adaptations of Dijkstra and Yen's algorithms [7]. In a scheduled DTN, the intervals of connectivity, referred to as contacts, are predetermined and known in advance. This knowledge is facilitated through orbit trajectory predictions and contact planning. The contact plan, which comprises the list of contacts for the network, is provided to each node. This contact plan enables the creation of a graph for efficient pathfinding within the network.

However, SABR is recognized to encounter scalability challenges [8], primarily attributed to its algorithmic complexity. The processing pressure of SABR is directly influenced by the size of the contact plan, imposing substantial limitations on its suitability for networks consisting of more than a few nodes or contact plans spanning a long time horizon.

The integration and utilization of Yen's K-shortest path algorithm in CGR are quite unusual as scheduling events occur during Yen's algorithm main loop and not after the computation of the K-shortest paths. Such entanglement justifies new hypotheses stating that the algorithmic complexity explanation alone is insufficient to explain its high processing pressure.

Furthermore, it is worth noting that the algorithmic complexity of Yen's algorithm is contingent on the specific implementation of Dijkstra's algorithm. Additionally, the increasing adoption of node multigraph utilization [9]–[12] as an approach to reduce the processing cost typically associated with CGR asks for a deeper analysis of the impact of Dijkstra and Yen's algorithm in CGR.

This paper proposes an in-depth analysis of CGR's functioning that allows a new level of understanding concerning CGR's scalability issues. This work highlights severe operational and security risks and the results show that Yen's algorithm is not the sole scalability issue.

In Section II, the state-of-the-art will be covered. The analysis of CGR will then be conducted in Section III. Evaluation of the core issues follows in Section IV, and lastly, a conclusion will be proposed in Section V.

II. State-of-the-art

A. Yen's Algorithm

Yen's k-shortest path algorithm, introduced in 1972 by Yen [13] and Lawler [14], is a technique used to find the k-shortest paths between a given source and destination in a graph. It operates iteratively by progressively exploring alternative routes, eliminating previously discovered paths at each iteration. The algorithm begins by finding the shortest path using a conventional algorithm such as Dijkstra's. It then examines the set of nodes encountered on this shortest path and identifies potential detours by temporarily removing edges from the graph. By iteratively repeating this process, the algorithm constructs a set of k shortest paths by considering different deviations and reconstructions from the original shortest path. Yen's algorithm provides a flexible and adaptive approach to finding alternative routes, allowing for a diverse range of path options based on the desired number of shortest paths (K).

B. Contact Graph Routing

CGR consists of two main phases: *route construction* and *route selection*.

a) Route Computation: During the route computation phase, routes are calculated by a delay-tolerant version of Dijkstra's algorithm and incorporating an alternative route search mechanism, typically Yen's algorithm. These routes are generated based on a specific destination without considering the characteristics of individual bundles, such as priority, size, expiration time, or node exclusions. Dijkstra's algorithm is adapted by assuming a bundle size of zero, thus disregarding the residual contact volumes. The resulting routes are then stored in routing tables on a per-destination basis.

b) Route Selection: During the route selection phase, when a bundle needs to be scheduled, the routing table is accessed for the specified destination to identify potential routes for the particular bundle. The bundle's specific aspects, such as priority, size, expiration time, and node exclusions, are considered in this phase. Unsuitable routes are disregarded, and the most appropriate route with the earliest estimated arrival time is selected. The route selection phase can be resumed to compute a new route if no suitable route is found in the routing table. This process continues until a computed route that meets the requirements of the bundle is found and selected or it is determined that no suitable route exists, leading to the abandonment of bundle scheduling.

C. Yen's Algorithm in CGR

Yen's K-shortest path algorithm is commonly employed in CGR with an adaptive approach. This is how CGR operates in ION, for instance. The parameter K, representing the number of paths to be discovered, is not explicitly specified. As a result, the main loop of Yen's algorithm does not terminate after a fixed number of iterations, leading to an infinite loop that can be temporarily suspended and resumed as required.

While suitable routes are available in the routing table, the optimal route will be selected for forwarding the bundle, effectively pausing the main loop of Yen's algorithm. If the existing routes are deemed unsuitable, the algorithm resumes its main loop to calculate the next alternative route for a single iteration. If the newly computed route is suitable, the bundle can be forwarded using this route. However, further iterations of Yen's main loop are necessary if the route remains unsuitable. Consequently, a single bundle scheduling event may trigger multiple route computations.

