Operator Calculus Approach to Minimal Paths: Precomputed routing in a Store and Forward Satellite Constellation

Hugo Cruz-Sanchez, Stacey Staples, René Schott, Ye-Qiong Song

Operator Calculus Approach to Minimal Paths: Precomputed routing in a Store and Forward Satellite Constellation

Hugo Cruz-Sánchez
INRIA
615, rue du Jardin botanique,
54600 Villers-lès-Nancy, FRANCE
Email: hugo.cruzsanchez@inria.fr

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

René Schott, Ye-Qiong Song
LORIA - University of Lorraine
Campus Scientifique, B.P. 239,
54500 Vandœuvre-lès-Nancy, FRANCE
Email: song@loria.fr, schott@loria.fr

***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 proposed here is applied to the problem of precomputed routing in a store-and-forward (S&F) 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 a greedy 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. In some cases, in which the tree pruning algorithm did not find a solution, the operator calculus does. These results correspond to a one-single constraint case considering the end-to-end delay as the cost of the links, if the case of multi constraints is considered (e.g. bandwidth, rapidity,...) the operator calculus approach can be similarly used.**

I. Introduction

Operator calculus (OC) methods on graphs have been developed in a number of works by Schott and Staples (cf. [6], [7], [9], [10]). 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 S&F 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. The OC approach leads, in fact, to the design of a "built-in" algorithm.

A. 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. [4]. The system considered in that work offers S&F services including messaging, tracking, and monitoring. Periods of satellites' visibility for targeted areas were precomputed using a visibility algorithm [1].

The system considered here is not unlike the Orbcomm satellite system–a store-and-forward satellite constellation for messaging based on relaying messages over a cellular network when it is available and on satellite relays in remote locations or out-of-coverage situations. Orbcomm satellites are not equipped with inter-satellite link technology, and any routing must be implemented by using GW-Sat-GW sequences of links.

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. As the satellites do not use inter-satellite links (ISL) the satellites are interconnected by gateways relaying messages, in this way only gateway-satellite connections are allowed in the model discussed. The proposed algorithm in [4] to compute minimal paths is inspired from the theory

framework developed in [5]. The network formed by the satellites and gateways is constantly evolving in time. The topology changes along the time and as the satellites and gateways are not connected permanently the connectivity of the network is strongly time-dependent. The minimal path computing must consider the existence of sequential networks as the representation of the topology dynamics. The time domain is included in the model by considering the time windows connections which are the periods during which a connection is able to be used.

The problem of the satellite system with time windows is similar to those belonging to the dynamic shortest path problem in transportation field. The dynamic shortest path problem is a consequence of the inclusion of the "time" factor in a static network. The problem was firstly addressed by Cooke and Helsey in 1966 [3], (for a better understanding of the transportation dynamic problems and its modeling see [2]).

The cost of routes may be composed by several parameters as the waiting time over the nodes, the quantity of memory used for storage, the energy consumed at each node, etc. All these costs have more importance on the space segment as each satellite is very costly and has limited resources.

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 [4]. The computed paths in this work are single-constraint and correspond to those with the minimum end-to-end delay. 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. The algorithm has been adapted to find optimal routes in a single constraint point of view over different routing metrics. Greedy Algorithm Results (GAR) from the heuristics-based algorithm are used here as benchmarks for comparison of minimal-delay paths obtained from the single-constraint OC approach.

II. 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*. A full path is computed as the sequence of paths at each frame.

