Operator calculus approach to minimal paths: Precomputed routing in a store-and-forward satellite constellation

Hugo Cruz-Sánchez, René Schott, and Ye Qiong Song
LORIA
Campus Scientifique, B.P. 239
54506 Vandoeuvre-lès-Nancy
FRANCE
cruzsanh@loria.fr
schott@loria.fr
song@loria.fr

G. Stacey Staples*
Department of Mathematics and Statistics
Southern Illinois University Edwardsville
Edwardsville, IL 62026-1653, USA
sstaple@siue.edu

ABSTRACT
An innovative minimal paths algorithm based on operator calculus in graded semigroup algebras is described. Classical approaches to routing problems invariably require construction of trees and the use of heuristics to prevent combinatorial explosion. The operator calculus approach presented herein, however, allows such explicit tree constructions to be avoided. Moreover, the implicit tree structures underlying the problem are pruned automatically by the inherent properties of the semigroup algebras used in this approach. The operator calculus algorithm is applied to the problem of precomputed routing in a store-and-forward satellite constellation, which provides message communication services by relaying messages between satellites through gateways on the ground. The minimum end-to-end delay paths obtained are compared with the best existing heuristics-based results. The best existing results were obtained from an algorithm designed to explore only a portion of the solution space in order to avoid combinatorial explosion and memory overload. In all test cases, the operator calculus is shown to return paths whose minimum end-to-end delay is either equal to or less than that of the best existing result.

Keywords
shortest paths, message routing, operator calculus, semigroup algebras

1. INTRODUCTION

Operator calculus (OC) methods on graphs have been developed in a number of works by Schott and Staples (cf. [3], [4], [5], [6]). The principal idea underlying the approach is the association of graphs with algebraic structures whose properties reveal information about the associated graphs. In particular, by constructing the “nilpotent adjacency matrix” associated with a finite graph, information about self-avoiding structures (paths, cycles, trails, etc.) in the graph are revealed by computing powers of the matrix.

In a dynamic problem, such as routing in a store-and-forward satellite constellation, continuous-time graph processes are discretized into “frames” representing graph connectivity within time windows. Nilpotent adjacency matrices are then constructed for distinct frames, and their properties are used to enumerate weighted paths in the process.

In the operator calculus approach, graded semigroup algebras are generated by “null-square” elements such that properties of the algebra “sieve out” paths; i.e., cycles are removed from consideration automatically. In the earlier operator calculus approach, the algebras were commutative so that path-identifying data (vertex sequences) was lost.

By defining a thresholded, non-commutative multiplication on the algebra, paths are preserved, paths containing cycles are removed from consideration, and paths with higher total weight than already existing paths can be removed from consideration (pruned) automatically by the inherent properties of the algebra itself. In this way, tree construction and pruning are implicit and the algorithm is reduced to matrix multiplication over the appropriate algebraic structures.

1.1 Store-and-forward satellite constellations
The application considered herein is based on the Low Earth Orbit (LEO) satellite constellation described by Cruz-Sánchez, et al. [1]. The system considered in that work offers store-and-forward (S&F) services including messaging, tracking, and monitoring. Time windows of satellite visibility for targeted areas were precomputed using a visibility algorithm [2].

In the routing problem under consideration, an ordered pair of gateways is chosen to serve as message source and message target. Messages are transported by satellites from
one gateway to another. Satellites are interconnected by gateways relaying messages. For simplicity, only gateway-satellite connections are allowed in the model discussed.

Full details of the context, space segment, terrestrial segment, geographical distribution of gateways, and types of routes are beyond the scope of the current work, and can be found in [1]. A description of the heuristics-based algorithm and a discussion of simulation results can also be found in that paper. Because the trees constructed by the algorithm were very large, heuristics were applied to avoid combinatorial explosion and memory overload. Consequently, the algorithm was designed to explore only a portion of the solution space for the precomputed routing problem. Best Existing Results (BER) from the heuristics-based algorithm are used as benchmarks for comparison of minimal-delay paths obtained from the OC approach.