This process, depicted in figure 1, is the core operational approach implemented in ION.

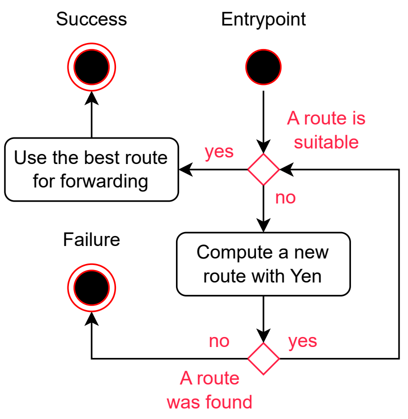


Figure 1: Route selection and computation phases in CGR with Yen's algorithm.

D. Determinism and Predictability

Deterministic algorithms consistently produce the same output for a given input, leading to predictable processing and memory utilization behavior. This absence of randomness holds considerable value in space missions, enabling effective and risk-free operations.

In this context, Dijkstra's algorithm shows a notable sensitivity to the graph size, determined by the number of nodes and edges. However, its algorithmic complexity does not exhibit discernible sensitivity to the specific source and destination inputs employed for pathfinding. In this case, the worst-case computational pressure remains consistent.

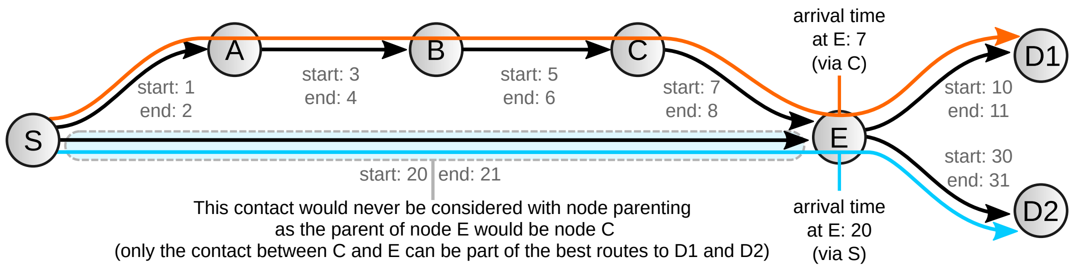
The practical behavior of Dijkstra's algorithm lends predictability to its routing capabilities, as graph update events occur less frequently than message forwarding events. Moreover, the sizes of the provided contact plans are anticipated to remain relatively stable between updates. As a result, each update of the contact plan is not expected to introduce substantial variations in the processing pressure associated with Dijkstra's algorithm.

Dijkstra and Yen's algorithms are characterized by their deterministic nature and predictability. Their predictability is particularly crucial at the individual message-forwarding event level. However, as discussed in Section II-C, it was conceptually illustrated that the number of iterations (K) needed to schedule a bundle using Yen's algorithm can vary. The subsequent sections will delve into a quantitative analysis of the predictability of CGR when Yen's algorithm is employed adaptively as its core component.

III. Analysis

A. Graph structure

In the context of CGR, the graph structure is defined such that the vertices represent contacts, and the edges represent the periods of bundle retention. Dijkstra's algorithm produces a path composed of a sequence of vertices. Accordingly, when applied within CGR, Dijkstra's algorithm yields a route as its



arrival time at E: 7 (via C)
S
A
B
C
E
D1
D2
start: 1 end: 2
start: 3 end: 4
start: 5 end: 6
start: 7 end: 8
start: 10 end: 11
start: 30 end: 31
start: 20 end: 21
This contact would never be considered with node parenting as the parent of node E would be node C (only the contact between C and E can be part of the best routes to D1 and D2)
arrival time at E: 20 (via S)

Figure 2: Pathfinding variations opposing node parenting against contact parenting and analog techniques (figure from [11])

output, which can be represented as a sequence of contact vertices, with the edges (absent but denoted by their order) representing the intervals of message retention.

The routes reflect pretty well the contact graph description. However, applying this description to a whole graph renders its construction unintuitive.