A. Single Constraint Operator Calculus

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

$$\omega_{\mathbf{u}} := \omega_{u_1}\omega_{u_2}\cdots\omega_{u_k}\ . \tag{1}$$

For fixed positive integer n, consider the alphabet $\Omega := \{\omega_i : 1 \le i \le n\}$. One thereby obtains a freely-generated semigroup Ω_n whose elements are the empty string, $\omega_\emptyset$, along with all *finite words* on distinct generators, i.e., finite sequences of distinct symbols from the alphabet Ω, if one defines the multiplication of two such sequences by

$$\omega_{\mathbf{u}}\omega_{\mathbf{v}} = \begin{cases} \omega_{\mathbf{u}.\mathbf{v}} & \text{if } \mathbf{u}\cap\mathbf{v} = \emptyset, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Here, . denotes sequence concatenation. Since there are only n generators, it is clear that the maximum multi-index size of semigroup elements is n. Moreover, each multi-index of size k can appear in $k!$ permutations so that the order of the semigroup is $|\Omega_n| = \sum_{k=0}^{n}(n)_k$, where $(n)_k$ denotes the falling factorial. Defining (vector) addition and real scalar multiplication on this semigroup yields the graded semigroup algebra $\mathbb{R}\Omega_n$ whose dimension is $|\Omega_n|$.

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 $\mathbf{u} = (u_1, \ldots, u_k)$ of total weight $x \in \mathbb{R}$ will be represented in $\mathbb{R}\Omega_n$ by an element of the form $\nu^x\,\omega_{\mathbf{u}}$. The concatenation of this path with another path $\mathbf{v} = (v_1, \ldots, v_\ell)$ of weight $y \in \mathbb{R}$ is then represented by the product

$$\nu^x\omega_{\mathbf{u}}\,\nu^y\omega_{\mathbf{v}} = \begin{cases} \nu^{x+y}\omega_{\mathbf{u}.\mathbf{v}} & \text{if } \mathbf{u}.\mathbf{v} \text{ is a path,} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

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:

$$a_{\mathbf{u}}\omega_{\mathbf{u}} \diamond_T a_{\mathbf{v}}\omega_{\mathbf{v}} = \begin{cases} 0 & \mathbf{u}\cap\mathbf{v} \neq \emptyset \text{ or } ab > T, \\ (ab)\omega_{\mathbf{u}.\mathbf{v}} & \text{otherwise.} \end{cases} \tag{4}$$

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

B. Multi Constraints Operator Calculus

In the general case, c is a vector of constraints and (3) becomes

$$\nu^x\omega_{\mathbf{u}}\,\nu^y\omega_{\mathbf{v}} = \begin{cases} \nu^{x+y}\omega_{\mathbf{u}.\mathbf{v}} & \text{if } \mathbf{u}.\mathbf{v} \text{ is a path of weight } c, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

where $c = (c_1, ..., c_m)$ and $x = (x_1, ..., x_m)$ $y = (y_1, ..., y_m)$. See [8] for details about the algebraic framework.
Below we focus on the single constraint case for pedagogical reasons (and lack of space).

III. Shortest Path Algorithms

As in the work of Cruz-Sánchez, et al. [4], each frame of the system is a graph $G = \langle V, E\rangle$ 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

$$GW = \{1, 2, \ldots, g\} \tag{6}$$

to be the set of vertices representing gateways, and define the set

$$S = \{g + 1, g + 2, \ldots, g + s\} \tag{7}$$

to be the set of vertices representing satellites. The vertex set of graph G is therefore

$$V = GW \cup S. \tag{8}$$

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

$$W_{ij} := \{[a_1, b_1]_{ij}, [a_2, b_2]_{ij}, \ldots, [a_k, b_k]_{ij} : i \in GW, j \in S\} \tag{9}$$

as the collection of time intervals during which gateway i is visible to and from satellite j. These intervals are the time windows associated to each feasible communication link. As the proposed system is not periodic the time windows do not follow any definition rule, the values of the intervals are computed in advance and the set of intervals may be enough to compute routes from and to any gateway.

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

$$E_t = \{(i, j) \in GW \times S : \exists [a_\ell, b_\ell]_{ij} \in W_{ij}, a_\ell \le t \le b_\ell\}. \tag{10}$$

A *hop-minimal* path from initial vertex v_0 to terminal vertex v_t is a sequence $(v_0, v_1, \ldots, v_k = v_t)$ 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 $v_0 \to v_t$ containing fewer vertices. The *length* of the path $(v_0, v_1, \ldots, v_k = v_t)$ is k.