2. OPERATOR CALCULUS APPROACH

The precomputed windows obtained from the visibility algorithm for LEO satellites are used to define a sequence of matrices having entries in a graded semigroup algebra described below. In this way, the system of gateways and satellites is modeled as a sequence of graphs whose topology is fixed throughout each time window. Each graph of the sequence is referred to henceforth as a frame.

2.1 Graded semigroup algebra

It will be convenient to adopt multi-index notation for subsequent discussion. In particular, letting \( u = (u_1, \ldots, u_k) \) for some \( k \), the notation \( \omega_u \) will be used to denote a sequence (or word) of distinct symbols of the form

\[
\omega_u := \omega_{u_1} \omega_{u_2} \cdots \omega_{u_k}.
\]  

(1)

For fixed positive integer \( n \), consider the alphabet \( \Omega := \{\omega_i : 1 \leq i \leq n\} \). One thereby obtains a freely-generated semigroup whose elements are the symbol 0 along with all finite words on distinct generators, i.e., finite sequences of distinct symbols from the alphabet \( \Omega \), if one defines the multiplication of two such sequences by

\[
\omega_u \omega_v = \begin{cases} 
\omega_{uv} & \text{if } u \cap v = \emptyset, \\
0 & \text{otherwise.}
\end{cases}
\]  

(2)

Here, \( \circ \) denotes sequence concatenation. Since there are only \( n \) generators, it is clear that the maximum multi-index size of semigroup elements is \( n \). Moreover, these \( n \) symbols can appear in \( n! \) permutations so that the order of the semigroup is \( n! + 1 \).

Defining (vector) addition and real scalar multiplication on the semigroup yields the graded semigroup algebra \( \mathbb{R} \Omega_n \) of dimension \( n! + 1 \).

Extending the nilpotent adjacency matrix construction to \( \mathbb{R} \Omega_n \) allows one to enumerate (list) all paths and cycles in a finite graph by considering powers of the matrix. The associated tree structure underlying the cycle/path enumeration problem is automatically “pruned” by the inherent properties of the algebra. This sort of pruning is all that is required for obtaining paths on minimal numbers of hops. However, additional pruning is useful when path minimality is based on path weights other than the number of hops.

In the OC approach, a path \( u = (u_1, \ldots, u_k) \) of total weight \( x \in \mathbb{R} \) will be represented in \( \mathbb{R} \Omega_n \), by an element of the form \( e^x \omega_u \). The concatenation of this path with another path \( v = (v_1, \ldots, v_l) \) of weight \( y \in \mathbb{R} \) is then represented by the product

\[
 e^x \omega_u e^y \omega_v = \begin{cases} 
e^x+y \omega_{uv} & \text{if } u \circ v \text{ is a path,} \\
o & \text{otherwise.}
\end{cases}
\]  

(3)

To facilitate weight-based pruning, additional operations are defined on the graded semigroup algebra \( \mathbb{R} \Omega_n \). Specifically, thresholded multiplications are defined on the algebra by linear extension of the following:

\[
a_u \omega_u \circ_T a_v \omega_v = \begin{cases} 
0 & u \cap v = \emptyset \text{ or } ab > T, \\
(ab) \omega_{uv} & \text{otherwise.}
\end{cases}
\]  

(4)

In this manner, the concatenation of two paths is removed from consideration if the combined weight of the two paths exceeds the threshold value.

3. SHORTEST PATH ALGORITHMS

As in the work of Cruz-Sánchez, et al. [1], each frame of the system is a graph \( G = (V, E) \) composed of a set \( V \) of vertices (or nodes), and a set \( E \) of edges. Letting \( g \) and \( s \) denote the numbers of gateways and satellites, respectively, define

\[
GW = \{1, 2, \ldots, g\}
\]  

(5)

and let \( S = \{g + 1, g + 2, \ldots, g + s\} \)

(6)

to be the set of vertices representing gateways, and define the set \( V = GW \cup S \).