This aspect is exemplified in ION, where a red-black tree stores the graph data. Even though a tree is leveraged internally, this data structure is a 1-dimensional ordered list with relatively fast insertion and access (logarithmic complexity). This choice deviates from a structure that readily aligns with the contact graph representation. In contrast, CGR implementations and alternative approaches that employ node multigraphs [9]–[12] facilitate a direct and intuitive mapping between the concept of node multigraphs and the data structure utilized in the implementation. A node multigraph is usually implemented with nested maps (with 2 levels, the first level needs to be implemented as a hashmap for constant complexity access). The inmost values are ordered lists of contacts for single sender-receiver pairs (the sender being the key for the first level and the receiver for the second level of the nested maps). Rapid access to the contacts within the multigraph offers a valuable feature for reducing processing pressure. However, it is important to note that the output may vary depending on the parenting strategy employed.

In ION, parenting is *contact based*, and each contact being explored is assigned a parent contact (to construct a reverse path from the last hop contact). This allows the algorithm to find the best route as intended by the SABR standard.

Some implementations switch to *node parenting* by attaching explored contacts to the receiver nodes of the contacts (nodes being vertices in the graph). If node parenting is leveraged, the computational pressure is minimal [9], [10]. Indeed, assigning and overriding the distances on vertices (nodes) requires less effort compared to assigning and overriding the distances on edges (contacts), drastically reducing the exploration scope and the number of elements eligible to be the next current element for the next iteration within Dijkstra (as they are way more contacts than nodes). Still, the computed routes might not be the shortest in the sense of SABR since they possibly show sub-optimal hop counts if the topology involves multiple paths sharing the same last contact (discussed later in Section III-B).

The differences in the contact and node parenting pathfinding representations are depicted in figure 2.

Note that the parenting strategy is independent of the underlying graph structure, and it can arguably be stated that the minimal modification to alter CGR is a shift in the parenting strategy rather than a shift in the graph structure.

To wrap up, contact parenting is preferred for optimizing networking performance, while node parenting is favored for enhancing computational performance.

However, making a definitive choice between these two strategies is not necessarily mandatory. For instance, in the context of SPSN (Shortest Path tree approach for Space Networks), an alternative approach aimed at achieving the objectives of SABR. Node parenting could still be observed in [15] but was addressed in [11] through a hybrid approach that combines contact parenting with node-based filtering. This hybrid approach balances the two strategies and leverages the strengths of each to optimize the overall performance of the system.

B. Distance calculation

According to the SABR standard, the shortest path to a destination is determined based on the earliest arrival time at the destination. In cases where multiple paths have the same optimal arrival time, the preference is given to the path with the fewest contacts. Suppose a definitive decision cannot be made based on these criteria. In that case, the tie is resolved by considering the expiration time of each route, with a later expiration time being preferred. Additionally, if necessary, the node number of the receiving node for the initial contact of each route is examined, with a smaller node number being preferred as a final tiebreaker.

During the exploration and selection process, the arrival time at a vertex (which represents the receiver of the contact if the vertices are associated with contacts) is calculated. This calculation considers the bundle's arrival time at the sending vertex (representing the contact's sender), the contact's start time, and the latency associated with signal propagation between the sending and receiving nodes. In scenarios where the search is volume-aware, as demonstrated in [10], [16], or during the route selection process, the bundle size is also considered a factor in the calculations.

In the Dijkstra algorithm, distance calculation involves assigning a distance value to each vertex, the sum of the distance from the previous vertex, and the cost of the edge connecting them. This value is updated if a shorter distance is found. However, in the context of SABR, the objective is to minimize the arrival time rather than the transit time.

To illustrate this, consider a scenario of flight connections, where the arrival time of a passenger is determined by the arrival time of the last flight, regardless of the cumulative transit times and arrival times of the previous flights taken to reach the penultimate airport. In this scenario, the shortest trip would be the one that allows the passenger to reach the destination airport as soon as possible, prioritizing arrival time over transit time.

Consequently, distance to the destination in CGR is quasi-exclusively determined by the last hop contact rather than being calculated in an additive manner due to the different metrics involved. The route arrival time is more likely to be constrained by the last contact arrival time, even if route construction is volume-aware. If the start time of the last contact is not later than the start time of the previous one, then the arrival time will not be constrained by the last contact but by the penultimate one, and the same comment applies to each contact upstream on the route.

C. Yen's algorithm unpredictability

This approach to distance calculation presents significant side effects when Yen's algorithm is used as the alternative route construction technique.