A. The Operator Calculus Algorithms

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

$$\langle v_i|\Psi|v_j\rangle := \begin{cases} \omega_j & \text{if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Straightforward induction establishes the *path enumeration theorem*.

Theorem 1 (Path enumeration theorem): Let v_0 and v_∞ denote any distinct pair of vertices in the graph $G = \langle V, E\rangle$ with nilpotent adjacency matrix Ψ over $\mathbb{R}\Omega_{|V|}$. Then,

$$\omega_{v_0}\langle v_0|\Psi^k|v_\infty\rangle = \sum_{k\text{-paths } \mathbf{u}=(v_0,\ldots,v_\infty)} \omega_{\mathbf{u}}. \tag{12}$$

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

An immediate consequence of Theorem 1 is that Algorithm 1 enumerates all hop-minimal paths from vertex v_i to vertex v_t in a fixed finite graph G.

Algorithm 1 HopMin[Ψ, v_0, v_∞]

Require: Ψ, v_0, v_∞
Ensure: Enumerate all hop-minimal paths from v_0 to v_∞
  $\langle\xi| = \omega_{\{v_0\}}\langle v_0|\,\Psi$
  while $(\langle\xi|v_\infty\rangle = 0 \wedge \langle\xi| \neq \langle 0|)$ **do**
    $\langle\xi| = \langle\xi|\,\Psi$
  end while
  return $[\langle\xi|v_\infty\rangle]$

A *minimum weight hop-minimal path* from initial vertex v_0 to terminal vertex v_∞ 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

$$\langle v_i|\Psi|v_j\rangle := \begin{cases} \nu^{d_{ij}}\,\omega_j & \text{, if } (v_i, v_j) \in E, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

For arbitrary path $\mathbf{u}$, denote the total additive weight of $\mathbf{u}$ by

$$\mathrm{wt}(\mathbf{u}) := \sum_{\text{edges } (v_i, v_j) \text{ of } \mathbf{u}} d_{ij}. \tag{14}$$

Corollary 1: Let v_0 and v_∞ denote any distinct pair of vertices in the graph $G = \langle V, E\rangle$ with weighted nilpotent adjacency matrix Ψ over $\mathbb{R}\Omega_{|V|}$. Then,

$$\omega_{v_0}\langle v_0|\Psi^k|v_\infty\rangle = \sum_{k\text{-paths } \mathbf{u}=(v_0,\ldots,v_\infty)} \nu^{\mathrm{wt}(\mathbf{u})}\omega_{\mathbf{u}}. \tag{15}$$

Let $\xi = \sum_{\mathbf{u}\in\Omega_n} a_{\mathbf{u}}\,\omega_{\mathbf{u}}$ denote an arbitrary element of the semigroup algebra $\mathbb{R}\Omega_n$. Define

$$\inf(\xi) = \begin{cases} \inf\{a_{\mathbf{u}} : \mathbf{u} \in \Omega_n, a_{\mathbf{u}} \neq 0\} & \text{if } \xi \neq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

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

$$\langle\inf\varphi| := (\inf(\varphi_1), \ldots, \inf(\varphi_n)). \tag{17}$$

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

$$\rho(\xi) = \sum_{\substack{\mathbf{v}\in\Omega_n:\\ a_{\mathbf{v}}=\inf(\xi)}} a_{\mathbf{v}}\omega_{\mathbf{v}}, \tag{18}$$

and define its component-wise vector extension by

$$\eta\langle\varphi| := (\rho(\varphi_1), \ldots, \rho(\varphi_n)) \tag{19}$$

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

Algorithm 2 enumerates all minimum weight hop-minimal paths from vertex v_0 to vertex v_t in a fixed finite graph G.