(7)

This vertex set is fixed for all graphs of the routing process. By design, the satellite constellation avoids the use of inter-satellite links, so that communication is only possible between satellites and gateways. Given a satellite-gateway pair of vertices, \((i, j)\) \( \in GW \times S \), define the set of visibility windows

\[
W_{ij} := \{[a_1, b_1]_{ij}, [a_2, b_2]_{ij}, \ldots, [a_k, b_k]_{ij} : i \in GW, j \in S\}
\]  

(8)

as the collection of time intervals during which gateway \( i \) is visible to and from satellite \( j \).

Graph topology (i.e., the edge set of \( G \)) is thereby determined for the frame valid at time \( t \) by

\[
E_t = \{(i, j) \in GW \times S : \exists [a_t, b_t]_{ij} \in W_{ij}, a_t \leq t \leq b_t\}.
\]  

(9)

A hop-minimal path from initial vertex \( v_0 \) to terminal vertex \( v_k \) is a sequence \( (v_0, v_1, \ldots, v_k = v_k) \) of \( k+1 \) vertices such that \((v_i, v_{i+1}) \in E(G)\) for each \( i = 0, \ldots, k - 1 \) and there exists no path \( v_0 \rightarrow v_i \) containing fewer vertices. The length of the path \((v_0, v_1, \ldots, v_k = v_k)\) is \( k \).

3.1 The OC algorithms

For convenience, Dirac notation is adopted to specify row and column vectors. In particular, when \( A \) is the adjacency matrix of a graph, \( \langle v_i | A \rangle \) denotes the row of \( A \) associated with vertex \( v_i \), while \( A | v_j \rangle \) denotes the column of \( A \) associated with vertex \( v_j \).
Given a graph $G = (V, E)$ on $n = |V|$ vertices, the nilpotent adjacency matrix of $G$ over $\mathbb{R}\Omega_n$ is defined by

$$
\langle v_i | \Psi | v_j \rangle := \begin{cases} 
\omega_j & \text{if } (v_i, v_j) \in E, \\
0 & \text{otherwise.}
\end{cases}
$$

(10)

Straightforward induction establishes the path enumeration theorem.

**Theorem 3.1.** Let $v_0$ and $v_\infty$ denote any distinct pair of vertices in the graph $G = (V, E)$ with nilpotent adjacency matrix $\Psi$ over $\mathbb{R}\Omega_{|V|}$. Then,

$$
\omega_{v_0}(v_0|^k|v_\infty) = \sum_{k\text{-paths } u=(v_0, \ldots, v_\infty)} \omega_u.
$$

(11)

In particular, all paths of length $k$ with initial vertex $v_0$ and terminal vertex $v_\infty$ are enumerated by the multi-indices of the terms in the summation.

An immediate consequence of Theorem 3.1 is that the algorithm below enumerates all hop-minimal paths from vertex $v_0$ to vertex $v_\infty$ in a fixed finite graph $G$.

```
proc HopMin[\Psi, v_0, v_\infty]
% Hop-minimal enumeration algorithm
% Enumerate all hop-minimal paths from v_0 to v_\infty
\langle \xi \rangle = \omega(\xi) \langle v_0 | \Psi
\While (\langle \xi | v_\infty \rangle = 0 \And \langle \xi \rangle \neq (0))
\langle \xi \rangle = \langle \xi \rangle \Psi
\EndWhile
Return [\langle \xi | v_\infty \rangle]
```

A minimum weight hop-minimal path from initial vertex $v_0$ to terminal vertex $v_\infty$ is a hop-minimal path whose additive weight is minimal among all such paths. Modifying the construction of the nilpotent adjacency matrix makes enumeration of these paths possible.