The adaptive implementation of Yen's algorithm is likely the most cost-efficient strategy in CGR. Computing all possible routes beforehand is impractical, as the number of paths needed cannot be determined in advance (as discussed in Section II-C). Instead, the adaptive strategy allows for the computation of only the routes necessary for the specific routing requirements at hand. By dynamically generating routes as needed, the adaptive approach minimizes unnecessary computations and optimizes allocating computational resources. This adaptive strategy balances computational efficiency and meets the routing demands of CGR.

However, while a single Dijkstra call typically incurs a minimal computational cost, cycling through Yen's main loop can involve multiple algorithm iterations. Each iteration, in turn, requires additional Dijkstra calls, with the total count proportional to the number of iterations multiplied by the average path length. Therefore, the computational cost associated with Yen's algorithm can accumulate as the main loop progresses, potentially leading to increased processing requirements during bundle scheduling.

The critical question revolves around the stability of the number of iterations required for scheduling a single bundle, as it directly impacts predictability. The number of iterations is influenced by network load and topology.

To illustrate a worst-case scenario, as discussed in [11] and depicted in Figure 3, let's consider two bundles to be scheduled between a source node S and a destination node D. The two final contacts with node D, denoted as C_1 and C_2, have node G as the transmitting node. The topology is intentionally dense between nodes S and G, and the start time of contacts C_1 and C_2 happens further in the future with respect to the dense section of the network.

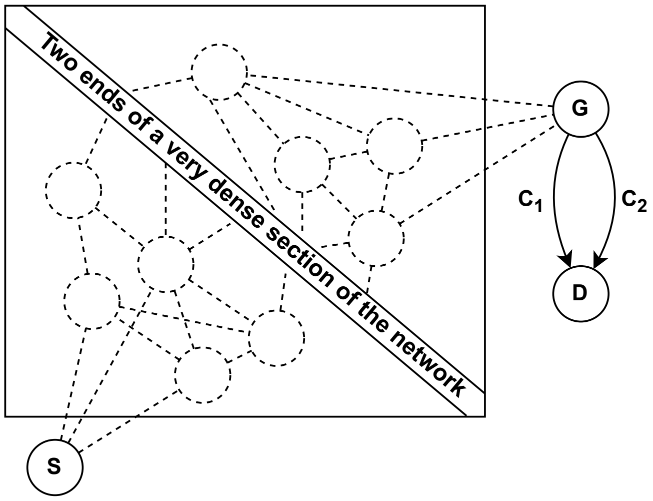


Figure 3: Problematic configuration example for Yen's algorithm (figure from [11])

The first bundle fully utilizes the volume of C_1, necessitating Yen's algorithm to find a route with C_2 as the last hop contact for the second bundle. Several Yen's iterations might be needed to arrive at a feasible route in such a configuration. Specifically, once the first bundle is scheduled with a route ending at C_1. Let set 1 be the list of routes sharing C_1 as the last contact. Any route sharing C_1 as the last hop will be considered shorter than any route ending with C_2. With set 2 the list of routes sharing C_2 as the last contact. Since Yen's algorithm discovers routes in increasing order of distance, CGR must identify all routes in set 1 before it can detect the first route in set 2. This scenario highlights the potential challenges and delays in determining suitable routes for bundle scheduling.

The refinement of the issue focuses on the size of set 1 and its potential operational consequences. Specifically, it raises two concerns: the memory pressure and the processing time associated with computing set 1. If set 1 can become arbitrarily large, it may impose unsustainable memory usage, potentially leading to resource exhaustion. Alternatively, if memory is not the bottleneck, the processing time required to handle set 1 could render the routing algorithm unresponsive, causing delays in route determination. Both scenarios highlight the importance of understanding the limitations and scalability of the routing algorithm, as excessively large set 1 could pose significant operational challenges and impact the network's overall performance.

Yen's algorithm needs two containers, usually named A (the output) and B (internal memory for candidates to be pushed in A). In CGR, the container A can be the routing table, and each Yen's algorithm iteration inserts a new route to the routing table. The container B is an internal memory for Yen's algorithm, and each iteration can insert several routes into this container. Consequently, the memory pressure is not only reflected in the routing table sizes and could be highly underestimated if only the routing tables are considered.