Algorithm 2 MinWtHopMin[Ψ, v_0, v_∞]

Require: Ψ, v_0, v_∞
Ensure: Enumerate all minimum weight hop-minimal paths from v_0 to v_∞ using weighted nilpotent adjacency matrix Ψ
  $\langle\xi| = \omega_{\{v_0\}}\langle v_0|\,\Psi$
  while $(\langle\xi|v_\infty\rangle = 0 \wedge \langle\xi| \neq \langle 0|)$ **do**
    $\langle\xi| = \langle\xi|\,\Psi$
  end while
  return $[\rho(\langle\xi|v_\infty\rangle)]$

A *minimal weight path* from initial vertex v_0 to terminal vertex v_∞ 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 S&F satellite constellation is just one example of this problem.

B. 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 $G_t = \langle V, E_t\rangle$.

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_∞ in a graph process, enumerate all minimal delay paths, i.e., paths of minimum end-to-end delay, from v_0 to v_∞ in the process.

Each frame of the process similarly has an existence time window of the form $[a_\ell, b_\ell] \subset \mathbb{R}$ 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.

It is assumed *a priori* that all frame lengths exceed all single hop delays. In the satellite problem this restriction is not significant, as visibility windows are typically measured in minutes, while hop delays are in milliseconds.

Once a path of total weight T from v_0 to v_∞ 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 $\diamond_T$ defined by (4).

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

$$\overline{\chi}(\langle\varphi|)_j = \begin{cases} 1 & \text{if } \varphi_j = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

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

$$\langle\varphi| \diamond_\tau |\psi\rangle := (\varphi_1 \diamond_{\tau_1} \psi_1, \ldots, \varphi_n \diamond_{\tau_n} \psi_n). \tag{21}$$

Let $*$ denote componentwise vector product.

Given a weighted nilpotent adjacency matrix Ψ over $\mathbb{R}\Omega_n$ associated with one frame of a graph process, Algorithm 3 is used to enumerate all minimal weight paths in that frame emanating from vertex v_0 and having total weight not exceeding$T$.

Algorithm 3 ThresholdEnumAll[$\Psi, \langle v_0|, T$]

Require: $\Psi, \langle v_0|, T$
Ensure: Enumerate all minimal paths with initial vertex v_0 having weight not exceeding T in the frame corresponding to Ψ
  $\langle\tau| := (T, \ldots, T)$
  $\langle\xi_0| := \omega_{\{v_0\}}\langle v_0|$
  $\langle\xi| := \langle\xi_0| \diamond_\tau \Psi$
  $\tau = \langle\tau| * \overline{\chi}(\xi) + \langle\inf\xi|$
  while $\langle\xi| \neq \langle\xi_0|$ **do**
    $\langle\xi_0| := \langle\xi|$
    $\langle\xi| := \rho(\langle\xi_0| \diamond_\tau \Psi)$
    $\langle\tau| = \langle\tau| * \overline{\chi}(\langle\xi|) + \langle\inf\xi|$
  end while
  return $[\langle\xi|]$

Note that the role of $\langle\tau|$ 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.

Algorithm 3 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_∞ is completed by ordering of frames and time thresholds.

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

Algorithm 4 enumerates minimal weight paths from initial vertex v_0 to terminal vertex v_∞ across a sequence of frames associated with nilpotent adjacency matrix sequence (Ψ_t).

Algorithm 4 DynamicMinimalPaths $[v_0, v_\infty]$

Require: v_0, v_∞
Ensure: Enumerate all minimal paths from v_0 to v_∞
  $[0, t'] := \{\text{first frame interval }\}$
  $\langle \Upsilon| = \text{ThresholdEnumAll}[\Psi_0, \omega_{v_0}\langle v_0|, e^{t'}]$
  $t_\infty = \langle \inf \Upsilon | v_\infty \rangle$
  while $t_\infty = 0$ **do**
    $\langle \Upsilon_0| = e^{t'}\langle \Upsilon| * \chi(\langle \Upsilon|)$
    $[t, t'] := \{\text{next frame }\}$
    $\langle \Upsilon'| = \text{ThresholdEnumAll}[\Psi_t, e^t \omega_{v_0}\langle v_0|, e^{t'}]$
    $\langle \Upsilon''| = \text{ThresholdEnumAll}[\Psi_t, \langle \Upsilon_0|, e^{t'}]$
    $\langle \Upsilon| = (\cdots, \eta(\langle \Upsilon'|v_i\rangle, \langle \Upsilon''|v_i\rangle), \cdots)$
    $t_\infty = \langle \inf \Upsilon | v_\infty \rangle$
    $t := t'$
  end while
  return $[\langle \Upsilon | v_\infty \rangle]$