Let $d_{ij} \in \mathbb{R}$ denote the additive weight of edge $(v_i, v_j) \in E$. Then, the weighted nilpotent adjacency matrix of $G$ over $\mathbb{R}\Omega_n$ is defined by

$$
\langle v_i | \Psi^k | v_j \rangle := \begin{cases} 
\exp(d_{ij})\omega_j & \text{if } (v_i, v_j) \in E, \\
0 & \text{otherwise.}
\end{cases}
$$

(12)

For arbitrary path $u$, denote the total additive weight of $u$ by

$$
\text{wt}(u) := \sum_{\text{edges } (v_i, v_j) \text{ of } u} d_{ij}.
$$

(13)

**Corollary 3.2.** Let $v_0$ and $v_\infty$ denote any distinct pair of vertices in the graph $G = (V, E)$ with weighted nilpotent adjacency matrix $\Psi$ over $\mathbb{R}\Omega_{|V|}$. Then,

$$
\omega_{v_0}(v_0|^k|v_\infty) = \sum_{k\text{-paths } u=(v_0, \ldots, v_\infty)} \exp(\text{wt}(u))\omega_u.
$$

(14)

Let $\xi = \sum_{u \in \Omega_n} a_u \omega_u$ denote an arbitrary element of the semigroup algebra $\mathbb{R}\Omega_n$. Define

$$
\inf(\xi) = \begin{cases} 
\inf\{a_u : u \in \Omega_n, a_u \neq 0\} & \text{if } \xi \neq 0, \\
0 & \text{otherwise.}
\end{cases}
$$

(15)

and for any vector $\langle \varphi \rangle = (\varphi_1, \ldots, \varphi_n) \in (\mathbb{R}\Omega_n)^n$ define its component-wise vector extension by

$$
\inf(\varphi) := \inf(\varphi_1), \ldots, \inf(\varphi_n).
$$

(16)

Further, define the mapping $\rho : \Omega_n \rightarrow \mathbb{R}$ by

$$
\rho(\xi) = \sum_{a_u = \inf(\xi)} a_u \omega_u.
$$

(17)

and define its component-wise vector extension by

$$
\eta(\varphi) := (\rho(\varphi_1), \ldots, \rho(\varphi_n)).
$$

(18)

for any vector $\langle \varphi \rangle = (\varphi_1, \ldots, \varphi_n) \in (\mathbb{R}\Omega_n)^n$.

The algorithm below enumerates all minimum weight hop-minimal paths from vertex $v_0$ to vertex $v_\infty$ in a fixed finite graph $G$.

```
proc MinWtHopMin[\Psi, v_0, v_\infty]
% Hop-minimal enumeration algorithm
% Enumerate all minimum weight hop-minimal paths
% from v_0 to v_\infty using weighted nilpotent
% adjacency matrix \Psi
\langle \xi \rangle = \omega(\xi) \langle v_0 | \Psi
\While (\langle \xi | v_\infty \rangle = 0 \And \langle \xi \rangle \neq (0))
\langle \xi \rangle = \langle \xi \rangle \Psi
\EndWhile
Return [\rho(\langle \xi | v_\infty \rangle)]
```

A minimal weight hop-minimal path from initial vertex $v_0$ to terminal vertex $v_\infty$ is a path whose additive weight is minimal among all such paths. In the remainder of this paper, paths represent data transmission, and the weight of a path is defined as the path’s end-to-end delay. The problem being considered is the enumeration of minimal delay paths in graph processes. Precomputed routing in a store-and-forward satellite constellation is just one example of this problem.

### 3.2 Minimal delay paths in graph processes

Given a fixed collection of vertices $V$, a graph process on $V$ is defined as a sequence of graphs $G_t = (V, E_t)$.

In particular, the edge weights considered for these graphs will represent transition time delays between pairs of adjacent vertices. For example, if the graph’s vertices represent communication nodes, the weight of an edge represents the time required to message transmission between the two nodes incident with that edge.

The minimal delay path problem is stated thusly: Given a source-target pair of vertices $v_0$ and $v_\infty$ in a graph process, enumerate all minimal delay paths, i.e., paths of minimum end-to-end delay, from $v_0$ to $v_\infty$ in the process.

Each frame of the process similarly has an existence time window of the form $[a_t, b_t] \subset \mathbb{R}$ during which the graph’s
topology is fixed. The total weight of any path thus depends on lengths of intervals of frames spanned by the path.

Once a path of total weight $T$ from $v_0$ to $v_\infty$ has been found, there is no need to continue enumerating paths whose weights (or partial weights) exceed $T$. To this end, one passes to the thresholded product $\circ_T$ defined by (4).