Furthermore, all the routes in set 1 will likely be disregarded in subsequent bundle scheduling after the two original bundles have been scheduled. This is because all these routes share the same issue of an exhausted contact (C_1). Consequently, the memory pressure resulting from storing routes in set 1 would not provide any benefit or contribute to scheduling future bundles. This underscores the potential inefficiency and wastage of system resources caused by such routes in the routing algorithm.

Detecting all routes in set 1 can potentially lead to a behavior comparable to a combinatorial explosion. The consequences may not be noticeable in simple scenarios with few contacts (as those typically analyzed in CGR-related literature). However, in scenarios involving dense networks with multiple nodes and large contact plan time horizons, the likelihood of encountering a configuration similar to the scenario depicted in Figure 3 increases. This is particularly true during operational periods when a specific destination has fewer and later connections than other nodes in the system. In such cases, the number of routes to be detected and processed by the routing algorithm can grow significantly, potentially causing computational and memory issues.

A ring road network [17], [18] is a network encompassing orbiters and surface nodes with DTN capabilities. This issue was detected on a realistic ring road scenario encompassing 15 satellites and 15 ground stations. This experience highlights two essential aspects.

Firstly, the issue discussed is not limited to networks with many nodes and contacts. The severity of the problem is determined by the density of the network between the source and the problematic contact rather than the overall number of nodes. Even in relatively small scenarios, if there is sufficient network density and the start time of the problematic contact is late enough, the issue can still arise.

Secondly, it is important to note that this issue is not restricted to unrealistic laboratory scenarios. The scenarios we use in the evaluation section were generated using real satellite orbit information and still present significant issues. This highlights that the problem can manifest in real-world scenarios, emphasizing the practical significance of addressing it.

D. Security threats

The identified vulnerability raises concerns about the potential for targeted attacks on specific nodes within the network. If an attacker has knowledge of the network's contacts and identifies a critical contact, they can exploit this information by sending bundles of carefully chosen sizes. This can result in the targeted nodes, and potentially the entire network, being paralyzed.

This attack might be easy to implement as only a few bundles of specific sizes are required to trigger a node failure. Additionally, the flexibility of Bundle encapsulation [19] allows the attacker to encapsulate the bundle that can cause the desired consequences for a particular destination within a bundle that can be sent to another target node. Once the encapsulated bundle is received on the target node, forwarding it may lead to a failure.

These findings highlight the importance of addressing the security vulnerabilities associated with bundle processing and the need for robust measures to protect against targeted attacks within the network.

Handling bundle expiration is crucial to avoid triggering the identified issue when resuming Yen's algorithm's main loop. When the bundle expiration time is earlier than the current time, the routing algorithm is expected to drop the bundle immediately. However, if the bundle expiration time is later than the present time, an important check needs to be performed before resuming Yen's algorithm's main loop: the bundle can be dropped if a newly route constructed route shows an arrival time later than the bundle expiration. Failure to implement this check properly can result in Yen's algorithm continuing to search for routes as long as new routes to the destination can be detected.

Such handling of the expiration time is not discussed in the standard, and the potential operational risks would motivate the introduction of such concerns in SABR.

IV. Evaluation

A. Algorithms and Scenarios

The study's evaluation demonstrates that using Yen's algorithm in the context of bundle routing presents scalability limitations and highlights the independent nature of the identified issue from the parenting strategy employed. In addition to evaluating Yen's algorithm, the study plans to assess the performance of contact parenting by replacing Yen's algorithm with a limiting contact approach, as described in [16]. This approach is known to exhibit minimal processing pressure. Furthermore, different flavors of routing strategies, including limiting contact approaches and node parenting, will be evaluated for control purposes.

This evaluation comprises four algorithms:

- *cgr-1st-dep*: node parenting, limiting contact approach (first depleted) [16]
- *cgr-1st-end*: node parenting, limiting contact approach (first ending) [16]
- *cgr-1st-end-cvic*: contact parenting, limiting contact approach (first ending)
- *cgr-yen-ion*: node parenting, Yen's algorithm (first depleted).