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_∞, the path with minimal scalar coefficient is returned, whereupon computing the natural logarithm reveals the total end-to-end delay of the path.

IV. 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 OSX.

In order to compare the results of the OC algorithm with the heuristics-based algorithmic approach, data from the simulations considered in [4] 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 $(t_0, t_1, \ldots, t_{11,550})$ from precomputed satellite visibility windows was used to generate the sequence of frames. For each interval $[t_i, t_{i+1}]$, a frame consists of a fixed graph with weighted nilpotent adjacency matrix Ψ_t.

Figure 1 details a collection of minimal paths for the satellite constellation precomputed routing problem. Each row represents a randomly selected source-target gateway pair. The columns 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.

	Path	End-to-end	Frames	CompTime	GAR Delay
1	{109, 124, 51, 113, 11, 128, 25, 115, 24, 125, 79}	1.5739790000×10^6	153	112.151	$2.\times10^9$
	{100, 126, 12, 110, 31, 117, 33}	254811.0000	1	1.14431	$2.\times10^9$
	{7, 114, 6, 111, 35}	177816.0000	16	17.5843	177806
	{42, 119, 12, 126, 11, 116, 40, 113, 51}	74613.0000	8	7.1625	74603
	{28, 119, 12, 126, 96}	40.	1	0.536073	0
	{68, 129, 1, 122, 59}	1.4359830000×10^6	180	66.6282	1.43597×10^6
	{27, 118, 25, 126, 13, 110, 2, 129, 33, 117, 5, 120, 7, 114, 38}	713722.0000	1	2.24743	$2.\times10^9$
	{75, 123, 95}	20.	1	0.532196	0
	{80, 126, 11, 116, 40}	40.	1	0.686869	0
	{69, 129, 1, 110, 12, 126, 98}	60.	1	0.685782	0
	{67, 128, 15, 113, 2, 116, 33, 119, 5, 110, 7, 117, 37, 120, 49}	1.7596330000×10^6	30	23.5207	$2.\times10^9$
	{104, 130, 47, 120, 5, 117, 33, 129, 2, 122, 74}	1.3808130000×10^6	171	60.0822	1.3808×10^6
	{6, 125, 9, 113, 40, 116, 11, 126, 12, 110, 1, 122, 31, 117, 3}	14068.0000	3	3.29195	13958
	{49, 121, 37, 125, 9, 113, 40, 116, 11, 126, 12, 110, 1, 122, 29}	14048.0000	3	3.70848	13958
	{49, 121, 48, 130, 54}	3.9735310000×10^6	586	427.521	$2.\times10^9$
	{98, 126, 11, 116, 40, 113, 9, 125, 22}	13998.0000	3	2.93792	13958
	{16, 122, 1, 110, 12, 126, 11, 116, 40, 113, 9, 123, 8}	13998.0000	3	3.22967	13958
	{32, 117, 31, 122, 60}	1.5754750000×10^6	201	210.148	1.57546×10^6
	{89, 129, 1, 110, 12, 128, 10, 113, 9, 125, 37, 121, 49, 130, 103}	483740.0000	33	41.7653	$2.\times10^9$
	{85, 129, 1, 110, 12, 126, 11, 116, 40, 113, 9, 123, 8}	13998.0000	3	3.22835	13958
	{96, 126, 11, 116, 40, 113, 51}	74613.0000	8	8.02067	74603
	{36, 114, 6, 125, 23}	40.	1	0.532063	0
	{104, 130, 47, 120, 5}	713602.0000	85	34.7893	1.0418×10^6
	{62, 124, 9, 117, 10, 111, 42, 114, 12}	4.0890910000×10^6	22	17.9143	$2.\times10^9$
	{102, 127, 46, 117, 33, 129, 4}	710656.0000	84	36.4281	710636
	{83, 128, 12, 116, 45}	1.4949800000×10^6	86	85.2079	$2.\times10^9$
	{44, 117, 2, 129, 1, 110, 12, 126, 11, 116, 40, 113, 51, 124, 109}	697891.0000	81	46.6312	697871
	{54, 130, 15, 121, 3, 112, 17, 123, 27, 120, 10, 117, 9, 124, 74}	4.0994060000×10^6	24	27.7137	$2.\times10^9$
	{72, 122, 2, 119, 1, 116, 13, 126, 25}	1.0233260000×10^6	1	2.35096	$2.\times10^9$
	{35, 111, 5, 120, 4}	177846.0000	1	1.03593	$2.\times10^9$
31	{34, 111, 6, 125, 9, 113, 40, 116, 11, 126, 12, 110, 1, 122, 31, 117, 32}	14068.0000	3	3.7212	38268
	{32, 117, 2, 129, 1, 110, 12, 126, 11, 116, 40, 113, 9, 125, 6, 111, 34}	14018.0000	3	3.65299	38268
	{102, 127, 46, 117, 5, 120, 7, 111, 36}	697941.0000	81	32.5583	697871
	{60, 122, 6, 117, 34, 129, 33, 119, 4}	1.6744160000×10^6	14	13.2357	$2.\times10^9$