For arbitrary $\langle \varphi \rangle = (\varphi_1, \ldots, \varphi_n) \in (\mathbb{R} \Omega_n)^n$, the anti-indicator map $\chi : (\mathbb{R} \Omega_n)^n \to \{0, 1\}$ is defined by its component-wise action

$$\chi(\varphi)_j = \begin{cases} 1 & \text{if } \varphi_j = 0, \\ 0 & \text{otherwise}. \end{cases}$$

(19)

Given a vector $\langle \tau \rangle \in \mathbb{R}_+^n$ of nonnegative reals, the componentwise thresholded inner product is defined on vectors of $(\mathbb{R} \Omega_n)^n$ by

$$\langle \varphi \rangle \circ_T \langle \psi \rangle := (\varphi_1 \circ_1 \psi_1, \ldots, \varphi_n \circ_n \psi_n).$$

(20)

Let $\ast$ denote componentwise vector product.

Given a weighted nilpotent adjacency matrix $\Psi$ over $\mathbb{R}_+^n$ associated with one frame of a graph process, the following algorithm is used to enumerate all minimal weight paths in that frame emanating from vertex $v_0$ and having total weight not exceeding $T$.

```
proc ThresholdEnumAll[\Psi, \{0\}, T]
% Enumerate all minimal paths from \(v_0\) to \(v_\infty\)
% having weight not exceeding \(T\) in the frame
% corresponding to \(\Psi\)
⟨τ⟩ := (T,...,T)
⟨ξ0⟩ := ω_{v0} ⟨0⟩
⟨ξ⟩ := ⟨ξ0⟩ \circ_T Ψ
τ := ⟨τ⟩ \ast χ(⟨ξ⟩)
While ⟨ξ⟩ \neq ⟨ξ0⟩
⟨ξ0⟩ := ⟨ξ⟩
⟨ξ⟩ := ρ(⟨ξ0⟩ \circ_T Ψ)
⟨τ⟩ := ⟨τ⟩ \ast χ(⟨ξ⟩) + (inf ⟨ξ⟩)
EndWhile
Return [⟨ξ⟩]
```

Note that the role of $\langle \tau \rangle$ in the algorithm is to maintain component-wise thresholds for thresholded multiplication. In this way, each new path of lesser weight establishes a new threshold, and paths leading to greater weights are automatically pruned by subsequent multiplications. Minimum weight paths are preserved in each iteration, and the algorithm is iterated until no new paths are formed.

The ThresholdEnumAll algorithm returns a vector containing all existing minimal paths from initial vertex $v_0$. The terminal vertices of these minimal paths can then serve as intermediate steps of minimal paths in subsequent frames of the process. However, one must always allow the possibility of new paths from $v_0$ being initiated in subsequent frames, rather than just extending these intermediate paths. Furthermore, the last frame considered will always be the first frame in which a path from $v_0$ to $v_\infty$ is completed by ordering of frames and time thresholds.

Letting $t,t'$ denote the absolute times of successive changes in graph topology, let $\Psi_t$ denote the weighted nilpotent adjacency matrix of the corresponding frame, i.e., the graph whose topology is fixed during the interval $[t,t']$.

```
proc DynamicMinimalPaths[\Psi_0, v_\infty]
% Enumerate all minimal paths from \(v_0\) to \(v_\infty\)
% in the frame in which a path from \(v_0\) to \(v_\infty\) is completed by ordering of frames and time thresholds.
% Letting \(t,t'\) denote the absolute times of successive changes in graph topology, let \(\Psi_t\)
% denote the weighted nilpotent adjacency matrix of the corresponding frame, i.e., the graph whose
% topology is fixed during the interval \([t,t']\).
\(\{0, t'\} := \{\text{first frame interval }\}
\langle \Upsilon_0 \rangle := \text{ThresholdEnumAll}[\Psi_0, \omega_{v0} ⟨0⟩, e^t]
\tau_\infty := (\inf \langle \Upsilon_0 \rangle | v_\infty)
While \(\tau_\infty = 0\)
\langle \Upsilon | v_\infty \rangle := e^t \langle \Upsilon \rangle \ast χ(⟨\Upsilon⟩)
\{t, t'\} := \{\text{next frame }\}
\langle \Upsilon' \rangle := \text{ThresholdEnumAll}[\Psi_t, e^t \omega_{v0} ⟨0⟩, e^t]
\langle \Upsilon'' \rangle := \text{ThresholdEnumAll}[\Psi_t, ⟨\Upsilon_0⟩, e^t]
\langle \Upsilon \rangle := (\cdots, η(⟨\Upsilon'⟩ | v_\infty), ⟨\Upsilon''⟩ | v_\infty), \cdots
\tau_\infty := (\inf \langle \Upsilon \rangle | v_\infty)
\(t := t'\)
EndWhile
Return [⟨\Upsilon⟩]
```