The algorithms utilized in the evaluation use a node multigraph (whatever parenting strategy leveraged). Also, various techniques to mitigate processing pressure of *cgr-yen-ion* have been applied to ensure that the simulation times remain tractable in practice. To control the computational complexity, a maximum value of K (the number of paths to find) has been set to 1000. If this upper bound is reached during the route computation, the bundles are dropped to avoid excessive

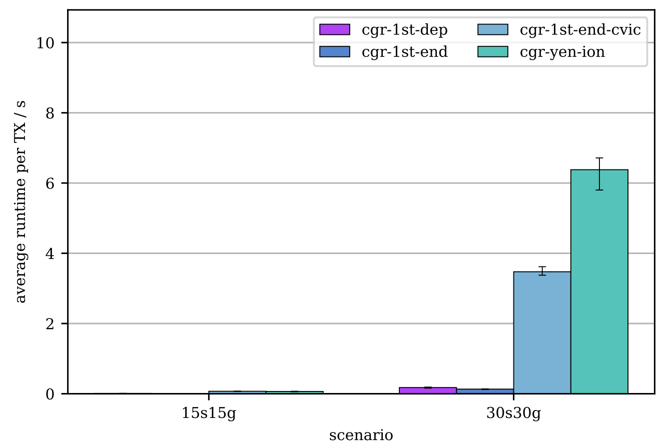


Figure 4: Avg. runtime per transmission (processing overhead).

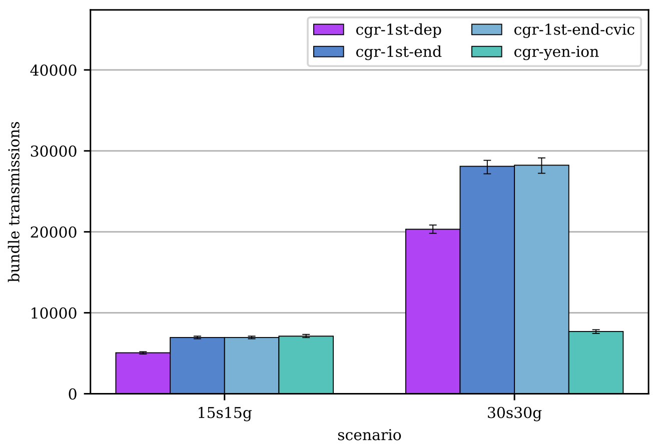


Figure 5: Transmission count (simulation overhead).

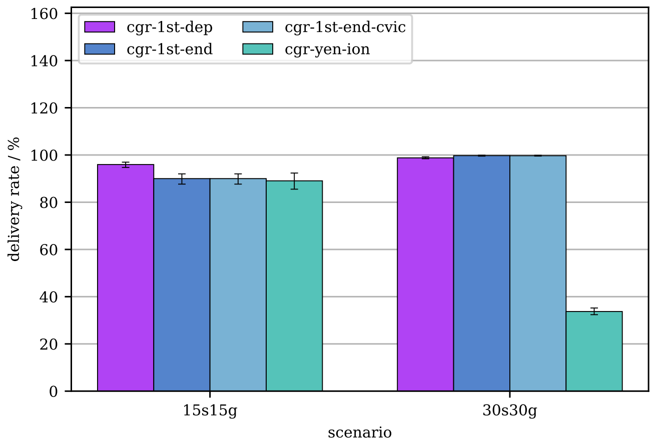


Figure 6: Delivery ratio.

computational burden. Additionally, two iterations at most are allowed for a single bundle scheduling. After two attempts, if a suitable route is not found, the bundle is dropped. This approach prevents prolonged execution times and, more importantly, prevents individual bundles from consuming a significant portion of the allocated 1000 iterations.

This evaluation presents two realistic ring road satellite network scenarios[1]:

- *15s15g*: 15 satellites and 15 ground stations, 1445 bundle injections, 3608 contacts.
- *30s30g*: 30 satellites and 30 ground stations, 5784 bundle injections, 22242 contacts.

The scenarios allow inter-satellite links, and the contacts have an average duration of about 7 minutes 30, and the contact plan covers an operational period of 48 hours.