Fig. 1. A sample of randomly chosen source-target gateway pairs with minimal (shortest end-to-end delay) paths found.

Figures 2 and 3 depict results obtained over 727 trials by generating random source-target pairs. 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. 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 GAR delay value 2×10^9, 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.

With these facts in mind, the results of the third row of Fig. 1 are in agreement, with the OC delay $177,816$ being equal to the GAR delay $177,806$ with one hop occurring in the final frame of consideration. Row 1 shows that in some cases the algorithm used to generate the GAR was unable to find a path between nodes nevertheless OC finds a solution. Row 31 shows that the OC approach leads to a better solution.

Figure 2 depicts the minimal end-to-end delays (in seconds)

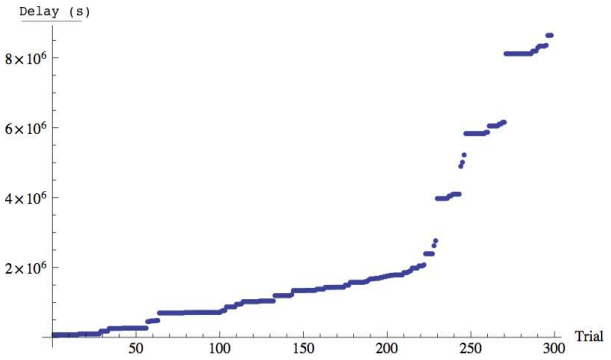

Fig. 2. End-to-end delays (in seconds) of paths obtained by the OC algorithm and non-existent in the GAR solutions. In 727 random trials, no GAR path was found for 298 source-target pairs. (Sorted by increasing min delay.)