Scalar coefficients of terms in the algebra correspond to minimal end-to-end delays in paths computed. The key to the approach is that when passing from one frame to the next, all scalar coefficients are reset to reflect the starting time of the latest frame. Subsequently, additive weights are accumulated based on the number of hops transpiring in the current frame. When one or more paths reach the terminal vertex $v_\infty$, the path with minimal scalar coefficient is returned, whereupon computing the natural logarithm reveals the total end-to-end delay of the path.

4. THE RESULTS

The operator calculus algorithms were implemented in Mathematica on a 2.4 GHz MacBook Pro with 4 GB of 667 MHz DDR2 SDRAM running Mathematica 8 for MAC OS X.

In order to compare the results of the OC algorithm with the heuristics-based algorithmic approach, data from the simulations considered in [1] are used. The graph process (satellite constellation) being considered consists of 21 satellites and 109 gateways. The vertex set is represented by integers $\{1, \ldots, 109\}$ for gateways and $\{110, \ldots, 130\}$ for satellites.

An ordered collection of 11, 550 distinct interval endpoints $(t_1, t_2, \ldots, t_{11,550})$ from precomputed satellite visibility windows was used to generate the sequence of frames. For each interval $[t_1, t_{i+1}]$, a frame consists of a fixed graph with weighted nilpotent adjacency matrix $\Psi_t$.

Figure 1 details a collection of minimal paths for the satellite constellation precomputed routing problem. Each row
Figure 1: A sample of randomly chosen source-target gateway pairs with minimal (shortest end-to-end delay) paths found.
represents a randomly selected source-target gateway pair. The columns then contain minimal paths found using the OC algorithm, the corresponding minimum end-to-end delay, the number of frames required by the minimal path, the time of computation in seconds, and the best existing minimum end-to-end delay.

The Mathematica implementation of the OC algorithm includes a path validation procedure to ensure that the OC algorithm does in fact return a valid path for each source-target gateway pair. This validation is done by looking up visibility windows associated with successive steps of paths found to verify that transitions are allowed in the process.

In the implementation of the OC approach, each hop is assumed to require 10 ms of time. This differs from the heuristics-based algorithm, which assumes no delay in individual hops. With these facts in mind, the results of the third row of Figure 1 are in agreement, with the OC delay 177,816 being equal to the BER delay 177,806 with one hop occurring in the final frame of consideration.

Figure 2 gives statistical analysis of minimal end-to-end delays found over 727 trials with randomly selected source-target gateway pairs.

```
Minimum end-to-end delay summary
---------------------------------
Minimum: 20.
First quartile: 14 018.
Median: 472 264.
Third quartile: 1.38081 ´ 10^6
Maximum: 8.6455430000 ´ 10^6
---
Mean: 1.1642 ´ 10^6
---
Standard deviation: 1.9257 ´ 10^6
```

**Figure 2: Analysis of shortest end-to-end delays found in 727 trials with OC approach.**

At the core of the heuristics-based algorithm is construction of a tree whose root is the initial gateway. In order to prevent combinatorial explosion, the depth of the constructed tree is limited. Consequently, many source-target pairs have the default BER delay value $2 \times 10^9$, indicating that no path was found by the heuristics-based algorithm. In fact, in 298 of the 727 trials, the heuristics-based algorithm found no path.