B. Results

a) Processing Overhead: Figure 4 gathers the results for the processing time per transmission (simulation time included). The *cgr-1st-end* and *cgr-1st-dep* algorithms act as controls as they do not use contact parenting or Yen's algorithm. The processing times per bundle transmission do not exceed 200 ms for the *cgr-1st-dep* and *cgr-1st-end* algorithms, while they exceed 3 seconds for *cgr-1st-end-cvic* and more than 6 seconds for *cgr-yen-ion* for the *30s30g* scenario. Results are compelling evidence that contact parenting (*cgr-1st-end-cvic*) and the use of Yen's algorithm (*cgr-yen-ion*) present two concrete causes for the scalability limitations of CGR. Concerning scalability, switching from the *15s15g* scenario to the 30s30g scenario represents a multiplication of the average runtime per transmission of 17 for *cgr-1st-dep*, 26 for *cgr-1st-end*, 49 for *cgr-1st-end-cvic*, and 98 for *cgr-yen-ion*. It shall be recalled that the processing baselines (i.e., on the *15s15g*) are way lower for the limiting contact approaches.

b) Simulation Overhead: The simulation overhead is not excluded from the runtime results but is negligible. In practice, simulation overhead is proportional to the number of transmissions. Figure 5 proves that this overhead cannot account for the differences observed in the processing pressure.

[1]Generated with https://gitlab.com/d3tn/dtn-tvg-util

c) Delivery Ratio: The algorithm *cgr-1st-dep* acts as a control to show that the load is high, as this flavor is known to perform better in congested environments compared with other CGR flavors [16]. Figure 6 confirms that some congestion is present for the *15s15g* scenario as the delivery rate of the *cgr-1st-dep* flavor is higher than the other flavors. Switching to the *30s30g* scenario highlights that the mitigation techniques used for the *cgr-yen-ion* algorithm, i.e., setting K to 1000, do not allow Yen's approach to functioning as intended. Lifting the mitigation technique would possibly increase networking performance by allowing CGR to find routes for the now-dropped bundles. In other words, lifting the mitigation techniques would increase the already problematic processing pressures with mitigation.

C. Analysis

a) Yen's Intractability: The results of our study underscore the severity of the issues identified. Even with implementing countermeasures to prevent extended processing delays, the average processing time per transmission remains considerable for Yen's algorithm. The impact of Yen's algorithm is validated by comparing *cgr-1st-end* with *cgr-yen-ion* on the *30s30g* scenario (from $\approx$0.2 to $\approx$6.3 seconds). Moreover, the processing load of Yen's algorithm was effectively controlled through our mitigation measures, which were triggered over 3800 times during the evaluation. Ad-hoc experiments in the *15s15g* scenario proved that more than an hour might be needed for Yen to deliver a valid route without mitigation. However, mitigation reduced the delivery ratio to approximately 37%, although the existence of routes for at least 99.7% of the generated traffic. This demonstrates the significant impact of the processing limitations on the overall performance of Yen's algorithm. Additionally, the average hop count for Yen's algorithm was 1.33, in contrast to the average hop count of 4.86 observed for the *cgr-1st-end* flavor. This disparity highlights that the mitigation measures are activated relatively early in the planned end-to-end path.

b) Contact Parenting: The use of contact parenting was found to have higher processing pressure than expected, significantly contributing to CGR's scalability issues while being essential to accurate path detection (see section III-A). This impact was validated through a comparison between the *cgr-1st-end* flavor and the *cgr-1st-end-cvic* variant in the *30s30g* scenario, multiplying the scheduling times by 26.

Furthermore, it was observed that implementing a node multigraph did not adequately alleviate the processing pressure associated with contact parenting. Despite this approach, the processing pressure in CGR persists. These findings underscore the importance of conducting further investigations and optimizations to address the identified issues in CGR effectively (for example with hybrid parenting like in [11]).

V. Conclusion

In this paper, we conducted a detailed analysis of the scalability issues in CGR and the associated operational risks. Our analysis identified two independent aspects contributing to the high processing pressure: the occurrence of long hangs during bundle scheduling due to the integration of Yen's algorithm in CGR and the processing pressure associated with contact parenting in Dijkstra's exploration.

Through our evaluation, we demonstrated the independent nature of these issues by observing problematic processing pressures in flavors that utilize contact parenting and Yen's algorithm separately. Furthermore, we emphasized the independence between the graph structure and parenting, challenging the traditional association of node multigraph implementations with strict node parenting and contact parenting with the concept of contact graph.

Considering the future Mars Communications Architecture report predictions from the IOAG, which indicate a network of 25 delay-tolerant nodes by 2026, our findings indicate that CGR will be highly challenged to provide robust routing capabilities for such a network.