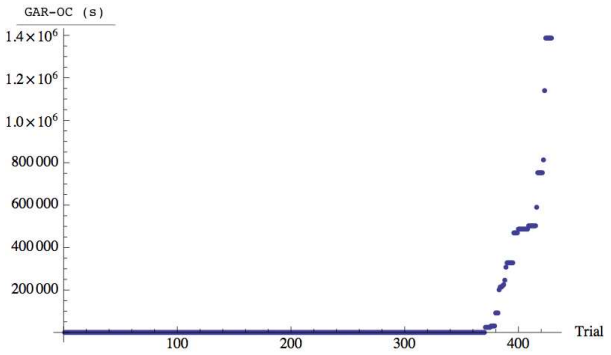

Fig. 3. Differences (in seconds) $GAR - OC$ between greedy algorithm minimum delays and OC delays among source-target pairs having GAR path data. (Sorted by increasing difference in min delay.)

obtained by the OC algorithm when no corresponding GAR paths were found. The values of end-to-end delay are very high. It means that the tree pruning algorithm was not able to build the branches associated to these solutions. The reason is that the routing tree does not include these solutions. Its size does not allow to store it in memory for analysis. Therefore the pruning process avoiding combinatorial explosion has eliminated the branches containing the solutions found by the OC algorithm.

The small values plotted in Fig. 3 indicate the agreement between GAR data and results from the OC algorithm, particularly in cases of paths found in the initial frame.

V. Conclusion

There are clear advantages to the OC approach to computing minimal paths as well as numerous avenues for further exploration:

A. Advantages of the Operator Calculus Approach

Exactness. As it is shown in the results the OC approach allows to solve efficiently the problem of the minimal end-to-end delay paths in the system of satellites communication. 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. [4]), 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. The S&F routing and in particular delay tolerant networks (DTN) can benefit of the OC approach. In these systems the construction of routes and the routing problem are dependent on availability of links between nodes along the time.

Multi-objective cost function. The OC algorithms presented here are single-constraint nevertheless the OC approach can be easily adapted to use multiple constraints to compute minimal paths. In the case of satellite systems and in communication systems in general it represents a great potential to construct multi-objective algorithms which may include multiple cost functions for efficient routing [8].

B. Ongoing Work

The optimization of the computation times for the OC algorithm is possible by developing heuristics that could still yield better results than existing methods. Of particular interest is obtaining a method of computing optimal paths in minimal time for dynamic routing.

Multi-constrained Quality of Service (QoS) problems can be modeled using sequences of graphs and can also benefit from the operator calculus approach. These results will be presented in a separate paper.

References

[1] I. Ali, N. Al-Dhahir, J.E. Hershey, Predicting the visibility of LEO satellites, *IEEE Transactions on Aerospace and Electronic Systems*, **35** (1999), 1183–1190.

[2] I. Chabini. Discrete Dynamic Shortest Path Problems In Transportation Applications: Complexity And Algorithms With Optimal Run Time, *Transportation Research Records*, **1695** (1998), 170–175.

[3] K.L. Cooke, E. Hasley, The shortest route through a network with time-dependent internodal transit times. *Journal of Mathematical Analysis and Applications*, **14**, (1966) 493–498.

[4] H. Cruz-Sánchez, L. Franck, A-L. Beylot, Precomputed routing in a Store and Forward satellite constellation, *2007 IEEE 66th Vehicular Technology Conference*, doi: 10.1109/VETECF.2007.64

[5] J. Desrosiers, P. Pelletier, F. Soumis, Plus court chemin avec contraintes d'horaires, *Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle*, tome 17, No. 4 (1983)

[6] R. Schott, G.S. Staples, Nilpotent adjacency matrices and random graphs, *Ars Combinatoria*, **98** (2011), 225–239.

[7] R. Schott, G.S. Staples, Nilpotent adjacency matrices, random graphs, and quantum random variables, *Journal of Physics A: Mathematical and Theoretical*, **41** (2008), 155205.

[8] R. Schott, G.S. Staples, Operator calculus in generalized zeon algebras: theory and application to multi-constrained path problems, *submitted* http://www.loria.fr/ schott/staceyQoSjune11.pdf.

[9] G.S. Staples, Clifford-algebraic random walks on the hypercube, *Advances in Applied Clifford Algebras*, **15** (2005), 213–232.

[10] G.S. Staples, A new adjacency matrix for finite graphs, *Advances in Applied Clifford Algebras*, **18** (2008), 997–991.