Figure 3 depicts the end-to-end delays obtained by the OC algorithm when no corresponding BER paths were found.

```
Figure 3: End-to-end delays from OC algorithm among paths having no BER path data.
```

**Figure 4: Differences BER − OC between best existing minimum delays and OC delays among source-target pairs having BER path data.**

The small values plotted in Figure 4 indicate the agreement between BER data and results from the OC algorithm, particularly in cases of paths found in the initial frame. Cases of disagreement are seen in greater detail in Figure 5.

```
Figure 4: Differences BER − OC between best existing minimum delays and OC delays among source-target gateway pairs.
```

In Figure 5, minimum end-to-end delays obtained from the OC approach are compared with best existing paths for which two criteria are satisfied: 1. path data exists in the BER data, and 2. OC and BER do not agree on the paths found. These criteria were satisfied in 59 of the 727 trials. Horizontal dashes indicate BER and red asterisks represent OC results. As expected, the minimal delay obtained by OC is shorter than that of BER in all cases.

Figure 6 represents the (sorted) differences of the delays (BER−OC) seen in Figure 5, while Figure 7 is a Mathematica-generated statistical analysis of the minimum-delay differences.

```
Figure 6 indicates a relatively small number of “bad” running times. These running times are associated with a relatively small number of “bad” target gateways, as seen in Figure 9. In fact, six targets account for the 33 longest running times of the OC algorithm. These running times correspond to targets in geographical regions lacking satellite coverage for a large number of initial frames.
```

5. CONCLUSION

In summary, there are clear advantages to the OC approach to computing minimal paths as well as numerous avenues for further exploration.
Figure 5: Delay comparisons where BER results exist, and $BER \neq OC$.

Figure 6: Delay differences: BER results exist, and $BER \neq OC$.

Minimum end-to-end delay comparison
---
59 delay differences found among existing paths.
Maximum delay difference: $1.38719 \times 10^5$
Third quartile: 502809.
Median delay difference: 487098.
First quartile: 214131.
Minimum delay difference: 24200.0000
---
Mean delay difference: 483467.
---
Standard deviation: 388310.

Figure 7: Analysis of minimum delay differences among existing paths.

Figure 8: OC algorithm running time in seconds over 727 trials in descending order.

Figure 9: Algorithm running times for source-target pairs.

Figure 10: OC algorithm running time in seconds versus number of frames considered after initial connectivity over 727 trials.
Maximum computation time: 1308.38
Third quartile: 58.0357
Median computation time: 10.8021
First quartile: 2.16793
Minimum computation time: 0.389963
---
Mean computation time: 68.0055
---
Standard deviation: 169.671

Figure 11: Analysis of sample computation times of OC algorithm over 727 trials.

5.1 Advantages of the operator calculus approach

- Exactness. Unlike previous heuristic-based algorithms, the operator calculus approach gives exact results; indeed, the minimal end-to-end delay paths obtained are optimal. In fact, in 298 of the 727 trials, the heuristics-based algorithm found no path, while the OC algorithm always obtained a minimal delay path. Moreover, as is the case in the heuristic approach (cf. [1]), the running time of the exact algorithm is linear in terms of the number of frames considered in finding the minimal path.

- Flexibility. The operator calculus approach can be used to compute minimal paths for any graph sequence characterized by a sequence of matrices. Modifying delays associated with hops is straightforward, allowing even random variables to be used. Graph connectivity and hop delays can easily be made time- and/or environment-dependent.

5.2 Ongoing work

- Detailed analysis of OC algorithm time and storage complexity and side-by-side comparisons with heuristics-based approaches are in progress.

- Other satellite constellation designs can be considered, allowing not only satellite-gateway connections, but satellite-satellite interlinks as well. Different traffic models can also be considered.

- Multi-constrained Quality of Service (QoS) problems can be modeled using sequences of graphs and can also benefit from the operator calculus approach.

6. REFERENCES