Future work will focus on developing mitigation techniques that preserve networking performance for Yen's algorithm or exploring alternative route search strategies. Additionally, there is a need to introduce mitigation techniques to address the processing pressure in Dijkstra's algorithm within CGR.

Acknowledgment

This research was conducted during doctoral studies supervised by Marius Feldmann and with a simulation platform provided by Felix Walter. This research has received support from the EU's H2020 R&D program under the Marie Skłodowska-Curie grant agreement No 101008233 (MISSION project) and the French National Research Agency (ANR) ANR-22-CE25-0014-01.

References

[1] "The future mars communications architecture," *Interagency Operations Advisory Group*, 2022. [Online]. Available: https://www.ioag.org/Public%20Documents/MBC%20architecture%20report%20final%20version%20PDF.pdf

[2] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss, "Rfc 4838," *Delay-Tolerant Networking Architecture, IRTF DTN Research Group, April*, 2007.

[3] S. Burleigh, K. Fall, and E. J. Birrane, "Bundle Protocol Version 7," RFC 9171, Jan. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9171

[4] CCSDS, "Schedule-aware bundle routing," *Consultative Committee for Space Data Systems*, 2019.

[5] S. Burleigh, "Contact graph routing," *http://tools.ietf.org/html/draft-burleigh-dtnrg-cgr-00*, 2009. [Online]. Available: http://tools.ietf.org/html/draft-burleigh-dtnrg-cgr-00

[6] S. Burleigh, "Interplanetary overlay network: An implementation of the dtn bundle protocol," 2007.

[7] A. W. Brander, M. C. Sinclair *et al.*, "A comparative study of k-shortest path algorithms," in *Proc. of 11th UK Performance Engineering Workshop*. Springer, 1995, pp. 370–379.

[8] G. Wang, S. C. Burleigh, R. Wang, L. Shi, and Y. Qian, "Scoping contact graph-routing scalability: Investigating the system's usability in space-vehicle communication networks," *IEEE Vehicular Technology Magazine*, vol. 11, no. 4, pp. 46–52, 2016.

[9] F. Walter, "Prediction-enhanced Routing in Disruption-tolerant Satellite Networks," Doctoral Dissertation, Technische Universität Dresden, 2020. [Online]. Available: https://nbn-resolving.org/urn:nbn:de:bsz:14-qucosa2-721622

[10] O. De Jonckère and J. A. Fraire, "A shortest-path tree approach for routing in space networks," *China Communications*, vol. 17, no. 7, pp. 52–66, 2020.

[11] O. De Jonckère, J. Fraire, and S. C. Burleigh, "Enhanced pathfinding and scalability with Shortest-Path tree routing for space networks," in *2023 IEEE International Conference on Communications (ICC): SAC Satellite and Space Communications Track (IEEE ICC'23 - SAC-05 SSC Track)*, Rome, Italy, May 2023.

[12] M. Moy, R. Kassouf-Short, N. Kortas, J. Cleveland, B. Tomko, D. Conricode, Y. Kirkpatrick, R. Cardona, B. Heller, and J. Curry, "Contact multigraph routing: Overview and implementation," in *2023 IEEE Aerospace Conference*, 2023, pp. 1–9.

[13] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[14] E. L. Lawler, "A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem," *Management science*, vol. 18, no. 7, pp. 401–405, 1972.

[15] O. De Jonckère and J. A. Fraire, "A shortest-path tree approach for routing in space networks," *China Communications*, vol. 17, no. 7, pp. 52–66, 2020.

[16] J. A. Fraire, P. G. Madoery, A. Charif, and J. M. Finochietto, "On route table computation strategies in delay-tolerant satellite networks," *Ad Hoc Networks*, vol. 80, pp. 31–40, 2018.

[17] C. Krupiarz, C. Belleme, D. Gherardi, and E. Birrane, "Using smallsats and dtn for communication in developing countries," in *Proc. International Astronautical Congress (IAC-08. B4. 1.8)*, 2008.

[18] M. Feldmann, J. A. Fraire, F. Walter, and S. C. Burleigh, "Ring road networks: Access for anyone," *IEEE Communications Magazine*, vol. 60, no. 4, pp. 38–44, 2022.

[19] S. C. Burleigh, "Bundle-in-Bundle Encapsulation," Internet Engineering Task Force, Internet-Draft draft-ietf-dtn-bibect-03, Feb. 2020, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-dtn-bibect/03/